**IBM** ®

IBM Processor for ATM Resources

Revision 2.1

**Databook**

# Table of Contents

## Features

- Optimized for server applications.
- Configurable for sustained performance of up to 400Mb/s at full duplex.
  - Up to 65,534 independent Logical Channels.
  - Individual or group allocation of resources.
  - Firewall protection of packet memory storage and channel bandwidth.
  - Extensive support for Virtual Paths with VC bandwidth sharing.
- Scalable PHY interface
  - 8 and 16-bit Utopia interface (1 to 622MHz).
  - Cell HEC generation, checking, and correction included.
- PCI 32-bit interface up to 33MHz.
- Streaming 32-bit bus transfers with peak rate of 132MB/s.
- Two internal memory controllers for packet and control memory. Each supports 1-8MB of SRAM or ZBT SRAM or 4-128MB of EDO DRAM, SDRAM or ESDRAM. The controllers are inde-

pendent: one can use SRAM devices while the other uses DRAM devices. A single array of memory can be used in systems whose sustained full-duplex total bandwidth requirement is less than 102Mb/s.

- Supports AAL1, 2, 5, and null AALs with framing and scheduling extensions for MPEG-II.
- Supports Cell, Stream, FIFO, and Frame based Queuing.
- Supports reception in cell, FIFO and Frame increments.
- Received frames can be queued by Logical Channel or Group.
- Received frames can be queued after full reception or after header reception.
- Event Queue warns of potential problems and predicts the need for data movement without interrupt overhead.
- Configurable interrupts on events.
- TCP/IP Checksum built into memory controller.
- JTAG Test Interface

## Description

IBMIBM2520L8767 is an Asynchronous Transfer Mode Resource Manager (IBM2520L8767). It acts as an interface and translator between a Peripheral Component Interconnect (PCI) Bus and an ATM utopia or similar interface to an ATM PHY. This

device supports an integrated packet/frame memory (integrated DRAM controller; no glue required) and performs the Segmentation And Reassembly (SAR) functions for several of the ATM Adaptation Layers (AALs).

## ATM Subsystem Block Diagram (Please see page 10 for descriptions of subsystems.)
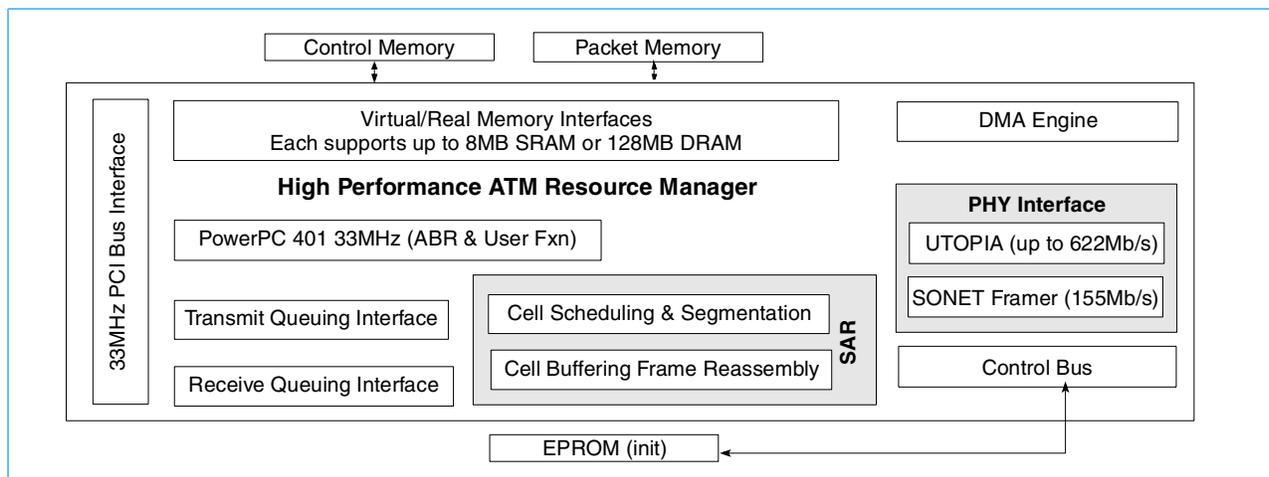
## Ordering Information

| Part Number | Description |
|---|---|
| IBM2520L8767 | Asynchronous Transfer Mode Resource Manager |

## Conventions

Throughout this book the bit notation is non-IBM, meaning that bit zero is the least significant bit and bit 31 is the most significant bit for a four-byte word.

The internal addressing view of IBM2520L8767 registers and memory is big-endian. In most cases, a system will wire its PCI bus interface to make the register view transparent, *i.e.,* the most significant bit in this specification will be the most significant bit in the register. If registers are read and written 32 bits at a time (which is the only way to access many of the registers), then the endian-ness should not be a programming issue with respect to the registers.

The IBM Processor for ATM Resources DMA controller can transfer data in either big-endian or little-endian mode. (See *GPDMA Control Register* on page 135 for details.)

Overbars, eg. $\overline{\text{CAS}}$, designate signals that are asserted "low".

## Standards Compliance

The IBM IBM Processor for ATM Resources, part number IBM2520L8767, has been designed with a number of standards in mind. For additional information, please see:

- At the network side, IBMIBM2520L8767 complies with the ATM standards and recommendations defined by ITU-TS (formerly CCITT), ANSI and ATM Forum. The following is a list of standard documents that the IBM Processor for ATM Resources design point is based on:
  - ITU Rec. I-361 - B-ISDN ATM layer specification
  - ITU Rec. I.362 - B-ISDN ATM Adaptation Layer (AAL) functional description
  - ITU Rec. I.363 - B-ISDN ATM Adaptation Layer (AAL) specification
  - ITU Rec. I.413 - B-ISDN user-network interface
  - ITU Rec. I-432 - B-ISDN user-network interface - Physical Layer specification
  - ITU Rec. I-610 - OAM principles of B-ISDN access
  - ANSI T1.ATM-199x Draft, Broadband ISDN - ATM Layer Functionality and Specification
  - ANSI T1.CBR-199x Draft, Broadband ISDN - ATM Adaptation Layer for Constant Bit Rate Service Functionality and Specification
  - ATM Forum 93-620R2 - ATM User-Network Interface Specification - Version 2.3 (July 27, 1993)
  - Bellcore TA-NWT-001248 Generic Requirements for Operations of Broadband Switching Systems (October 1993)

- At the system interface side, IBMIBM2520L8767 complies with the following PCI-bus architecture:
  - PCI Local Bus Specification, Production Version, Revision 2.1, June 1, 1995. Interface Technical Reference, 11/89, Part number 15F2160.

- The PHY interface of IBMIBM2520L8767 conforms to the following specifications:
  - SATURN User Network Interface, PMC-Sierra, Inc., February 1995
  - ATM Forum 93-727 An ATM PHY Data path interface, Version 2.01, March 24, 1994
  - Am7968/Am7969 TAXIchip(tm) Handbook, Transparent Asynchronous Transmitter/Receiver Interface, published by Advanced Micro Devices, 1994.

# Environmental Ratings

## Absolute Maximum Ratings

| Parameter | Rating | Unit | Note |
|---|---|---|---|
| Supply Voltage | -0.5 to 3.6 | Volt | 1 |
| Input Voltage Applied | -0.5 to 5.5 | Volt | 1 |
| Storage Temperature | -65 to 150 | °C | 1 |
| Ambient Temperature with Power Applied | -40 to 100 | °C | 1 |
| ESD Voltage | 3000 | Volt | 1 |
| 1. These are the maximum ratings that can be applied to the device without damage. The device function and specifications are valid only within the Recommended Operating Conditions. | | | |

## Recommended Operating Conditions

| Parameter | Rating | Unit |
|---|---|---|
| Junction Temperature | 0 to 85 | °C |
| Supply Voltage ($V_{DD}$) with respect to Ground | $3.3 \pm 2\%$ | Volt |

## Power Dissipation

| Parameter | Rating | Unit |
|---|---|---|
| Total (nominal) | 2 | Watt |
| Total (maximum) | 3 | Watt |

## Package Diagram

**Top View**

20L8767

A01 Locator
also No Ball

25mm ± 0.2

32.5mm ± 0.2

**Side View**

THIS VIEW

Linear Tolerances: +- 0.5mm

0.9 ± 0.1

1.75 min

2.15 max

**IBM2520L8767**
**IBM Processor for ATM Resources**
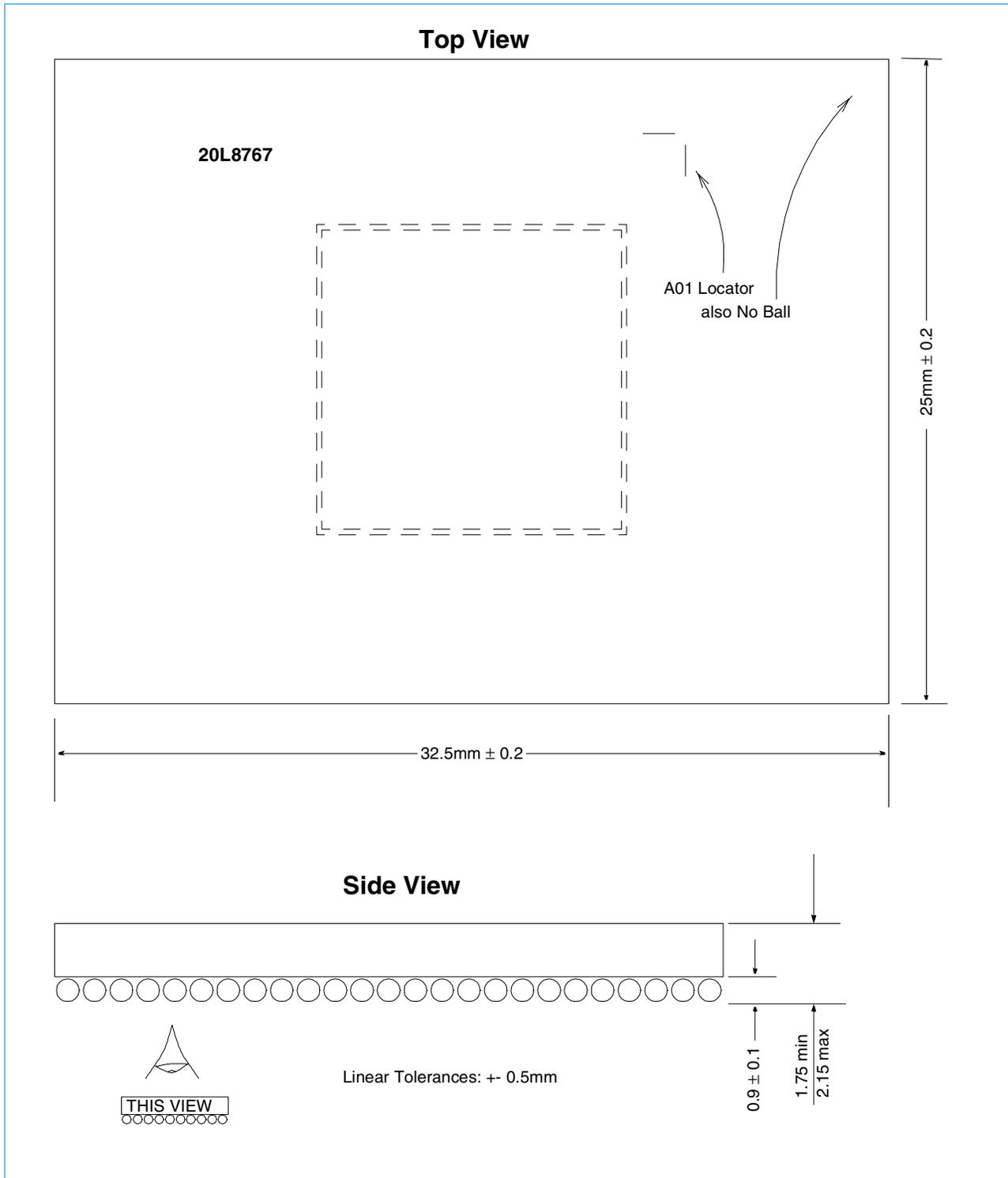
## Pinout Viewed from Above
Pin functions are listed in *Input/Output Definitions* on page 19. Complete lists of pin assignments are in *Pin Assignments and DC Characteristics* on page 516.



This View

○ GND    ⊗ No Connect    ◍ V$_{DD}$    Linear Tolerances: ± 0.5mm
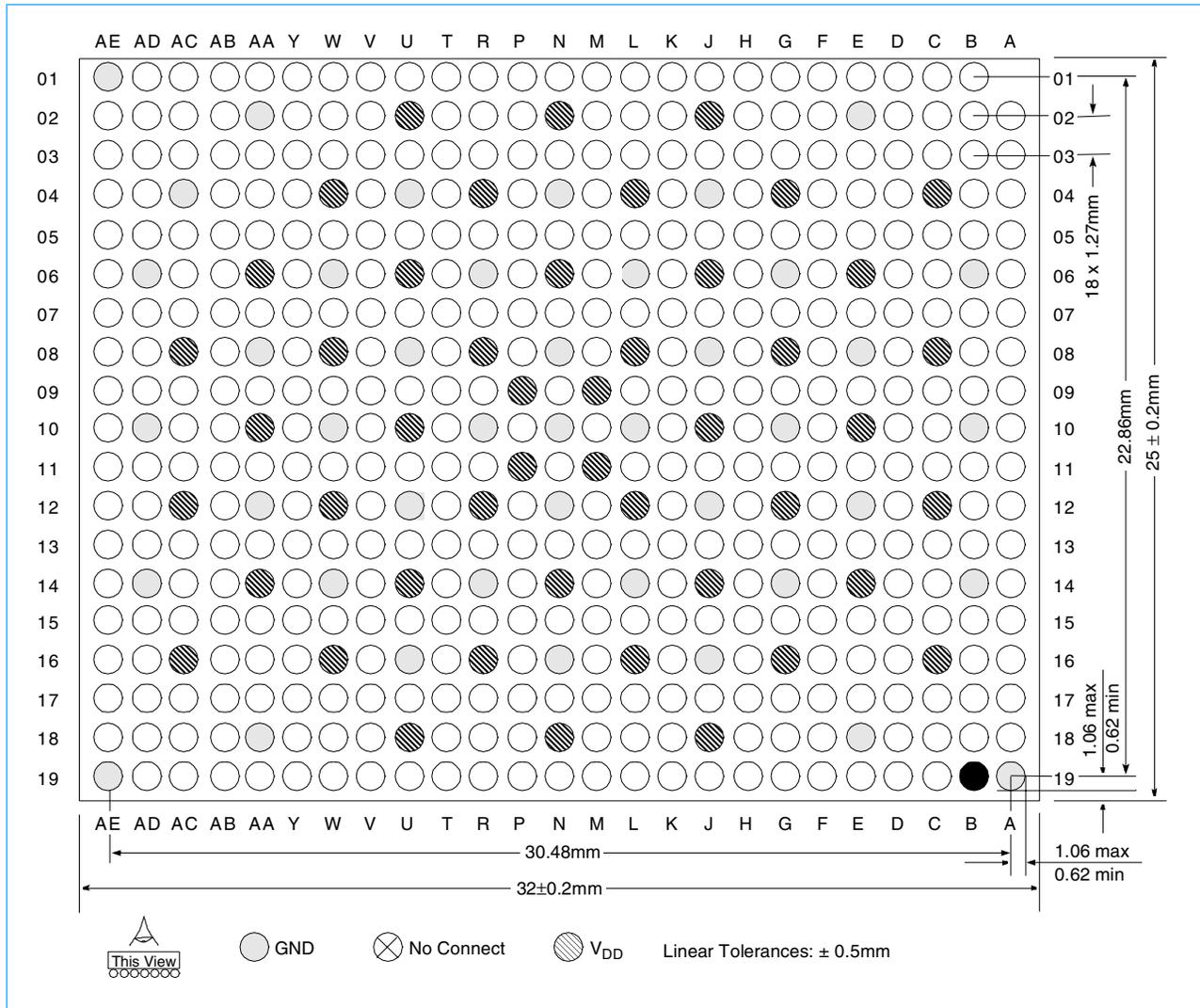
## Ground Pin Locations

| | | | | | |
|---|---|---|---|---|---|
| 0A19 | 0B06 | 0B10 | 0B14 | 0E02 | 0E08 |
| 0E12 | 0E18 | 0G06 | 0G10 | 0G14 | 0J04 |
| 0J08 | 0J12 | 0J16 | 0L06 | 0L10 | 0L14 |
| 0N04 | 0N08 | 0N10 | 0N12 | 0N16 | 0R06 |
| 0R10 | 0R14 | 0U04 | 0U08 | 0U12 | 0U16 |
| 0W06 | 0W10 | 0W14 | AA02 | AA08 | AA12 |
| AA18 | AD06 | AD10 | AD14 | AE01 | AE19 |

**Pinout Viewed from Below** Pin functions are listed in *Input/Output Definitions* on page 19. Complete lists of pin assignments are listed in *Pin Assignments and DC Characteristics* on page 516.

| AE | AD | AC | AB | AA | Y | W | V | U | T | R | P | N | M | L | K | J | H | G | F | E | D | C | B | A |

GND     No Connect     V$_{DD}$

This View
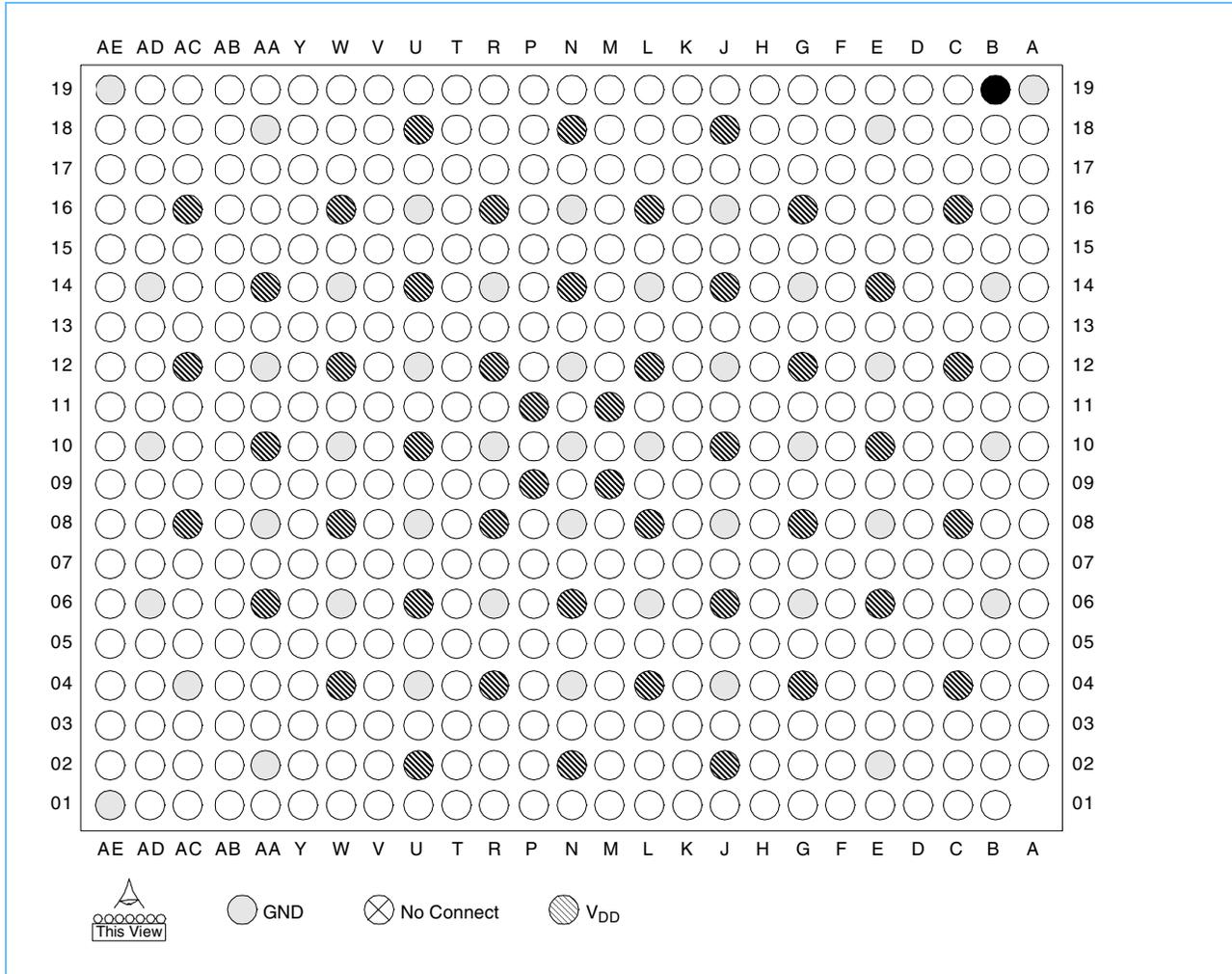
# V$_{DD}$ Pin Locations

| | | | | | |
|---|---|---|---|---|---|
| 0C04 | 0C08 | 0C12 | 0C16 | 0E06 | 0E10 |
| 0E14 | 0G04 | 0G08 | 0G12 | 0G16 | 0J02 |
| 0J06 | 0J10 | 0J14 | 0J18 | 0L04 | 0L08 |
| 0L12 | 0L16 | 0M09 | 0M11 | 0N02 | 0N06 |
| 0N14 | 0N18 | 0P09 | 0P11 | 0R04 | 0R08 |
| 0R12 | 0R16 | 0U02 | 0U06 | 0U10 | 0U14 |
| 0U18 | 0W04 | 0W08 | 0W12 | 0W16 | AA06 |
| AA10 | AA14 | AC04 | AC08 | AC12 | AC16 |

# Block Diagram

IBM

PCI Bus

```
CRSET
CBIST
SCLCK
CJTAG
```

```
PCINT
```

```
DMAQ
DMA
GPDMA
```

```
SRAM
```

```
401
CORE
```

```
BCACH
Bus Cache
```

```
Interrupt
Status
INTST
```

```
DRAM
Control
```

```
Control
DRAM
```

```
Memory
Pool
Control
POOLS
```

Internal Memory Bus

```
Arbiter
COMIT/VMEM
```

```
Frame
DRAM
```

```
RXQUE
Rx Queue Management
```

```
Checksum
and
DMA CHKSM
```

```
RAAL
AAL Processing
```

```
CSKED
Cell Scheduler
```

```
REASM
Reassembly
```

```
SEGBF
Segmentation
```

```
Internal
SONET
Framer
```

```
NPBUS
```

```
LINKC
Asynchronous Cell Interface
```

PHY Control/
EPROM access

8/16 bit
PHY Interface

Serial
Interface

# Functional Description

The IBM Processor for ATM Resources acts as a conversion unit from a bus memory interface (which is Work Queue oriented) to a PHY level ATM. To accomplish this, the IBM2520L8767 contains the major functional units listed below and shown in the *Block Diagram* on page 8.

Control Processor Bus Interface

| | |
|---|---|
| PCINT | PCI Interface entity |
| INTST | Interrupt Status entity |
| GPDMA | General Purpose DMA entity |
| DMAQS | Queue control for DMA activity |

Memory Control

| | |
|---|---|
| COMET/PAKIT | DRAM Controlling entity |
| VIMEM | Virtual Memory controller |
| ARBIT | Memory Subsystem Arbitration requestor |
| BCACH | Bus Cache entity |
| POOLS | Memory Pool manager |

Transmit Data Path

| | |
|---|---|
| CSKED | Cell Scheduler |
| SEGBF | Cell Segmentation entity |

Receive Data Path

| | |
|---|---|
| REASM | Cell Re-assembly entity |
| RAALL | AAL processor |
| RXQUE | Receive Queue manager |

PHY Level Interfaces

| | |
|---|---|
| LINKC | Asynchronous Physical Layer interface |
| NPBUS | Nodal Processor Bus interface |
| FRAMR | Full SONET framing support logic |

Hardware Protocol Assist

| | |
|---|---|
| CHKSM | TCP/IP Checksum Logic |
| PCORE | Embedded 401 Processor Core |

Base Device Functions

| | |
|---|---|
| SCLCK | The System Clock Generation and Repowering entity |
| CRSET | Hardware and Software Reset Controlling entity |
| CBIST | Built-In Self Test logic entity |
| CJTAG | JTAG test interface logic entity |

## Subsystem Blocks

The IBM IBM Processor for ATM Resources has the following four interfaces.

**The IBM Processor for ATM Resources** provides the host bus interfacing, memory management for buffers and control, cell segmentation and reassembly, and PHY hardware control for an ATM adapter.

**External Memory**, consisting of two DRAM, SDRAM, or SRAM arrays used for the storage of packet data and the control structures used by the IBM Processor for ATM Resources. Both the packet and control memory arrays consist of two, 32-bit wide banks.

When running at 102Mb/s or slower (full duplex aggregate throughput), a single array of memory can be used. Both control and data store would be contained in this single array of memory. For a detailed description of the external memory organization refer to *The DRAM Controllers (COMET/PAKIT)* on page 141.

**The PHY (Physical) Layer**, which connects to several available hardware support devices. This layer of hardware converts a parallel data stream into a serial data stream to be shipped to and from the PMD layer.

The PHY and PMD end of a card design can be implemented as one of several encoding schemes and speeds, supporting both copper and fiber optic serial links. The interface will support the ATM Forum "Utopia spec," The PMC chip, the AMD Taxi Chip set, and possibly a 25Mb/s serial interface to the IBM UTP solution. (See *Standards Compliance* on page 3 for documents which describe these interfaces.)

**The PMD (Physical Media Dependent) Layer**, which connects to the line drivers and receivers. This could be either a copper or a fiber optic transceiver.

## External Architecture

The IBM IBM Processor for ATM Resources has four major interfaces:

**A System Bus** which will act as an actively cached memory slave and and as a master for the PCI 32-bit bus.

**The Physical (PHY) Interface** which supports several physical layer hardware devices that perform parallel to serial data conversion and the rest of the Transmission convergence.

**An External DRAM Interface** that controls one or two arrays of 2-bank interleaved DRAM with 60-ns access time for packet and control memory. The interface is direct drive to the DRAM.

**The Control and Configuration Interface** which covers a number of functions. It gives access from the system bus to the PHYs and to EPROM. The EPROM can also be used to hold initial device configuration, up to and including PVC configurations.

These four interfaces allow the IBM Processor for ATM Resources to be used in both "deep" and "shallow" adaptors with minimal external logic. (See *Block Diagrams of Possible Systems* on page 12 for examples.)

## Internal Architecture

### Logical Channel Support

The Logical Channel is the unit of resource allocation in ATM. At one level, the End Station negotiates with the Network Interface to determine the characteristics of each end Station to End Station connection. The resources that may be reserved in the network are defined in the ATM UNI (User Network Interface) Specification (see references in *Standards Compliance* on page 3). These resources include (but are not limited to) the peak and average bandwidth to be used by the logical channel, the maximum burst length that may be transmitted at the burst rate, the latency and variance of the connection, and the loss probability.

The term Logical Channel rather than virtual circuit or VPI/VCI is used in this databook to provide a level of abstraction from these specific instances.

A Switched Virtual Circuit (SVC) can be negotiated with specific characteristics specifically for it.

A virtual path can be negotiated with the network, and several virtual circuits within that path can then be multiplexed using the VCI on that single VPI without having to renegotiate for each additional VCI. The Logical channel with respect to the network would be the Virtual Path. There would be multiple logical channels internal to the end station based on the Virtual Circuits used within the path.

Using ATM Adaptation Layers 3 and 4, a Multiplexing IDentifier (MID) can be used to provide multiple Logical Channels across a single VPI/VCI.

All of these Logical Channels are dealt with uniformly in the IBM2520L8767. A hierarchy of Logical Channel Descriptors can be built up, and Frames or buffer can be queued to each of the LCDs. See *Transmit Cell Scheduler (CSKED)* on page 223 for details.

### Virtual Memory Support

The Packet memory space appears on the bus as if it is up to 64K buffers each of which can (architecturally) appear to be 64K bytes long. A level of indirection has been added to the addressing of Packet memory to provide these large frame buffers without requiring memory behind all of them at the same time. This has been done for a number of reasons:

- The Frames on the network can be up to 64KB long.
- The receiver does not know how long a frame will be until it is completely received.
- Software generally has a much easier time of dealing with contiguous memory.

The memory does not page or swap. There two major efficiencies used internally:

- The first N bytes of memory in a buffer are directly referenced.
- The blocks that make up the buffers are of multiple sizes.

**Queues**

The IBM2520L8767 makes extensive use of three types of cached single memory operation atomic queues:

| | |
|---|---|
| Transmit queues | The interface to the scheduling entity. Blocks and Frames can be queued to Logical Channels. |
| Receive queues | Based on the settings in the Logical Channel Descriptor (receive side), cells arriving can be queued individually, collected into frames, of stored in FIFO buffers. |
| Event queues | When a frame is transmitted, its memory can be "garbage collected" or a reference to the frame can be placed on an event queue for software to handle. If either a FIFO buffer scheme or frame buffer scheme is used to source or sink data on a logical channel, it is possible to set thresholds on the buffering that will cause events to be queued. When a threshold is crossed, for instance if a transmitting LC is about to run out of data to transmit, an event will be queued. Software can read these events either by polling or by being interrupted and can schedule tasks to provide more data. Events can be scheduled on the reception of the first N bytes of a frame so that header processing can begin even before the complete frame is received. This will allow "cut-through" routing to be supported. |

**Scheduling**

There is extensive support for transmit scheduling. Please see *Transmit Descriptor Data Structures* on page 39 and *Transmit Packet Header Structure* on page 35 for details.

# Block Diagrams of Possible Systems



**Shallow Adapter for 155Mb/s**

PCI Bus

Memory 2-256MB ECC — ATM-RM Device → Optical Transceiver

Short PCI Card

**100 Mb/sec Card Using a TAXI Chipset**

PCI Bus

Memory 2-256MB ECC — ATM-RM Device → Transmit Device / Receive Device → Optical Transceiver

Short PCI Card
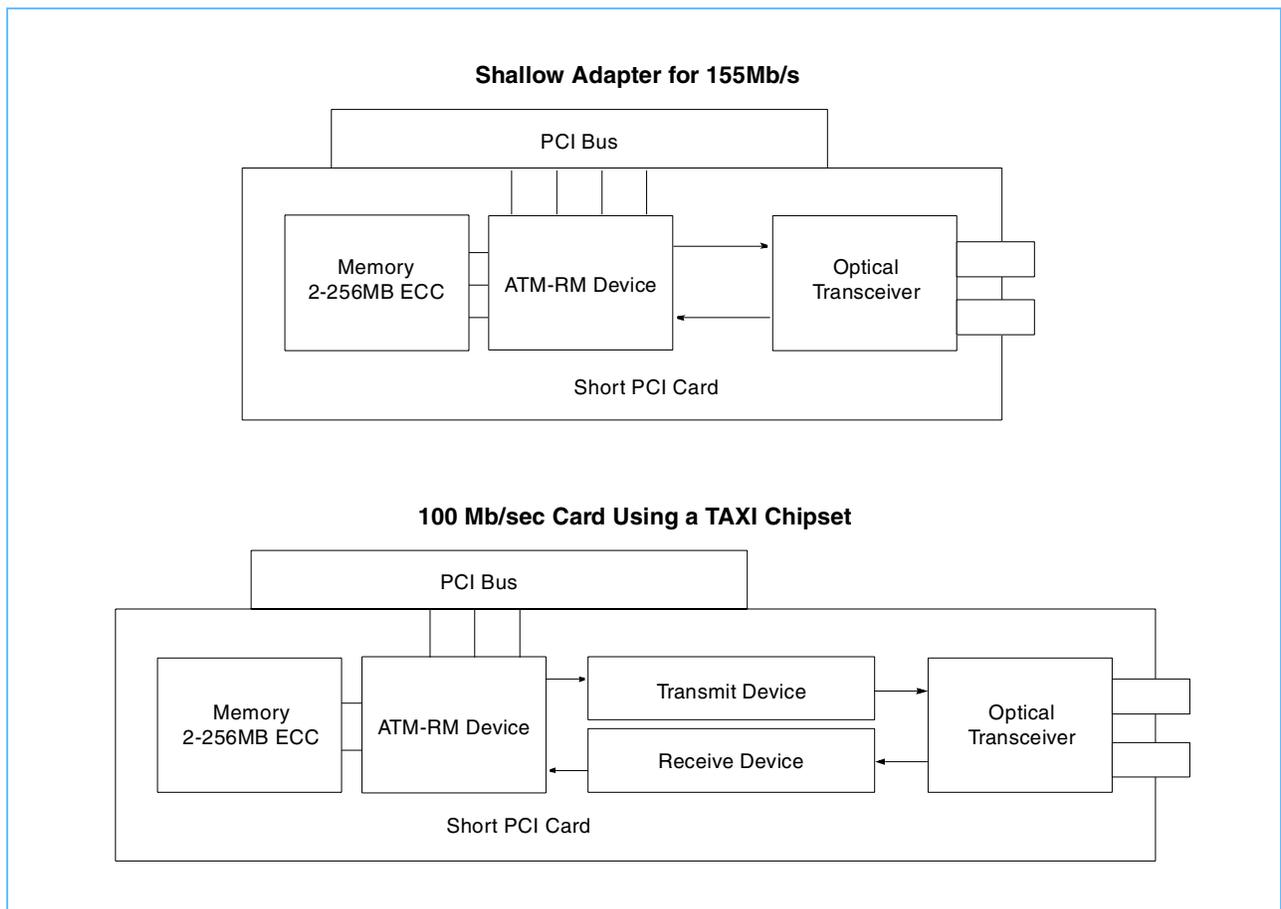
## MPEG Video Compression and Distribution Considerations

The ATM Forum has specified mechanisms to be used for the delivery of video streams. They are based on the MPEG-II video compression standard which produces a variable rate video stream. The challenge is to ensure that the delivery of this stream is performed effectively.

The figure below shows a six-second sequence of an MPEG encoded video. It clearly shows the scheme employed MPEG encoding; every twelfth frame transmitted is an "I" or Image frame. This frame is a complete compressed version of its basis frame. In between each "I" frame there are three "P" or Predicted frames that predict, based on the previous "I" and "P" frames, what the image should look like, and then compare the prediction with the actual basis frame. What is transmitted in the "P" frame is the difference between the prediction and the actual data. The third frame type is the "B" or between/both frame. These frames are also difference-from-prediction frames, but they use information about frames in both directions in time. Because the "B" frames are interpolations rather than projections, the predictions are usually much more accurate than the "P" frames, so the differences from the predictions are generally smaller. The variations in frame size can be seen due to differences in the compression quality, or compressibility, of the sepecific video.

A number of issues require consideration in the attempt to support digital delivery of video. One "hook" that the IBM Processor for ATM Resources includes is a delivery mode that stays within the traffic specification for traffic shaping, but attempts to deliver these variable sized frames on fixed time intervals.

## ATM Subsystem Dataflow
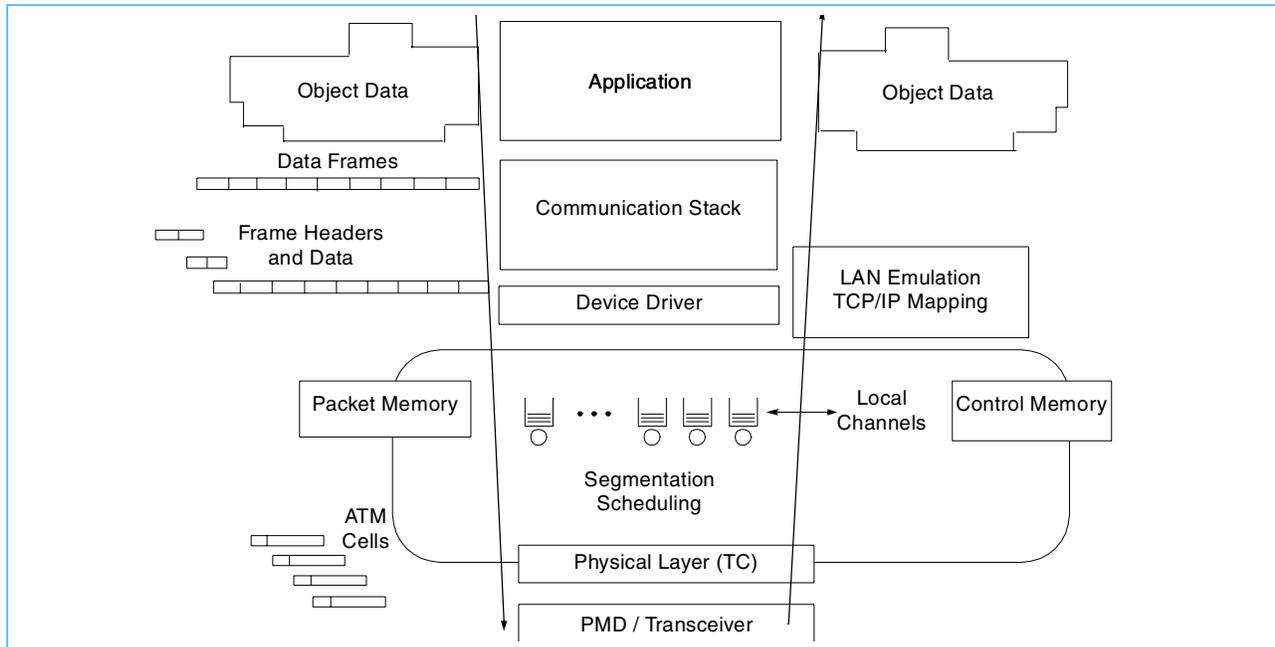


As shown in the figure, Data, in the form of application objects or control structures, are divided into communication frames at the communication stack interface. The stack may further partition the frames to fit reliability, efficiency, latency, and protocol requirements.

In most cases, the communication stack encapsulates the data frame with protocol headers and/or trailers. These header blocks are often located in memory in areas apart from the Data frames. A device driver is often given the task of moving this scattered memory to the actual transmission device. Scatter DMA is often used to make this operation efficient.

With an IBM2520L8767 in the ATM Subsystem, the data can be DMAed into virtually contiguous buffers connected to and controlled by the IBM2520L8767. It is also possible to write the frame headers directly from the processor to the IBM2520L8767 memory. The fully assembled frame is queued for transmission over a particular logical channel. (See more on the richness of logical channels in ATM and the IBM2520L8767 in *Data Structures* on page 35).

The logical channels with pending work are serviced by the ATM Segmentation Layer which breaks the enqueued data into 48-byte chunks (depending on the ATM Adaptation Layer (AAL)) and prefixes it with a five-byte header in preparation for transmission (yielding 53-byte packets).

A Transmission Convergence (TC) Sublayer appropriate for the Physical Layer (PHY) and Physical Media Dependent (PMD) connection is then exercised, making ATM cells suitable for transmission.

The receiving process is the reverse of the transmission process, except that the scheduling performed during transmission is replaced by an identification-demultiplexing step during the reception of cells.

*Isochronous/time-based support:* Note that not all of these separate parts or steps described in this section are necessary for a dedicated function system. The IBM2520L8767 can easily be used in dedicated systems due to it goal of minimal processor intervention for steady state operations.

# Data Flows

This section describes the data and control flow to and through the IBM Processor for ATM Resources. In order for cell traffic to flow through an ATM interface, the cells require that Logical Channels be allocated. For information on Logical Channels, please see *Data Structures* on page 35.

Feature Summary
- Virtual memory
- Memory pools
- Register read/write interface for memory allocation
- Transmit path scheduling
- Receive path demultiplexing
- Event Queues

Operation Summary
- Basic Assurance Tests (BATs)
- Initialize and cConfigure
- Test Path to Switch
- Permanent Virtual Circuit Setup
- Identify LAN Servers
- Initialize SVCs
- Run, initializing circuits (Q.93B) and transmitting data.

## Transmit Path

A typical transmit operation begins with the software requesting a buffer from POOLS and filling it with data via slave DMA, master DMA, or processor writes. If virtual buffers are being used, the data write operation can fail due to lack of physical buffers. In the event of a failure, the header of the packet is updated to indicate the failure. The software can audit the header after the buffer has been completely transferred, and either take action to recover the data immediately, or allow CSKED to generate an event later in the transmit cycle for any buffers that have had a data write failure.

Before the data can be transmitted, the buffer header must be updated to contain information required for correct transmission. Information such as data length, starting offset, and Logical Channel (LC) address are just a few of the fields that must be correctly reflected in the buffer header. For a complete list of the fields in the buffer header refer to *Packet Header* on page 35.

In addition to the fields in the buffer header, the scheduling and segmentation sections of the Logical Channel Descriptor (LCD) such as peak rate, average rate, and AAL type must also be set up correctly prior to transmission. For a complete list of the fields in the LCD, refer to *Transmit Descriptor Data Structures* on page 39.

After the data has been transferred into packet storage and both the buffer header and the LCD structure have been correctly initialized, the buffer address is queued to CSKED. When it receives a buffer, CSKED checks the buffer header (packet memory) to make sure that the data transfer operation that filled the buffer completed without error. If it finds an error, CSKED posts an event to software and does nothing further with this buffer. If it does not find an error, CSKED fetches several fields from the LCD (control memory) indicated in the buffer header to determine the current state of that LCD. If the LCD is busy sending another buffer, the new buffer is queued to this LCD and will be processed when all previously enqueued buffers have been transmitted. If the LCD is not busy, CSKED updates the LCD based on several fields in the buffer header and queues the LCD to the next time slot on the time wheel (control memory).

When CSKED detects a previously enqueued LCD on the time wheel, several fields are retrieved from the LCD. Among other things, these fields are used by CSKED to determine where on the time wheel to reschedule this LCD. The LCD address is then provided to SEGBF for processing.

When CSKED provides an LCD address to SEGBF, the segmentation portion of the LCD is retrieved from control memory to determine both the current address at which to continue buffer segmentation and the type of cell to construct. Depending on the AAL type bits in the segmentation portion of the LCD, the cell is constructed in an internal array using data from the LCD as well as data fetched from packet memory. When the cell construction is complete, status is raised to LINKC indicating that a new cell is available for transmission.

Transmit opportunities are repeatedly provided to SEGBF by CSKED at the desired rate until all the data in the buffer has been passed to LINKC via the cell buffer array. When SEGBF detects that no more data exists for a buffer, the LCD address is passed back to CSKED, indicating buffer completion. If no more buffers are queued, CSKED removes the LCD from the time wheel. If more buffers are queued, the LCD is updated and the segmentation process continues until all buffers on the LCD queue are used. A bit in the buffer header generates a transmit complete event when no buffers remain in the queue.

## Transmit Scheduling Capabilities



**Transmit Queues and Logical Channel Traffic Shaping**
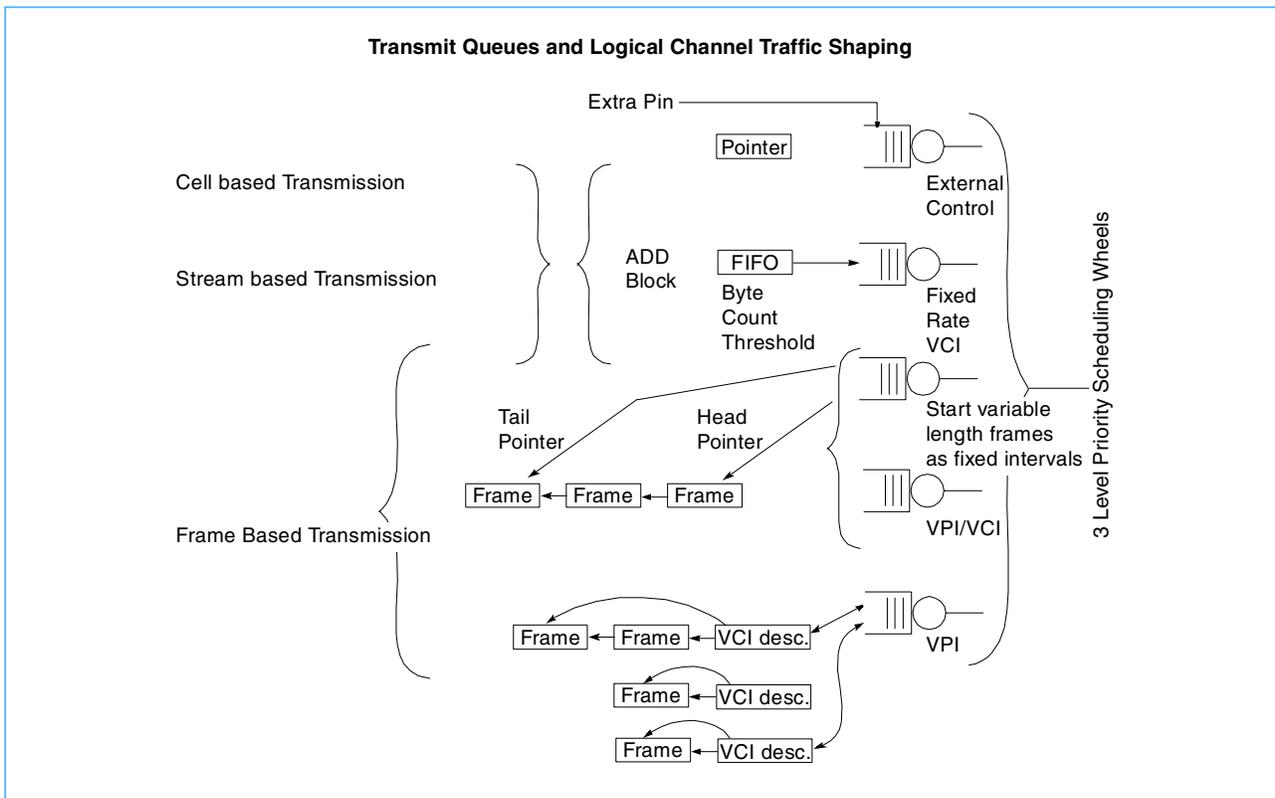
## Receive Path

As cells arrive, they pass from LINKC to REASM. REASM uses a portion of the ATM header to look up the LCD address for this cell. The LCD address is then passed to RAALL. RAALL reads the receive portion of the LCD, and then processes the cell based on the LCD information. For example, the LCD specifies what AAL to use and maintains the current reassembly state. Using the current reassembly state, the cell data is written to packet memory. While the data is written to packet memory, other functions such as CRC generation and verification are performed in parallel. If a packet is complete, all trailer verification is performed. If the packet is good, an event is placed on a receive queue in the RXQUE entity. For error scenarios, see Entity 14: *Receive Queues (RXQUE)* on page 300 and Entity 13: *Receive AAL Processing (RAAL)* on page 274. At this point, software can dequeue the packet event from RXQUE using the dequeue operation. It can then examine headers, DMA the data into user space, and perform TCP checksums. When these actions are complete, the buffer is returned to the IBM2520L8767 by performing a POOLS free buffer operation.

# Input/Output Definitions

The several interfaces to the IBM Processor for ATM Resources are described in the following sections. There are 383 active I/O pins, assigned as follows:

- 89 for the PCI bus,
- 21 for the NPBUS,
- 57 for the PHY bus,
- 142 for the DRAM memory interface, and
- 42 strictly for configuration and testing.

Cross references between the physical pin assignments and pin signal names are listed in *Pin Assignments and DC Characteristics* on page 516.

## PCI Bus Connections

## PCI Bus Interface Pin Descriptions

| Quantity | Pin Name | Input/Output | Pin Description |
|---|---|---|---|
| 1 | MFRAME | S/T/S | Cycle Frame is driven by the current master to indicate the beginning and duration of an access. |
| 4 | PCBE(3-0) | T/S | Bus Command and Byte Enables are multiplexed on the same PCI pins. During address phase they define the bus command and during the data phase they define the byte enables. |
| 1 | MSERR | O/D | System Error reports address parity errors, data parity errors on the Special Cycle command, or any other system error where the result will be catastrophic. |
| 32 | PAD(31-0) | S/T/S | Address and Data are multiplexed on the same pins. A bus transaction consists of one address phase and one or more data phases. |
| 1 | PPAR | T/S | Parity is even parity across ad(31-0) and C/BE(3-0). Parity generation is required by all PCI agents. |
| 1 | MPERR | S/T/S | Parity Error is for reporting data parity errors during all PCI bus transactions except Special Cycle. |
| 1 | MINTA | O/D | Interrupt A is used to request an interrupt. |
| 1 | MINT2 | O/D or S/T/S | The is an interrupt line that will go active low when sources within the IBM2520L8767 go active. It can be optionally   connected to PCI interrupt B. See Entity 2: *Interrupt and Status/Control (INTST)* on page 95 for more details. |
| 1 | PIDSEL | Input | Initialization Device Select is a chip select during configuration transactions. |
| 1 | MDEVSEL | S/T/S | Device Select indicates the driving device has decoded its address as the target of the current transaction. |
| 1 | MTRDY | S/T/S | Target Ready signals the target agent's ability to complete the current data phase of the transaction. |
| 1 | MIRDY | S/T/S | Initiator Ready indicates the bus master's ability to complete the current data phase. |
| 1 | MLOCK | S/T/S | Lock indicates an atomic operation that may require multiple transactions to complete. If this is IBM2520L8767 cascade mode, this bit functions as MSREQ (secondary request), which the primary IBM2520L8767 will receive from the secondary IBM2520L8767 to then request the main PCI bus, or as MSGNTGI (secondary grant gate in). See PCINT Cascade Control Register for more details. |
| 1 | MSTOP | S/T/S | Stop indicates the current target is requesting the master to stop the current transaction. |
| 1 | MGNT | IN | Receives the Bus Grant line after a request has been made. |
| 1 | MREQ | S/T/S | Requests the Bus for an Initiator transfer. |
| 32 | PAD64(63-32) | S/T/S | Address and Data are multiplexed on the same pins and provide 32 additional bits. Also, this pins are multiplexed with the ENSTATE outputs, that allow debug of various internal state machines and   signals. |
| 4 | PCBE64(7-4) | T/S | Bus Command and Byte Enables are multiplexed on the same PCI pins for 64 bit transfer support. |
| 1 | MREQ64 | S/T/S | Request 64-bit transfer. Has the same timing as MFRAME. |
| 1 | MACK64 | S/T/S | Acknowledge 64-bit transfer. Has the same timing as MDEVSEL. |
| 1 | PPAR64 | S/T/S | Parity Upper DWORD is the even parity bit that protects MAD64(63-32) and PCBE(7-4). When not on a PCI bus supporting 64 bits, this will drive ENSTATE outputs. |

## DRAM Memory Bus Interface

One control memory and one packet memory bus provide the attachment to the external DRAM. Up to two arrays of 32 data bits plus potential error detection bits may be connected to each bus. Each bus consists of address, data, and control lines as illustrated below. See *Memory I/O Cross Reference By Device Type* on page 23 for the use of a particular signal with a particular memory type.

## DRAM Memory Bus Connections

## DRAM Memory Bus Interface Pin Descriptions

| Quantity | Pin Name | Input/Output | Pin Function | Pin Description |
|---|---|---|---|---|
| 2 | $\overline{PMRAS}$(1:0) | Output | $\overline{PMRAS}$(1:0) | Used for the two packet memory arrays. |
| 2 | $\overline{CMRAS}$(1:0) | Output | $\overline{CMRAS}$(1:0) | Used for the two control memory arrays. |
| 2 | $\overline{PMCAS}$(1:0) | Output | $\overline{PMCAS}$(1:0) | Used by the two packet memory arrays. |
| 2 | $\overline{CMCAS}$(1:0) | Output | $\overline{CMCAS}$(1:0) | Used by the two control memory arrays. |
| 1 | $\overline{PMSYNRAS}$ | Output | $\overline{PMSYNRAS}$ | The RAS signal for packet synchronous DRAM. |
| 1 | $\overline{CMSYNRAS}$ | Output | $\overline{CMSYNRAS}$ | The RAS signal for control synchronous DRAM. |
| 1 | $\overline{PMSYNCAS}$ | Output | $\overline{PMSYNCAS}$ | The CAS signal for packet synchronous DRAM. |
| 1 | $\overline{CMSYNCAS}$ | Output | $\overline{CMSYNCAS}$ | The CAS signal for control synchronous DRAM. |
| 2 | $\overline{PMWE}$(1:0) | Output | | Packet memory write enable. |
| 2 | $\overline{CMWE}$(1:0) | Output | | Control memory write enable. |
| 4 | $\overline{PMCS}$(7:4) | Output | | Packet memory SRAM chip selects. |
| 4 | $\overline{CMCS}$(7:4) | Output | | Control memory SRAM chip selects. |
| 1 | PMCLK | Output | | Packet memory clock. |
| 1 | PMCLKRP | Output | | Packet memory clock repowered. |
| 1 | CMCLK | Output | | Control memory clock. |
| 1 | CMCLKRP | Output | | Packet memory clock repowered. |
| 18 | PMADDR(17:0) | Output | Address signals to packet memory. | When 1Mx16 DRAM modules are used:12 row-address bits latched by the row address strobe (RAS). 8 column-address bits latched by the column address strobe (CAS). When 256Kx16 modules are used: 9 row-address bits latched by the row address strobe (RAS). 9 column-address bits latched by the column address strobe (CAS). |
| 18 | CMADDR(17:0) | Output | Address bus to control memory. | When 1Mx16 DRAM modules are used:12 row-address bits latched by the row address strobe (RAS). 8 column-address bits latched by the column address strobe (CAS). When 256kx16 DRAM modules are used: 9 row-address bits latched by the row address strobe (RAS). 9 column-address bits latched by the column address strobe (CAS). |
| 36 | PMDATA(35:0) | Input/Output | Data signals to and from the packet memory. | |
| 3 | PMDATA2(38:36) | Input/Output | Data signals to and from the packet memory. | |
| 36 | CMDATA(35:0) | Input/Output | Data signals to and from the control memory. | |
| 3 | CMDATA2(38:36) | Input/Output | Data signals to and from the control memory. | |

## Memory I/O Cross Reference By Device Type

| IBM2520L8767 I/O | EDO DRAM | Sync DRAM 2-Bank Device | Sync DRAM 4-Bank Device | SRAM |
|---|---|---|---|---|
| xxADDR(17:13) | Address(17:13) | Address(16:12) | Address(15:11) | Address(17:13) |
| xxADDR(12) | Address(12) | Address(11) | Bank Select 1 | Address(12) |
| xxADDR(11) | Address(11) | Bank Select 0 | Bank Select 0 | Address(11) |
| xxADDR(10:0) | Address(10:0) | Address(10:0) | Address(10:0) | Address(10:0) |
| $\overline{\text{xxRAS}}$(1:0) | RAS(1:0) | Chip Select(1:0) | Chip Select(1:0) | Chip Select(1:0) |
| $\overline{\text{xxCAS}}$(1:0) | CAS(1:0) | DQM(1:0) | DQM(1:0) | Chip Select(3:2) |
| $\overline{\text{xxCS}}$(6:4) | N/A | N/A | N/A | Chip Select(6:4) |
| $\overline{\text{xxCS}}$(7) | N/A | CKE | CKE | Chip Select(7) |
| $\overline{\text{xxWE}}$(1:0) | WE(1:0)* | WE(1:0)* | WE(1:0)* | WE(1:0)* |
| $\overline{\text{xxSYNRAS}}$ | N/A | RAS | RAS | N/A |
| $\overline{\text{xxSYNCAS}}$ | N/A | CAS | CAS | Byte Enable(3) |
| xxDATA(31:0) | Data(31:0) | Data(31:0) | Data(31:0) | Data(31:0) |
| xxDATA(35:32) | ECC(3:0) | ECC(3:0) | ECC(3:0) | Parity(3:0) |
| xxDATA2(38:36) | ECC(6:4) | ECC(6:4) | ECC(6:4) | Byte Enable(2:0) |

**Notes:** xx = CM for Control Memory or PM for Packet Memory.

All signal groups marked by an asterisk are active at the same time.

For SDRAMs with shared ECC configurations, the DQM signals are active independently.

For SDRAMs with split ECC, the DQMs are usually active unless doing burst length 2 and the DQM is needed to terminate a burst.

IBM

## Possible Memory Configurations Using DRAM With Shared ECC

| Module Size | 1 Array | | | 2 Arrays | | |
|---|---|---|---|---|---|---|
| | Storage Size | ECC Size | Number of Devices | Storage Size | ECC Size | Number of Devices |
| 256Kx16 | 1MB | 1MB | 3 | 2MB | 512KB | 5 |
| 1Mx16 | 4MB | 4MB | 3 | 8MB | 2MB | 5 |
| 2Mx8 (16MB) | 8MB | 8MB | 5 | 16MB | 4MB | 10 |
| 4Mx4 (16MB) | 16MB | 16MB | 10 | 32MB | 8MB | 20 |
| 4Mx16 (64MB) | 16MB | 16MB | 3 | 32MB | 8MB | 5 |
| 8Mx8 (64MB) | 32MB | 32MB | 5 | 64MB | 16MB | 10 |
| 16Mx4 (64MB) | 64MB | 64MB | 10 | 128MB | 32MB | 20 |

## Possible Memory Configurations Using SRAM

| Module Size | Number of Devices for 1MB | Number of Devices for 2MB | Number of Devices for 4MB | Number of Devices for 8MB | Notes |
|---|---|---|---|---|---|
| 256Kx18 | 2 | 4 | 8 | 16 | |
| 128Kx36 | 2 | 4 | 8 | N/A (See note below) | 2 |

1. For x18 SRAM modules, half the data bus goes to one module, and the other half goes to a second module. The chip select to the two modules is common. Therefore, one chip select is needed per MB of memory.
2. 2. With x36 modules, a chip select is required for each module (0.5MB of memory). With eight chip selects available, the maximum memory allowed is 8MB for x18 modules and 4MB for x36 modules.

## NPBUS

The NPBUS consists of an eight-bit multiplexed addr/data bus plus generic transfer control signals to work with the PHY level hardware microprocessor interface.

### NPBUS Connections

## NPBUS Pin Descriptions (Page 1 of 2)

| Quantity | Pin Name | Input/Output | Pin Function | Pin Description |
|---|---|---|---|---|
| 1 | $\overline{PBPHY1}$ | Output | $\overline{Select\ PHY\ 1}$ | When low, indicates that IBM2520L8767 has selected PHY 1 to write to control registers inside PHY 1 or to read either the control or status registers. |
| 1 | $\overline{PBPHY2}$ | Output | $\overline{Select\ PHY\ 2}$ | When LOW, indicates that IBM2520L8767 has selected PHY2 to write to control registers inside PH 2 or to read either the control or status registers. See NPBUS Control Register for more details. If configured, this pin can also be:<br><br>Odd parity across the eight-bit wide bidirectional data bus. This pin can also be configured as MPMDSEL - this control pin, under register bit control, can drive a logical value out. The intention is to select between the different PMD types on the 155 Mb/s copper card (UTP verses STP). If it is in cascade mode, this bit functions as PIDSELO (+idsel out), which the primary IBM2520L8767 will drive to the secondary IBM2520L8767 when trying to update configuration space via configuration cycles. This multiplexec pin also carries the PBDATAP signal. |
| 32 | ENSTATE (63-32) | Output | | When programmed, drives out the real-time state of entity state machines, counters, etc. for debug purposes. The lower 16 bits of this bus are also PBADDR(15 - 0), which are the address lines for the external parallel EPROM. |
| 1 | $\overline{PBEPRM}$ | Output | $\overline{EPROM\ Select}$ | When LOW, indicates that IBM2520L8767 has selected the external EPROM to read from. After reset, IBM2520L8767 will start accessing the optional on-card ROM/EPROM and do the chip initialization function if it does not find a serial EPROM attached. |
| 1 | PBALE1 | Output | Address Latch Enable 1 | When high, indicates that IBM2520L8767 has generated an address on the PBDATA bus and should be latched by either a PHY that supports this muxing or an external octal latch TTL part. For an external EPROM, it will also latch bits 7-0 of the address for an external EPROM access. |
| 1 | PBALE2 | Output | Address Latch Enable 2 | When high, indicates that IBM2520L8767 has generated an address on the PBDATA bus and should be latched by an external octal latch TTL part that holds bits 15-8 of the address for an external EPROM access. |
| 1 | PBADDR16 | Output | Address Send 16 | Supplies address 16 to an external EPROM. The pin will also function as PBALE3, an address latch enable, that indicates that the IBM2520L8767 has generated an address on the PBDATA bus and should be latched by an external octal latch TTL part that holds bits 23-16 of the address for an external EPROM access. The mechanism used to set this mode is to put a pull-down resistor on this pin. At reset time, it will be detected and set this bit in PBALE3 mode. Otherwise it will be in PBADDR16 mode. |
| 1 | PBADDR17 | Output | Address Send 17 | Supplies address 17 to an external EPROM. If it is in cascade mode, this bit functions as MSGNTGO (secondary grant gate out) that the primary IBM2520L8767 will drive to allow a bus grant to the secondary IBM2520L8767 . See PCINT Cascade Control Register for more details. |

1. S/T/S = a sustained tri-state pin owned and driven by one and only one agent at a time. The agent that drives the s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a s/t/s signal any sooner that one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

## NPBUS Pin Descriptions (Page 2 of 2)

| Quantity | Pin Name | Input/Output | Pin Function | Pin Description |
|---|---|---|---|---|
| 1 | PBSCLK | Output | Clock for the I$^2$C serial EPROM accesses. | |
| 1 | PBSDATA | Output or Data | | This is the data bit that connects to the external serial EPROM to read from or write to. It must have a pullup resistor attached and supports the I$^2$C protocol. The range of suported serial EPROM is from 256 to 2K bytes. After reset, the IBM2520L8767 will start accessing the optional on-card serial EPROM and do the chip initialization function. If this chip is pulled down (or no pullup), the IBM2520L8767 will assume that no serial EPROM is attacted and will go try to fetch from a parallel EPROM. |
| 1 | PBRNWRT | Output | Read or Write | This pin allows the IBM2520L8767 to read from or write to internal registers of the PHY parts. This signal acts as the write strobe when talking to PMC-Sierra chips such as the Suni-Lite . |
| 1 | PBRDRDY | S/T/S[1] | | This pin allows the IBM2520L8767 to read from or write into internal registers of the PHYs by acting as a data acknowledge signal from the memory slaves. |
| 8 | PBDATA(7-0) | Input or Output | | The PB-Bus is an eight-bit wide bidirectional data bus used to interface the PHYs to the IBM2520L8767 When a data transfer is not happening, the lower four bits act as: MLED(3-0) - four control pins that, under register bit control, can drive general status to LED devices. |
| 1 | PBINTRA | Input | | This input from PHY A is an attention line that, when LOW, indicates that one or more unmasked flags are set in the status registers of PHY 1<br><br>If additional PHY parts are added, they should also dot their interrupt line onto this input. |
| 1 | PBPHYRST | Output | Implements the network safety features of the device | This signal is the ORed value of RESET and all of the status bits that cause the IBM2520L8767 to stop transferring data. It is asserted for a pulse, and then removed. This signal is asserted low. |

1. S/T/S = a sustained tri-state pin owned and driven by one and only one agent at a time. The agent that drives the s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a s/t/s signal any sooner that one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

IBM

## ATM PHY Bus Interface

The PHY Bus consists of a transmit data path, receive data paths, and control signals.

## ATM PHY Bus Interface Connections

## PHY Bus Pin Descriptions (Page 1 of 2)

| Quantity | Pin Name | Input/Output | Pin Function | Pin Description |
|---|---|---|---|---|
| 16 | FYTDAT (15 - 0) | Output | PHY Transmit Data. | When using an external PHY, this 16 pin bus carries the ATM CELL octets that are loaded in the PHY Transmit FIFO. When using the internal framer, the lower eight bits carry the SONET/SDH octets bound for the network, while bits 15, 14, and 13 are used for the RX HDLC interface signals OFPrxR1Data, OFPrxR1DS and OFPrxRclk, respectively. |
| 2 | FYTPAR (1 - 0) | Output | Transmit Data Parity. | When using an external PHY, these are byte parity signals for FYTDAT. When using the internal framer, bit one provides the RX Out-Of-Frame indication, OOF, and bit 0 provides the optical/electrical module transmit shutdown control signal, OFPtxSDown. |
| 1 | FYTSOC | Output | Transmit start of Cell. | When using an external PHY, this indicates the start of cell on FYTDAT. When using the internal framer, this provides the TX HDLC interface signal, OFPtxT1Dclk. |
| 1 | FYTWRB | Output | Transmit write strobe. | When using an external PHY, this signal is used to write ATM cells to the transmit FIFO. When using the internal framer, this signal provides the 19.44 MHz TX clock, RefClkT. |
| 1 | $\overline{\text{FYTENB}}$ | Output | Transmit write enable. | When using an external PHY, this indicates that transmit data to the PHY is valid. When using the internal framer, this provides the TX HDLC interface signal, OFPtxT1DS. When interfacing to the IBM2520L8767, it should be connected to -TDVal. |
| 1 | $\overline{\text{FYRENB}}$ | Output | Receive write enable. | When using an external PHY, this indicates to the PHY that the IBM2520L8767 is ready to accept data. When using the internal framer, this provides the clock recovery reset signal, RSTCRec1. When using the IBM ATM-TC PHY, this should be connected to +RLoad. |
| 1 | FYRRDB | Output | Receive ready strobe. | When using an external PHY, this is used to read ATM cells from the PHY receive FIFO. When using the internal framer, this signal provides the 19.44 MHz RX clock, RxByClk. When using the IBM ATM-TC (25 Mb/s), this should be connected to RBCLK. |
| 16 | FYRDAT(15 - 0) | Input | PHY Receive Data . | When using an external PHY, this 16 pin bus carries the ATM CELL octets that are read from the PHY Receive FIFO. When using the internal framer, the lower eight bits carry the SONET/SDH octets received from the network. |
| 2 | FYRPAR(1 - 0) | Input | PHY Receive Data Parity. | When using an external PHY, these are byte parity signals for FYRDAT. When using the internal framer, bit 1 provides the optical/electrical module low power indication signal, OFPtxLPow, and bit 0 is not used. |
| 1 | FYRSOC | Input | Receive start of Cell. | When using an external PHY, This signal indicates the start of cell on the FYRDAT bus. When using the IBM ATM-TC phy, this input should be pulled down to the inactive state. When using the internal framer, this is the receive frame pulse input signal, FPulse. indicates when a cell is available in the receive FIFO. When using the internal framer, this signal is not used. When using IBM ATM-TC, it should be connected to +RDVal. |

## PHY Bus Pin Descriptions (Page 2 of 2)

| Quantity | Pin Name | Input/Output | Pin Function | Pin Description |
|----------|----------|--------------|--------------|-----------------|
| 1 | FYTCA | Input | Transmit Cell Available. | When using an external PHY, this indicates that a cell is available in the PHY transmit FIFO. When using the internal framer, this provides the TX HDLC interface signal, OFPtxT1Data. When interfacing to IBM ATM-TC, it should be connected to +TLoad. |
| 1 | $\overline{\text{FYFUL}}$ | Input | PHY Transmit Full. | When using an external PHY, this is asserted LOW by the PHY when it can accept no more than four more data transfers before it is full. This pin should be pulled up to the inactive state when using a PHY that does not drive it. When using the internal framer, this provides the TX HDLC interface signal, OFPtxT1DFrm. |
| 1 | $\overline{\text{FYEMP}}$ | Input | PHY Receive Empty. | When using an external PHY, this is asserted LOW by the PHY to indicate that in the current cycle there is no valid data for delivery to the IBM2520L8767. When the PHY does not drive FY0EMP, this input should be tied to the inactive state. When using the internal framer, this signal is not used. |
| 1 | FYTSDATP/N | Input | SERDES Transmit Data (Differential). | When using the internal framr and the internal serdes, these signals provide the serial transmit data stream. |
| 1 | FYTSCLKP/N | Input | SERDES Transmit Clock (Differential). | When using the internal framr and the internal serdes, the reference 155.52MHz clock is supplied on these signals. When not in use, these should be tied to (TBD). |
| 1 | FYRSDATP/N | Input | SERDES Receive Data (Differential). | When using the internal framr and the internal serdes, the recovered receive data is supplied on these signals. When not in use, these should be tied to (TBD). |
| 1 | FYRSCLKP/N | Input | SERDES Receive Clock (Differential). | When using the internal framr and the internal serdes, the recovered 155.52MHz clock is supplied on these signals. When not in use, these should be tied to (TBD). |
| 1 | FYDTCT | Input | PHY Carrier Detect. | When using an external PHY, the PHY uses this signal to indicate carrier detect. When using the internal framer, this signal provides the deserializer lock detect signal, ELockDet, from the deserializer. |
| 1 | FYDISCRD | Input | PHY Cell Discard. | When using an external PHY, this signal causes the current cell being received to be discarded. In this case it should only be asserted for the duration of one of the 53 bytes of the ATM cell. When using the internal framer, this signal provides the optical/electrical module Loss-Of-Signal indication, LossSig. |
| 1 | FYSETCLP | Input | PHY CLP Bit Set | When HIGH, causes the current cell being received to have its CLP bit set to 1. This signal should only be asserted for the duration of one of the 48 data bytes of the ATM cell. |

**Note:** Because some of the PHY transmit I/Os are used for receive framer functions and vice versa, there are some restrictions on how the interfaces can be used.

1. If the transmit path is using an external PHY and the receive path is using the internal framer, FYTPAR(1) will assume the OOF function and not be available as a parity output. This is only a concern if the PHY uses a 16-bit data interface and parity is being used.

2. If the receive path is using an external PHY and the transmit path is using the internal framer, FYRPAR(1) will assume the OFPtxLPow function and not be available as a parity input. This is only a concern if the PHY uses a 16-bit data interface.

3. If the transmit path is using an external PHY and the receive path is using the internal framer and the external PHY has a 16-bit data interface, then the receive HDLC interface cannot be used. The three I/O for the RX HDLC interface will instead take on the function of FYTDAT(15-13).

## Transmit PHY I/O Cross Reference

| IBM2520L8767 I/O | Suni-Lite | Utopia | ATM-TC | Internal Framer |
|---|---|---|---|---|
| FYTWRB | TFCLK | TxCLK | N/A | RefClkT |
| FYTCA | TCA | TxClav | TLoad | OFPtxT1Data |
| $\overline{\text{FYFUL}}$ | N/A | $\overline{\text{TxFull}}$ | N/A | OFPtxT1DFrm |
| $\overline{\text{FYTENB}}$ | $\overline{\text{TWRENB}}$ | $\overline{\text{TxEnb}}$ | $\overline{\text{TDVal}}$ | OFPtxT1DS |
| FYTSOC | TSOC | TxSOC | N/A | OFPtxT1Dclk |
| FYTDAT(15) | N/A | TxData(15) | N/A | OFPrxR1Data |
| FYTDAT(14) | N/A | TxData(14) | N/A | OFPrxR1DS |
| FYTDAT(13) | N/A | TxData(13) | N/A | OFPrxR1Dclk |
| FYTDAT(12-8) | N/A | TxData(12-8) | N/A | N/A |
| FYTDAT(7-0) | TDAT(7-0) | TxData(7-0) | TData(7-0) | TxExtDat(7-0) |
| FYTPAR(1) | N/A | TxPrty(1) | N/A | OOF |
| FYTPAR(0) | N/A | TxPrty(0) | TDPrty | OFPtxSDown |

Signals marked with an overbar are active low.

Inputs listed as N/A should be tied to their inactive Utopia state.

## Receive PHY I/O Cross Reference

| IBM2520L8767 I/O | Suni-Lite | Utopia | ATM-TC | Internal Framer |
|---|---|---|---|---|
| FYRRDB | RFCLK | RxCLK | RBClk | RxByClk |
| FYRCA | RCA | RxClav | RDVal | N/A |
| $\overline{\text{FYEMP}}$ | N/A | $\overline{\text{RxEmpty}}$ | N/A | N/A |
| $\overline{\text{FYRENB}}$ | $\overline{\text{RRDENB}}$ | $\overline{\text{RxEnb}}$ | RLoad | RSTCRec1 |
| FYRSOC | RSOC | RxSOC | N/A | FPulse |
| FYRDAT(15-8) | N/A | RxData(15-8) | N/A | N/A |
| FYRDAT(7-0) | RDAT(7-0) | RxData(7-0) | RData(7-0) | RxExtDat(7-0) |
| FYRPAR(1) | N/A | RxPrty(1) | N/A | OFPtxLPow |
| FYRPAR(0) | N/A | RxPrty(0) | RDPrty | N/A |

Signals marked with an overbar are active low.

Inputs listed as N/A should be tied to their inactive Utopia state.

# Clock, Configuration, and LSSD Connections

Normal Value

IBM2520L8767

NPBUS Signals

| Signal | Normal Value |
|---|---|
| MPCIRST | 1 |
| PCICLK | |
| TXCLK | |
| RXCLK | |
| MPEGCLK | |
| DTR | |
| CTS | |
| TXD | |
| RXD | |
| RTS | |
| DSR | |
| JTAGRSTC | |
| JTAGTCKC | |
| TESTM | 1 |
| TESTCT | 1 |
| TMMCORE | 1 |
| JTAGTDIC | 1 |
| MHALT401 | 1 |
| MBYPASS | 1 |
| PFFCFG(2:0) | P |
| PFFOSC | |
| PLLTI | 0 |
| PVDDA | 1 |
| NSELT | 1 |
| JTAGRST | 1 |
| JTAGTCK | 0 |
| JTAGTMS | 0 |
| JTAGTMSC | 0 |
| JTAGDI | 0 |
| JTAGDO | |
| JTAGDOC | |
| PINTCLK | |
| PABISTO | |
| PPLLOUT | |
| BISTODI | |
| MFORCEBP | |
| ENSTATE(63-32)/PBADDR(15-0) | |
| IBDINH1 | 1 |
| IBDINH2 | 1 |
| IBDRINH | 1 |
| LEAKTST | 1 |

## Clock, Configuration, and LSSD Pin Descriptions (Page 1 of 2)

| Quantity | Pin Name | Input/Output | Pin Description |
|---|---|---|---|
| 1 | $\overline{\text{MPCIRST}}$ | Input | This signal will cause a hardware reset when asserted low. See Entity 20: *Reset and Power-on Logic (CRSET)* on page 383 for more details on resets. |
| 1 | PCICLK | Input | The PCICLK is a 40-50% duty cycle 30-ns clock. |
| 1 | TXCLK | Input | This is the LinkC asynchronous transmit clock. |
| 1 | RXCLK | Input | This is the LinkC asynchronous receive clock. |
| 1 | MPEGCLK | Input | This is the MPEG asynchronous clock. |
| 1 | $\overline{\text{TESTM}}$ | Input | When the test mode pin is not asserted, this chip will run as specified. When the test mode pin is asserted, the chip is in LSSD test mode. Transparent latches become clocked latches and I/Os change to primary test inputs and test outputs. This signal is asserted low and should be tied to a '1'b for normal operation. |
| 1 | $\overline{\text{TESTCT}}$ | Input | When the Test Clock Tree pin is not asserted, this chip will run as specified. When the TestClock Tree pin is asserted, the clock tree will use this input to control the clock tree outputs. This signal is asserted low and should be tied to a '1'b for normal operation. |
| 1 | $\overline{\text{TMMCORE}}$ | Input | Test Mode Matrix for the 401 Core. This signal is asserted low and should be tied to a '1'b for normal operation. |
| 1 | JTAGTDIC | Input | This is the TDI to the 401 JTAG TAP controller. |
| 1 | MHALT401 | Input | Used by RISCWatch to halt the 401 core for debug purposes. |
| 1 | $\overline{\text{MBYPASS}}$ | Input | When tied to '0'b on the card, the PLL function will not multiply the chip input (PCI clock). Instead, it will just pass the clock input frequency to the internal clock tree. Normal mode for this pin is '1'b. A 1-K pullup must be used. |
| 3 | PFFCFG (2 - 0) | Input | These bits control the "find frequency" function which sets the range bits of the PLL. Below is the encoded meaning of these bits. table, but some examples are provided here.<br>- 000 = Force to 66MHz operation: set range to 11 and adjust ROM fetch speed<br>- 001 = Disable auto range function: set range to 01 (<16MHz bypass mode)<br>- 010 = Disable auto range function: set range to 10 (16-31.5MHz)<br>- 011 = Disable auto range function: set range to 11 (31.5-66MHz)<br>- 100 = Enable auto range function for 19.44MHz<br>- 101 = Reserved<br>- 110 = Enable auto range function for 25.00MHz<br>- 111 = Enable auto range function for 32.00MHz |
| 1 | PFFOSC | Input | This input is the auto range known frequency input that is used to time the PCI clock input. This should be connected to some oscillator on the the card. A typical example would be the PHY oscillator. |
| 1 | PLLTI | Input | When tied to '1'b, this input will cause the PLL to do a parametric testing at the wafer and module level. Normal mode for this pin is a '0'b. |
| 1 | PVDDA | Input | Filtered Vdd source to the PLL logic. See technology application notes for filter circuit. |
| 1 | $\overline{\text{NSELFT}}$ | Input | Minus active SELFTEST input. Normal mode is a '1'b. |
| 1 | $\overline{\text{JTAGRST}}$ | Input | JTAG Test Reset provides an asynchronous initialization of the TAP controller. |
| 1 | JTAGTCK | Input | JTAG Test Clock is used to clock state information and test data into and out of the device during operation of the TAP. |

IBM

## Clock, Configuration, and LSSD Pin Descriptions (Page 2 of 2)

| Quantity | Pin Name | Input/Output | Pin Description |
|---|---|---|---|
| 1 | JTAGTMS | Input | JTAG Test Mode Select is used to control the state of the TAP controller in the device. |
| 1 | JTAGTMSC | Input | JTAG Test Mode Select is used to control the state of the TAP controller in the 401 core. |
| 1 | JTAGTDI | Input | JTAG Test Data Input is used to serially shift test data and test instructions into the device during TAP operation. |
| 1 | JTAGTDO | Output | Test Data Output is used to serially shift test data and test instructions out of the device during TAP operation. |
| 1 | JTAGTDOC | Output | Test Data Output - Core is used to serially shift test data and test instructions during Processor core TAP operations. |
| 1 | PINTCLK | Output | This is the external test point to measure the jitter effects of the phase-lock loop circuit. |
| 1 | PDBLCLK | Output | This is the external test point that is double the frequency of the PINTCLK. It is used to clock enstate state signals at this frequency. |
| 1 | PPLLOUT | Output | This is an observation output only. This will make the output of the PLL observable. This is also the DTR signal when the SELRS232 is active. |
| 1 | $\overline{\text{BISTDI1}}$ | Output | Drives the DI input during BIST |
| 1 | MFORCEBP | Output | Allows IBM2520L8767 to bypass the internal PLL. See *Printed Circuit Board Considerations* on page 515. |
| 1 | DTR | Input or Output | RS232 DTR for the core debugger. |
| 1 | CTS | Input or Output | RS232 CTS for the core debugger. |
| 1 | TXD | Input or Output | RS232 TXD for the core debugger. |
| 1 | RXD | Input or Output | RS232 RXD for the core debugger. |
| 1 | RTS | Input or Output | RS232 RTS for the core debugger. |
| 1 | DSR | Input or Output | RS232 DSR for the core debugger. |
| 1 | $\overline{\text{JTAGRSTC}}$ | Input or Output | JTAG Test Reset provides an asynchronous initialization of the Processor core TAP controller. |
| 1 | JTAGTCKC | Input or Output | JTAG Test Clock is used to clock state information and test data into and out of the device during operation of the Processor core TAP controller. LatchTclkc in test mode. |
| 1 | IBDINH1 | Input | This is the Boundary Scan input for BSINH1. |
| 1 | IBDINH2 | Input | This is the Boundary Scan input for BSINH2(*). |
| 1 | IBDRINH | Input | This is the Boundary Scan input for rinh. |
| 1 | LEAKTST | Input | This is the STI driver/receiver leak test input. |

# Data Structures

These structures reside in control memory for each of the logical channels that are set up for transmission or reception.

## Packet Header

Each packet buffer consists of two parts. The first part is the control information used by the IBM2520L8767. The second portion of the packet buffer is used to hold the actual packet data. The following figures show the structure of the transmit and receive packet headers:

## Transmit Packet Header Structure

```
struct tx_min_packhead {     struct tx_packhead {
    bit32 next_buffer;           bit32 next_buffer;
    bit8  AAL5_user_byte1;   bit8  AAL5_user_byte;
    bit8  buffer_offset;         bit8  buffer_offset;
    bit16 buffer_length;         bit16 buffer_length;
    bit25 lc_address;            bit25 lc_address;
    bit1  reserved;              bit1  reserved;
    bit1  reserved;              bit1  reserved;
    bit1  dma_on_xmit;           bit1  dma_on_xmit;
    bit1  generate_CRC10;        bit1  generate_CRC10;
    bit1  free_on_xmit;          bit1  free_on_xmit;
    bit1  queue_on_xmit;         bit1  queue_on_xmit;
    bit1  cell_loss_priority; bit1  cell_loss_priority;
    };                           bit32 dma_desc_addr;
                                 bit24 reserved;
                                 bit8  AAL5_user_byte2;
                                 };
```

The minimum transmit packet header size (and transmit offset) is 0xC bytes.

## Receive Packet Header Structure

```
struct rx_packhead {              struct rx_packhead {
    bit16 rx_label;                   bit16 rx_label;
    bit4 reserved;                    bit4 reserved;
    bit1  toobig_status;              bit1  toobig_status;
    bit1  memchk_status;              bit1  memchk_status;
    bit1  fabort_status;              bit1  fabort_status;
    bit1  badlen_status;              bit1  badlen_status;
    bit1  badcpi_status;              bit1  badcpi_status;
    bit1  badcrc_status;              bit1  badcrc_status;
    bit1  timout_status;              bit1  timout_status;
    bit1  fifopk_status;              bit1  fifopk_status;
    bit1  congestion_status;          bit1  congestion_status;
    bit1  route_status;               bit1  route_status;
    bit1  error_status;               bit1  error_status;
    bit1  done_status;                bit1  done_status;
    bit8  AAL5_user_byte;             bit8  AAL5_user_byte;
    bit8  buffer_offset;              bit8  buffer_offset;
    bit16 buffer_length;              bit16 buffer_length;
    bit25 lc_address;                 bit25 lc_address;
    bit6  reserved;                   bit6  reserved;
    bit1  cell_loss_priority;         bit1  cell_loss_priority;
    bit32 rx_atm_header;              bit32 rx_atm_header;
    bit32 host_data;                  bit24 host_data;
                                      bit8  AAL5_user_byte2;
    bit32 cut_thru_addr;                   bit32 cut_thru_addr;
   };                                };
```

## Transmit and Receive Packet Header Field Descriptions (Page 1 of 2)

| Field Name | Field Description |
|---|---|
| next_buffer | This field is used by the hardware to chain buffers together on queues. It contains the address of the next buffer if one exists. For transmit buffers allocated in virtual memory, this field will be written by the hardware with a distinctive pattern ('zzzzzBAD'x) where zzzzz is the offset of the failure when a write operation was not able to complete due to a shortage of the real buffers needed to map into the virtual address space. This field can be checked after all buffer write operations and the appropriate recovery actions taken immediatly, or when a buffer that has had a write failure is enqueued to CSKED, an event will be generated and the buffer will not be processed by CSKED. A status bit also exists in the BCACH status register indicating that a write to virtual memory has failed. With cache performance in mind, this status bit could be checked first, and if it is not set, there is no need to access the header of the packet. <br><br> **Note:** This automatic error recovery mechanism results in the restriction that this first four bytes of a transmit packet must never be written via programmed IO or DMA during preparation for transmission. If this field is written by a software or DMA operation, the automatic error detection will not work properly and undesirable results are likely. |
| AAL5_user_byte1 | This field contains the value to be sent in the user byte in the last cell of an AAL5 packet if INTST is configured for one user byte. |
| dma_on_xmit | If this bit is set, a DMA descriptor address placed in the packet header (offset 'C'x) will be queued for execution. |
| generate_CRC10 | If this bit is set, CRC10 will be generated over the cell(s) in this packet. |
| free_on_xmit | If this bit is set, the buffer will be freed after the transmission completes. |
| queue_on_xmit | If this bit is set, the buffer will be queued on the transmit complete queue after the transmission completes. |

## Transmit and Receive Packet Header Field Descriptions (Page 2 of 2)

| Field Name | Field Description |
|---|---|
| cell_loss_priority | This bit is used on both transmit and receive:<br>txIIf this bit is set, the cell loss priority bit in the ATM cell header will be set for each cell in this packet<br>rxThis bit contains theOR'd cell loss priority bits across all the cells that comprised this packet if this LCD is using AAL5. |
| buffer_offset | This field contains the offset into the buffer where the data starts. |
| buffer_length | This field contains the length of the packet. |
| lc_address | This is the address of the logical connection descriptor that this packet was received on. |
| rx_atm_header | On reception, the four-byte ATM header (no HEC) is copied from the first and last cell into this area. |
| AAL5_user_byte2 (tx) | This field contains the value to be sent in the user bytes in the last cell of an AAL5 packet if INTST is configured for two-user byte. |
| AAL5_user_byte2 (rx) | This field contains the second AAL5 user byte in the last cell of an AAL5 packet if INTST is configured for two-user byte. |
| bit16 rx_label | This field is written with "RA" in ASCII (0x5241) to signal that this buffer was used by RAALL. |
| bit4 reserved | This field is always zero |
| bit1 routed_status | This bit is set if this packet or cell was internally routed. |
| bit1 toobig_status | Indicates the current packet exceeded the maximum packet size. |
| bit1 memchk_status | Indicates the current packet had a memory check (real size exceeded or virtual error). |
| bit1 fabort_status | Indicates the current packet was aborted (AAL5 forward abort). |
| bit1 badlen_status | Indicates the current packet had a bad AAL5 length in the trailer. |
| bit1 badcpi_status | Indicates the current packet had a bad AAL5 CPI field (not zero). |
| bit1 badcrc_status | Indicates the current packet had a bad AAL5 CRC. |
| bit1 timout_status | Indicates the current packet had a reassembly timeout error. |
| bit1 fifopk_status | Indicates the current packet is a FIFO packet (see MPEG FIFO mode). |
| bit1 congestion_status | This bit is written when the packet is completed. It contains the OR'd congestion bit across an AAL5 packet. |
| bit1 route_status | This bit is written when the packet is completed if it is internally routed. |
| bit1 error_status | This bit is written when the packet is completed if an error condition occurs. |
| bit1 done_status | This bit is written when the packet is completed. It can be used when thresholding. |
| host_data | If host data is enabled in RAALL, then the 32/24 bits of host data is read from the LCD and written to this area for each packet. The size is based on how many user bytes are used. |
| cut_thru_addr | This field is only used in one of the cut-through modes and has two purposes. When the packet is first received, the packet address is written to this field. This information can then be used by software to do a further cut-through operation or free the packet. When a cut-through operation is performed, and the packet is not complete yet, the descriptor address is placed in this field .<br>When in scatter mode, the low-order bits specify how many pages are in the DMA list that follows the packet header. |
| dma_desc_addr | If the dma_on_xmit bit is set in the packet header, this field contains the address of the DMA descriptor that will be queued when transmission is complete. |

## Logical Channel Data Structure

If LCD-based memory management is used, a 64-bit section with the following layout is inserted at byte offset 18x:

| Threshold 1 | Threshold 2 | Pool ID1 | Pool ID1 | Bytes waiting to be transmitted |
|---|---|---|---|---|
| 16 | 16 | 4 | 4 | 24 |

Four different formats, determined by whether scheduling is based on virtual path parameters or is configured for ABR. See Scheduling Portion of a Transmit Descriptor on page 68.

**CSKED Scheduling Information (3) 64-bit words**

| Tx Length | Misc. Flags | ATM Header |
|---|---|---|
| 16 | 16 | 32 |

| Segmentation Pointer | Partial CRC |
|---|---|
| 16 | 32 |

**CSKED/SEGBF Shared Information (2) 64-bit words**

**SEGBF Segmentation Information (0-3) 64-bit words**

for AAL type 0x7

| MPEG-2 or ABR Data | |
|---|---|
| 32 | 32 |

| OAM Data | Fixed Blocking Data |
|---|---|
| 32 | 32 |

| total user cells       LCD Cell Status | total user cells with CLP=0 |
|---|---|
| 32 | 32 |

for AAL types 0x4 and 0x6

| OAM Data | Fixed Blocking Data |
|---|---|
| 32 | 32 |

| total user cells       LCD Cell Status | total user cells with CLP=0 |
|---|---|
| 32 | 32 |

for all other AAL types

| total user cells       LCD Cell Status | total user cells with CLP=0 |
|---|---|
| 32 | 32 |

| OAM Data | Fixed Blocking Data |
|---|---|
| 32 | 32 |

ABR code portion

| Unused | ABR Variables |
|---|---|
| 56 | 8 |

| ABR Code Variables |
|---|
| 64 |

On ABR Connections, the ABR code running in PCORE uses 10 bytes of the LCD starting at offset 0x46 for managing the connection.

Offset

| | |
|---|---|
| 0x00 | Transmit Portion |
| 0x50 | Receive Portion |
| 0x78 | User Data |

128-byte LCD

**Transmit Descriptor Data Structures**

*Logical Channel Data Structure* on page 38 and *Scheduling Portion of a Transmit Descriptor* on page 40 show the layout of the transmit portion of a Logical Channel Descriptor. When initializing an LCD, any locations that are not written to a specific value should be initialized to zeroes. Fields that typically need to be initialized to a non-zero value are flagged with a # in the structure below.

**Note:** This is only one possible layout of the transmit portion of the LCD. Some field locations vary and are further defined later in this section.

Care must be taken when updating fields in the LCD and then immediatly causing the updated fields to be accessed by other IBM2520L8767 entities. For example, it is possible, although not likely, under the right conditions, for a normal LCD update followed by a SEGBF cell enqueue operation to actually execute in reverse order. This is due to IBM2520L8767 internal priority levels and could result in SEGBF fetching the LCD data before it has been updated to the new value. For this reason, it is highly advisable to use the LCD update mechanism in RAALL (*RAALL LCD Update Data Registers* on page 295, *RAALL LCD Update Mask Registers* on page 295, and *RAALL LCD Update Op Register* on page 296) to guarantee that any LCD update operation completes before any subsequent ops can execute.

The transmit portion of the LCD can be subdivided into three distinct parts based on which chip functions or entities access that particular part of the LCD. The first three 64-bit words are scheduling related and are accessed only by CSKED. The next two 64-bit words are related to both scheduling and segmentation and are accessed and shared by both CSKED and SEGBF. The words following these shared locations are related only to segmentation and are accessed only by SEGBF. The number of 64-bit words in this portion of the LCD can vary from zero to three. The actual number being used in an LCD is determined by the AAL type bits.

The three 64-bit words containing CSKED scheduling information can have four different formats depending upon whether scheduling is based on virtual path parameters or is configured for ABR. These four formats are shown in *Scheduling Portion of a Transmit Descriptor* on page 40:

## Scheduling Portion of a Transmit Descriptor

| Next descriptor in chain | Peak Interval | Average Interval |
|---|---|---|
| 32 | 16 | 16 |

| Time Stamp | Timing Data |
|---|---|
| 32 | 32 |

| Transmit Packet Queue Head Pointer | Transmit Packet Queue Tail Pointer |
|---|---|
| 32 | 32 |

for a standard virtual circuit

| Next Descriptor in Chain | Peak Interval | ABR Parameters |
|---|---|---|
| 32 | 16 | 16 |

| ABR Values and Timing Data |
|---|
| 64 |

| Transmit Packet Queue Head Pointer | Transmit Packet Queue Tail Pointer |
|---|---|
| 32 | 32 |

using ABR

| Next Descriptor in Chain | Peak Interval | Average Interval |
|---|---|---|
| 32 | 16 | 16 |

| Virtual Path Descriptor Address | Reserved |
|---|---|
| 32 | 32 |

| Transmit Packet Queue Head Pointer | Transmit Packet Queue Tail Pointer |
|---|---|
| 32 | 32 |

for a Virtual Circuit on a Virtual Path

| Next Descriptor in Chain | Peak Interval | Average Interval |
|---|---|---|
| 32 | 16 | 16 |

| Time Stamp | Timing Data |
|---|---|
| 32 | 32 |

| Transmit LCD Queue Head Pointer | Transmit LCD Queue Tail Pointer |
|---|---|
| 32 | 32 |

for a Virtual Path

CSKED Scheduling Information (3) 64-bit words

## Transmit Logical Channel Descriptor Structure

```
    .
 typedef struct {
     bit32 next_lcd;
     bit16 #peak_interval;
     bit16 #average_interval;

     bit32 #timestamp;
     bit11 Reserved;
     bit1  remove_lcd;
     bit1  lc_on_timewheel;
     bit3  #alter_sched;
     bit2  #transmit_priority;
     bit1  #max_resolution;
     bit3  #max_burst_mult;
     bit10 #max_burst_value;

     bit26 head_packet_pointer;
     bit1  free_on_xmit;
     bit1  queue_on_xmit;
     bit1  dma_on_xmit;
     bit3  reserved;
     bit26 tail_packet_pointer;
     bit6  reserved;

     bit16 transmit_length;
     bit8  buffer_offset;
     bit1  #enable_blocking;
     bit1  #enable_statistics;
     bit1  flush_LCD;
     bit1  #generate_CRC10;
     bit1  #OAM_CLP_source
     bit3  #AAL_type;
     bit32 #ATM_header;

     bit32 segmentation_pointer;
     bit32 current_CRC;

     bit32 total_user_cells;
     bit32 total_user_cells_CLP0;

     bit16 BIP-16;
     bit8  Monitor_sequence_number;
     bit1  OAM_cell_transmitted
     bit2  previous_PCR_bits17_16
     bit1  #OAM_PTI_bit0;
     bit1  previous_packet_contained_PCR
     bit1  #OAM_CLP_value;
     bit2  #OAM_block_size;
     bit32 reserved;

 } tx_lcd_struct, *tx_lcd_struct_ptr
```

## Redefinition of Shared and Segmentation Portion of Transmit LCD for ABR

```
typedef struct {
    bit16 transmit_length;
    bit8  buffer_offset;
    bit1  #enable_blocking;
    bit1  #enable_statistics;
    bit1  flush_LCD;
    bit1  #generate_CRC10;
    bit1  #OAM CLP source
    bit3  #AAL_type;

    bit32 #ATM_header;
    bit32 segmentation_pointer;

    bit32 current_CRC;
    bit16 explicit_rate;
    bit16 current_rate;
    bit16 minimum_rate;
    bit16 reserved;

    bit16 BIP-16;
    bit8  Monitor_sequence_number;
    bit1  OAM_cell_transmitted
    bit2  reserved
    bit1  #OAM_PTI_bit0;
    bit1  reserved
    bit1  #OAM_CLP_value;
    bit2  #OAM_block_size;

    bit32 backward_ptr;
    bit32 total_user_cells;
    bit32 total_user_cells_CLP0;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

Owing to the use of certain LCD fields, a connection running ABR can not be set up for segmentation AAL types 0x6 (fixed-sized blocking) or 0x7 (MPEG-2 assist)

## Redefinition of Segmentation Portion of Transmit LCD for fixed size AAL5 blocking (segmentation type 0x6)

```
typedef struct {

    bit16 transmit_length;
    bit8  buffer_offset;
    bit1  #enable_blocking;
    bit1  #enable_statistics;
    bit1  flush_LCD;
    bit1  #generate_CRC10;
    bit1  #OAM CLP source
    bit3  #AAL_type;
    bit32 #ATM_header;

    bit32 segmentation_pointer;
    bit32 current_CRC;

    bit16 BIP-16;
    bit8  Monitor_sequence_number;
    bit1  OAM_cell_transmitted
    bit2  previous_PCR_bits17_16
    bit1  #OAM_PTI_bit0;
    bit1  previous_packet_contained_PCR
    bit1  #OAM_CLP_value;
    bit2  #OAM_block_size;
    bit8  Current_Blocking_Count (4 bytes)
    bit8  #Blocking_size (4 bytes x'2F' for MPEG-2)
    bit1  PID_field_valid
    bit2  PID_bits_12_11
    bit5  Current_transport_stream_packet
    bit1  Check_PCR
    bit1  PCR_present
    bit1  PID_matches
    bit5  #Packets_per_AAL5_frame

    bit32 total_user_cells;
    bit32 total_user_cells_CLP0;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

## Redefinition of Segmentation Portion of Transmit LCD for MPEG2 (segmentation type 0x7)

```
typedef struct {

    bit16 transmit_length;
    bit8  buffer_offset;
    bit1  #enable_blocking;
    bit1  #enable_statistics;
    bit1  flush_LCD;
    bit1  #generate_CRC10;
    bit1  #OAM CLP source
    bit3  #AAL_type;
    bit32 #ATM_header;

    bit32 segmentation_pointer;
    bit32 current_CRC;

    bit16 previous_PCR_bits15_0
    bit11 PID_bits_10_0
    bit1  PCR_delta_valid
    bit36 PCR_delta

    bit16 BIP-16;
    bit8  Monitor_sequence_number;
    bit1  OAM_cell_transmitted
    bit2  previous_PCR_bits17_16
    bit1  #OAM_PTI_bit0;
    bit1  previous_packet_contained_PCR
    bit1  #OAM_CLP_value;
    bit2  #OAM_block_size;
    bit8  Current_Blocking_Count (4 bytes)
    bit8  #Blocking_size (4 bytes x'2F' for MPEG-2)
    bit1  PID_field_valid
    bit2  PID_bits_12_11
    bit5  Current_transport_stream_packet
    bit1  Check_PCR
    bit1  PCR_present
    bit1  PID_matches
    bit5  #Packets_per_AAL5_frame

    bit32 total_user_cells;
    bit32 total_user_cells_CLP0;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

## Redefinition of Scheduling Portion of Transmit LCD for ABR

```
typedef struct {
    bit32 next_lcd;
    bit16 #peak_interval;
    bit3  #Nrm;
    bit3  #Trm;
    bit10 #Tadtf;

    bit8  #Nc;
    bit8  #Ncrm;
    bit16 Reserved;
    bit11 Tlrm1;
    bit1  remove_lcd;
    bit1  lc_on_timewheel;
    bit3  #alter_sched;
    bit2  #transmit_priority;
    bit1  #max_resolution;
    bit13 Tlrm2;

    bit26 head_packet_pointer;
    bit1  free_on_xmit;
    bit1  queue_on_xmit;
    bit4  reserved;
    bit26 tail_packet_pointer;
    bit6  reserved;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

## Redefinition of Scheduling Portion of Transmit LCD for timers

```
typedef struct {
    bit32 next_lcd;
    bit32 #timer_period;

    bit32 #timestamp;
    bit12 reserved;
    bit1  lc_on_timewheel;
    bit1  reserved;
    bit2  #timer_type;
    bit2  #transmit_priority;
    bit1  #max_resolution;
    bit13 reserved;

    bit32 #dma_desc_addr;
    bit32 reserved;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

## Definition of LCD-Based Memory Management of Transmit LCD

```
typedef struct {
    bit16 #threshold_1;
    bit16 #threshold_2;
    bit4  #pool_id1;
    bit4  #pool_id2;
    bit24 #bytes_queued;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

# Definition of ABR code variables

```
typedef struct {
    bit8  #CRM;
    bit8  #iCDF;
    bit16 #iMCR;
    bit16 #PCR;
    bit8  #iRDF;
    bit8  #iRIF;
    bit16 #ICR;
} tx_lcd_struct, *tx_lcd_struct_ptr;
```

**Field Definitions**

The following is a detailed description of the fields listed above. This data structure should be initialized at connection setup but not modified while transmission is occurring on the connection. Only those fields marked with a # typically need to be initialized to something other than zero.

## ABR Code Variables Definitions (Page 1 of 7)

| Field Name | Field Description |
|---|---|
| next_lcd | This field is used by the hardware to chain LCDs together on queues. It contains the address of the next LCD if one exists. |
| peak_interval # | This field contains the minimum spacing allowed between consecutive cells on this connection. This spacing is expressed in cell times. A connection that can transmit every cell time would have a value of 1 for this field. |
| average_interval # | This field contains the minimum average spacing allowed between cells transmitted on this connection. It is the inverse of the Sustainable Cell Rate. The value for this field is expressed in cell times. |
| Nrm # | This field specifies the maximum number of cells a source may send for each forward RM-cell. Number of cells = (2**Nrm)+1. |
| Trm # | This field provides an upper bound on the time between forward RM-cells for an active source. Time = 100•(2**-Trm) msec. |
| Tadtf # | The ACR Decrease Time Factor is the time permitted between sending RM-cells before the rate is decreased to ICR. Time = Tadtf• 0.01 sec. |
| Nc # | This field is used as a counter to determine when Nrm cells have been sent. It should be initialized at connection setup time to 0. |
| Ncrm # | This field is used as a counter to determine when CRM RM-cells have been sent. It should be initialized at connection setup time to CRM. |
| timestamp # | This field contains a timestamp used by the hardware to determine if transmit opportunity credits exist and if the Burst Tolerance has been exceeded. It should be initialized at connection setup time to the value in the current timeslot counter. |
| Tlrm1 & 2 | These fields are used by the hardware to determine when the last RM cell was sent. They should be initialized to 0. |
| lc_on_timewheel # | This field indicates if the LCD is currently queued to the timewheel. It should be initialized ot 0. |
| remove_lcd # | If this bit is set the LCD will be removed from the time wheel at next transmission opportunity. It should be initialized to 0. |
| lc_on_timewheel # | This field indicates if the LCD is currently queued to the timewheel. It should be initialized to 0. |

## ABR Code Variables Definitions  (Page 2 of 7)

| Field Name | Field Description |
|---|---|
| alter_sched # | These encoding bits will alter the scheduling of cells on a Virtual Circuit (VC) |
| | 000 = Normal Scheduling — Scheduling is not altered. |
| | 001 = VC on VP — This VC is contained on a virtual path and will share the VP bandwidth after one packet is sent. The scheduling parameters are contained in the descriptor for the virtual path that is pointed to by the Virtual Path Discriptor Address field in this LCD. |
| | 010 = MPEG-2 Scheduling — The cells being sent out on this connection are monitored for a Peak Cell Rate (PCR). If a PCR is found, the AAL5 packet is terminated at the end of the MPEG-2 frame and the last cell is scheduled to go out at the time specified in the PCR. |
| | 011 = Packet-based scheduling — Packets will be scheduled at the average interval and cells within the packet will be scheduled at the peak interval. This is useful for sending information where variably-sized packets need to be sent at regular intervals. |
| | 100 = ABR scheduling — This VC will send Resource Management cells and adjust its transmisin rate according to the behaviors specified in the ATM Forum Traffic Management Specification, Version 4.0. |
| | 101 = Fair VC on VP — This VC is contained on a virtual path and will share the VP bandwidth after one cell is sent. The scheduling parameters are contained in the descriptor for the virtual path which is pointed to by the Virtual Path Descriptor Address field in this LCD. |
| | 110 = Reserved — For MPEG-2 scheduling. |
| | 111 = Reserved |
| transmit_priority # | This field specifies the priority of transmission on this connection. 0=high, 1=medium, 2=low. |
| max_resolution # | If this bit is set, the lower eight bits of the average interval and peak interval parameters contain a fractional component. This allows a finer resolution for scheduling. For example, for a peak interval of 1.5 time units, the value written to the peak_interval field should be hex 0180. If this bit is set, the initial value of timestamp should contain the current timeslot counter shifted 16 bits to the left. |
| max_burst_mult # | The values in this field and the next field are used to limit the number of cells that can be transferred at the peak rate. The max_burst_value will be multiplied by 4 to the power of the value in this field to yield the maximum credit time. This time is expressed in cell times and represents the time it would take to acquire the maximum number of cell credits. This maximum credit time should equal the maximum number of cells that can be transferred at the peak rate (MBS) times the difference between the average and intervals. Maximum credit time = MBS * (AI-PI) where MBS = maximum burst size, AI = average interval, and PI = peak interval. MBS must be at least 1 to transmit at peak rate. If MBS is not at least 1 the peak interval should be set to the average interval. |
| max_burst_value # | The value in this field will be multiplied by 4 to the power of the value in the max_burst_mult field to yield the maximum credit time. |
| head_packet_pointer | This field is used to chain buffers to LCDs. |
| tail_packet_pointer | This field is used to chain buffers to LCDs. |
| transmit_length | This field contains the length of the currently transmitted packet. |
| free_on_xmit | This bit is set if the header of the currently transmitted packet has specified that the packet is to be freed after transmission. |
| queue_on_xmit | This bit is set if the header of the currently transmitted packet has specified that the packet is to be queued after transmission. |
| dma_on_xmit | This bit is set if the header of the currently transmitted packet has specified that a DMA descriptor is to be queued after transmission. |

## ABR Code Variables Definitions (Page 3 of 7)

| Field Name | Field Description |
|---|---|
| buffer_offset | This field contains the offset into the buffer that the transmit data starts. |
| enable_blocking # | When set, this bit enables OAM blocking cells to be sent on the associated VC by the segmentation logic. Other fields in the LCD define the content and frequency of these frames. Setting this bit also forces statistics to be kept for the associated LCD regardless of the state of the enable_statistics bit. This same function can be globally controlled in the SEGBF control register. |
| enable_statistics # | When set, this bit enables statistics keeping for the associated LCD. Another eight bytes of the associated LCD is used to maintain counts of the total number of user cells and the total number of user cells with CLP=0 that have been sent over this VC. This same function can be globally controlled in the SEGBF control register. |
| flush_LCD # | When set, this bit causes the segmentation logic to flush all frames currently queued to the LCD without performing any segmentation on them. |
| generate_CRC10 # | This bit is set if the header of the currently transmitted packet has specified that the cells in this packet should have CRC-10 generated. When set, this bit overrides the other AAL select bits in the LCD and forces the segmentation logic to generate a CRC-10 terminated cell. In this mode, the segmentation logic fetches 52 bytes of data from memory. The HEC is calculated over the first four bytes fetched, and appended following these four bytes. The next 48 bytes are used to calculate the CRC-10 and the zero-padded two-byte result is appended after the 48 bytes of data. This function is intended to make it easier to send OAM cells. |
| OAM_CLP_source # | When reset, the CLP bit for all OAM cells is retrieved from the ATM header in the LCD. When set, the CLP bit for all OAM cells is retrieved from the OAM_CLP_value field in the LCD. This allows OAM traffic to be sent with a different CLP value than other traffic on the VC. |

## ABR Code Variables Definitions (Page 4 of 7)

| Field Name | Field Description |
|---|---|
| AAL_type # | This field specifies the AAL type to be used with this connection. The following values are decoded by the hardware: <br><br> '0'x = Raw 48-byte mode. When an LCD with this value is encountered, the segmentation logic assembles a cell from the four ATM header bytes in the LCD, a calculated HEC, and the 48 bytes pointed to by the transmit offset. <br><br> '1'x = Raw 52-byte mode. When an LCD with this value is encountered, the segmentation logic assembles a cell from the four ATM header bytes fetched using the transmit offset, a calculated HEC, and the next 48 bytes following the ATM header. <br><br> '2'x = Raw 53-byte mode. When an LCD with this value is encountered, the segmentation logic assembles a cell directly from the 53 bytes pointed to by the transmit offset. <br><br> '3'x = Reserved <br><br> '4'x = MPEG blocking with PCR termination. This mode is handled the same as Fixed-size blocking mode with one additional feature. An MPEG transport stream packet that contains a PCR causes the current AAL5 frame to be terminated even if the predefined number of packets has not been assembled into an AAL5 frame. <br><br> '5'x = AAL5 mode. When an LCD with this value is encountered, the segmentation logic assembles a cell in accordance with the AAL5 specification. The four-byte ATM header is retrieved from the LCD, then the HEC is calculated and appended. The next 48 bytes of data are fetched from the current segmentation pointer and the CRC-32 is calculated over all 48 bytes. If fewer than 48 bytes of data are available, the data is zero-padded to complete the cell. When all data in the buffer has been exhausted, the last cell will contain the AAL5 trailer. The CPCS user to user field in the AAL5 trailer depends on the state of the two bits defined in INTST. <br><br> '6'x = Fixed-size blocking. When an LCD with this value is encountered, the segmentation logic assembles a predefined number of packets of a predefined size into an AAL5 packet. The number and size of the packets are defined in two other fields of the LCD. The length field of the LCD must be initialized with a value that is an integral multiple of the predefined size and the predefined number of packets. That is, if the predefined size is x'47' (188 bytes) and the programmed number is 2, then the length must be initialized to n•188•2, or 188, 376, 564 etc. <br><br> '7'x = MPEG blocking with PCR delay and termination. This mode is handled the same as MPEG blocking with termination but with one added feature. As well as terminating an AAL5 frame, a transport stream packet that contains a PCR will be delayed by the segmentation logic until an internal time base indicates that the correct time has come for the packet to be delivered to the remote end. In this mode, the first PCR containing transport stream packet causes the segmentation logic to derive a PCR adjust value relative to an internal time-base. All PCR-containing packets processed in the future use this derived PCR adjust value to determine if the correct amount of time has passed indicating that the packet should be forwarded to it's destination. If the correct amount of time has not passed, the last cell of the AAL5 frame is delayed until the correct time. If the transport stream contains multiple PID values, a bit in the SEGBF control register can be used to globally enable only matching PID PCR recognition and delaying. |
| ATM_header # | This field contains the first four bytes of the ATM header. |
| segmentation_pointer | This field contains a pointer to the next data to be transmitted. In normal operation this field is initialized by the cell scheduler when a new frame is queued for segmentation. |
| current_CRC | This field contains the CRC as it is being built. |
| BIP-16 | This 16-bit field contains the Bit Interleaved Parity accumulated over the last block of data. After initialization, this field should only be accessed by the hardware. |

## ABR Code Variables Definitions (Page 5 of 7)

| Field Name | Field Description |
|---|---|
| Monitor_sequence_number | This eight-bit field contains the Monitor Sequence Number that is used to construct the hardware generated OAM cells. After initialization, this field should only be accessed by the hardware. |
| OAM_cell_transmitted | This one-bit field contains a flag indicating that an OAM cell has been sent. After initialization, this field should only be accessed by the hardware. |
| Previous_PCR_bits_17_16 | This two-bit field contains bits 17 and 16 of the PCR in the most recently segmented MPEG transport stream packet. After initialization, this field should only be accessed by the hardware. |
| OAM_PTI_bit0 # | This one-bit field is copied directly to the low bit of the payload type field in the ATM header as the segmentation logic is building an OAM cell. |
| Previous packet contained PCR | This one-bit field is used during MPEG processing by the hardware to indicate that a previous transport stream packet contained a PCR. After initialization, this field should only be accessed by the hardware. |
| OAM_CLP_value # | This one-bit field can be used to provide the value for the CLP bit in the hardware generated OAM cells. If the OAM_CLP_source bit elsewhere in the LCD is in the correct state, this bit will be copied directly to the CLP bit in the ATM header of the OAM cell. |
| OAM_block_size # | This two-bit field defines the number of user cells that are sent between OAM cells.<br>00 = 128 cells<br>01 = 256 cells<br>10 = 512 cells<br>11 = 1024 cells |
| Current_Blocking_Count | This eight-bit field contains the current count of four-byte values that have been assembled into cells and sent out on this LCD for all fix block or MPEG AAL types. Other than initialization, this field should only be accessed by the hardware. |
| Fixed_Blocking_size # | This eight-bit field should be initialized by the software to contain the number of four-byte values that constitute a packet. For MPEG2 this register should be set to x'47' (4•x'47' = 188 byte transport stream packet). |
| PID_field_is_valid | When set, this one-bit field indicates that the PID field in the LCD contains a valid PID. After initialization, this field should only be accessed by the hardware. |
| PID_bits_12_11 | This two-bit field contains bits 12 and 11 of the previously saved PID. After initialization, this field should only be accessed by the hardware. |
| Current_transport_stream_packet | This five-bit field contains the number of the current transport stream packet that is being segmented. After initialization, this field should only be accessed by the hardware. |
| Check_PCR | This one-bit field is set when the segmentation hardware has determined that it is time to check and delay a cell if the PCR indicates. Other than initialization, this field should only be accessed by the hardware. |
| PCR_present | This one-bit field is set by the hardware to indicate that the current transport stream packet contained a PCR. Other than initialization, this field should only be accessed by the hardware. |
| PID_matches | This one-bit field is set by the hardware to indicate that the PID in the transport stream being segmented matches the PID which was previously locked on. |
| Packets_per_AAL5_frame # | This five-bit field should be initialized by the software to indicate how many packets should be concatenated into an AAL5 frame. |
| previous_PCR_bits_15_0 | This 16-bit field contains the least significant 16 bits of the last PCR that has been encountered. After initialization, this field should only be accessed by the hardware. |
| PID_bits_10_0 | This two-bit field contains bits 10 - 0 of the previously saved PID. After initialization, this field should only be accessed by the hardware. |

## ABR Code Variables Definitions  (Page 6 of 7)

| Field Name | Field Description |
|---|---|
| PCR_delta_valid | This one-bit field is set by the hardware to indicate that the 36-bit PCR delta field is valid. After initialization, this field should only be accessed by the hardware. |
| PCR_delta | This 36-bit field is set by the hardware the first time a PCR-containing transport stream packet is encountered by the segmentation logic. The high 36 bits of the PCR in the packet, along with an internal time base, are used to determine a PCR delta for future use. After initialization, this field should only be accessed by the hardware. |
| explicit_rate | This 16-bit field contains the explicit cell rate as defined for ABR traffic on this LCD. |
| current_rate | This 16-bit field contains the current cell rate as defined for ABR traffic on this LCD. |
| minimum_rate | This 16-bit field contains the minimum cell rate as defined for ABR traffic on this LCD. |
| backward_ptr | When software needs to send a backward RM cell, this 32-bit field should be updated with the address of a buffer that contains the desired backward RM cell. After the segmentation logic transmits the cell, this field will be cleared by the hardware. |
| total_user_cells | This 32-bit field contains a count of the total number of user cells that have been sent on this LCD, after either statistics or blocking are enabled either globally or locally for this LCD. This field should be set to zero when the connection is initialized. An event will be generated when this count wraps. |
| total_user_cells_CLP0 | This 32-bit field will contain a count of the total number of user cells that have been sent on this LCD with CLP = 0, if either statistics or blocking are enabled either globally or locally for this LCD. This field should be set to zero when the connection is initialized. An event will be generated when this count wraps. When configured, this field can also be total packets transmited. |
| threshold_1&2 | These fields are compared to the upper 24 bits of the bytes_queued field to determine when a threshold is crossed and the pool ID for the received LCD should be changed. |
| pool_id1&2 | These fields are used to change the POOL ID when a threshold is crossed. |
| bytes_queued | This field is used to keep track of the number of bytes queued for transmission on this LCD. |
| timer_type # | These encoded bits determine the time of timer.<br><br>00 = relative non-periodic timer — The expiration time will be in one timer period. The timer will not be scheduled again automatically.<br><br>01 = relative periodic timer — The expiration time will be in 1 timer period. The timer will be automatically scheduled again .<br><br>10 = absolute non-periodic timer — The timer will expire at the time specified by the timestamp field. The timer will not be automatically scheduled again .<br><br>11 = absolute periodic timer — The timer will expire at the time specified by the timestamp field. The timer will be scheduled again, automatically, using the time specified in the timer_period field. |
| timer_period # | This field specifies the number of timeslots before the timer expires. |
| dma_desc_addr # | The DMA descriptor pointed to by this field will be queued for execution when the timer expires. |
| CRM | Missing RM-cell count. CRM limits the number of forward RM-cells which may be sent in the absence of received backward RM-cells. CDF is written to the NCRM field whenever a backward RM cell is detected. |
| iCDF | Cutoff Decrease Factor (CDF) controls the decrease in ACR associated with CRM. CDF is zero or a power of 2 value in the range of 1/64 to 1. iCDF represents the power of 2 that is in the denominator of CDF. CDF = 1/(2**iCDF) so iCDF = log (base 2) of 1/CDF. Range = 1 to 6. A zero value for CDF should be represented as 0xFF for iCDF. |

## ABR Code Variables Definitions (Page 7 of 7)

| Field Name | Field Description |
|---|---|
| iMCR | This is the reciprocal of the Minimum Cell Rate (MCR). MCR is represented in the ABR Rate format. iMCR needs to be an integer representing the interval between cells, in units of timeslots per cell. The following formula illistrates the conversion from MCR to iMCR. iMCR = 1/(MCR) * 53 * 8 * (TSP/CI*(2**23)). The Timeslot Prescalar (TSP) is defined by the CSKED Timeslot Prescalar Register. The clock interval (CI) is determined by the CRSET Clock Control Register. With the TSP set to one timeslot per cell transmission time, iMCR =1 for a full bandwidth, 2 for a half bandwidth, etc. |
| PCR | The Peak Cell Rate (PCR) is the cell rate that the source may never exceed. PCR should be in the ABR Rate format. |
| iRDF | Rate Decrease Factor (RDF) controls the decrease in the cell transmission rate. RDF is a power of 2 value in the range of 1/32,728 to 1. iRDF represents the power of 2 that is in the denominator of RDF. RDF = 1/(2**iRDF) so iRDF = log (base 2) of 1/RDF. Range = 1 to 15. |
| iRIF | Rate Increase Factor (RIF) controls the increase in the cell transmission rate. RIF is a power of 2 value in the range of 1/32,728 to 1. iRIF represents the power of 2 that is in the denominator of RIF. RIF = 1/(2**iRIF) so iRIF = log (base 2) of 1/RIF. Range = 1 to 15. |
| ICR | The Initial Cell Rate is the rate at which a source should send initially and after an idle period. ICR should be in the ABR Rate format. |

## Transmit Data Structure Linkage
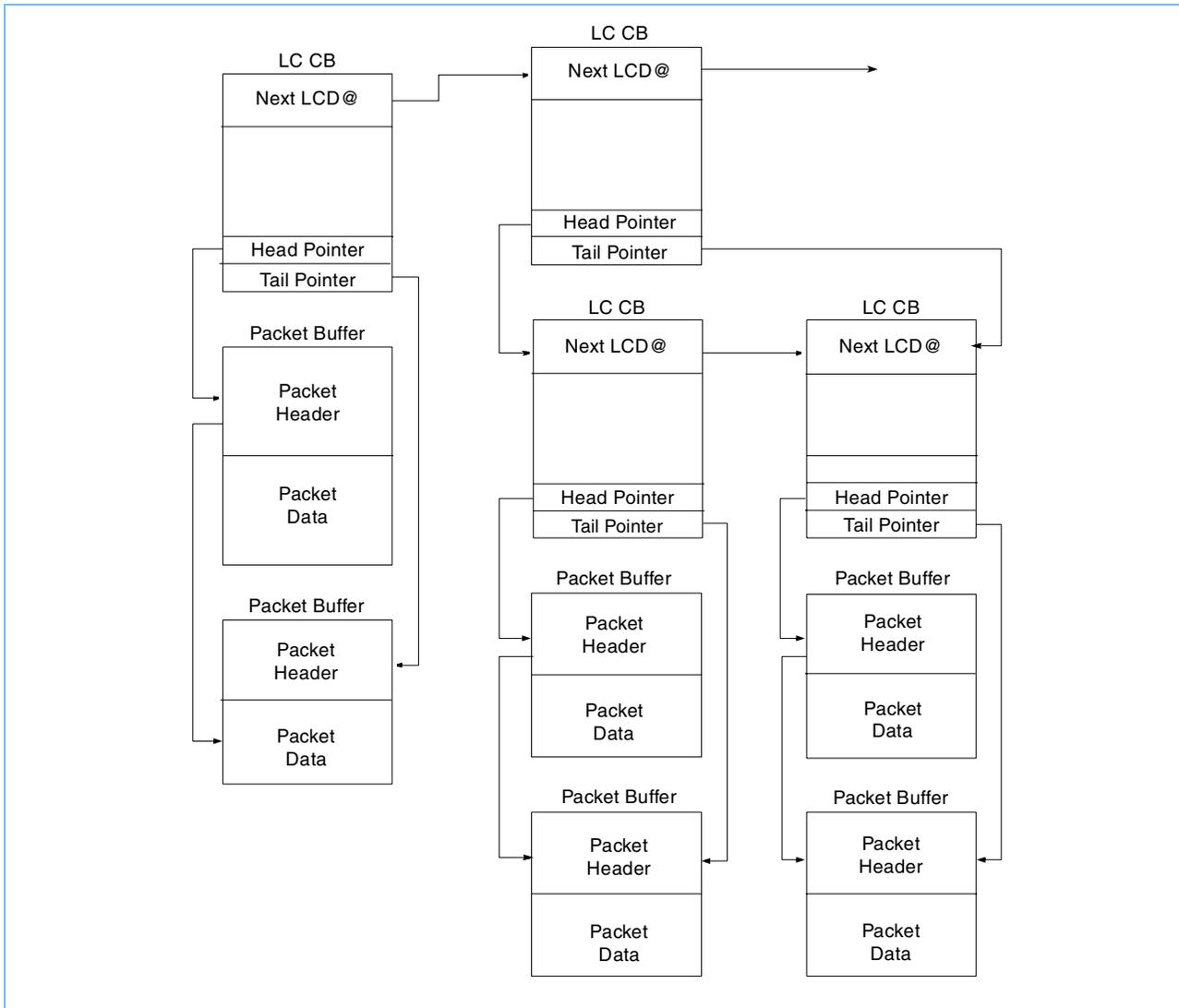


### Receive LCD Data Structure and Modes

The following are the major differences in this pass:
- The packed portion (first 32 bits) is almost completely different, but is now easier to set up and to follow.
- The host data and OAM fields are swapped to improve performance.
- Several options can no longer be set on an LCD basis; they are set on a chip basis instead. See RAALL for details.
- New modes and options.

The format of the receive LCD structure depends on which AAL is being configured and which options are used. The following sections detail the receive portion of the LCD for each major option:

**Raw LCD**

A raw LCD allows raw ATM cells to be received with no reassembly. The user can select to receive 53-, 52-, or 48-byte cells. The packet header contains the ATM header regardless of the cell length selected. The cell data is then placed after the packet header at the configured receive offset. The 53-byte mode stores the entire cell including HEC, the 52-byte mode stores the entire cell minus the HEC, and the 48-byte mode stores only the ATM cell payload. Optional CRC-10 checking is available in raw modes.

## Raw LCD Layout

```
class RawLcd {
  // packet portion - bit32
  bit2   aalType;
    // 00 - Raw
  bit2   mode;
    // 00 - none
  bit2   size;
    // 00 - 53 byte cell
    // 01 - 52 byte cell
    // 10 - 48 byte cell
    // 11 - res
  bit2   state;
    // 00 - down
    // 01 - idle/enabled
  bit1   res;
  bit1   res;
  bit1   res;
  bit1   storeCrc10;
  bit1   res;
  bit3   res;
  bit1   res;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;
  bit32 res;
  bit32 res;
  bit32 res;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**Raw Routed LCD**

A raw routed LCD receives data in the same way that a raw LCD does. Once received, the cell buffer is routed internally to the scheduler and rescheduled for transmission. Normally when a cell is received, the receive LCD address is written into the packet header and the buffer is surfaced to the user. When a cell is routed, the routedLcd field is used to fill in the LCD address in the packet header. This allows cells to be routed out the transmit interface with the same or different vp/vc.

When routing cells, the user can choose to surface non-user data (NUD) cells to the user, or to route them in line with the user-data cells. This is controlled with the routeNud bit in the LCD. A cell is considered NUD if the most significant bit of the PTI field in the ATM header is turned on.

If the routed cell stream is actually an AAL5 packet stream, then earlyDrop mode might be considered. In this mode, a cell being dropped due to resource causes the LCD to go into error mode until the cell that contains the user indicate (UIND) bit is received. All cells received in error mode are dropped, except the final cell which is forwarded. This conserves bandwidth while maintaining the AAL5 integrity. In RAALL, an additional POOL can be specified to ensure that these final cells are always forwarded even when resources are low.

The low order bits in the routedLcd field should be set correctly to free the buffer on transmission. These bits corespond to the flag bits in the packet header. Raw routing is also called forwarded or fast forward mode.

## Raw Routed LCD Layout

```
class RawRoutedLcd {
  // packet portion - bit32
  bit2   aalType;
    // 00 - Raw
  bit2   mode;
    // 01 - routed
  bit2   size;
    // 00 - 53 byte cell
    // 01 - 52 byte cell
    // 10 - 48 byte cell
    // 11 - res
  bit2   state;
    // 00 - down
    // 01 - idle/enabled
    // 11 - error        // only valid in early drop mode
  bit1   res;
  bit1   res;
  bit1   res;
  bit1   earlyDrop;
  bit1   routeNud;
  bit3   res;
  bit1   res;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;
  bit32 routedLcd;
  bit32 res;
  bit32 res;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

### Raw Cut Through LCD

A raw cut-through LCD receives data in the same way that a raw LCD does. Once received, the ctRxqNum field is used to get a DMA descriptor address or a buffer address (direct cut-through enabled) from the coresponding receive queue. The DMA descriptor is then built using the cell buffer address, the data length, and the cut-through flags specified in RAALL. After being built, it is queued to the DMA queue specified using dmaQsel and the configuration in DMAQS. If there is no DMA descriptor available, then a no descriptor event is queued (see Entity 14: *Receive Queues (RXQUE)* on page 300).

## Raw Cut Through LCD Layout

```
class RawCutThruLcd {
  // packet portion - bit32
  bit2   aalType;
    // 00 - Raw
  bit2   mode;
    // 10 - cut thru
  bit2   size;
    // 00 - 53 byte cell
    // 01 - 52 byte cell
    // 10 - 48 byte cell
    // 11 - res
  bit2   state;
    // 00 - down
    // 01 - idle/enabled
  bit1   res;
  bit1   res;
  bit1   res;
  bit1   storeCrc10;
  bit1   res;
  bit3   ctRxqNum;
  bit1   dmaQsel;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit32 res;
  bit32 res;
  bit32 res;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**Raw FIFO LCD**

This mode is depreciated and should not be used. In the IBM2520L8767, a new FIFO mode similar to the AAL5 FIFO mode will be implemented.

## Raw FIFO LCD Layout

```
class RawFifoLcd {
   // packet portion - bit32
   bit2   aalType;
     // 00 - Raw
   bit2   mode;
     // 11 - fifo mode
   bit2   size;
     // 00 - 53 byte cell
     // 01 - 52 byte cell
     // 10 - 48 byte cell
     // 11 - 48 byte cell contiguous
   bit2   state;
     // 00 - down
     // 01 - idle/enabled
     // 10 - reasm
     // 11 - error
   bit1   rtoTest;
   bit1   rtoEnable;
   bit1   res;
   bit1   res;
   bit1   res;
   bit3   res;
   bit1   res;
   bit3   rxqNum;
   bit4   res;
   bit8   res;

   bit6   res;
   bit10  fifoThresh;
   bit10  fifoSizeMask;
   bit6   res;
   bit32  baseAddr;
   bit11  maxCellCnt;
   bit10  currRxCellCnt;
   bit11  totalRxCellCnt;
   bit32  tucCLP0;
   bit32  tuc;
   bit32  hostData;
   bit16  oamTUC;
   bit16  oamBIP;
};
```

There are four FIFO modes supported, and the mode is specified in the protocol enable bits of the receive portion of the LCD. The four modes are:

- 53-byte cells
- 52-byte cells (no HEC)
- 48-byte cells
- Contiguous 48-byte cells

The first three modes are logically the same and differ only in how much data is delivered to the user. In these modes, cells are received into each 64-byte area of the receive FIFO. Another way to think of it is the receive pointer increments by 64 bytes with the reception of each cell. When the end of the FIFO is reached, it wraps back to the begining of the FIFO buffer.

The last mode is fundamentally different in that the 48-byte payload from each cell is copied into the FIFO buffer in a contiguous fashion. Every cell is contiguous in memory, so the tail portion of the FIFO buffer is not used if it cannot hold a complete cell payload. Again, if the end of the buffer is reached, then it wraps back to the begining of the buffer. The following figure shows the two FIFO layouts:

## Receive FIFO Buffer Layouts

Non-Contiguous

Base Address - 0

Cell Data

Base Address - 64

Cell Data

Cell Data

Contiguous

Base Address + 0

Cell Data

Base Address + 48

Cell Data

Cell Data

Unused Area
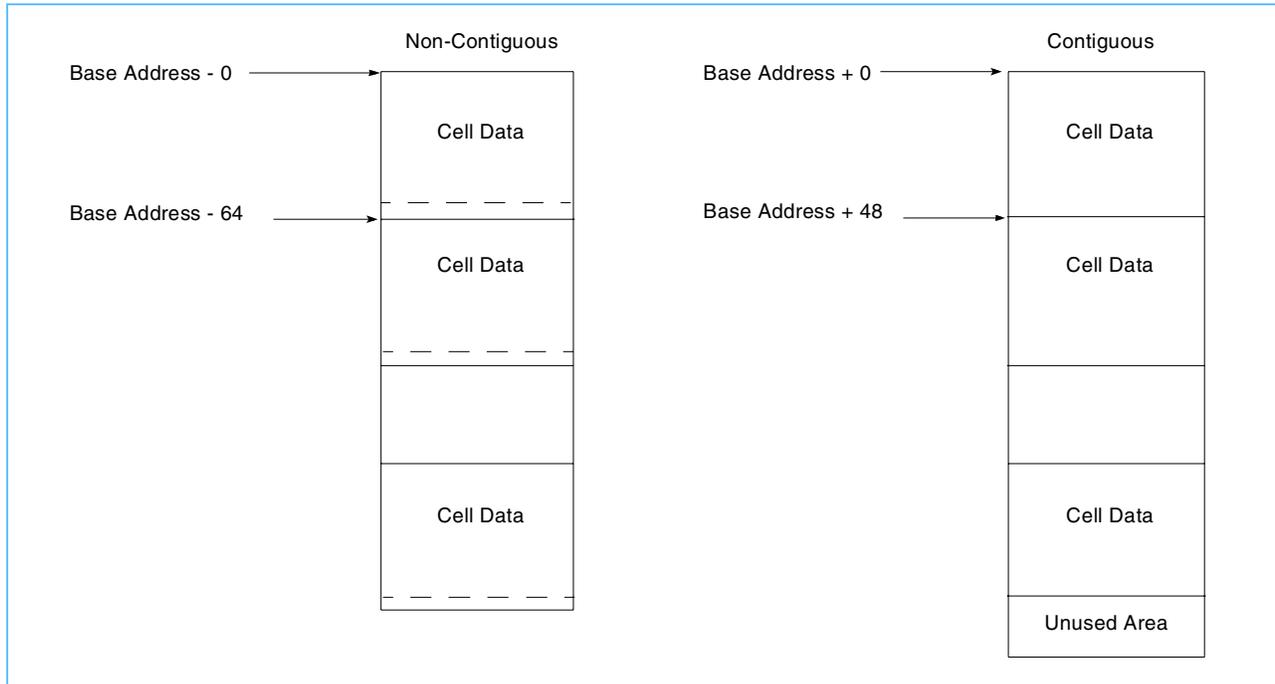
The receive FIFO is maintained by RAALL using six number fields in the receive portion of the LCD:

- Cell Threshold Count
- FIFO Size Mask
- Base Addr and Rx Ptr
- Maximum Cell Count
- Current RX Cell Count
- Total RX Cell Count

Once set up, cells are received into the FIFO buffer. FIFO threshold and/or FIFO full events are posted as cells are received. The user can then process the cells that are surfaced with these events. If a threshold and full event coincide, then the threshold event is surfaced.

The user informs RAALL that cells have processed by using the synchronization operation. The synchronization operation allows the user to specify either a specific number of cells to synchronize, or to use the threshold to synchronize cells. RAALL provides a status bit to indicate that a bad synchronization operation has been attempted (more cells synchronized than are in FIFO).

If status is enabled in the LC, then cell drop events are raised when a cell is dropped because the FIFO was full.

Statistics and OAM blocking support is supported in FIFO mode.

If enabled in the RAALL control register, then FIFO LCs are included in the LCs reassembling counts.

**AAL5 LCD**

An AAL5 LCD allows AAL5 packets to be received with no special processing. The headerThresh can be used to allow packet header thresholding events to surface.

## AAL5 LCD Layout

```
class Aal5Lcd {
  bit2   aalType;
  // 01 - Aal 5
  bit2   mode;
    // 00 - none
  bit1   res;
  bit1   res;
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit1   descState;
  bit3   res;
  bit1   res;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit16 headerThresh;
  bit16 maxLength;
  bit32 rxBuffAddr;
  bit32 rxCrc;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**AAL5 Routed LCD**

An AAL5 Routed LCD allows AAL5 packets to be received. Once received, the packet buffer is then routed internally to the scheduler and rescheduled for transmission. Normally when a packet is received, the receive LCD address is written into the packet header and the buffer is surfaced to the user. When a packet is routed, the routedLcd field is used to fill in the LCD address in the packet header. This allows packets to be routed out the transmit interface with the same or different VP/VC.

The low order bits in the routedLcd field should be set correctly to free the buffer on transmission. These bits corespond to the flag bits in the packet header.

AAL5 routing is also called forwarded or fast forward mode.

## AAL5 Routed LCD Layout

```
class Aal5RoutedLcd {
  bit2   aalType;
  // 01 - Aal 5
  bit2   mode;
    // 01 - routed
  bit1   res;
  bit1   res;
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit1   routeNud;
  bit3   res;
  bit1   res;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit32 routedLcd;
  bit32 rxBuffAddr;
  bit32 rxCrc;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**AAL5 Cut-Through Mode 6 LCD**

An AAL5 Cut-Through Mode 6 LCD allows AAL5 packets to be received with additional cut-through processing. Once the entire packet is received, the ctRxqNum field is used to get a DMA descriptor address or a buffer address (direct cut-through enabled) from the corresponding receive queue. The DMA descriptor is then built using the buffer address, the data length, and the cut-through flags specified in RAALL. After being built, the descriptor is enqueued to the DMA queue specified using dmaQsel and the configuration in DMAQS. If there is no DMA descriptor available, then a no descriptor event is queued (see Entity 14: *Receive Queues (RXQUE)* on page 300).

The amount of data DMA'd includes the packet header, any padding bytes specifed with the receive offset, and up to cutThruThresh bytes of data.

## AAL5 Cut-Through Mode 6 LCD Layout

```
class Aal5Ct6Lcd {
  bit2   aalType;
  // 01 - Aal 5
  bit2   mode;
    // 10 - cut thru
  bit1   ctMode;
    // 0 - mode 6
  bit1   useFifoRegs;
    // 0 - use normal cut thru desc
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit1   res;
  bit3   ctRxqNum;
  bit1   dmaQsel;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit16 cutThruThresh;
  bit16 maxLength;
  bit32 rxBuffAddr;
  bit32 rxCrc;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

### AAL5 Cut-Through Mode 6 LCD Using Hardware FIFO Registers

This is the same as AAL5 Cut-Through Mode 6, except for how DMA descriptors are built. This mode is meant to be used with a hardware FIFO device. Two DMA descriptors are built using the hardware FIFO registers in RAALL. The first descriptor is built to DMA the data (same as Mode 6) to the first FIFO address. The second DMA descriptor is built to write the buffer address to the second FIFO address, thus forming a complete command.

## AAL5 Cut-Through Mode 6 Using Hardware FIFO Registers LCD Layout

```
class Aal5Ct6Lcd {
  bit2   aalType;
  // 01 - Aal 5
  bit2   mode;
    // 10 - cut thru
  bit1   ctMode;
    // 0 - mode 6
  bit1   useFifoRegs;
    // 1 - use hardware fifos regs to build cut thru desc
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit4   channelNum;
  bit1   dmaQsel;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit16 cutThruThresh;
  bit16 maxLength;
  bit32 rxBuffAddr;
  bit32 rxCrc;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**AAL5 Cut-Through Mode 7 LCD**

An AAL5 Cut-Through Mode 7 LCD allows AAL5 packets to be received with additional cut-through processing. Once cutThruThresh bytes of data are received the ctRxqNum field is used to get a DMA descriptor address or a buffer address (direct cut-through enabled) from the coresponding receive queue. The DMA descriptor is then built using the buffer address, the data length, and the cut-through flags specified in RAALL. After being built, the descriptor is enqueued to the DMA queue specified using dmaQsel and the configuration in DMAQS. If there is no DMA descriptor available, then it is retried on the next cell reception. If there is no DMA descriptor available when the packet completes, then a no descriptor event is queued (see Entity 14: *Receive Queues (RXQUE)* on page 300).

The amount of data DMA'd includes the packet header, any padding bytes specifed with the receive offset, and up to cutThruThresh bytes of data.

## AAL5 Cut-Through Mode 7 LCD Layout

```
  class Aal5Ct7Lcd {
    bit2   aalType;
    // 01 - Aal 5
    bit2   mode;
      // 10 - cut thru
    bit1   ctMode;
      // 1 - mode 7
    bit1   dmaedHeader;
    bit2   state;
      // 00 - down
      // 01 - idle
      // 10 - reasm
      // 11 - error
    bit1   rtoTest;
    bit1   rtoEnable;
    bit1   tmpCLP;
    bit1   tmpCongestion;
    bit1   descState;
    bit3   ctRxqNum;
    bit1   dmaQsel;
    bit3   rxqNum;
    bit4   rxPoolId;
    bit8   rxOffset;

    bit16 cutThruThresh;
    bit16 maxLength;
    bit32 rxBuffAddr;
    bit32 rxCrc;
    bit32 tucCLP0;
    bit32 tuc;
    bit32 hostData;
    bit16 oamTUC;
    bit16 oamBIP;
  };
```

### AAL 5 Cut-Through Scatter Mode LCD

Cut-through scatter mode allows received data to be "scatter DMAed" to host memory via a list of host pages. When enough data on a particular VC has arrived to fill a host page, a DMA operation is automatically initiated to transfer that data from IBM2520L8767 memory to a page in host memory. As this occurs, subsequent receive data is accumulated until another host-page-sized page of data is collected, at which time a DMA is initiated and the process continues.

Upon receiving the last data in a packet, the packet header along with all the DMA descriptors used to transfer the 'scattered' data is passed up to the host. The host can now access the received data since it is already in host memory and it has the packet header and the list of DMA descriptors describing where each page of the data packet resides in host memory.

## AAL5 Cut-Through Scatter Mode LCD Layout

```
class Aal5ScatterLcd {
  bit2   aalType;
  // 01 - Aal 5
  bit2   mode;
    // 11 - scatter mode
  bit1   res;
  bit1   res;
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit1   res;
  bit3   ctRxqNum;
  bit1   dmaQsel;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit6  numDesc;
  bit10 numHeadBytes;
  bit16 maxLength;
  bit32 rxBuffAddr;
  bit32 rxCrc;
  bit32 tucCLP0;
  bit32 tuc;
  bit32 hostData;
  bit16 oamTUC;
  bit16 oamBIP;
};
```

**AAL5 FIFO Mode LCD**

An AAL5 FIFO Mode LCD allows AAL5 packets to be concatenated in a single receive buffer. The receive buffer is the FIFO. This is useful with applications such as MPEG where small fixed-size packets are received, but the overhead of receiving a large number of packets per second is too high. This mode allows the packets to be gathered together into the FIFO (or super packet) and then be processed with a single receive interrupt.

The receive packet sizes do not need to be fixed-size, but the user needs to be able to parse the receive packets if they are not. The fifoThresh is used to specify the largest packet that will be received. When the free space in the FIFO falls to this threshold, the end of the current packet will terminate the FIFO packet. Subsequent cells are gathered into another packet.

The encoded fifoSize field specifies how large the FIFO buffer can grow. This value should be less than or equal to the size of the receive buffers as configured in VIMEM.

The fifoPtr should be initialized to zero and used by the IBM2520L8767 to keep track of the current location in the FIFO buffer between packets.

When a FIFO packet completes, there are two modes of operation. First, the default mode operates just like cut-through Mode 6. When the packet completes, a cut-through DMA descriptor is used to move the data to system storage. This can be disabled in the RAALL control register. When disabled, the packet is surfaced to software via a super-packet event (0xe) just like a normal AAL5 packet.

## AAL5 FIFO Mode LCD Layout

```
class Aal5Ct6Lcd {
  bit2   aalType;
  // 01 - Aal 5
  bit4   mode;
    // 1000 - same as cut thru mode 6 (see fifo mode bit below)
  bit2   state;
    // 00 - down
    // 01 - idle
    // 10 - reasm
    // 11 - error
  bit1   rtoTest;
  bit1   rtoEnable;
  bit1   tmpCLP;
  bit1   tmpCongestion;
  bit1   fifoMode;
    // must be set to 1 to use this mode
  bit3   ctRxqNum;
  bit1   dmaQsel;
  bit3   rxqNum;
  bit4   rxPoolId;
  bit8   rxOffset;

  bit3   fifoSize;
  bit13  fifoThresh;
  bit16  fifoPtr;
  bit32  rxBuffAddr;
  bit32  rxCrc;
  bit32  tucCLP0;
  bit32  tuc;
  bit32  hostData;
  bit16  oamTUC;
  bit16  oamBIP;
};
```

### LCD Field Definitions

The following are the definitions of the LCD fields grouped by major function:

A * after the name specifies a field that software should set up.

All reserved fields should be set to zero.

## Common LCD Field Definitions

| Field Name | Field Description | Note |
|---|---|---|
| aalType | Specifies the AAL for this LCD.<br>00 = Raw Mode<br>01 = AAL5 | 1 |
| state | Specifies the reassembly state for this LCD. This field is used by the IBM2520L8767, but in order to receive cells, an LCD must be initialized to idle state. The following are the valid values:<br>00 = Down State<br>01 = Idle State<br>10 = Reassembling State<br>11 = Error State | 1 |
| ctRxqNum | Specifies which rxque contains cut DMA descriptors. This field is only valid in cut-through and scatter modes. | 1 |
| channelNum | Specifies which hardware channel is to be used. This field is only valid in cut-through mode h. | 1 |
| dmaQsel | Specifies which RAAL DMA queue should be selected when the dma is sent to DMAQS. See DMAQS Raall/Csked Queue Number Register. | 1 |
| rxqNum | Specifies which rxque normal events should be posted. Note: some events may be routed to the error queue based on your RXQUE setup. | 1 |
| rxPoolId | Specifies which POOL ID whould be used when getting buffers for received packets. | 1 |
| rxOffset | Specifies the offset into the IBM2520L8767 buffer where the received packet should be placed. | 1 |
| tucCLP0 | LC statistic that counts the total users cells with CLP=0 received on this LC. For accurate counts, this should initialized to zero. **Note:** this field can be changed to count packets received. See RAALL Control Register. | 1 |
| tuc | LC statistic that counts the total users cells received on this LC. For accurate counts, this should initialized to zero. | 1 |
| hostData | If enabled, the contents of this field are placed in packet of each received packet for this LCD. One use of this, is to place a correlator to a host specific data structure for this LCD. | 1 |
| oamTUC | OAM blocking total user cells received on this LC. This is a rolling count that is updated when OAM blocking is enabled. A new value is set from each PM cell as it is received. | |
| oamBIP | OAM blocking 16-bit Bit-Interleaved-Parity. This is the calculated bip over the current block of cells for this LC. It is reset to zero on each PM cell reception for this LC. | |

1. Software should set up this field.

## LCD Raw Mode Field Definitions

| Field Name | Field Description | Note |
|---|---|---|
| mode | This field selects the major mode for this LCD. The other fields are based on this value.<br>00 = Normal<br>01 = Routed<br>10 = Cut Thru<br>11 = FIFO Mode | 1 |
| size | This field specifes how many bytes are stored when cell is received.<br>00 = 53-Byte cell<br>01 = Byte cell (no HEC)<br>10 = 48-Byte cell<br>11 = 48-Byte cells (contiguous mode in FIFO mode only) | 1 |
| storeCrc10 | When set, the CRC-10 state bit is written into the packet header. A one is written in the error status bit in word 0 if a bad CRC-10 is detected. | 1 |
| earlyDrop | When set, the cells for this LCD are assumed to form an AAL5 stream; early drop mode is enabled. | 1 |
| routeNud | When set, non-user data cells are routed just like other routed cells. When cleared, non-user data cells are terminated at this node. | 1 |
| routedLcd | When routing cells, this field is used to fill in the LCD field of the packet header. This allows the user to dynamically route cells back out the interface using a different LCD. The user should be sure to set the free on transmit bit in this field as if it was in a packet header. | 1 |
| fifoThresh | Specifies how many cells should be received in the receive FIFO before a FIFO threshold event should be surfaced. If this field is set to zero, no threshold events are surfaced. | 1 |
| fifoSizeMask | Used by RAALL to determine how large the FIFO buffer is.<br><br>SizeField (15-6)   FIFO Size   64-Byte cells   48-Byte cells<br>0000000000   64 bytes   1   1<br>0000000001   128 bytes   2   2<br>0000000011   256 bytes   4   5<br>0000000111   512 bytes   8   10<br>0000001111   1K bytes   16   21<br>0000011111   2K bytes   32   42<br>0000111111   4K bytes   64   85<br>0001111111   8K bytes   128   170<br>0011111111   16K bytes   256   341<br>0111111111   32K bytes   512   682<br>1111111111   64K bytes   1K   1365 | 1 |
| baseAddr | Specifies the base address of the FIFO buffer at initialization. Once enabled, RAALL uses this field to maintain a pointer to where the next cell is to be received. | 1 |
| maxCellCnt | Specifies the maximum number of cells that the FIFO buffer should contain. This does not need to be the maximum specified by the buffer size. | 1 |
| currRxCellCnt | Specifies how many cells have been received into the FIFO since the last threshold was crossed. This should be initialized to zero. After initialization, RAALL maintains this field. | 1 |
| totalRxCellCnt | Specifies how many active cells are contained in the FIFO (cells not acknowledged by user). This should be initialized to zero. After initialization, RAALL maintains this field. | 1 |

1. Software should set up this field.

## AAL5 Field Definitions (Page 1 of 2)

| Field Name | Field Description | Note |
|---|---|---|
| mode | This field selects the major mode for this LCD. The other fields are based on this value.<br>00 = Normal Mode<br>01 = Routed Mode<br>10 = Cut Thru Mode<br>11 = Scatter Mode | 1 |
| rtoTest | This is the reassembly timeout processing test-and-set bit. It is used by the IBM2520L8767, but should be initialized to zero. | 1 |
| rtoEnable | If set, reassembly processing is enabled for this LC, if the LC is running AAL5 or is in FIFO mode. | 1 |
| tmpCLP | Used by the IBM2520L8767 to track the current state of the OR'd CLP bit for the current AAL5 packet. This field should be set to zero at initialization. After initialization, the IBM2520L8767 maintains this field. | 1 |
| tmpCongestion | Used by the IBM2520L8767 to track the current state of the OR'd congestion bit for the current AAL5 packet. This field should be set to zero at initialization time. | 1 |
| descState | This is used by the IBM2520L8767 for cut-through processing. This should be initialized to zero. | 1 |
| headerThresh | Specifies how much data should be received before poping a packet start event. If it is set to zero, only complete packet events will be popped. | 1 |
| maxLength | Specifies the maximum amount of data that can be received per packet on this LC. If set to zero, the value in RAALL Max SDU Length Register is used. This allows the maximum packet size to be set on an LCD basis. | 1 |
| rxBuffAddr | This field is used by the IBM2520L8767, but should be initialized to zero by software. This field is used to track the current packet under reassembly. | 1 |
| rxCcrc | This field is used by the IBM2520L8767 to maintain the CRC residue as the current packet is reassembled. This field does not need to be initialized. | |
| routedLcd | When routing cells, this field is used to fill in the LCD field of the packet header. This allows the user to dynamically route cells back out the interface using a different LCD. The user should be sure to set the free on transmit bit in this field as if it was in a packet header. | 1 |
| ctMode | Specifies cut-through Mode 6 when cleared and Mode 7 when set. Mode 6 DMAs the header when the packet is complete, while Mode 7 DMAs the headers when the threshold is met. | 1 |
| cutThruThresh | Used to determine how much cut-through data is DMA'd. | 1 |
| dmaedHeader | This is used by the IBM2520L8767 for cut-through Mode 7 processing. This should be initialized to zero. | 1 |
| numDesc | This is used by the IBM2520L8767 for scatter processing, and sould be initialized to zero. | 1 |
| numHeadBytes | Specifies how many bytes of data should kept with the packet header when DMAing the final DMA list for a completed scatter packet. Must be less than the page size. | 1 |

1. Software should set up this field.

## AAL5 Field Definitions  (Page 2 of 2)

| Field Name | Field Description | Note |
|---|---|---|
| fifoSize | Used to specify the size of AAL5 FIFO buffers.<br>000 = 512 bytes<br>001 = 1KB<br>010 = 2KB<br>011 = 4KB<br>100 = 8KB<br>101 = 16KB<br>110 = 32KB<br>111 = 64KB | 1 |
| fifoThresh | Specifes maximum-sized packet to be received into the receive FIFO. | 1 |
| fifoPtr | Used by the IBM2520L8767. Initialize to zero. | 1 |

1. Software should set up this field.
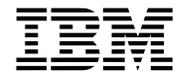
# Internal Organization: Entity Descriptions

This part contains detailed descriptions of the entities which, working together, make up the IBM2520L8767. The data flows through the chip have already been described; now the details of the registers and algorithms will be revealed. The entity descriptions are numbered for easy reference.

### Note on Set/Clear/Read Type Registers

There are many registers in the IBM2520L8767 that operate as a set/clear type. These registers have two addresses. The base address is for clearing bits in the register, and base address +4 bytes is for setting bits in the register. The setting or clearing operations occur only for those bits that have the value of '1' on the write of the register. Either of the addresses can be used for reading the register.

# Control Processor Bus Interface Entities

## Entity 1: The IOP Bus Specific Interface Controller (PCINT)

This entity provides PCI specific interfacing between the external connection and the internal entities. It will support the following functions:
- PCI memory target
- PCI master
- Address and data latching
- Provide parity error detection and generation
- Provide configuration space registers

**PCI Options Taken**
- Medium address decode design point
- Locking as a memory target supported
- Interrupt A will be supported, with interrupt 2 as a the sideband signal
- Registers will not burst, but cause retries when a burst is attempted
- BIST defaults set at the PCI 2 second maximum

**PCI Target Response**
- A Target Retry is issued if a burst crosses the end of the IBM2520L8767's memory space.
- A Target Abort will be issued if AD and command bus have bad parity (address phase parity error). Optionally, if SERR# is enabled, it will also be returned.
- If enabled, the PERR# signal will be driven on bad parity during data write cycles (data phase parity error) when the IBM2520L8767 is the target of the command.
- A Target Retry will be issued by the IBM2520L8767 if internal contention will cause a large bus access delay.

**PCI Master Response**
- A Master Abort will be issued if DEVSEL# is not asserted after five clocks.
- If enabled, the PERR# signal will be driven on bad parity during data read cycles (data phase parity error) when the IBM2520L8767 is the initiator of the command.

**PCI Master Retry**
- The IBM2520L8767 will retry when requested by the slave.

### 1.1: PCINT Config Word 0

Identifies this device and vendor type, allocated by PCI SIG.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0000 |
| **Restrictions** | Can be read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'00A11014', but alterable at power-up/reset time with Crisco code. See Entity 16: *Nodal Processor Bus Interface (NPBUS)* on page 340 for details. |
| **Power on Reset value (Little Endian)** | X'1410A100' |

| Device ID | Vendor ID |
|---|---|

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-16 | 15-0 | Device ID | This is a unique two-byte device ID assigned to this adapter. |
| 15-0 | 15-0 | Vendor ID | This is a unique two-byte vendor ID. |
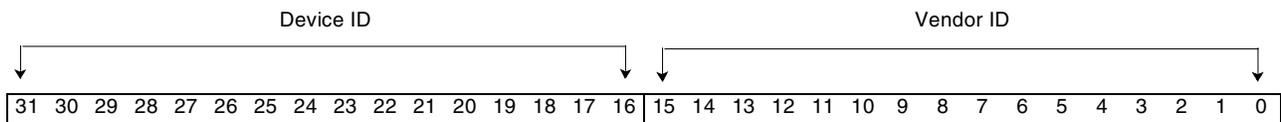
### 1.2: PCINT Config Word 1

The Status register is used to record status information for the PCI bus related events. Writing '1' to a bit in this register will reset that bit. The Command register provides coarse control over a device's ability to generate and respond to PCI cycles. Access type of the Command register is read/write. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write and Read/Reset |
| **Address** | XXXX 0004 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'02800000' |
| **Power on Reset value (Little Endian)** | X'00008002' |



| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31 | 15 | Detected Parity Error. | This bit is set by the device whenever it detects a parity error, even if parity error handling is disabled (as controlled by bit 6 of PCINT Configuration Word 1). |
| 30 | 14 | Signaled System Error. | This bit is set whenever the device asserts $\overline{SERR}$. |
| 29 | 13 | Received Master Abort. | This bit is set by a master device whenever its transaction is terminated with master-abort, except for Special Cycle. |
| 28 | 12 | Received Target Abort. | This bit is set by a master device whenever its transaction is terminated with target-abort. |
| 27 | 11 | Signaled Target Abort. | This bit is set by a target device whenever its transaction is terminated with target-abort. |
| 26-25 | 10-9 | DEVSEL Timing. | These bits are hard-wired to '01', assuming medium address decode. |
| 24 | 8 | Data Parity Detected. | This bit implemented by this bus master. It is set when this agent asserts $\overline{PERR}$ or observed $\overline{PERR}$ asserted, AND this agent setting the bit acted as the bus master for the operation in which the error occurred, AND bit 6 of PCINT Configuration Word 1 is set. |

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 23 | 7 | Fast Back-to-Back Capable. | This bit is hard-wired to '1'. |
| 22 | 6 | UDF Supported | Defaults to zero unless set by Crisco. |
| 21 | 6 | 66 MHz Capable | Defaults to zero unless set by Crisco. |
| 20-16 | 6-0 | Reserved. | Reserved |
| 15-10 | 15-10 | Reserved. | Reserved |
| 9 | 9 | Fast Back-to-Back Enable. | This bit can be set to a value, but is ignored by internal logic. |
| 8 | 8 | $\overline{SERR}$ Enable. | If this bit is '1', the $\overline{SERR}$ driver is enabled. |
| 7 | 7 | Wait Cycle Control. | This bit is hard-wired to zero because stepping is not supported by this master. |
| 6 | 6 | Parity Error Response. | When this bit is '1', normal action is taken when a parity error is detected. When it is '0', any parity errors detected are ignored and normal operation continued. |
| 5 | 5 | VGA Palette Snoop. | This bit is not implemented. |
| 4 | 4 | Memory Write and Invalidate Enable. | This bit is not implemented. |
| 3 | 3 | Special Cycles. | This bit is set to a '0', and will not monitor Special Cycle operations. |
| 2 | 2 | Bus Master Enable. | If this bit is '1', this device will be allowed to act as a bus master. |
| 1 | 1 | Memory Space Enable. | If this bit is '1', this device will respond to memory space accesses. |
| 0 | 0 | I/O Space Enable. | If this bit is '1', this device will respond to I/O space accesses. |

### 1.3: PCINT Config Word 2

The Class Code is used to identify the generic function for this device. The Revision ID is used to identify the level of function for this device. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0008 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'02030004' |
| **Power on Reset value (Little Endian)** | X'04000302' |

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-24 | 23-16 | Upper Byte. | The upper byte of the Class Code is a base code that broadly classifies the type of function this device performs. Code chosen is: X'02' - Network controller |
| 23-16 | 15-8 | Middle Byte. | The middle byte of the Class Code is a sub-class code that identifies more specifically the function of this device. Code chosen is: X'03' - ATM controller |
| 15-8 | 7-0 | Lower Byte. | The lower byte of the Class Code identifies a specific register-level programming interface so that device independent software can interact with this device. Code chosen is: X'00' |
| 7-0 | 7-0 | Revision ID. | This is the revision level of this chip. |
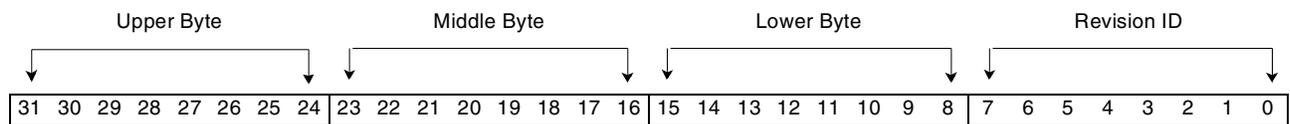
**IBM**

### 1.4: PCINT Config Word 3

This word specifies the system cache size in units of 32-bit words, the value of the Latency Timer for this PCI bus master, the Header Type which identifies the layout of bytes in configuration space, and the register for the control and status of BIST (Built-in self-test). See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 000C |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'00000000' |
| **Power on Reset value (Little Endian)** | X'00000000' |

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31 | 7 | BIST Capable. | This bit is a '1' because this device supports BIST. |
| 30 | 6 | Start BIST. | Writing this bit '1' will invoke BIST. This bit is reset after BIST is complete. This bit has two seconds to reset after a start BIST action. |
| 29-28 | 5-4 | Reserved. | Reserved |
| 27-24 | 3-0 | Completion Code. | A value of '0' means this device has passed BIST. If bit 27 is on, the PRPG value failed. If bit 26 is on, the MISR value failed. Bits 25 and 24 will always be zero. |
| 23-16 | 7-0 | Header Type (Read Only). | The encoding chosen is X '00'. |
| 15-8 | 7-0 | Latency Timer. | This register specifies a value of latency in units of PCI bus clocks. |
| 7-0 | 7-0 | Cache Line Size. | This register is used to best determine what read command should be used by this master. Any cache line size is supported. |

**1.5: PCINT Base Address 1 (I/O for regs)**

This register specifies the base address of where in PCI I/O space the IBM2520L8767 registers will be mapped. When written with ones and read back, the least significant bits read back as zero will indicate the amount of I/O space required for this device to operate. For example, when a value of 'FFFFFFFF' is written, a value read of 'FFFFFF00' indicates that 256 bytes of address space this required. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0010 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. Bit 17 in the PCINT Base Address Control Register must be set to allow the IBM2520L8767 to decode addresses for this range. |
| **Power on Reset value (Big Endian)** | X'00000001' |
| **Power on Reset value (Little Endian)** | X'01000000' |

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-2 | 31-2 | Base Address. | This register is used to hold the address where the target device will decode for I/O accesses. The size is 16K of addressing, naturally aligned. This means that only bits 31-14 are writable. The PCI specification only allows 256 bytes of I/O Base Address, so this address is only for special applications. Using the feature of non-postable writes for I/O cycles must accompany enough I/O space in the system memory map. |
| 1 | 1 | Reserved. | Reserved and set to '0'. |
| 0 | 0 | I/O Space. | This is I/O space, so this bit is set to a '1'. |

## 1.6: PCINT Base Address 2 (Mem for regs)

This register specifies the base address of where in PCI memory space the IBM2520L8767 registers will be mapped. When written with ones and read back, the least significant bits read back as zero will indicate the amount of memory space required for this device to operate. For example, when a value of 'FFFFFFFF' is written, a value read of 'FFFFFF00' indicates that 256 bytes of address space this required. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0014 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. Bit 16 in the PCINT Base Address Control Register must be set to allow the IBM2520L8767 to decode addresses for this range. |
| **Power on Reset value (Big Endian)** | X'00000000' |
| **Power on Reset value (Little Endian)** | X'00000000' |

Base Address — bits 31-4. Prefetchable — bit 3. Memory Space — bit 0.

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-4 | 31-4 | Base Address | This register is used to hold the address where the target device will decode for memory accesses. The size is 16K of addressing, naturally aligned. This means that only bits 31-14 are writable. |
| 3 | 3 | Prefetchable. | This memory space is non-prefetchable, so this bit is set to '0'. This means that there are side effects on reads. |
| 2-1 | 2-1 | | This base address can be mapped anywhere in 32 bit address space. The value of these bits is '00'. |
| 0 | 0 | Memory Space. | This is memory space, so this bit is set to '0'. |

### 1.7: PCINT Base Addresses 3-6 (Memory)

This register specifies the base address of where in PCI memory space the IBM2520L8767 memory will be mapped. When written with ones and read back, the least significant bits read back as zero will indicate the amount of memory space required for this device to operate. For example, when a value of 'FFFFFFFF' is written, a value read of 'FFFFFF00' indicates that 256 bytes of address space this required. See bit definitions.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Reg 3 | XXXX 0018 |
| | Reg 4 | XXXX 001C |
| | Reg 5 | XXXX 0020 |
| | Reg 6 | XXXX 0024 |
| **Power on Value** | X'00000008' | |

**Restrictions**  Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register.

If one of these registers is not enabled (see PCINT Base Address Control Register), then a read of that register will return all zeros. The power on value stated below assumes that the register is enabled. Normally, configuration code will just read these registers to find out what is there. To enable more that the default of registers 3 and 4, the use of Crisco code could be used. See Entity 16: *Nodal Processor Bus Interface (NPBUS)* on page 340 for details.

Base Address — bits 31 down to 4; Prefetchable — bit 3; Type — bits 2-1; Memory Space — bit 0.

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-4 | 31-4 | Base Address | This register is used to hold the address where the target device will decode for memory accesses. The size of addressing is naturally aligned and determined by what is set in the PCINT Base Address Control Register. |
| 3 | 3 | Prefetchable | This memory space is prefetchable, so this bit is set to '1'. This means that there are no side effects on reads, all bytes are returned on reads regardless of byte enables, and host bridges can merge processor writes into this range without causing errors. |
| 2-1 | 2-1 | Type | This base address can be mapped anywhere in 32-bit address space. The value of these bits is 00b. |
| 0 | 0 | Memory Space | This is memory space, so this bit is set to a '0'. |

**Note:** These registers power up to X'08000000' if accessed Little Endian.

### 1.8: PCINT CardBus CIS Pointer

This register contains the an offset to where the Card Information Structure (CIS) is located. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0028 |
| **Restrictions** | Cannot be written unless by Crisco, or the PCI configuration space override write bit is on. |
| **Power on Reset value** | X'00000000' |

Reserved

Address Space Indicator

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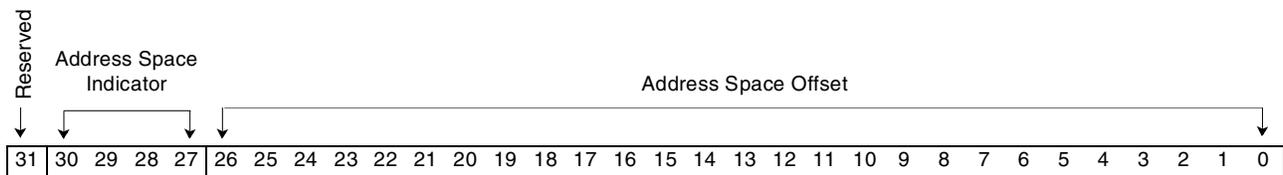
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 31 | Reserved | Reserved |
| 30-27 | Address Space Indicator | Can be set by Crisco code, likely to be in expansion ROM space. |
| 26-0 | Address Space Offset | This field has the offset into expansion ROM that is the location of the CIS. See the PCMCIA v2.10 specification for details of the CIS. |

**1.9: PCINT Subsystem ID/Vendor ID**

This register contains the Subsystem ID and Subsystem Vendor ID. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 002C |
| **Restrictions** | Cannot be written unless by Crisco, or the PCI configuration space override write bit is on. |
| **Power on Reset value (Big Endian)** | X'xxxx1014' |
| **Power on Reset Value (Little Endian)** | X'1410xxxx' |

| Bit(s) | Name | Description |
|---|---|---|
| 31 | Reserved | Reserved |
| 31-16 | Subsystem ID | Generally will be set by Crisco code. Other possible codes that could be returned for the Subsystem ID are listed in the table below. The correctness of their value is superseded by higher (IOA card) levels of documentation. |
| 15-0 | Subsystem Vendor ID | Default value is the IBM vendor ID. |

*Alternate Codes for Subsytem ID Bits in PCINT Subsystem ID/Vendor ID Register*

| Subsystem ID | Card Name | Function | Control Memory | Packet Memory |
|---|---|---|---|---|
| 00A2 | Caribou | 25 Mbs, 4/5 Token Ring, UTP-3,4,5 | 0 Meg | 2 Meg |
| 00A3 | Reindeer-D | 45 Mbs, DS3, 75 Ohm Coax | 0 Meg | 1 Meg |
| 00A4 | Reindeer-E | 34 Mbs, E3, 75 Ohm Coax | 0 Meg | 1 Meg |
| 00A5 | Okapi | 155 Mbs, SONET OC3c, STP/UTP05 | 0-16 Meg | 4-32 Meg |
| 00A6 | Gazelle | 155 Mbs, SONET OC3c, MM-Fiber | 16 Meg | 32 Meg |
| 00A8 | Moose | 155 Mbs, SONET OC3c, SM-Fiber | 16 Meg | 32 Meg |
| 00BE | Sugarpine | 622 Mbs, SONET OC12c, Fiber | 4 Meg | 4 Meg |
| 0051 | Pioneer | 622 Mbs, SONET OC12c, Fiber | 8 Meg | 8 Meg |

### 1.10: PCINT ROM Base Address

This register specifies the base address of where in PCI memory space the IBM2520L8767 ROM will be mapped. When written with ones and read back, the least significant bits read back as zero will indicate the amount of memory space required for this device to operate. For example, when a value of 'FFFFFFFF' is written, a value read of 'FFFFFF00' indicates that 256 bytes of address space is required. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0030 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value** | X'00000000' |

```
                    Base Address                              Reserved          Address Decode Enable

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

| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-10 | 31-10 | Base Address. | This register is used to hold the address where the target device will decode for expansion ROM. The size is fixed at 1K of addressing, naturally aligned. |
| 9-1 | 9-1 | Reserved. | Reserved and set to a '0'. |
| 0 | 0 | Address Decode Enable. | This bit set to '1' will enable accesses to expansion ROM only if Memory Space Enable bit (bit one in PCINT Configuration Word 1) is also set. |

### 1.11: PCINT Config Word 15

This register is used to communicate interrupt line routing information, tells which interrupt pin this device uses, and specifies the desired setting for Latency Timer values. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 003C |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'00010100' |
| **Power on Reset value (Little Endian)** | X'00010100' |

```
   Max_Lat (Read Only)        Min_Gnt (Read Only)      Interrupt Pin (Read Only)        Interrupt Line

 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9  8 | 7  6  5  4  3  2  1  0
```
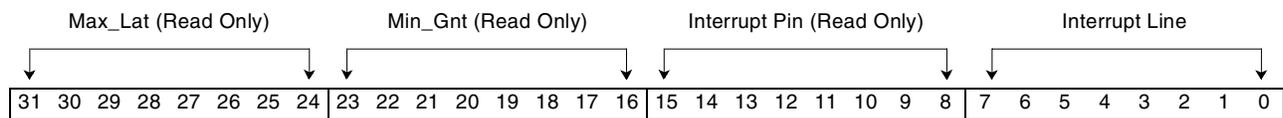
| Bit(s) | PCI Spec | Name | Description |
|---|---|---|---|
| 31-24 | 7-0 | Max_Lat (Read Only). | This value specifies a period of time in units of 1/4 microsecond. Max_Lat is used for specifying how often this device needs to gain access to the PCI bus. |
| 23-16 | 7-0 | Min_Gnt(Read Only). | This value specifies a period of time in units of 1/4 microsecond. Min_Gnt is used for specifying how long a burst period this device needs, assuming a 33-MHz clock rate. |
| 15-8 | 7-0 | Interrupt Pin (Read Only). | This device used $\overline{INTA}$ for its PCI bus interrupt. Value of this field is 01h. |
| 7-0 | 7-0 | Interrupt Line. | Software will write the routing information into this register as it initializes and configures the system. |

## 1.12: PCINT Endian Control Register

This register allows control and status to the Big/Little Endian per address selection. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0058 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. |
| **Power on Reset value** | X'00000000' |

| Bit(s) | Name | Description |
|---|---|---|
| 31-29 | Reserved. | Reserved |
| 28-24 | | Same as the definitions for bits 4-0. |
| 23-5 | Reserved. | Reserved |
| 4 | Byte Swap for Expansion ROM (on-card flash). | When this bit is set to '1', the bytes of an internal Expansion ROM access (Big Endian View) will be swapped to and from the PCI interface. |
| 3 | Reserved | Reserved |
| 2 | Byte Swap for Configuration Registers. | When this bit is set to '1', the bytes of an internal Configuration register access (Big Endian View) will be swapped to and from the PCI interface. |
| 1 | Byte Swap for Register Accesses (Memory or I/O Space). | When this bit is set to '1', the bytes of an internal register access (Big Endian View) will be swapped to and from the PCI interface. |
| 0 | Byte Swap for Memory. | When this bit is set to '1', the bytes of an internal Packet Memory access (Big Endian View) will be swapped to and from the PCI interface. |

## 1.13: PCINT Base Address Control Register

This register controls all the base address registers that map to memory. See bit definitions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 005C |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset value (Big Endian)** | X'0001000F' |
| **Power on Reset value (Little Endian)** | X'0F000100' |



| Bit(s) | Function | Description |
|---|---|---|
| 31 | Disable devto errors to GPDMA | Setting this bit to a '1' will disable device time-out errors from stopping a GPDMA transfer. |
| 30 | Disable master detected target abort errors to GPDMA | Setting this bit to a '1' will disable master detected target abort errors from stopping a GPDMA transfer. |
| 29 | Disable master detected perr errors to GPDMA | Setting this bit to a one will disable master detected parity errors from stopping a GPDMA transfer. |
| 28 | Disable slave target abort errors from PCINT | Setting this bit to a '1' will disable target abort errors to the requesting PCI master. |
| 27 | Disable master detected parity errors to GPDMA | Setting this bit to a '1' will disable master detected parity errors from stopping a GPDMA transfer. |
| 26 | Disable slave memory incremental retry cycles | Setting this bit to a '1' will disable slave memory retry attempts and will wait until the data transfer has completed. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 25-24 | Encoded Control for DMA reads | Encoding of bits:<br>X'0':  Let the IBM2520L8767 pick the best memory read command based on the cacheline size bits and the DMA count.<br>X'1':  Fix the read DMA command to Memory Read Multiple<br>X'2':  Fix the read DMA command to Memory Read Line<br>X'3':  Fix the read DMA command to Memory Read. |
| 23 | Allow decoding for zero Base Address values. | Setting this bit to a '1' will enable decoding of a BAR address that is set to zero. Normally, the PCI specification does not allow for a zero address to be a valid decode. |
| 22 | Enforce sequential PCI register writes. | Setting this bit to a '1' will make sure that PCI register writes will occur in sequential order of prior memory accesses or register reads. The cost for doing this is possible extra retry cycles for accesses not dependent on other posted accesses to complete. |
| 21 | Enforce sequential PCI register reads. | Setting this bit to a '1' will make sure that PCI register reads will occur in sequential order of prior memory accesses or register writes. The cost for doing this is possible extra retry cycles for accesses not dependent on other posted accesses to complete. |
| 20 | Disable retrying on the 1st cycle of a memory access. | Setting this bit to a '1' will disable the retrying of a memory access to the IBM2520L8767. This will cause a PCI spec violation, but not a data integrity problem. It will solve the rare case where two masters are accessing control memory at the same time and retries happen to both endlessly. |
| 19 | Enable writing to special config registers. | Setting this bit to a '1' will enable writing to certain registers that are normally read-only. An example of this would be the vendor and function ID register (PCINT Configuration Word 0). |
| 18 | Disable Incremental Latency time-out retries | Setting this bit to a '1' will disable PCI retries due to cycles taking more than eight cycles on burst accesses after the first access. |
| 17 | Enable PCINT Base Address 1 (I/O for regs). | Setting this bit to a '1' will enable PCINT Base Address 1 (I/O for registers). This does the same function as bit zero in the PCINT Configuration Word 1 register, but also make the PCINT Base Address 1 (I/O for regs) read back zeros even when written to with values. It guards against anything that BIOS code may do to PCINT Configuration Word 1 register bit zero if I/O accesses are not desired. |
| 16 | Enable PCINT Base Address 2 (Mem for regs). | This bit set will enable PCINT Base Address 2 (Mem for regs) such that the IBM2520L8767 registers can be accessed by PCI memory cycles. |
| 15-12 | Encoded Control for PCINT Base Address 6 (Memory). | Encoding of bits:<br>X'0':  Disable this Base Address. |
| 11-8 | Encoded Control for PCINT Base Address 5 (Memory). | X'1':  Configured to respond to a 2GB address size.<br>X'2':  Configured to respond to a 1GB address size. |
| 7-4 | Encoded Control for PCINT Base Address 4 (Memory). | X'3':  Configured to respond to a 512MB address size.<br>X'4':  Configured to respond to a 256MB address size.<br>X'5':  Configured to respond to a 128MB address size. |
| 3-0 | Encoded Control for PCINT Base Address 3 (Memory). | X'6':  Configured to respond to a 64MB address size.<br>X'7':  Configured to respond to a 32MB address size.<br>X'8':  Configured to respond to a 16MB address size.<br>X'9':  Configured to respond to a 8MB address size.<br>X'A':  Configured to respond to a 4MB address size.<br>X'B':  Configured to respond to a 2MB address size.<br>X'C':  Configured to respond to a 1MB address size.<br>X'D':  Configured to respond to a 64KB address size, and internal windowing of memory is enabled.<br>X'E':  Configured to respond to a 32KB address size, and internal windowing of memory is enabled.<br>X'F':  Configured to respond to a 16KB address size, and internal windowing of memory is enabled. |

### 1.14: PCINT Window Offsets for Base Addresses 3-6

These registers specify the amount of memory space required for this device to operate. See bit definitions.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Reg 3 | XXXX 0060 |
| | Reg 4 | XXXX 0064 |
| | Reg 5 | XXXX 0068 |
| | Reg 6 | XXXX 006C |
| **Power on Value** | X'00000000' | |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. | |

Windowing Offset Range — bits 31-14

Reserved — bits 13-0

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

| Bit(s) | Function | Description |
|---|---|---|
| 31-14 | Windowing Offset Range. | This register is used to hold the address offset, which is added to the PCI address (when windowing is enabled) to form the internal memory address. Bits 15 and 14 may or may not be used, depending on how bits are set in the PCINT Base Address Control Register. <br><br> When bit 20 of PCINT Count Time-out Register is set, Window Offset register three can be updated with the address returned from a good get buffer from POOLS. This will save a write from code to this register. <br><br> When bit 20 of PCINT Count Timeout Register is set, Window Offset register four can be updated with the address returned from a dequeue from the receive queue. This will save a write from code to this register. |
| 13-0 | Reserved. | Reserved |

**1.15: PCINT Count Timeout Register**

This register holds the count limit of PCI slave retry cycles. See bit definitions.

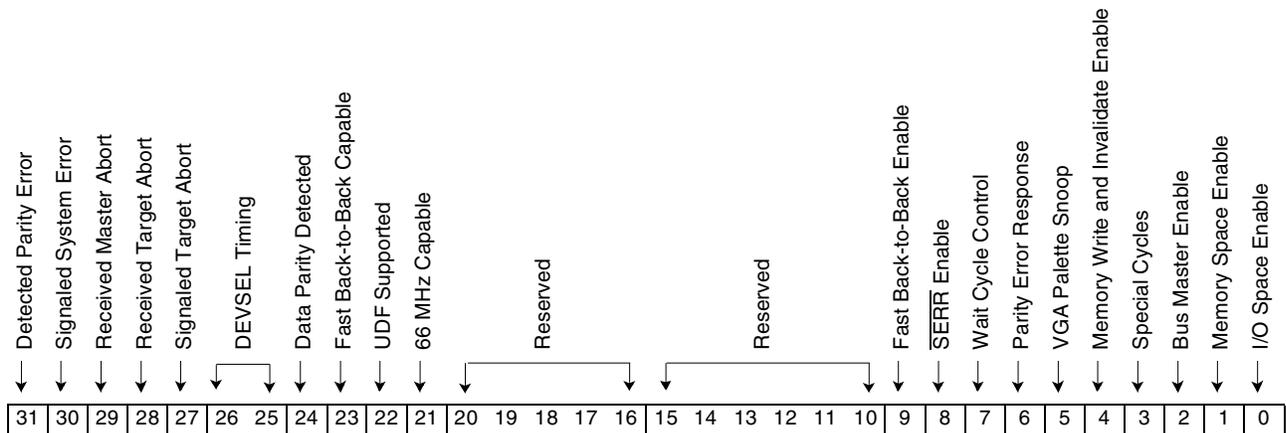| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0070 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see *PCINT Base Address Control Register* on page 85), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset Value (Big Endian)** | X'0200FFFF' |
| **Power on Reset Value (Little Endian)** | X'FFFF0002' |



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

Bit labels (left to right): Reserved; Register read retry timeout value; Reserved; Enable Dynamic Window Offset Updates; Disable Register Retry Accesses; Disable PCI Locking Function; Disable Slave Machine; Crisco retry active; Slave Transaction Timeout; Retry Timeout Count

| Bit(s) | Function | Description |
|---|---|---|
| 31-27 | Reserved | Reserved |
| 26-24 | Register read retry timeout value | The bits can be set to determine how many PCI cycles a register access will wait for an internal cycle to complete for a read access. It can be programmed to wait for up to seven cycles. A value of zero will not timeout this access with a retry. |
| 23-21 | Reserved | Reserved |
| 20 | Enable Dynamic Window Offset Updates | This bit will enable the values of PCINT Window Offsets for Base Addresses 3-6 so that it updated with a good get primitive or certain receive queue dequeues. |
| 19 | Disable Register Retry Accesses | This bit will disable PCI retry signaling during a register or primitive access. |
| 18 | Disable PCI Locking Function | This bit will disable this PCI locking function when set to '1' |
| 17 | Disable Slave Machine | This bit is for Crisco code use. It will disable all responses to the PCI bus in slave mode. In general, never turn this bit on. Bit 19 of the PCINT Base Address Control Register must be set before this bit can be changed. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 16 | Crisco retry active | This bit is for Crisco code use. It powers up to a '1' at reset. When this bit is set, a retry will be signalled to all config accesses. It can be reset to a '0' either by a register write by Crisco, or by a reset pulse signalled from NPBUS when Crisco execution is complete. A Crisco write allows quicker access to configure space, even if Crisco is not done writing to other parts of the chip. When this bit is on, the counter related to bits 7-0 of the register does not increment since the retry count could be excessive in this case. |
| 15-8 | Slave Transaction Timeout | These bits hold a value that is used to count the number of PCI clocks times 256 when a PCI slave cycle is in progress. If the count is reached, due to some internal chip hang condition, a target abort is issued. A value of '0' will disable target aborts from this function |
| 7-0 | Retry Timeout Count | These bits hold a value that is used to count the number of PCI retries. The max count is 256 times 16 retries. If the count is reached, a target abort is issued. A value of '0' will disable target aborts from this function. |

**1.16: PCINT 64bit Control Register**

This register contains miscellaneous control bits.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0078 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see PCINT Base Address Control Register), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset Value (Big Endian)** | X'00000002' |
| **Power on Reset Value (Little Endian)** | X'02000000' |



| Bit(s) | Function | Description |
|---|---|---|
| 2 | Swap Word mode | This bit set to a '1' will enable word swapping of the each of the four groups of data bytes in an 8-byte transfer. |
| 1 | Disable 64bit data phase parity checking. | This bit set to a '1' will disable the data phase parity checking on bits 32 to 63 of the AD PCI bus. |
| 0 | $\overline{REQ64}$ indicates that a 64 bit PCI bus is connected | This bit will set when the $\overline{REQ64}$ I/O pin was low bus when $\overline{RST}$ went inactive. When set to a '0', the enstate lines will actively be driving the PCI pins AD(63 - 32). |
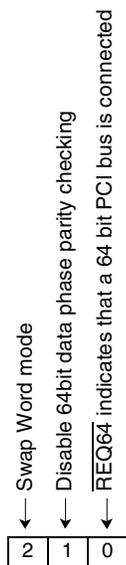
### 1.17: PCINT Perf Counters Control Register

This register contains control bits for the PCINT performance Counter 1 and PCINT Performance Counter 2.

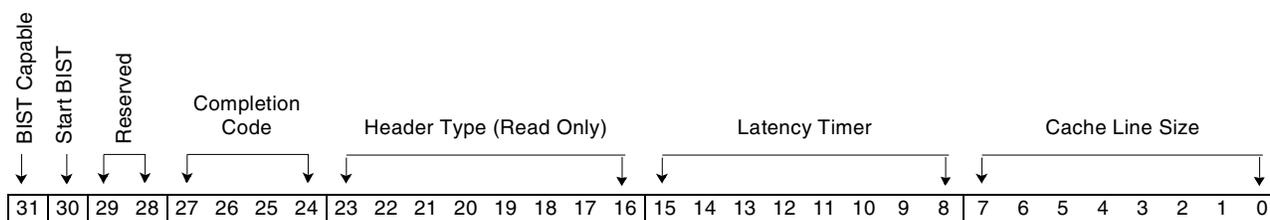| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 007C |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see PCINT Base Address Control Register), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset Value (Big Endian)** | X'00000000' |
| **Power on Reset Value (Little Endian)** | X'00000000' |

| Bit(s) | Function | Description |
|---|---|---|
| 31-18 | Reserved | Reserved. |
| 17-16 | Counter Modes | These bits will determine which kind of mode both counters will operate in.<br>X'0'  Stop on overflow<br>X'1'  Interrupt on wrap<br>X'2'  Event on wrap<br>X'3'  Reserved |
| 15-12 | Master/Slave Types - Counter 2 | These bits will determine which kind of PCI cycle owners to be counted for counter 2. The defines are the same as bits 7-4. |
| 11-8 | Cycles Types - Counter 2 | These bits will determine what kind of PCI events are to be counted for counter 2. The defines are the same as bits 3-0. |
| 7-4 | Master/Slave Types - Counter 1 | These bits will determine which kind of PCI cycle owners to be counted for counter 1.<br>X'0'  All Devices on the PCI bus<br>X'1'  All Devices but the IBM2520L8767<br>X'2'  A only (master or slave)<br>X'3'  IBM2520L8767 master<br>X'4'  IBM2520L8767 slave (all types)<br>X'5'  IBM2520L8767 slave register accesses<br>X'6'  IBM2520L8767 slave memory accesses |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 3-0 | Cycles Types - Counter 1 | These bits will determine what kind of PCI events are to be counted for counter 1.<br>X'0'  Off<br>X'1'  All PCI clock cycles<br>X'2'  Active PCI bus cycles (frame + irdy + trdy)<br>X'3'  PCI Data Xfer Opportunities ((irdy + trdy) & devsel)<br>X'4'  PCI Data Xfers (irdy & trdy)<br>X'5'  PCI Retries (irdy & no trdy & devsel & stop)<br>X'6'  PCI Target Aborts (irdy & no trdy & no devsel & stop)<br>X'7'  PCI Disconnects (irdy & trdy & devsel & stop) |

### 1.18: PCINT Perf Counter 1

This register contains PCI performance counter 1.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0080 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see PCINT Base Address Control Register), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset Value (Big Endian)** | X'xxxxxxxx' (Indeterminate) |
| **Power on Reset Value (Little Endian)** | X'xxxxxxxx' (Indeterminate) |

Counter 1

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

| Bit(s) | Name | Description |
|---|---|---|
| 31-0 | Counter 1 | See PCINT Performance Counters Control Register on how this counter will increment. |

**1.19: PCINT Perf Counter 2**

This register contains PCI performance counter 2.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0084 |
| **Restrictions** | Can be written or read during configuration cycle, memory cycle when enabled (see PCINT Base Address Control Register), or an I/O cycle. This register is documented as Big Endian, but how data is presented on the PCI bus depends on how the controls are set in the PCINT Endian Control Register. |
| **Power on Reset Value (Big Endian)** | X'xxxxxxxx' (Indeterminate) |
| **Power on Reset Value (Little Endian)** | X'xxxxxxxx' (Indeterminate) |

Counter 2

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

| Bit(s) | Name | Description |
|---|---|---|
| 31-0 | Counter 2 | See PCINT Performance Counters Control Register on how this counter will increment. |

## Entity 2: Interrupt and Status/Control (INTST)

This entity contains the masking registers that choose which interrupt/status source will be gated onto one of the two available interrupt I/O pins. A new delayed interrupt function has been added. This function allows IBM2520L8767 status registers to be read and placed in system memory before the interrupt signal is raised. For details, see the DMAQS entity.

A bus timer function is provided in this entity that times a single bus access to make sure that the cycle is terminated before the system timer times out. This allows the user code an opportunity to recover from the error as opposed to the subsystem common code.

Below is a summary of this entity's functions:
- Interrupt Prioritized Status Registers
- Interrupt Source Register
- Interrupt Enable Registers
- Bus timer function
- Control Processor error register with enable register

### 2.1: INTST Interrupt 1 Prioritized Status

Used to help quickly parse which interrupting entity of the IBM2520L8767 is active.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0400 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

Prioritized Status

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

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 31-0 | Prioritized Status. | Reading this register will give a prioritized value of this bits in the INTST Interrupt Source and INTST Enable for Interrupt 1 (MINTA) registers ANDd together, returning a value that will be a hex number equal to bit number n + 1. For example, if bit 31 is on, X'20' will be read back. |

## 2.2: INTST Interrupt 2 Prioritized Status

Used to help quickly parse which interrupting entity of the IBM2520L8767 is active.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0404 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

Prioritized Status

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

| Bit(s) | Function | Description |
|---|---|---|
| 31-0 | Prioritized Status. | Reading this register will give a prioritized value of these bits in the INTST Interrupt Source and INTST Enable for Interrupt 2 (MINT2) registers AND'd together, returning a value that will be a hex number equal to bit number n + 1. For example, if bit 31 is on, X'20' will be read back. |

### 2.3: INTST Control Register

Used to control various IBM2520L8767 functions. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

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

Fields (left to right):
- Reserved (bits 31–11)
- CPI Bytes (bits 10–9)
- Reserved (bit 8)
- Disable the ENSTATE output pins (bit 7)
- Master Chip Enable for Receiving. (bit 6)
- Master Chip Enable for Transmitting (bit 5)
- Master Chip Enable (bit 4)
- Zeros on Data Parity (bit 3)
- Treat MINT2 as push-pull (bit 2)
- Enable the PLL output (hardware test only) (bit 1)
- Disable the ENSTATE clocks output pins (bit 0)

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0408 and 0C |
| **Restrictions** | None |
| **Power on Reset value** | X'0000202' |

| Bit(s) | Function | Description |
|---|---|---|
| 31-11 | Reserved | Reserved |
| 10-9 | CPI Bytes. | These bits are encoded to tell how many bytes long the AAL5 CPI field is. The following are the encodings:<br>00    CPI field is zero bytes long. In this case, the two bytes containing the CPI field and the AAL5 user-to-user byte are copied into the packet header. See the definition of the packet header for the locations.<br>01    CPI field is one byte long and is always zero. In this case, the one-byte AAL5 user-to-user byte is copied into the packet header.<br>10    CPI field is two bytes long and is always zero.<br>11    treated the same as '00' |
| 8 | Reserved | Reserved |
| 7 | Disable the ENSTATE output pins | When this bit is set to '0', the chip I/O ENSTATES will be driven with the output of the internally multiplexed debug states. When set to '1', these outputs will be quiet. |
| 6 | Master Chip Enable forReceiving. | When this bit is set to '1', various state machines in the receive part of the chip will be enabled. |
| 5 | Master Chip Enable for Transmitting. | When this bit is set to '1', various state machines in the transmit part of the chip will be enabled. |
| 4 | Master Chip Enable. | When this bit is set to '1', various state machines in the chip will be enabled. This must be set to '1' to transmit or receive anything. |
| 3 | Zeros on Data Parity. | When this bit is set to '1', zeros will be forced on the data bus parity line(s) during a slave read data phase or a master address phase or a master write data phase. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 2 | Treat MINT2 as push-pull | When this bit is set to '1', the chip I/O MINT2 will be driven active high as well as low, like a push-pull driver. This is for use as a specific sideband application, not as a general shared open-drain interrupt line. |
| 1 | Enable the PLL output (hardware test only) | When this bit is set to '1', the chip I/O PPLLOUT will be driven with the output of the internal PLL. When set to '0', this output will be quiet. This should be normally reset by Crisco. |
| 0 | Disable the ENSTATE clocks output pins | When this bit is set to '0', the chip I/O PINTCLK and PDBLCLK will be driven with the output of the internal clock tree. When set to '1', these outputs will be quiet. |

### 2.4: INTST Interrupt Source

This register will indicate the source(s) of the interrupt(s) pending, or used as a status register when the bits are enabled. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. It should be noted that bits in this register always reflect the state of the source register bit. Writing a value will have no effect. Reserved bits will not take on the written value. The delay of running through a latch has been removed.

**Length**              32 bits

**Type**                Read Only

**Address**             XXXX 0410 and 14

**Restrictions**        None

**Power on Reset value**  X'00000000'
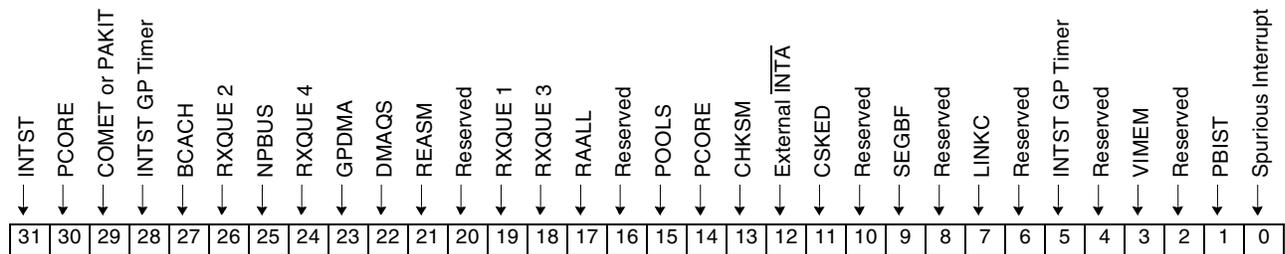


| Bit(s) | Name | Description |
|--------|------|-------------|
| 31 | INTST | A Control Processor related condition has occurred. A read of the INTST CPB Status and INTST CPB Status Enable must be done for more information. See *INTST CPB Status on page 102* and *INTST CPB Status Enable on page 104*. |
| 30 | PCORE | The PCORE entity has hardware interrupts that need handling. |
| 29 | COMET or PAKIT | The COMET or PAKIT entities have interrupts that need handling. |
| 28 | INTST GP Timer | The INTST General Purpose Timer Counter has reach the INTST General Purpose Timer Compare value and caused an interrupt. |
| 27 | BCACH | The BCACH entity has interrupts that need handling. |
| 26 | RXQUE 2 | The RXQUE entity has interrupts that need handling. |
| 25 | NPBUS | The NPBUS entity has interrupts that need handling. |
| 24 | RXQUE 4 | The RXQUE entity has interrupts that need handling. |
| 23 | GPDMA | The GPDMA entity has interrupts that need handling. |
| 22 | DMAQS | The DMAQS entity has interrupts that need handling. |
| 21 | REASM | The REASM entity has interrupts that need handling. |
| 20 | Reserved | Reserved |
| 19 | RXQUE 1 | The RXQUE entity has interrupts that need handling. |
| 18 | RXQUE 3 | The RXQUE entity has interrupts that need handling. |
| 17 | RAALL | The RAALL entity has interrupts that need handling. |
| 16 | Reserved. | Reserved. |
| 15 | POOLS | The POOLS entity has interrupts that need handling. |

| Bit(s) | Name | Description |
|---|---|---|
| 14 | PCORE | The PCORE entity has User Defined interrupts that need handling. |
| 13 | CHKSM | The CHKSM entity has interrupts that need handling. |
| 12 | External $\overline{\text{INTA}}$ | This bit will be set when the IBM2520L8767 detects that MINTA is low and, conditionally, when the same bit in INTST Enable for PCORE Normal Interrupt or INTST Enable for PCORE Critical Interrupt is set. This bit is for use by the PCORE entity, and recommended that interrupts directed out that drive this output (MINTA) are disabled. |
| 11 | CSKED | The CSKED entity has interrupts that need handling. |
| 10 | Reserved | Reserved |
| 9 | SEGBF | The SEGBF entity has interrupts that need handling. |
| 8 | Reserved | Reserved |
| 7 | LINKC | The LINKC entity has interrupts that need handling. |
| 6 | Reserved | Reserved |
| 5 | INTST GP Timer | The INTST General Purpose Timer Counter has reached the INTST General Purpose Timer Compare value and caused an interrupt. |
| 4 | Reserved | Reserved |
| 3 | VIMEM | The VIMEM entity has interrupts that need handling. |
| 2 | Reserved | Reserved |
| 1 | PBIST | This bit is set when the PBIST entity did not indicated that it was done. It also not clearable. |
| 0 | Spurious Interrupt | Under normal conditions, this bit should never be set. However, if one of the other bits in this register turn on, then off, then a spurious interrupt condition would occur. The manual vector passed to the processor would point to this bit being on. |

## 2.5: INTST Enable for Interrupt 1 (MINTA)

This register serves as an enable for interrupt 1. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST Interrupt Source* on page 99 register for the bitwise description that the corresponding bit in this register will enable.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0418 and 1C |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

**2.6: INTST Enable for Interrupt 2 (MINT2)**

This register serves as a enable for interrupt 2. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST Interrupt Source* on page 99 for the bitwise description that the corresponding bit in this register will enable.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0420 and 24 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

**2.7: INTST Interrupt Source without Enables**

Used to help quickly parse which interrupting bit of INTST Interrupt Source is active. It does not matter what state the Enable registers are set to since the value returned does not depend on them.

| | |
|---|---|
| **Length** | 32 bits |
| **Type:** | Read Only |
| **Address** | XXXX 0428 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

Prioritized Status

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

| Bit(s) | Function | Description |
|---|---|---|
| 31-0 | Prioritized Status | Reading this register will give a prioritized value of this bits in the INTST Interrupt Source, returning a value that will be a hex number equal to bit number n + 1. For example, if bit 31 is on, X'20' will be read back. |

IBM

## 2.8: INTST CPB Status

This register holds the status bits for errors on the Control Processor bus. These bits, when disabled, will set a bit in the INTST Interrupt Source register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 19 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0430 and 34 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000' |

| Bit(s) | Function | Description |
|---|---|---|
| 18 | Target Abort: Register Access Retry Timeout | This bit is set when this slave does more retry cycles than the specified amount in the PCINT Count Timeout Register during a register access. |
| 17 | Target Abort: Slave Access Timeout | This bit is set when this slave does not access the IBM2520L8767 in the specified amount in the PCINT Count Timeout Register. |
| 16 | PCI timing changed | The PCI bus clock has changed operating range enough that the IBM2520L8767 must be adjusted to run with the new bus frequency. The IBM2520L8767 should be reset and re-initialized. |
| 15 | ARBIT Detected Memory Errors | This bit is set when error conditions detected by ARBIT are enabled. Note: this bit is a reflection of the arbitrator status bits and does not need to be reset if the arbitrator condition has been reset. |
| 14 | Reserved | Reserved |
| 13 | Reserved | Reserved |
| 12 | Master PCI Parity Error Detected | This bit is set when a PCI bus data parity error is detected in master mode. |
| 11 | Master Detected Perr Active | This bit is set when a target has driven parity error. |
| 10 | Master Termination: Target Abort Received | This bit is set when in master mode and the transfer is aborted by the target. |

| Bit(s) | Function | Description |
|---|---|---|
| 9 | Master Termination: Master-initiated Abort | This bit is set when in master mode and the transfer is aborted by this master. |
| 8 | Target Detected PCI 64 Bit Data Parity Error | This bit is set when a PCI data parity error is detected in 64 bit target mode (the upper DWORD has the data parity error). |
| 7 | Target Disconnect: Memory Addressing | This is set when a memory access is occurring and bits 0 and 1 of the address are not zero. |
| 6 | Target Detected PCI Data Parity Error | This bit is set when a PCI data parity error is detected in target mode. |
| 5 | Target Abort: Memory Access Retry Timeout | This bit is set when this slave does more retry cycles than the specified amount in the PCINT Count Timeout Register during a memory access. |
| 4 | Target Abort: Address Parity Error | This bit is set when an address parity error is detected. |
| 3 | Target Disconnect or Retry: End of Memory | This bit is set when a termination condition occurs due to reaching the end of the configured memory space. |
| 2 | Performance Counter 2 | The PCINT Performance Counter 2 has overflowed. |
| 1 | Performance Counter 1 | The PCINT Performance Counter 1 has overflowed. |
| 0 | Illegal Register Access | This bit is set when an IBM2520L8767 register is being accessed by fewer than four bytes at a time. This is not true for configuration registers during a configuration cycle. |

IBM

## 2.9: INTST CPB Status Enable

This register serves as a enable for the INTST CPB Status register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST CPB Status on page 102* for the bitwise description that the corresponding bit in this register will enable. This enable will initialize to the disabled state.

| | |
|---|---|
| **Length** | 19 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0438 and 3C |
| **Restrictions** | None |
| **Power on Reset value** | X'00000' |

## 2.10: INTST IBM2520L8767 Halt Enable

This register serves as a enable for the INTST CPB Status register and will gate which errors will reset bit 4 (Master chip enable), bit 5 (Master chip enable for Transmitting), and bit 6 (Master chip enable for Receiving), all in the INTST Control Register register. This allows selected bits to disable the IBM2520L8767, especially in the case of severe hardware detected errors.

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST CPB Status on page 102* for the bitwise description that corresponding bit in this register will enable. This enable will initialize to the disabled state.

| | |
|---|---|
| **Length** | 19 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0440 and 44 |
| **Restrictions** | None |
| **Power on Reset value** | X'69F71' |

### 2.11: INTST CPB Capture Enable

This register serves as a enable for the INTST CPB Status that will determine on which error type the INTST CPB Captured Address register will be updated. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST CPB Status on page 102* for the bitwise description that corresponding bit in this register will enable.

| | |
|---|---|
| **Length** | 19 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0450 and 54 |
| **Restrictions** | None |
| **Power on Reset value** | X'7FFFF' |

### 2.12: INTST CPB Captured Address

This information can be used to attempt a retry in the exception handling microcode. This register will hold the value of the IBM2520L8767 register address on the PCI during a bus error condition. This will only latch values from sources that are enabled in the INTST CPB Capture Enable register.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0458 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000001' |

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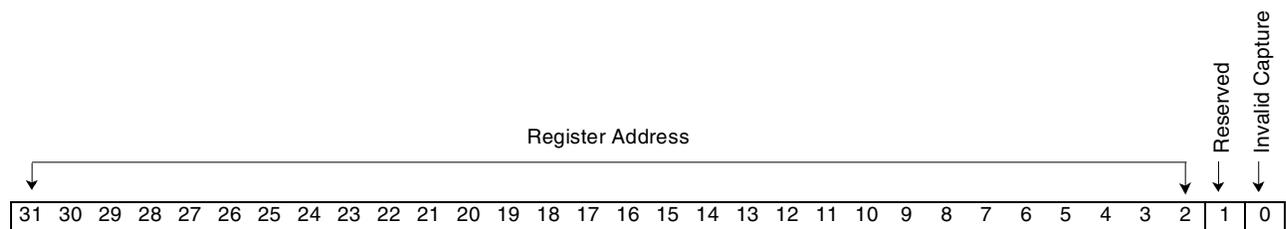
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Function | Description |
|---|---|---|
| 31-2 | Register Address | Captured IBM2520L8767 register address |
| 1 | Reserved | Reserved |
| 0 | Invalid Capture | When this bit is reset to '0', a valid capture has been made. |

### 2.13: INTST General Purpose Timer Pre-scaler

This is the pre-scaler for the INTST General Purpose Timer Compare. Owing to a physical design problem, the function of this register was lost. It should be set to a non-zero value, so that the INTST General Purpose Timer Counter can be used with a prescale of only the default clock (one tick every 30ns, assuming a 33-MHz system clock).

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0464 |
| **Restrictions** | None |
| **Power on Reset value** | X'0000014C' |

### 2.14: INTST General Purpose Timer Compare

This is the compare value for the general purpose timer.This register will hold the value of the data that is compared to the count value in the INTST General Purpose Timer Counter, setting the INTST General Purpose Timer Status bits. See *INTST General Purpose Timer Mode Control* on page 108 for details on operation of this register.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0468 |
| **Restrictions** | None |
| **Power on Reset value** | X'0800 0000' |

### 2.15: INTST General Purpose Timer Counter

This is the general purpose timer counter. This register will hold the value of the counter. It always counts up. See *INTST General Purpose Timer Mode Control* on page 108 for details on operation of this register.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 046C |
| **Restrictions** | None |
| **Power on Reset value** | X'0000 0000' |

### 2.16: INTST General Purpose Timer Status

This is the status of the general purpose timer counter.

**Length**            2 bits

**Type**             Clear/Set

**Address**          XXXX 0470 and 74

**Restrictions**      None

**Power on Reset value**  X'0'

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| Bit(s) | Function | Description |
|---|---|---|
| 1 | Timer Wrapped | This bit is set when the INTST General Purpose Timer Counter wraps around to a zero count value. |
| 0 | Timer Interrupt | See *INTST General Purpose Timer Mode Control* on page 108 for details on how this bit is set. For mode 0: This bit is set when the INTST General Purpose Timer Counter matches the value in the INTST General Purpose Timer Compare register. The comparing condition must be changed (write INTST General Purpose Timer Counter or INTST General Purpose Timer Compare) before resetting this bit, or the bit will set again. |

IBM

### 2.17: INTST General Purpose Timer Mode Control

This register controls the operating modes of the general purpose timer counter. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0478 and 7C |
| **Restrictions** | None |
| **Power on Reset value** | X'4' |

Timer Modes
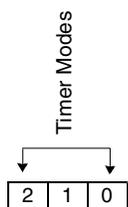
| 2 | 1 | 0 |
|---|---|---|

| Bit(s) | Function | Encodings | Description |
|---|---|---|---|
| 2-0 | Timer Modes | | These bits are encoded to provide eight different timer operation modes. |
| | | Mode 0 | The INTST General Purpose Timer Counter is a free-running up-counter and sets bit zero of INTST General Purpose Timer Status when equal to INTST General Purpose Timer Compare |
| | | Mode 1 | The INTST General Purpose Timer Counter is a free-running up-counter and sets bit 0 of INTST General Purpose Timer Status when equal to INTST General Purpose Timer Compare A write to INTST General Purpose Timer Compare will reset bit zero of INTST General Purpose Timer Status. |
| | | Mode 2 | The INTST General Purpose Timer Counter is a free-running up-counter and sets bit 0 of INTST General Purpose Timer Status when equal to or greater than INTST General Purpose Timer Compare A write to INTST General Purpose Timer Compare will reset bit zero of INTST General Purpose Timer Status. |
| | | Mode 3 | The INTST General Purpose Timer Counter is a up-counter and sets bit 0 of INTST General Purpose Timer Status when equal or greater than INTST General Purpose Timer Compare The INTST General Purpose Timer Counter is also reset when a comparison is made. A write to INTST General Purpose Timer Compare will reset bit zero of INTST General Purpose Timer Status and INTST General Purpose Timer Counter. |
| | | Mode 4 | The INTST General Purpose Timer Counter is disabled and no status bits will be set. |
| | | Modes 5-7 | Reserved |

### 2.18: INTST Enable for PCORE Normal Interrupt

This register serves as a enable for the PCORE normal interrupt input. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST Interrupt Source* on page 99 for the bit-wise description that the corresponding bit in this register will enable.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0480 and 84 |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

### 2.19: INTST Enable for PCORE Critical Interrupt

This register serves as a enable for the PCORE critical interrupt input. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the *INTST Interrupt Source* on page 99 for the bit-wise description that the corresponding bit in this register will enable.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0488 and 8C |
| **Restrictions** | None |
| **Power on Reset value** | X'00000000' |

### 2.20: INTST Debug States Control

This register serves as the control for external debug states.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0490 |
| **Restrictions** | None |
| **Power on Reset value** | X'04030201' |

Entity State Mux Control 4 (Hard-ware debug)  Entity State Mux Control 3 (Hard-ware debug)  Entity State Mux Control 2 (Hard-ware debug)  Entity State Mux Control 1 (Hard-ware debug)
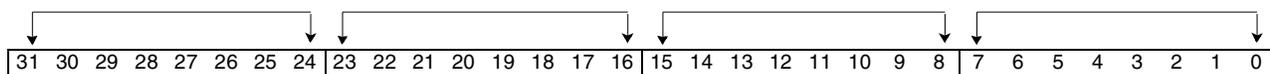
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 31-24 | Entity State Mux Control 4 (Hardware debug) | Selection of these bits allows internal state machines, counters, etc. to show up on chip outputs enstate (63-48). Selection encoding is the same as multiplexer 1 control. |
| 23-16 | Entity State Mux Control 3 (Hardware debug) | Selection of these bits allows internal state machines, counters, etc. to show up on chip outputs enstate (4 -32). Selection encoding is the same as multiplexer 1 control. |
| 15-8 | Entity State Mux Control 2 (Hardware debug) | Selection of these bits allows internal state machines, counters, etc. to show up on chip outputs enstate (31 -16) which are multiplexed over ad64 (31-16), also known as AD (63-48). Selection encoding is the same as multiplexer 1 control. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 7-0 | Entity State Mux Control 1 (Hardware debug) | Select of these bits allow internal state machines, counters, etc. to show up on chip outputs enstate (15 - 0) which are multiplexed over ad64 (15 - 0), also known as AD (47 - 32).<br><br>X'00'  Disabled (no transition on outputs). X'20'  Select POOLS 95-80 states.<br>X'01'  Select CRSET 15-0 states.     X'21'  Select POOLS 111-96 states.<br>X'02'  Select NPBUS 15-0 states.     X'22'  Select POOLS 127-112 states.<br>X'03'  Select PCINT 15-0 states.     X'23'  Select VIMEM 15-0 states.<br>X'04'  Select PCINT 31-16 states.     X'24'  Select VIMEM 31-16 states.<br>X'05'  Select COMET 15-0 states.     X'25'  Select VIMEM 47-32 states.<br>X'06'  Select COMET 31-16 states.     X'26'  Select ARBIT 15-0 states.<br>X'07'  Select ± 15-0 states.     X'27'  Select ARBIT 31-16 states.<br>X'08'  Select ± 31-16 states.     X'28'  Select ARBIT 47-32 states.<br>X'09'  Select RXQUE 15-0 states.     X'29'  Select ARBIT 63-48 states.<br>X'0A'  Select RXQUE 31-16 states.     X'2A'  Select PCORE 15-0 states.<br>X'0B'  Select RAALL 15-0 states.     X'2B'  Select PCORE 31-16 states.<br>X'0C'  Select RAALL 31-16 states.     X'2C'  Select PCORE 47-32 states.<br>X'0D'  Select RAALL 47-32 states.     X'2D'  Select PCORE 63-48 states.<br>X'0E'  Select RAALL 63-48 states.     X'2E'  Select PCORE 79-64 states.<br>X'0F'  Select REASM 15-0 states.     X'2F'  Select PCORE 95-80 states.<br>X'10'  Select LINKC 15-0 states.     X'30'  Select PCORE 111-96 states.<br>X'11'  Select SEGBF 15-0 states.     X'31'  Select PCORE 127-112 states.<br>X'12'  Select SEGBF 31-16 states.     X'32'  Select DMAQS 15-0 states.<br>X'13'  Select SEGBF 47-32 states.     X'33'  Select DMAQS 31-16 states.<br>X'14'  Select SEGBF 63-48 states.     X'34'  Select DMAQS 47-32 states.<br>X'15'  Select CSKED 15-0 states.     X'35'  Select DMAQS 63-48 states.<br>X'16'  Select CHKSM 15-0 states.     X'36'  Select SCLCK 15-0 states.<br>X'17'  Select CHKSM 31-16 states.     X'37'  Select SCLCK 31-16 states.<br>X'18'  Select GPDMA 15-0 states.     X'38'  Select SCLCK 39-32 states.<br>X'19'  Select BCACH 15-0 states.     X'39'-X'FF'  Reserved (do not toggle as well)<br><br>X'1A'  Select BCACH 31-16 states.<br>X'1B'  Select POOLS 15-0 states.<br>X'1C'  Select POOLS 31-16 states.<br>X'1D'  Select POOLS 47-32 states.<br>X'1E'  Select POOLS 63-48 states.<br>X'1F'  Select POOLS 79-64 states. |

## Entity 3: DMA QUEUES (DMAQS)

DMAQS provides the interface to the IBM2520L8767's DMA master capability (described in *General Purpose DMA (GPDMA)* ). It provides three DMA queues that hold DMA descriptor chains that are executed in a multiplexed fashion. Together with GPDMA, a very powerful interface is provided to software to complete complex tasks including TCP/IP checksumming for transmit and receive packets. The following sections describe the features of DMAQS, how to set up DMAQS, and some trouble shooting tips.

DMAQS also provides the delayed interrupt function.

### DMA Descriptors

DMA descriptors can reside in either PCI/system memory space or the IBM2520L8767 memory space. Certain types of descriptors must be located in the IBM2520L8767 memory space. These are the cut-through DMA descriptors. DMA descriptors that are located in the IBM2520L8767 memory space are more efficient to process because they do not need to be moved across the PCI bus. However, it is more costly for software to update across the bus. The best option is to mix descriptors in both locations. DMA descriptors that are infrequently changed, should reside in the IBM2520L8767 memory while dynamic descriptors should be placed in system memory. Descriptors located in the IBM2520L8767 memory space must fall in a definable address range. See DMAQS Local Descriptor Range Registers.

## DMA Descriptor Layout
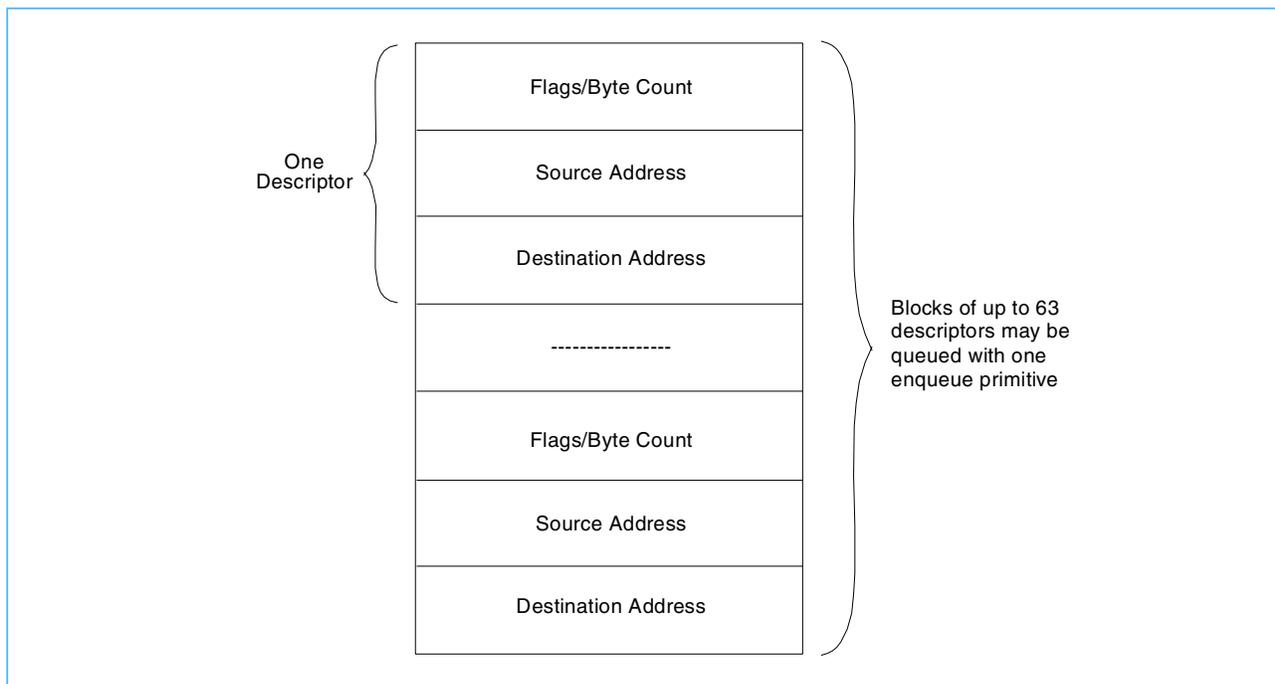
### DMA Types/Options

The DMA descriptor is very versatile, and can perform many actions. The following list shows some examples and possible flags to use (other combos are possible, see &regdmcntn.):

## DMA Types and Flags

| Hex Flags | DMA Operation |
|---|---|
| 3000 | Clear the current TCP checksum and include this DMA in the TCP checksum. |
| 1000 | Include this DMA in the TCP checksum and use previous checksum as seed. |
| 0800 | This DMA transfer is done in Little Endian mode. |
| 0400 | Upon completion of this DMA descriptor, the destination address from this descriptor is used as a packet address to be enqueued to transmit. |
| 0100 | Queue a DMA complete event when DMA is complete. |
| 0080 | Status in the status register is inhibited for this descriptor. This can be useful if ints/polling are being used to track when a particular DMA is complete. |
| 0001 | Move system memory to the IBM2520L8767 memory. |
| 0010 | Move the IBM2520L8767 memory to system memory. |
| 0012 | Move a single IBM2520L8767 register to system memory. |
| 0013 | Move IBM2520L8767 memory to system memory and free buffer. Upon DMA completion, the source address is used to free the IBM2520L8767 buffer as a get POOL ID. |
| 0017 | Auto-increment source address and move IBM2520L8767 memory to system memory and free buffer. Upon DMA completion, the source address is used to free the IBM2520L8767 buffer as a get POOL ID. |
| 0002 | Move single IBM2520L8767 register to IBM2520L8767 memory. |
| 0020 | Move IBM2520L8767 Memory to single IBM2520L8767 register. |
| 0021 | Move system Memory to single IBM2520L8767 register. |
| 0031 | Move system memory to a new IBM2520L8767 buffer. A get buffer operation will be done to fill in the destination address using the low four bits of the destination address as a get pool ID. |
| 0050 | Move something to source address of next descriptor. Allows indirection. |
| 0062 | Move single IBM2520L8767 register to destination address of next descriptor. Allows a get buffer operation in descriptor chain. (see get buff flag for a better option) |
| 0008 | Use source address as immediate data. Allows up to four bytes of immediate data in the DMA descriptor. |
| 0004 | Auto-increment the source address. The source address picks up right were it left off from the previous DMA descriptor. |
| 000C | Auto-increment the source address and use as immediate data. One use is to free a packet after DMAing data. (see free buff flags for better option) |
| 0040 | Auto-increment the destination address. The destination address picks up right were it left off from the previous DMA descriptor. One use is transmit scatter into an IBM2520L8767 virtual buffer. |
| 2200 | Hold the destination address. Useful for freeing a scatter DMA list, or doing a repetitive write to an IBM2520L8767 register. |
| 1200 | Hold the source address. Useful for doing a repetitive read from an IBM2520L8767 register. |

**Note:** These are not the only options. Some of the above can be OR'd together also.

Using the above, you can efficiently do TCP checksumming, place user events in receive queues, do register reads/writes, free buffers, and get buffers. Be creative.

## Descriptor Based DMAs

This is the recommended approach to processing DMAs. A single descriptor or a descriptor chain is built that describes the actions to take. The descriptor is then enqueued to the proper DMA queue. The number of the descriptor in the DMA chain is placed in the lower six bits of the descriptor address as it is enqueued.

## Register Based DMAs

While register based DMAs can be enabled and used, they are not recommended because they are not as efficient and they do not leave a debug trail as the descriptors do in the DMA queue. These should not be used concurrently with descriptor-based DMAs for a particular queue, but register-based and descriptor-based DMAs can be used on different queues. One possible use for register-based DMAs is doing DMAs from the core.

## Polling, Interrupts, or Events

There are several choices for handling DMA completion. First, the status register can be polled. While not very efficient, it is the easiest option. Second, you can use interrupts to tell when a DMA is done. Again, not very efficient. However, interrupts should be used to tell when a DMA error has happened.

One way to deal with DMA completes is the use the RXQUE event mechanism. By generating events, the user can dump in DMA descriptor and clean up at a later time when it is convenient. The user can use the automatic DMA events using the queue on DMA complete flag, or the user can place a user event on an arbitrary queue by writing a DMA descriptor that does an explicit RXQUE enqueue with user data.

## Error Detection and Recovery

Ideally, there should not be any errors. Errors are usually user-errors in the DMA descriptor which need to be fixed and are not recoverable. Errors on the PCI bus (i.e. parity) should not be happening in a normal working DMA and must be recovered in GPDMA. Upon successful completion of the recovered DMA, DMAQS will resume operation.

## DMA/Queue Scheduling Options

There are three DMA queues. Queue zero is higher priority then the other two. This high priority queue is always scheduled to go if the current descriptor is ready. The other two queues (Q1/Q2) are of equal priority and are scheduled in a round robin fashion when the descriptor is ready. This was meant to provide a transmit DMA queue, and receive DMA queue, and a high priority DMA queue. However, these queues can be used for any purpose by setting the routing registers properly.

Arbitration of Queues: The queues can be arbitrated after each DMA request length operation, after complete DMA descriptor chains complete, after single DMA descriptor in a chain completes. The queues can also be placed in true round robin mode, where all three queues have equal priority. No matter how the queues are arbitrated, the delayed interrupt transfer and DMA descriptor transfer (from system to queue) are always highest priority and are arbitrated after every DMA request length operation.

**Initialization of DMAQS**

DMAQS is very simple to set up. The following steps should be followed to set up DMAQS:

1. Set up each of the three DMA queues.

   To do this, you need to know the size of each queue (see *DMAQS Upper Bound Registers* for choices). Given this information, the DMA queue is set up with two register writes in diagnostic mode (see DMAQS Control Register).

   ```
   dmaqs->lowerBound[q] = baseAddress
     // should be aligned with size of queue
   dmaqs->upperBound[q] = encodedSize;    // set encoded size of dma queue
   ```

   The data structure for the DMA queue is now set up.

2. Set up the queue thresholds if they are being used:

   ```
   dmaqs->threshold[q] = threshold
         // set threshold size to be interrupted on
   // may also need to set int mask
   ```

3. Set up the local DMA descriptor range if local descriptors are being used:

   ```
   dmaqs->localDescLowerBound = localDescBase
    // set base addr of local desc in charm memory
   dmaqs->localDescUpperBound = localDescEnd;  // set ending addr of local desc in charm memo
   ry
   ```

4. Set up any options that are being used in the DMAQS Control Register:

   ```
   dmaqs->control[set] = ENABLE_DMA_QUEUES ö CLR_CHECKSUM_TO_FOXES
   // set options/modes
   ```

5. Finally, clear the diagnostic bit:

   ```
   dmaqs->control[clr] = DIAG_MODE
   // clear the diag mode bit
   ```

6. Need to set up memory bank selection if necessary, but normally control memory is used.

**Delayed Interrupts**

When enabled, an IBM2520L8767 register can be moved to system memory before the interrupt is raised to the system. The register and destination address are specified in the DMAQS Delayed Interrupt Source and Destination Registers. This allows the RXQUE status register or the INTST status register to be read into system memory before the interrupt is raised, thus removing PCI bus latency from your interrupt handler. Another option is to use both sets of interrupt masks, but use only a single hardware interrupt. When this is done, both delayed interrupt sources are read for interrupt two before the interrupt is raised. This allows the user to setup mask 2 for errors only and mask 1 for normal mainline interrupts only.

The RXQUE enabled status registers should also be considered as they show only the status that you are interested in.

The DMA transfer to move the registers is the highest priority in the DMA scheduling mechanism. However, you still may need to tune your DMA scheduling so these interrupts are not delayed behind a 64-K transfer.

**3.1: DMAQS Lower Bound Registers**

These registers specify the lower bound of the corresponding DMA queue data structure. These registers specify the lower bound of the corresponding DMA queue data structure. The head, tail, and length of the DMA queue are initialized when this register is written. When the DMA queue wraps past the upper bound, it wraps back to the value in the lower bound register, thus implementing the DMA queue as a circular buffer.

When this register is written, the corresponding DMA queue is essentially reset. This is because the head, tail, and length of the queue are all reset.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0600 |
| | Queue 1 | XXXX 0640 |
| | Queue 2 | XXXX 0680 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the DMAQS Control Register. | |
| | The alignment should correspond to the size specified in the upper bound register. For example, it should be 4-K aligned if the upper bound specifies 4-K size. | |
| | The low order nine bits are not writable and read back zero. | |

### 3.2: DMAQS Upper Bound Registers

These registers specify the encoded size/upper bound of the corresponding DMA queue data structure. The actual upper bound is calculated by adding the decoded queue size to the lower bound. When the DMA queue wraps past the upper bound, it wraps back to the lower bound register, thus implementing the DMA queue as a circular buffer.

| | | |
|---|---|---|
| **Length** | 3 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0604 |
| | Queue 1 | XXXX 0644 |
| | Queue 2 | XXXX 0684 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the DMAQS Control Register. | |

| Bit(s) | Description |
|---|---|
| 2-0 | 000   512 bytes of memory<br>001   1K of memory<br>010   2K of memory<br>011   4K of memory<br>100   8K of memory<br>101   16K of memory<br>110   32K of memory<br>111   64K of memory |

### 3.3: DMAQS Head Pointer Registers

These registers point to the head element of the corresponding DMA queue. During normal operations, these registers do not need to be read or written; they are used by the IBM2520L8767 to implement the DMA queues. These registers are initialized when the DMAQS Lower Bound Registers for the corresponding DMA queue is written. The head pointer registers are 4-byte aligned. (low order two bits always zero). Bits 31-17 are calculated internally

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0608 |
| | Queue 1 | XXXX 0648 |
| | Queue 2 | XXXX 0688 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations, these registers are read only. The registers can only be written when the diagnostic bit has been set in the DMAQS Control Register. | |

### 3.4: DMAQS Tail Pointer Registers

These registers point to the next free element of the corresponding DMA queue.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 060C |
| | Queue 1 | XXXX 064C |
| | Queue 2 | XXXX 068C |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations, these registers are read only. The registers can only be written when the diagnostic bit has been set in the DMAQS Control Register. | |

### 3.5: DMAQS Length Registers

These registers specify the length in bytes of the corresponding DMA queue. These registers specify the length in bytes of the corresponding DMA queue. This register is cleared when the corresponding DMAQS Lower Bound Registers is written.

| | | |
|---|---|---|
| **Length** | 17 bits | |
| **Type** | Read | |
| **Address** | Queue 0 | XXXX 0610 |
| | Queue 1 | XXXX 0650 |
| | Queue 2 | XXXX 0690 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | The lengths are calculated and cannot be written. | |

### 3.6: DMAQS Threshold Registers

These registers specify a queue length threshold at which the corresponding status bit is generated. These registers should be set equal to the queue length that should cause status to be generated. For example, if the value was set to five, then no interrupt would be generated until the queue was length five or more for the corresponding DMA queue. The threshold is level sensitive, so as long as the length is greater than or equal to the threshold, the corresponding status bit is set. When this register is set to zero, no thresholding is done.

| | | |
|---|---|---|
| **Length** | 17 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0614 |
| | Queue 1 | XXXX 0654 |
| | Queue 2 | XXXX 0694 |
| **Power on Value** | X'0000' | |
| **Restrictions** | Must be a multiple of 4. | |

IBM

### 3.7: DMAQS Interrupt Status

This register indicates the source(s) of the interrupt(s) pending. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 25 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0708 and 70C |
| **Restrictions** | None |
| **Power on Value** | X'00009200' |

Bit field (bits 24 down to 0):

| Bit | Label |
|---|---|
| 24 | Zero Address in Src/Dst |
| 23 | Enq Fifo Thresh Exceeded |
| 22 | Error During Delayed Int Transfer |
| 21 | Enq Fifo Full |
| 20 | Zero Length DMA Loaded |
| 19 | Error Enqueuing CSKED Descriptor |
| 18 | Error Enqueuing RAALL Descriptor |
| 17 | Error Enqueuing Descriptor |
| 16 | DMA Descriptor Error |
| 15 | DMA Descriptor Queue 2 Not Full |
| 14 | DMA Descriptor Queue 2 Threshold Exceeded |
| 13 | DMA Descriptor Queue 2 Full |
| 12 | DMA Descriptor Queue 1 Not Full |
| 11 | DMA Descriptor Queue 1 Threshold Exceeded |
| 10 | DMA Descriptor Queue 1 Full |
| 9 | DMA Descriptor Queue 0 Not Full |
| 8 | DMA Descriptor Queue 0 Threshold Exceeded |
| 7 | DMA Descriptor Queue 0 Full |
| 6 | Error Occurred During Descriptor Transfer |
| 5 | Error Occurred During DMA Transfer Q2 |
| 4 | Error Occurred During DMA Transfer Q1 |
| 3 | Error Occurred During DMA Transfer Q0 |
| 2 | DMA Transfer Complete Q2 |
| 1 | DMA Transfer Complete Q1 |
| 0 | DMA Transfer Complete Q0 |

| Bit(s) | Function | Description |
|---|---|---|
| 24 | Zero Address in Src/Dst | When set, a zero address was detected in the source or destination field of a DMA descriptor. The remainder of the descriptor chain was skipped and an event was enqueued to the DMA complete queue. This may or may not be an error condition. It is not an error if the get buffer mode is being used and no buffer was available. In this case, the descriptor can be retried or discarded by software. |
| 23 | Enq Fifo Thresh Exceeded | When set, the DMA enqueue FIFO length threshold has been exceeded. |
| 22 | Error During Delayed Int Transfer | When set, an error occurred while doing a delayed interrupt transfer. Generally this is a user error in the delayed interrupt setup. When this happens, the delayed interrupt enables are reset so interrupts flow to the system. |
| 21 | Enq Fifo Full | When set, the DMA enqueue FIFO is full and further enqueues will be held off. This bit is hot and cannot be reset. |
| 20 | Zero Length DMA Loaded | A descriptor was loaded that had a DMA length equal to zero. This will not stop the DMA engine, but it is technically a user error. |
| 19 | Error Enqueuing CSKED Descriptor. | A descriptor was enqueued from CSKED with a chain length of zero. |
| 18 | Error Enqueuing RAALL Descriptor. | A descriptor was enqueued from RAALL with a chain length of zero. |
| 17 | Error Enqueuing Descriptor. | A descriptor was enqueued with a chain length of zero. |
| 16 | DMA Descriptor Error. | An invalid transfer was described by the value loaded into the Transfer Count and Flag register. |

| Bit(s) | Function | Description |
|---|---|---|
| 15 | DMA Descriptor Queue 2 Not Full. | The DMA descriptor queue 2 is not full. This bit will always contain the status of the queue and is therefore is not writable. |
| 14 | DMA Descriptor Queue 2 Threshold Exceeded. | The threshold for DMA descriptor queue 2 has been exceeded. |
| 13 | DMA Descriptor Queue 2 Full. | The DMA descriptor queue 2 is full. This bit will always contain the status of the queue and is therefore is not writable. |
| 12 | DMA Descriptor Queue 1 Not Full. | The DMA descriptor queue 1 is not full. This bit will always contain the status of the queue and is therefore is not writable. |
| 11 | DMA Descriptor Queue 1 Threshold Exceeded. | The threshold for DMA descriptor queue 1 has been exceeded. |
| 10 | DMA Descriptor Queue 1 Full. | The DMA descriptor queue 1 is full. This bit will always contain the status of the queue and is therefore is not writable. |
| 9 | DMA Descriptor Queue 0 Not Full. | The DMA descriptor queue 0 is not full. This bit will always contain the status of the queue and is therefore is not writable. |
| 8 | DMA Descriptor Queue 0 Threshold Exceeded. | The threshold for DMA descriptor queue 0 has been exceeded. |
| 7 | DMA Descriptor Queue 0 Full. | The DMA descriptor queue 0 is full. This bit will always contain the status of the queue and is therefore is not writable. |
| 6 | Error Occurred During Descriptor Transfer. | Hardware errors occurred transferring the DMA descriptor. The transfer stopped after detecting the error. If the descriptor transfer is finished or is to be terminated, the byte count register must be written to clean up the failed descriptor transfer. Before this bit is reset, the DMA descriptor queue must contain the valid descriptor data or the &regdmt-dqcn. must be written to the value it contained prior to the descriptor enqueue. |
| 5 | Error Occurred During DMA Transfer Q2. | Hardware errors occurred during the last transfer on queue 2. The transfer stopped after detecting the error. Inspect GPDMA registers for actual location of error. |
| 4 | Error Occurred During DMA Transfer Q1. | Hardware errors occurred during the last transfer on queue 1. The transfer stopped after detecting the error. Inspect GPDMA registers for actual location of error. |
| 3 | Error Occurred During DMA Transfer Q0. | Hardware errors occurred during the last transfer on queue 0. The transfer stopped after detecting the error. Inspect GPDMA registers for actual location of error. |
| 2 | DMA Transfer Complete Q2. | The DMA transfer has completed for queue 2. |
| 1 | DMA Transfer Complete Q1. | The DMA transfer has completed for queue 1. |
| 0 | DMA Transfer Complete Q0. | The DMA transfer has completed for queue 0. |

### 3.8: DMAQS Interrupt Enable

This register serves as a mask for DMAQS Interrupt Status. The structure of this register is identical to the *DMAQS Interrupt Status* on page 120. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 25 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0700 and 704 |
| **Restrictions** | None |
| **Power on Value** | X'006f0078' |

### 3.9: DMAQS Control Register

Used to set options for DMAQS. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0710 and 714 |
| **Restrictions** | See bit descriptions. |
| **Power on Value** | X'000C0001' |



| Bit(s)0 | Function | Description |
|---|---|---|
| 31 | Reset FIFO | When this bit is set, the internal DMA enqueue FIFO is flushed, and this bit is reset. The result is this bit will always be read as a zero. This bit can only be set in diagnostic mode. |
| 30 | Restart DMA | When this bit is set, the internal DMA state machine will restart the current DMA that is stopped, and this bit is reset. The result is this bit will always be read as a zero. This bit should only be used if you really know what you are doing (translation: an IBM2520L8767 developer told you to use it!). |

| Bit(s)0 | Function | Description |
|---|---|---|
| 29-23 | Reserved | Reserved. |
| 22 | Disable desc snooping | When set, the DMA descriptor snooping logic is disabled. When enabled, performance may be enhanced. |
| 21 | Disable desc prefetch | When set, the next descriptor prefetch logic is disabled. This must be disabled if the next_src or next_dest flags are going to be used, otherwise performance may be enhanced by enabling this function. |
| 20 | Enable Cache Flushes of Local Desc | When set, all local DMA descriptors are flushed out of BCACH before being used. This only needs to be used if local DMA descriptors are in packet memory and are updated via the slave interface. Cut-through descriptors do not fall in this category. |
| 19-17 | FIFO Length Threshold | This value * 2 is used to set the FIFO length threshold. When this threshold is exceeded, moving descriptors across the PCI bus becomes a higher priority until the length moves below the threshold. |
| 16 | Enable Full Round Robin Scheduling | When set, all three DMA queues are of equal priority. When cleared, queue 0 is higher priority then queues 1 & 2. |
| 15 | Rearbitrate on Desc Completion | When set, the DMA queues are rearbitrated after each individual DMA descriptor completes. |
| 14 | Rearbitrate on Desc Chain Completion | When set, the DMA queues are rearbitrated after full DMA descriptor chains complete. This bit takes precedence over bit 15. When both bits 14 and 15 are cleared, the queues are rearbitrated after each DMA request length operation. Note: No matter how the queues are arbitrated, delayed interrupts and descriptor moves are highest priority and are arbitrated after every DMA request length operation completes. |
| 13 | Delayed Int Endian Bit | This bit determines the endian of the status word DMA transfer for delayed interrupts. |
| 12 | Route Int 2 to Int 1 | When set, the interrupt 2 signal is routed and raised as interrupt 1. This bit allows both sets of interrupt masks in the Interrupt Status register to be used, while still using only a single hardware interrupt. When set, both delayed interrupts should be enabled if they are being used. |
| 11 | Enable Delayed Int 1 | When set, the delayed interrupt mechanism for interrupt 1 is enabled. |
| 10 | Enable Delayed Int 2 | When set, the delayed interrupt mechanism for interrupt 2 is enabled. |
| 9 | Memory select for IBM2520L8767 DMA Desc | When this bit is set, the DMA descriptors that are located in the IBM2520L8767 are located in packet memory, otherwise they are located in control memory. |
| 8 | Memory select for DMA Queues | When this bit is set, the DMA Queues are located in packet memory, otherwise they are located in control memory. |
| 7 | Enable Register based DMAs | When set, source, destination, count, and system descriptor address (SDA) registers can be written to start a DMA. |
| 6 | Clear Checksum to All Ones. | When this bit is set, the DMAQS Checksum Register is set to 0xffff when it is cleared. When this bit is cleared, the DMAQS Checksum Register is set to zero when it is cleared. This option should be used if the TCP/IP checksum should never be set to zero (0xffff is zero also). |
| 5 | Queue on Error. | This bit on will cause any DMA error to log an error event. |
| 4 | Endian of DMA Descriptors. | This bit on will indicate DMA descriptors in system memory are in Little Endian format. The default is Big Endian. |
| 3 | Enable Queue 2 DMAs. | This bit enables DMA Queue 2. |
| 2 | Enable Queue 1 DMAs. | This bit enables DMA Queue 1. |
| 1 | Enable Queue 0 DMAs. | This bit enables DMA Queue 0. |
| 0 | Diagnostic Mode. | When set DMAQS is in diagnostic mode. |

### 3.10: DMAQS Enqueue DMA Descriptor Primitive

Enqueues a DMA descriptor chain to the corresponding DMA queue. This register is used to enqueue a DMA descriptor chain to the corresponding DMA queue. The write data is the address of the descriptor chain that describes the DMA transfers. The low six bits contain a count of the number of DMA descriptors in this chain. After the DMA descriptors are enqueued by writing to this register, the chain of descriptors are fetched from system memory and the DMA transfers described by the chain of descriptors are performed

| **Length** | 32 bits | |
|---|---|---|
| **Type** | Write | |
| **Address** | Queue 0 | XXXX 0618 |
| | Queue 1 | XXXX 0658 |
| | Queue 2 | XXXX 0698 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | None | |

### 3.11: DMAQS Source Address Register

Used to set and keep track of the Source Address during a DMA transfer. This is the source for the current DMA transfer. A bit in the Transfer Count and Flag Register will determine if the source address is internal to the IBM2520L8767 or is a system address.

| **Length** | 32 bits | |
|---|---|---|
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0620 |
| | Queue 1 | XXXX 0660 |
| | Queue 2 | XXXX 06A0 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | None | |

### 3.12: DMAQS Destination Address Register

Used to set and keep track of the Destination address during a DMA transfer. This is the Destination address for the current DMA transfer. A bit in the Transfer Count and Flag Register will determine if the destination address is internal to the IBM2520L8767 or is a system address.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 061C |
| | Queue 1 | XXXX 065C |
| | Queue 2 | XXXX 069C |
| **Power on Value** | X'00000000' | |
| **Restrictions** | None | |

### 3.13: DMAQS Transfer Count and Flag Register

Specifies the type and number of bytes transferred during a DMA transfer. The lower 16 bits are a counter of the number of bytes transferred during a DMA transfer. The upper 16 bits specify the type of transfer as follows.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0624 |
| | Queue 1 | XXXX 0664 |
| | Queue 2 | XXXX 06A4 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | None | |

Reserved — bits 31, 30
Clear Checksum/Hold Dest — bit 29
Compute Checksum/Hold Src — bit 28
Little Endian Mode — bit 27
Tx on DMA Complete — bit 26
Hold Mode — bit 25
Queue on DMA Complete — bit 24
Inhibit Status Update when DMA Complete — bit 23
Destination Address Specifier — bits 22, 21, 20
Source Address Specifier — bits 19, 18, 17, 16
Byte Transfer Count — bits 15–0

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

| Bit(s) | Function | Description |
|---|---|---|
| 31-30 | Reserved. | These bits must be zero. |
| 29 | Clear Checksum/Hold Dest. | When this bit is on the checksum and the alignment state are cleared. |
| 28 | Compute Checksum/Hold Src. | When this bit is set a checksum will be computed over this DMA segment. |
| 27 | Little Endian Mode. | When this bit is written to zero, this DMA channel will operate in Big Endian mode. When one, will operate in Little Endian mode. When in Little Endian mode both the source and destination must be aligned on four-byte boundaries. |
| 26 | Tx on DMA Complete | When set, the destination address is used as the packet address that is to be enqueued to CSKED to be transmitted. The lower bits are zeroed so the buffer base is used for the CSKED enqueue operation. |
| 25 | Hold Mode | When set, bit 28-29 are redefined to allow the source or destination address to be held instead of incremented. Bit 29 becomes hold destination address and bit 28 becomes hold source address. This allows a single DMA descriptor to do a N-to-1 or 1-to-N transfer. For example, an entire scatter DMA list can be freed to a RXQUE ENQ register. The address being held must be a register address. When holding, the maximum length is 252 bytes. When holding, the source or destination is incremented by four when the DMA completes (for auto-increment mode). |
| 24 | Queue on DMA Complete. | When this bit is set, the upper 26 bits of the DMAQS System Descriptor Address register will be queued to the DMA event queue when the DMA completes. If descriptors are not being used to set up the DMA, the DMAQS System Descriptor Address register should be loaded before starting the DMA with a value to identify this transfer. If descriptors are being used, the DMAQS System Descriptor Address register will be loaded automatically with the system address of the descriptor block at the time it is processed. |
| 23 | Inhibit Status Update when DMA Complete. | Normally a bit will be set in the status register when the DMA completes without error. If this bit is set, this update will not be done. This bit is useful when multiple DMAs are to be done and an interrupt is only desired on the last transfer. The DMA error status bits are unaffected by this bit. |
| 22-20 | Destination Address Specifier. | These bits specify how the destination address should be used for this DMA descriptor. The following are the valid patterns:<br>000    IBM2520L8767 Memory Address – The destination address specifies an IBM2520L8767 internal memory address.<br>001    PCI bus address – The destination address specifies a PCI bus address.<br>010    IBM2520L8767 Register Address – The destination address specifies an IBM2520L8767 register address. Only the low 16 bits must be specified.<br>011    Get IBM2520L8767 Buffer – The low four bits of the destination address specifies a pool ID from which to get a buffer. If a buffer is not available, a zero destination address event or appropriate status is raised, otherwise the buffer address is used as an IBM2520L8767 memory address.<br>100    Auto Increment Destination Address – The destination address is sourced from the previous DMA instead of the destination address specified in the descriptor.<br>101    Next Source Address – The destination address is the address of the source address field of the next descriptor in the current DMA chain. Using this feature allows indirection.<br>110    Next Destination Address – The destination address is the address of the destination address field of the next descriptor in the current DMA chain. Using this feature allows operations like doing a get buffer in the DMA descriptor chain.<br>Others  Reserved – Reserved, flagged as errors. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 19-16 | Source Address Specifier. | These bits specify how the source address should be used for this DMA descriptor. The following are the valid patterns:<br><br>0000   IBM2520L8767 Memory Address – The source address specifies an IBM2520L8767 internal memory address.<br><br>0001   PCI bus address – The source address specifies a PCI bus address.<br><br>0010   IBM2520L8767 Register Address – The source address specifies an IBM2520L8767 register address. Only the low 16 bits must be specified.<br><br>0011   IBM2520L8767 Memory Address and Free Buffer when DMA Complete – The source address specifies an IBM2520L8767 internal memory address, and this address will be freed to POOLS when the DMA is complete.<br><br>-100   Auto Increment Source Address – The source address is sourced from the previous DMA instead of the source address specified in the descriptor.<br><br>-111   Auto Increment Source Address and Free Buffer when DMA Complete – The source address is sourced from the previous DMA instead of the source address specified in the descriptor. The source address specifies an IBM2520L8767 internal memory address, and this address will be freed to POOLS when the DMA is complete.<br><br>1-00   Immediate Data – Use the Source Address Field as immediate data. Length must be $\leq$ 4.<br><br>1-11   Immediate Data and Free Buffer when DMA Complete – Use the Source Address Field as immediate data. Length must be $\leq$ 4. The source address will be freed to POOLS when the DMA is complete.<br><br>Others   Reserved – Reserved, flagged as errors. |
| 15-0 | Byte Transfer Count. | These bits indicate the number of bytes to transfer. A non-zero value in this field will start the DMA transfer. |

### 3.14: DMAQS System Descriptor Address

The upper 26 bits contain the address of the current descriptor block and the lower six bits contain the number of descriptors in the chain that remain to be processed. If DMA descriptors are used for DMA transfers, this register will contain the system address of the current descriptor block and the number of descriptors that remain to be processed. This address may be queued on DMA completion to correlate DMA transfers with system control blocks.

When doing register-based DMAs, the low six bits are set to "000001" when the DMAQS Transfer Count and Flag Register is written.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 0628 |
| | Queue 1 | XXXX 0668 |
| | Queue 2 | XXXX 06A8 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | This register should not be written if descriptors are going to be used to set up DMA transfers. If it is used, it must be written to 0 before descriptors are enqueued. | |

### 3.15: DMAQS Checksum Register

This register contains the accumulated checksum. This register contains the accumulated checksum value. It can also be used to initialize the checksum with a seed value. The most significant bit contains the alignment state (1 = odd, 0 = even alignment). The alignment state is significant between subsequent checksummed DMAs.

This register can be read at four different addresses. The base address will return the unmodified accumulated checksum. The base address +4 will return the inverted accumulated checksum. The base address + 8 will return the byte-swapped accumulated checksum. The base address + 12 will return the inverted byte-swapped accumulated checksum.

| | | |
|---|---|---|
| **Length** | 17 bits | |
| **Type** | Read/Write | |
| **Address** | Q0 Sum | XXXX 0630 |
| | Q0 Inv Sum | XXXX 0634 |
| | Q0 Swapped Sum | XXXX 0638 |
| | Q0 Inv Swapped | XXXX 063C |
| | Q1 Sum | XXXX 0670 |
| | Q1 Inv Sum | XXXX 0674 |
| | Q1 Swapped Sum | XXXX 0678 |
| | Q1 Inv Swapped | XXXX 067C |
| | Q2 Sum | XXXX 06B0 |
| | Q2 Inv Sum | XXXX 06B4 |

|                    | Q2 Swapped Sum | XXXX 06B8    |
|--------------------|----------------|--------------|
|                    | Q2 Inv Swapped | XXXX 06BC    |
| **Power on Value** | Q0 Sum         | X'00000000'  |
|                    | Q0 Inv Sum     | X'0000ffff'  |
|                    | Q0 Swapped Sum | X'00000000'  |
|                    | Q0 Inv Swapped | X'ffff0000'  |
|                    | Q1 Sum         | X'00000000   |
|                    | Q1 Inv Sum     | X'0000ffff'  |
|                    | Q1 Swapped Sum | X'00000000'  |
|                    | Q1 Inv Swapped | X'ffff0000'  |
|                    | Q2 Sum         | X'00000000'  |
|                    | Q2 Inv Sum     | X'0000ffff'  |
|                    | Q2 Swapped Sum | X'00000000'  |
|                    | Q2 Inv Swapped | X'ffff0000'  |
| **Restrictions**   | Only the base address accepts write data. All four addresses return read data. | |

### 3.16: DMAQS Delayed Int Src/Dst Registers

These registers contain the source and destination operands for the delayed interrupt function. These registers contain the source and destination operands for the delayed interrupt function.

| **Length**         | 16 bits           |           |
|--------------------|-------------------|-----------|
| **Type**           | Read/Write        |           |
| **Address**        | Delayed Int Src 1 | XXXX 0730 |
|                    | Delayed Int Dst 1 | XXXX 0734 |
|                    | Delayed Int Src 2 | XXXX 0738 |
|                    | Delayed Int Dst 2 | XXXX 073C |
| **Power on Value** | X'00000000'       |           |
| **Restrictions**   | The low two bits of the Source registers are not writable. | |

### 3.17: DMAQS Local Descriptor Range Registers

These registers specify the lower and upper bounds of the memory range for local DMA descriptors. These registers contain the address of the lower and upper bound of the memory range of descriptors that are in the IBM2520L8767. If a descriptor block is enqueued, it is compared to these registers. If it falls within this range, only the descriptor address is placed on the queue. When the descriptor is to be loaded into the DMA registers, and it falls within this range, it will not be taken from the queue but loaded directly from the descriptor address. These registers are 2-K aligned.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Lower Bound | XXXX 0720 |
| | Upper Bound | XXXX 0724 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | Can be written in diagnostic mode only. | |

### 3.18: DMAQS RAALL/CSKED Queue Number Register

This register specifies which DMAQS queue should be used when DMA descriptors are enqueued from RAALL (i.e. cut-through, scatter, etc.) or CSKED (DMA on transmit comp).

This register specifies which DMAQS queue should be used when DMA descriptors are enqueued from RAALL (i.e. cut-through, scatter, etc.) or CSKED (DMA on transmit comp). The low two bits specify RAALL queue 0, bits 3-2 specify CSKED queue, and bits 5-4 specify RAALL queue 1. The two RAALL queues are specified on an LCD basis using the queue selection bit.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0728 |
| **Restrictions** | Can be written in diagnostic mode only. Invalid values (i.e. 3), force queue number 2. |
| **Power on Value** | X'00000009' |

### 3.19: DMAQS Dma Request Size Register

This register specifies the maximum request size for DMA descriptor scheduling.This register specifies the maximum request size for DMA descriptor scheduling. This is the amount of data that DMAQS will request GPDMA to move in a single request. For example, if a descriptor wants to move 2K of data and the request size is set to 512 bytes, then DMAQS will request 512 bytes to be moved and then rearbitrate the DMA queues. Value of zero is the same as 0xffff.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 06C0 |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

### 3.20: DMAQS Enq FIFO Head Ptr Register

Used to maintain the enqueue FIFO. Points to the head FIFO entry in the FIFO array. The MSB bit, is used to determine if the head is chasing the tail, and is inverted each time the head pointer wraps

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0778 |
| **Power on Value** | X'00000000' |
| **Restrictions** | Can only be written in diagnostic mode. |

### 3.21: DMAQS Enq FIFO Tail Ptr Register

Used to maintain the enqueue FIFO. Points to the next free FIFO entry in the FIFO array. The MSB bit, is used to determine if the head is chasing the tail, and is inverted each time the tail pointer wraps

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Read/Write |
| **Address** | XXXX 077C |
| **Power on Value** | X'00000000' |
| **Restrictions** | Can only be written in diagnostic mode. |

### 3.22: DMAQS Enq FIFO Array

Holds DMA descriptor waiting to be placed on a DMA queue. Holds DMA descriptor waiting to be placed on a DMA queue. Array is organized as a 16x34 array.

To access the upper two bits of each word (holds the DMA queue number for descriptor), the array word number should be used as the address. To access the low-order 32 bits (the descriptor portion), the array word number times two plus four should be used. For example, address zero accesses the DMA queue number portion of array word zero and address four accesses the descriptor portion of array word zero.

| | |
|---|---|
| **Length** | 16 words X 34 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0780-7FC |
| **Power on Value** | X'00000000' |
| **Restrictions** | Can only be read/written in diagnostic mode. When read in non-diagnostic mode, zero is returned. |

## Entity 4: General Purpose DMA (GPDMA)

This entity provides DMA control between System Memory and IBM2520L8767 Packet Memory.

DMA transfers must be enabled in the GPDMA control registers for transmit and/or receive. There are two ways to initiate DMA transfers. The first is by directly writing the Source Address, Destination Address, and Transfer Count and Flag Registers. The second is by using DMA descriptors and enqueueing them using DMAQS. These two methods should not be used simultaneously. If using descriptors, refer to the DMAQS section beginning on page 112 for more information.

DMA transfers to system I/O space are not allowed.

### 4.1: GPDMA Interrupt Status

This register will indicate the source(s) of the interrupt(s) pending, or used as a status register when the bits are enabled. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 9 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0108 and 0C |
| **Power on Reset value** | X'000' |
| **Restrictions** | None |



| Bit(s) | Function | Description |
|---|---|---|
| 8 | DMA transaction timeout. | The DMA transaction timeout specified in the GPDMA Interrupt Enable timed out. |
| 7 | DMA Command Error. | An invalid transfer was described by the value loaded into the Transfer Count and Flag register. |
| 6 | Reserved | Reserved |
| 5 | Reserved | Reserved |
| 4 | Zero length DMA request from DMAQS. | DMAQS has requested a DMA with a length of zero. This bit is for information use only. This bit is not an error that will prevent GPDMA from processing additional DMA requests. |
| 3 | Error Occurred During Receive Transfer. | Hardware errors occurred during the last transfer. The transfer stopped after detecting the error. |

| Bit(s) | Function | Description |
|---|---|---|
| 2 | Error Occurred During Transmit Transfer. | Hardware errors occurred during the last transfer. The transfer stopped after detecting the error. |
| 1 | Receive Transfer Complete. | The receive transfer has completed. |
| 0 | Transmit Transfer Complete. | The transmit transfer has completed. |
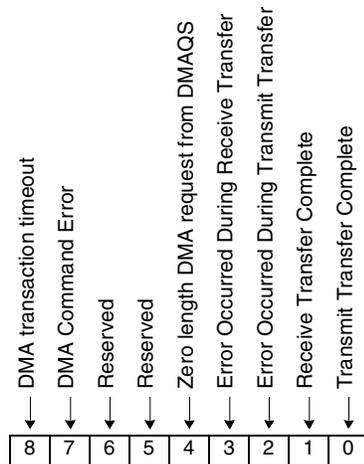
### 4.2: GPDMA Interrupt Enable

This register serves as a mask for GPDMA Interrupt Status. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See the GPDMA Interrupt Status register for the bitwise description that the corresponding bit in this register will mask.

| | |
|---|---|
| **Length** | 9 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0110 and 14 |
| **Power on Reset value** | X'9C' |
| **Restrictions** | None |

### 4.3: GPDMA Control Register

Used to set options for DMA operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 24 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0118 and 1C |
| **Power on Value** | X'880007' |
| **Restrictions** | None |



| Bit(s) | Function | Description |
|---|---|---|
| 23-20 | PCI Transfer Threshold | The value of these bits multiplied by eight determines the number of bytes that must be ready to transfer before a DMA transfer is initiated on the PCI bus. This can be used to tune the performance of the PCI bus. If the number of bytes left to transfer is less than the threshold, the transfer will start when all remaining bytes are ready to be transferred. |
| 19-16 | Memory Transfer Threshold | The value of these bits multiplied by eight determine the number of bytes that must be ready to transfer before a transfer is initiated on the internal memory bus. This can be used to tune the performance of the memory subsystem. |
| 15-8 | DMA Transaction Timeout | These bits hold a value that is used to count the number of cycles that an unacknowledged DMA cycle is in progress. If the count is reached, due to an internal chip hang condition, the DMA is terminated. A value of zero will disable this function. |
| 7 | Reserved. | Was Clear Checksum to All Ones. Now handled in DMAQS. When this bit is set, the GPDMA Checksum Register is set to 0xffff when it is cleared. When this bit is cleared, the GPDMA Checksum Register is set to zero. This option should be used if the TCP/IP checksum should never be set to zero (0xffff is zero also). |
| 6 | Disable access to internal array. | When this bit is set, the internal array cannot be read or written. This can be used to ensure that the array is not inadvertently read or written while DMAs are in progress, causing unpredictable results. |
| 5 | Enable limiting DMA burst to cache line size. | This bit on will cause a DMA burst to terminate upon crossing a cache line boundary of the PCI target. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 4-3 | PCI target cache line size. | This field will indicate the cache line size if aligning DMAs to the cache line size of the PCI target (see bit 5).<br>00   32 bytes<br>01   64 bytes<br>10   128 bytes<br>11   256 bytes |
| 2 | Reserved. | Was Enable Descriptor DMAs. Now handled in DMAQS. This bit on will enable DMA transfers to be initiated by DMA descriptors. |
| 1 | Enable Receive DMAs. | This bit on will enable DMA transfers out of the IBM2520L8767. |
| 0 | Enable Transmit DMAs. | This bit on will enable DMA transfers into the IBM2520L8767. |

### 4.4: GPDMA Source Address Register

Used to set and keep track of the Source Address during a DMA transfer. This is the system address that will increment during a DMA transfer. A bit in the Transfer Count and Flag Register will determine if the source address is internal to the IBM2520L8767 or is a system address.

**Length**          32 bits

**Type**           Read/Write

**Address**        XXXX 0128

**Power on Value**    X'00000000'

**Restrictions**     None

### 4.5: GPDMA Destination Address Register

Used to set and keep track of the Destination address during a DMA transfer. This is the Destination address that will increment during a DMA transfer. A bit in the Transfer Count and Flag Register will determine if the destination address is internal to the IBM2520L8767 or is a system address.

**Length**          32 bits

**Type**           Read/Write

**Address**        XXXX 0130

**Power on Value**    X'00000000'

**Restrictions**     None

### 4.6: GPDMA Transfer Count and Flag Register

Specifies the type and number of bytes transferred during a DMA transfer. The lower 16 bits are a counter of the number of bytes transferred during a DMA transfer. It is a count down counter; when zero is reached, the transfer ends. ***Writing a nonzero value to the lower 16 bits will start the DMA transfer.*** The upper 16 bits specify the type of transfer as follows.

**Length**          32 bits

**Type**            Read/Write

**Address**         XXXX 0138

**Power on Value**  X'000000'

**Restrictions**    None

Bit fields (bits 31–0):

- 31, 30: Reserved
- 29: Reserved. Was Clear Checksum/Hold Dest
- 28: Reserved. Was Compute Checksum/Hold Src
- 27: Little Endian Mode
- 26: Reserved
- 25: Hold Mode
- 24: Reserved
- 23: Reserved
- 22: Reserved
- 21: Register/Memory Destination Address
- 20: System/IBM2520L8767 Destination Address
- 19: Data/Address Source Address
- 18: Reserved
- 17: Register/Memory Source Address
- 16: System/IBM2520L8767 Source Address
- 15–0: Byte Transfer Count

| Bit(s) | Function | Description |
|---|---|---|
| 31-30 | Reserved. | These bits must be zero. |
| 29 | Reserved. Was Clear Checksum/Hold Dest. | When this bit is on, the checksum and the alignment state are cleared. |
| 28 | Reserved. Was Compute Checksum/Hold Src. | When this bit is set, a checksum will be computed over this DMA segment. |
| 27 | Little Endian Mode. | When this bit is written to zero, this DMA channel will operate in Big Endian mode. When one, it will operate in Little Endian mode. When in Little Endian mode both the source and destination must be aligned on four-byte boundaries. |
| 26 | Reserved. | Reserved for word swap mode. |
| 25 | Hold Mode | When set, bits 28-29 are redefined to allow the source or destination address to be held instead of incremented. Bit 29 becomes the hold destination address and bit 28 becomes the hold source address. An address being held must be a register address. When holding, the maximum length is 240 bytes. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 24 | Reserved. | Was Queue on DMA Complete. Now handled by DMAQS. When this bit is set, the upper 26 bits of the GPDMA System Descriptor Address register will be queued to the DMA event queue when the DMA completes. If descriptors are not being used to set up the DMA, the GPDMA System Descriptor Address register should be loaded, before starting the DMA, with a value to identify this transfer. If descriptors are being used, the GPDMA System Descriptor Address register will be loaded automatically with the system address of the descriptor block at the time it is processed. |
| 23 | Reserved. | Was Inhibit Status Update when DMA Complete. Normally a bit will be set in the status register when the DMA completes without error. If this bit is set, this update will not be done. This bit is useful when multiple DMAs are to be done and an interrupt is only desired on the last transfer. The DMA error status bits are unaffected by this bit. |
| 22 | Reserved. | Was Auto Increment Destination Address. This bit is only used when enqueuing descriptors. If this bit is set the destination address will be sourced from the previous DMA instead of the destination address specified in the descriptor. |
| 21 | Register/Memory Destination Address. | If this bit is set, the destination address is a register address. If this bit is not set, the destination address is a memory address. If the destination address is a system address this bit should cleared. I/O DMA cycles on the PCI bus are not implemented. |
| 20 | System/IBM2520L8767 Destination Address. | If this bit is set, the destination address is a PCI bus address. If this bit is not set, the destination address is internal to the chip. |
| 19 | Data/Address Source Address. | If this bit is set, the Source Address Register contains the source data. If this bit is not set, the Source Address Register contains the source address. |
| 18 | Reserved. | Was Auto Increment Source Address. This bit is only used when enqueuing descriptors. If this bit is set, the source address will be sourced from the previous DMA instead of the source address specified in the descriptor. |
| 17 | Register/Memory Source Address. | If this bit is set, the source address is a register address. If this bit is not set, the source address is a memory address. If the source address is a system address, this bit should cleared. I/O DMA cycles on the PCI bus are not implemented. |
| 16 | System/IBM2520L8767 Source Address. | If this bit is set, the source address is a PCI bus address. If this bit is not set, the source address is internal to the chip. |
| 15-0 | Byte Transfer Count. | These bits indicate the number of bytes to transfer. A non-zero value in this field will start the DMA transfer. |

## 4.7: GPDMA DMA Max Burst Time

Used to limit the number of cycles a master can burst on the PCI bus. When a DMA burst is started a counter is loaded with the value in this register. When the counter expires and the current access completes, the PCI bus will be released for use by another bus master. Writing a non zero value to this register enables this function

| | |
|---|---|
| **Length** | 24 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0158 |
| **Power on Value** | X'000' |
| **Restrictions** | None |

### 4.8: GPDMA Checksum Register

This register contains the accumulated checksum. This register will contain the accumulated checksum value. It can also be used to initialize the checksum with a seed value. The most significant bit contains the alignment state (1 = odd, 0 = even alignment). This register can be read at four different addresses. The base address will return the unmodified accumulated checksum. The base address +4 will return the inverted accumulated checksum. The base address + 8 will return the byte-swapped accumulated checksum. The base address + 12 will return the inverted byte-swapped accumulated checksum.

| | |
|---|---|
| **Length** | 17 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0160 |
| **Power on Value** | X'00000' |
| **Restrictions** | None |

### 4.9: GPDMA Read DMA Byte Count

Counts bytes DMAed into the IBM2520L8767. This register will count the bytes transferred into the IBM2520L8767 by the DMA controller. Descriptor bytes can optionally be included. (See GPDMA Control Register for details).

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0178 |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

### 4.10: GPDMA Write DMA Byte Count

Counts bytes DMAed out of the IBM2520L8767 Description. This register will count the bytes transferred out of the IBM2520L8767 by the DMA controller.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 017C |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

## 4.11: GPDMA Array

Reads the contents of the internal array. The internal array is used to hold data for the DMA.

**Length**            32 words x 32 bits

**Type**              Read/Write

**Address**           XXXX 0180-FF

**Power on Value**    X'00000000'

**Restrictions**      This address space is for diagnostic use only. It should not be read or written during normal operation.

# Memory Controlling Entities

## Entity 5: The DRAM Controllers (COMET/PAKIT)

This section describes the function of the COMET/PAKIT entities. COMET is the memory controller for control memory and PAKIT is the memory controller for packet memory.

Each controller can support the following types of memory:

- Extended Data Out (EDO) DRAM with a 60ns row access time and a 30ns page cycle. Supports memory sizes of 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB.

- Synchronous DRAM's running at 66MHz (15ns cycle time) with a CAS latency of 2 or 3, a burst length of 1 or 2. Supports memory sizes of 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB. Note that the cycle time of the SDRAM clock is a constant on the IBM2520L8767. Any SDRAM part selected must be capable of running at 66MHz or faster at the desired CAS latency.

- Synchronous SRAM running at 66MHz (15ns cycle time) with a read latency of 1 or 2 and a write latency of 0 or 1 (late write). Supports memory sizes of 1MB, 2MB, 4MB, or 8MB.

- For any memory configuration, modules must be selected such that the loading on any memory net (including card wiring) does not exceed 120pF.

The number of column address lines is programmable, allowing both DRAMs with symmetric address (same number of row and column address lines) and asymmetric (typically having more row than column address lines).

The memory may be operated with one RAS line or two. If the memory is configured to have two RAS lines (arrays), the memory address range is split equally between the two RAS lines (arrays).

Memory checking can be enabled/disabled and the method of checking selected can be either ECC or parity. IF ECC is selected, seven data bits are used for ECC over the 32 data bits. If parity is selected, four data bits are used to provide parity over the 32 data bits.

COMET/PAKIT are designed so that memory contents will be preserved over a reset. If the IBM2520L8767 is reset while a memory write cycle is in progress, the cycle will be completed in an orderly fashion to ensure that valid ECC/parity is written. Memory timings are not violated when reset goes active. Refresh is maintained during the reset.

**5.1: Memory Reset Sequence**

1. After a reset, onboard ROM or external firmware must properly configure the control registers for COMET/PAKIT.

2. If using EDO or SRAM, the reset sequence is complete. If using SDRAM, bit 0 of the memory controller's SDRAM Command and Status register must be written to a '1' to force the SDRAMs out of the self refresh state.

3. When the SDRAMs exit the self refresh state, bit 4 of the memory controller's SDRAM Command and Status register will read as '0' When this is detected, bit 2 of the same register must be written to '1' to initiate the SDRAM POR sequence.

4. After the POR sequence is complete, bit 5 will read '0'. The SDRAM POR sequence is now complete.

**Note:** Memory configuration errors will occur if an attempt is made to use memory that is configured incorrectly or if attempting to use SDRAM before the POR sequence is completed.

**Note:** Accesses to the first 0x20 bytes of memory (Control or Packet) are not allowed unless bit 26 of the corresponding memory control register is set. With this restriction in place, accesses with zero-valued pointers will cause the zero address error bit in the memory controller's status register to be set.
The two entities have identical registers so they are only described once.

## 5.2: COMET/PAKIT Control Register

This register contains the information which controls the functions of the entity. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. Before this register can be altered, writing it must be enabled in *COMET/PAKIT Memory Controller Write Enable Register* (described on page 152).

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0900 AND 04 |
| **PAKIT Address** | XXXX 0980 AND 84 |
| **Restrictions:** | Power On Value: X'00000000' |



| Bit(s) | Function | Description |
|---|---|---|
| 31-29 | Reserved. | Reserved |
| 28 | Memory Data Clock Control. | When set to '1', the returned memory output clock will be used to latch memory data. When a set to '0', an internal IBM2520L8767 C clock will be used. |
| 27 | SDRAM Mode Register Programming Control. | When using SDRAM, this bit, set to '0' will use the latency and the burst information available in this register to program the mode register. Set to '1', bits 23-8 of the SDRAM Command and Status Register will be used. |
| 26 | Disable Zero Address Error Detection. | When set to '1', this bit will disable the detection of zero address errors to memory. |
| 25-24 | State Information Selection. | These bits control what will be visible on the ENSTATE outputs if COMET/PAKIT are selected for observation on the ENSTATE pins. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 23 | ZBT SRAM. | If using Zero Bus Turnaround (ZBT) SRAMs, this bit should be set to '1'. Otherwise, it should be set to '0'. ZBT is supported only for a latency of 2 and is mutually exclusive with SRAM Late Write Mode. |
| 22 | SDRAM Split ECC. | When set to '1', this bit indicates that the ECC for multiple arrays of memory are in separate modules. If this bit '0', the ECC is in a shared module. This bit applies only when SDRAM is being used. |
| 21 | SDRAM Burst Length of 2. | When set to '1', this bit indicates that the SDRAM should be driven assuming a burst length of 2. This bit set to '0' indicates a burst length of 1. |
| 20 | EDO $T_{sch}$. | When set to '1', this bit will increase the DRAM timing parameter $T_{csh\ to}$ 60 nanoseconds. Otherwise, $T_{csh}$ is equal to 45ns. |
| 19 | Freeze DRAM Error Registers. | When set to '1', this bit will freeze the Memory Address Register and the DRAM ECC Syndrome Register when a memory error occurs. When this bit is set to '0', the error registers are updated whenever an error is encountered. For this bit to have any meaning with single bit errors, bit 18 must also be '1'. |
| 18 | Latch DRAM Error Registers on Single Bit Errors. | When set to '1', this bit will allow error data to be latched into the Memory Error Address Register and the DRAM ECC Syndrome Register when a single bit error occurs. When this bit is set to '0', single bits errors do not latch data into the error registers. |
| 17 | Enable ECC or Parity. | This bit set to '1' will enable ECC detection/correction or parity error detection. |
| 16 | SRAM Byte Enables for Writes Only. | This bit set to '1.' will cause byte enables to only be driven on writes to SRAM. The enables will be driven inactive for reads. If the bit is set to '0', the byte enables are valid on both reads and writes. |
| 15 | Disable SDRAM Overlapped Bank Accesses/Shorten SRAM Write Duration. | When the memory controller is configured for SDRAM, setting this bit to '1' will disable the overlapping of bank accesses. When configured for SRAM, setting this bit to '1' will shorten the time the IBM2520L8767 drives data on writes. |
| 14 | Parity or ECC | This bit set to '1' will cause parity to be generated. This bit set to '0' causes ECC to be generated. ECC is supported for DRAM only. |
| 13 | SRAM Late Write Mode | When set to '1', this bit will allow SRAM write data to be driven one cycle after the control signals. This bit set to '0.' indicates write data must be driven simultaneously with the control signals. |
| 12 | SRAM Module Width | This bit set to a '1' indicates that SRAM modules are 36 bits wide. This bit set to '0' indicates the SRAM modules are 18 bits wide. |
| 11-10 | SRAM or SDRAM Latency | These bits indicate the delay between performing a read and the memory returning data. The bits are encoded as follows:<br>00  1 Cycle (SRAM only)<br>01  2 Cycles<br>10  3 Cycles (SDRAM only)<br>11  Reserved |
| 9-8 | Memory Type | These bits indicate the type of memory being used for memory. The bits are encoded as follows:<br>00  SRAM<br>01  $T_{RAC=60,\ CAS=\ 15.0\ on/15.0\ off\ EDO\ DRAM}$<br>10  66 MHz Synchronous DRAM (SDRAM)<br>11  66 MHz Enhanced Synchronous DRAM (SDRAM) |
| 7 | Memory Unpopulated. | If this bit is '1', there is no physical memory connected to this controller. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 6-4 | DRAM Column Address Size | These bits indicate the number of column address lines. The bits are encoded as follows:<br>000   8 column address lines (256 words/row)<br>001   9 column address lines (512 words/row)<br>010   10 column address lines (1K words/row)<br>011   11 column address lines (2K words/row) - EDO only<br>100   12 column address lines (4K words/row) - EDO only<br>101   7 column address lines (128 words/row)<br>110   Reserved<br>111   Reserved |
| 3-1 | Memory Size | These bits indicate the amount of memory present. The bits are encoded as follows:<br>000   1 MB - SRAM or EDO DRAM<br>001   2 MB - SRAM or EDO DRAM<br>010   4 MB - SRAM or DRAM<br>011   8 MB - SRAM or DRAM<br>100   16 MB - DRAM<br>101   32 MB - DRAM<br>110   64 MB - DRAM<br>111   128 MB - DRAM |
| 0 | Number of DRAM Arrays. | This two bit indicates the number of arrays of DRAM present. This bit set to '0' indicates 1 array and the bit set to '1' indicates 2 arrays. |

## 5.3: COMET/PAKIT Status Register

This register contains status information for COMET/PAKIT. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0908 AND 0C |
| **PAKIT Address** | XXXX 0988 AND 8C |
| **Restrictions** | Power On Value: X'00000000' |

| Bit(s) | Function | Description |
|---|---|---|
| 31-6 | Reserved. | Reserved |
| 5 | Zero Address Error | This bit will be set if COMET/PAKIT is presented an address of zero. |
| 4 | Memory Configuration Error | This bit will be set if COMET/PAKIT is configured in an invalid combination. |
| 3 | Uncorrectable Error Detected. | This bit will be set if an uncorrectable error is detected. |
| 2 | Single Bit Error | This bit will be set if a single bit ECC error is detected. |
| 1 | Memory Address Out of Range Error. | This bit will be set if the address presented to the memory controller is out of the defined range. |
| 0 | Reserved. | Reserved |
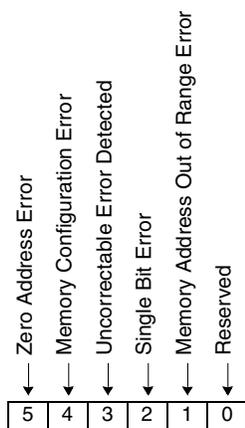
### 5.4: COMET/PAKIT Interrupt Enable Register

This register contains bits corresponding to the bits in the COMET/PAKIT status register. If a bit in this register is set and the corresponding bit is set in the COMET/PAKIT status register, an interrupt is generated. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0910 AND 14 |
| **PAKIT Address** | XXXX 0990 AND 94 |
| **Restrictions** | Power On Value: X'0000003A' |

### 5.5: COMET/PAKIT Lock Enable Register

This register contains bits corresponding to the bits in the COMET/PAKIT status register. If a bit in this register is set and the corresponding bit is set in the COMET/PAKIT status register, a signal is sent to VIMEM indicating that memory should be locked. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0918 AND 1C |
| **PAKIT Address** | XXXX 0998 AND 9C |
| **Restrictions** | Power On Value: X'0000003A' |

### 5.6: COMET/PAKIT Memory Error Address Register

This register holds the address at which the last memory error occurred.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **COMET Address** | XXXX 0920 |
| **PAKIT Address** | XXXX 09A0 |
| **Restrictions** | Power On Value: X'00000000' |

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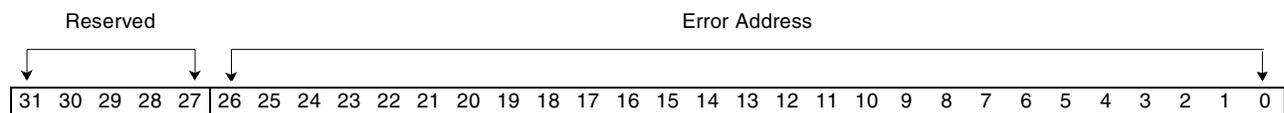
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Function | Description |
|---|---|---|
| 31-27 | Reserved. | Reserved |
| 26-0 | Error Address. | The real address of the last memory error. |

IBM

### 5.7: COMET/PAKIT SDRAM Command and Status Register

This register is used to issue various commands to the Synchronous DRAMS when they are attached to the IBM2520L8767. If the IBM2520L8767 is not configured for SDRAMS, any writes to this register will be ignored (except for bits 23 - 8). This register is also used to reflect the status of the Synchronous DRAMS. When a command bit in this register is set (bits 2-0 only), the command will execute and reset the bit upon completion. Only one bit (2-0 only) may be set during any write. Software should poll this register to make sure the previous command has completed before issuing another write to this register. If more than one bit at a time is written to this register (2-0 only) the results may be unpredictable.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0924 |
| **PAKIT Address** | XXXX 09A4 |
| **Restrictions** | Power On Value: X'00003030' |



| Bit(s) | Function | Description |
|---|---|---|
| 31-24 | Reserved. | Reserved |
| 23-8 | Command Register Data. | The value of these bits will be placed on synchronous DRAM address bits A15-A0 when the synchronous DRAM command register is written during the POR sequence (see bit 2 of this register). These bits power up to X'0030'. |
| 23-15 | | Should be written to zero. |
| 14-12 | | Should be set to the desired CAS latency. Only latency 2 and 3 are supported. |
| 11 | | Should be set to '0' for sequential addressing. |
| 10-8 | | Should be set to the burst length. Only burst length 1 and 2 are supported. |
| 7-6 | Reserved. | Reserved |
| 5 | POR | When set to '1', this bit indicates the POR sequence has not been performed on the SDRAMs. This bit will automatically reset to '0' when the POR sequence has been performed. |
| 4 | Self Refresh | This bit will read '1' when the SDRAMs are in the self refresh state. This bit will read '0' when the SDRAMs are not in the self refresh state. This bit will be a '1' after a POR or reset. The exit self refresh operation must be performed before the POR sequence is initiated. |
| 3 | Reserved. | Reserved |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 2 | Start Power on Reset Sequence. | When set to '1', this bit will cause the DRAM controller to initiate the SDRAM power on sequence. This includes an all banks precharge, following by a command register write that will set the CAS latency to 3, the wrap type to sequential, and the burst length to 1, followed by two refresh cycles. After this sequence is initiated, this bit will reset. When bit 5 has cleared, the SDRAMs are ready for normal use. |
| 1 | Enter the Self Refresh State. | When set to '1', this bit will cause the SDRAM controller to signal the SDRAMs to go into the self refresh state. All memory activity is suspended. This bit resets nearly instantly; bit 4 should be polled to determine if the SDRAMs have entered the self refresh state. |
| 0 | Exit the Self Refresh State. | When set to '1', this bit will cause the SDRAM controller to signal the SDRAMs to exit the self refresh state. This bit resets nearly instantly; bit 4 should be polled to determine if the SDRAMs have exited the self refresh state. |

## 5.8: COMET/PAKIT DRAM Refresh Rate Register

This register holds the value of a counter used to control the rate of refresh for the DRAM.

**Length**            32 bits

**Type**              Read/Write

**COMET Address**     XXXX 0928

**PAKIT Address**     XXXX 09A8

**Restrictions**      Power On Value: X'000009E6'

Reserved                                                                 Refresh Rate

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

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 31-12 | Reserved. | Reserved |
| 11-0 | Refresh Rate. | These bits are used to compare against the output of a polynomial counter. A polynomial counter is a shift register with an XOR feedback. It does not count in the same order as a binary counter, but uses less logic to implement. When the comparison is a match, a refresh is required by the DRAM. This register powers up to a refresh interval of 15.6 microseconds. |

**5.9: COMET/PAKIT Syndrome Register**

This register holds the syndrome bits that can be used to isolate the data or check bit in error when ECC is used. When parity is used, this register will indicate which of the four bytes of the memory bus had a parity error.

| | |
|---|---|
| **Length** | 7 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 092C |
| **PAKIT Address** | XXXX 09AC |
| **Restrictions** | Power On Value: X'00000000' |

Syndrome bits

```
┌──────────────────────┐
│ 6  5  4  3  2  1  0 │
└──────────────────────┘
```

| Bit(s) | Function | Description |
|---|---|---|
| bits 31-7 | Reserved | Reserved. |
| 6-0 | Syndrome bits | When using ECC, a single bit error can be identified by matching the contents of this register to the corresponding bit in the table below. When using parity, only bits 3-0 are valid and are interpreted as follows:<br>0000    No parity error<br>0001    Parity error on bits 7-0<br>0010    Parity error on bits 15-8<br>0100    Parity error on bits 23-16<br>1000    Parity error on bits 31-24<br>Other   Reserved |

## ECC Syndrome Bits

| Bit in Error | Syndromes | Bit in Error | Syndromes |
|---|---|---|---|
| ECC(6) | '1000000' | ECC(5) | '0100000' |
| ECC(4) | '0010000' | ECC(3) | '0001000' |
| ECC(2) | '0000100' | ECC(1) | '0000010' |
| ECC(0) | '0000001' | N/A | N/A |
| DATA(31) | '0111000' | DATA(30) | '0110100' |
| DATA(29) | '0110010' | DATA(28) | '0101100' |
| DATA(27) | '1110000' | DATA(26) | '1101000' |
| DATA(25) | '1100100' | DATA(24) | '1100010' |
| DATA(23) | '0100101' | DATA(22) | '0010101' |
| DATA(21) | '0001101' | DATA(20) | '1100001' |
| DATA(19) | '0110001' | DATA(18) | '0101001' |
| DATA(17) | '0011001' | DATA(16) | '1000101' |
| DATA(15) | '1010001' | DATA(14) | '1001100' |
| DATA(13) | '1001010' | DATA(12) | '1000110' |
| DATA(11) | '1000011' | DATA(10) | '1011000' |
| DATA(09) | '1010100' | DATA(08) | '1010010' |
| DATA(07) | '0100011' | DATA(06) | '0010011' |
| DATA(05) | '0001011' | DATA(04) | '0000111' |
| DATA(03) | '0011010' | DATA(02) | '0100110' |
| DATA(01) | '0010110' | DATA(00) | '0001110' |

### 5.10: COMET/PAKIT Checkbit Inversion Register

This register can be used for diagnostic purposes to invert the ECC/parity check bits that are written to memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0930 |
| **PAKIT Address** | XXXX 09B0 |
| **Restrictions** | Power On Value: X'00000000' |

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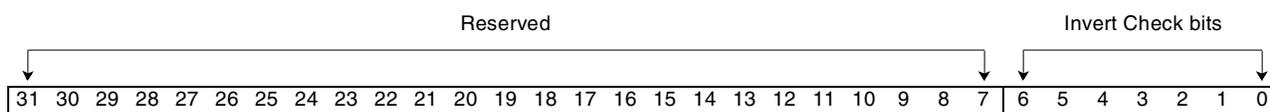
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Function | Description |
|---|---|---|
| 31-7 | Reserved | Reserved. |
| 6-0 | Invert Check bits | Setting any of these bits will invert the corresponding check bit that is written to memory. Only bits 3-0 are valid when parity is used as a checking mechanism. |

### 5.11: COMET/PAKIT Memory Controller Write Enable Register

This register must be written to a specific pattern before the Memory Control Register can be written.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **COMET Address** | XXXX 0934 |
| **PAKIT Address** | XXXX 09B4 |
| **Restrictions** | Power On Value: X'000000B4' |

Reserved           Lock bits

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

| Bit(s) | Function | Description |
|---|---|---|
| 31-8 | Reserved | Reserved. |
| 7-0 | Lock bits | This register must be written to a X'B4' before the Memory Control Register can be written. This register will POR to X'B4', but Crisco code will set up the memory controller and clear this register back to X'0'. |

# Entity 6: ATM Virtual Memory Logic (VIMEM)

This entity is responsible for adjustment of all addresses provided to the memory control entities. All addresses can be categorized into three distinct types, based entirely upon the location of the requested address with respect to the three base registers defined in this entity. The three types of addresses will be referred to as control, real packet, and virtual packet addresses. All memory requests arriving on the control memory bus are handled as control memory accesses, and simply have the contents of the control memory base register subtracted from them before being passed on to the control memory entity. When the processor accesses memory, the cache controller compares the requested address to the real packet memory base register and if the address is less than the base register, the request is routed to the control memory bus, else it is routed to the packet memory bus. All requests arriving on the packet memory bus are compared to the virtual memory base address register. If the address of the request is less than the base register, the contents of the real packet memory base register are subtracted from the address and this address is passed on to the packet memory control entity. If the requested address is greater than or equal to the base register, a more complex, but flexible scheme is used to determine the real address to provide to the packet memory control entity. For a detailed explanation of the virtual address generation scheme refer to *Virtual Memory Overview* on page 197 and the accompanying figures.

### 6.1: VIMEM Virtual Memory Base Address

This register defines the starting address of the virtual address space used to manage incoming and outgoing frames. Any time an access is made to virtual memory, that falls within the defined bounds of virtual memory, the contents of this register are subtracted from the virtual address to derive the true offset into virtual memory. This true offset, along with the known length of all virtual buffers, allows the index of the specific virtual buffer to be derived by the virtual memory access hardware. This index can then be used to access the real buffer map associated with this virtual buffer.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D10 |
| **Power On Value** | X'0040 0000' |
| **Restrictions** | The start of virtual address space must begin on a 128-KB boundary. For this reason, the lowest 17 bits of this register are forced to zero and are not implemented. Writes of any value to the low 17 bits of this register will be ignored, and a read will always return zero for the low 17 bits. |

Base Address of Virtual Memory / 128KB Boundary Restriction - Not Implemented

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

| Bit(s) | Description |
|---|---|
| 31-17 | These bits contain the upper 15 bits of the base address of virtual memory. |
| 16-0 | These bits will be forced to zero because the virtual memory base address must start on a 128K byte boundary. |

**6.2: VIMEM Control Memory Base Address**

This register defines the starting address of the control memory address space. Any time an access is made to control memory, the contents of this register is subtracted from the address before an access to memory occurs.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D14 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | The start of real control address space must begin on a 128-KB boundary. For this reason the lowest 17 bits of this register are forced to zero and are not implemented. Writes of any value to the low 17 bits of this register will be ignored, and a read will always return zero for the low 17 bits. |

Base Address of Real Control Memory            128KB Boundary Restriction - Not Implemented
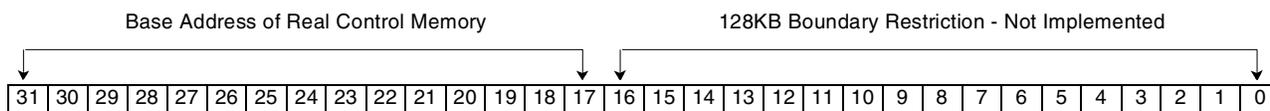
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-17 | These bits contain the upper 15 bits of the base address of real control memory. |
| 16-0 | These bits will be forced to zero because the real control memory base address must start on a 128K byte boundary. |

### 6.3: VIMEM Packet Memory Base Address

This register defines the starting address of the packet memory address space. Any time an access is made to packet memory, the contents of this register is subtracted from the address before an access to memory occurs.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D18 |
| **Power On Value** | X'0020 0000' |
| **Restrictions** | The start of real packet address space must begin on a 128-KB boundary. For this reason the lowest 17 bits of this register are forced to zero and are not implemented. Writes of any value to the low 17 bits of this register will be ignored, and a read will always return zero for the low 17 bits. This register must also be set up before any of the real buffer base registers, or the virtual buffer map registers are written. |

Base Address of Real Packet Memory          128KB Boundary Restriction - Not Implemented

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

| Bit(s) | Description |
|---|---|
| 31-17 | These bits contain the upper 15 bits of the base address of real packet memory. |
| 16-0 | These bits will be forced to zero because the real packet memory base address must start on a 128-KB boundary. |

**6.4: VIMEM Virtual Memory Total Bytes**

This register defines the total number of bytes in the address space being allocated for virtual memory. The contents of this register, divided by the configured size of virtual buffers, will yield the total number of virtual buffer indices that should be used to initialize POOLs. The value of the indices should range from this calculated value minus 1, down to zero. If an address is determined to be above or equal to the virtual memory base register it is assumed to be a virtual access. If the virtual buffer index derived from the requested address indicates that the virtual buffer space being accessed is above the limit defined by this register an error will be generated.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D0C |
| **Power On Value** | X'0001 0000' |
| **Restrictions** | The maximum value that should be set in this register is (65535 * virtual buffer size). For example, if 64-byte virtual buffers are configured, the maximum value that should be loaded into this register is X'3FFFC0'. |

```
          Virtual Memory Address Space                                              Not Implemented

 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
```
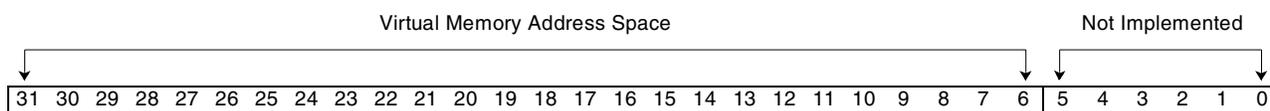
| Bit(s) | Description |
|---|---|
| 31-6 | These bits contain the upper 26 bits of the total number of bytes of address space being reserved for virtual memory. |
| 5-0 | These bits are not implemented and will be forced to zero because the virtual memory block can only be allocated in increments of the current virtual buffer size (minimum size is 64 bytes). |

### 6.5: VIMEM Virtual/Real Memory Buffer Size

This register defines the total number of bytes to be occupied by each of the virtual or real buffers as well as the spacing from one buffer to the next.

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D04 |
| **Power On Value** | X'2' |
| **Restrictions** | Care must be taken to set this register to a large enough value to contain the entire frame being sent as well as certain control information that the hardware stores in the buffer header. For example if the maximum frame being sent or received is 1024 bytes long, then this register should be set to indicate 2048-byte frames to allow sufficient room for the buffer header information added by the hardware. |

Virtual/Real
Buffer Size
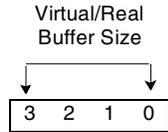
| 3 | 2 | 1 | 0 |
|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 3-0 | These bits contain the encoded 4-bit value that defines the virtual/real buffer size. The encoding is as follows:<br>0000      64 bytes<br>0001      128 bytes<br>0010      256 bytes<br>0011      512 bytes<br>0100      1024 bytes<br>0101      2048 bytes<br>0110      4096 bytes<br>0111      8192 bytes<br>1000      16384 bytes<br>1001      32768 bytes<br>1010      65536 bytes<br>1011      131072 bytes<br>1100 -1111      Reserved |

**6.6: VIMEM Packet Memory Offset**

This register contains the number that will be added by the VIMEM access logic to all accesses of real packet memory that occur. In a high performance configuration (separate control and packet store), this register should be written to all zeros to indicate that all accesses of real packet memory do not require any additional offset to be added. In a medium performance configuration (combined control and packet store), this register should be loaded with a value that indicates the logical partitioning between control and packet storage. If for instance, a single bank of 2 meg was configured and this register was loaded with X'00100000' (1 meg), then all accesses to real packet memory would be forced into the 1-meg to 2-meg range.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D3C |
| **Power On Value:** | X'0000 0000' |

| | |
|---|---|
| **Restrictions** | This register should only be loaded with a non-zero value if a medium performance configuration (combined control and packet store) exists. The value loaded must be between zero and the maximum of the total amount of memory in the single bank, and it must be on a 128-KB boundary. Any time the value in this register is changed, the related base registers must be reloaded because the value loaded into them is affected by the contents of this register during the load operation. The related registers are the virtual buffer map base address register and all five real buffer base registers. |

**6.7: VIMEM Maximum Buffer Size**

This register is used by the virtual memory logic to determine if an access to a virtual buffer falls into the region of the buffer that can be accessed. If a virtual buffer read or write accesses an offset in a virtual buffer that is greater than the contents of this register, the virtual memory logic can be configured to halt and generate an interrupt. The power up value of all '1's will cause this check to be disabled. This register is intended to provide the user with a means of providing additional protection to accesses of the virtual buffers. For example, if this register is loaded with X'FF8', all memory access up to and including the byte at address X'FFF' will be allowed. Any access of offset X'1000' or above will cause an exception.

| | |
|---|---|
| **Length** | 17 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D34 |
| **Power On Value** | X'1 FFF8' |
| **Restrictions** | All address logic based on this register only recognizes eight-byte words in memory. For this reason, the low three bits of this register are not implemented and will always be forced to zeros. |

### 6.8: VIMEM Access Control Register

The bits in this register control the configurable features of the virtual memory logic. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D80 and 84 |
| **Power On Value** | X'0' |
| **Restrictions** | None |

| Bit(s) | Description |
|---|---|
| 31-16 | Reserved. |
| 15 | When set, this bit will force the virtual memory logic to ignore the virtual buffer map validity indication, and force all maps to appear valid. |
| 14 | When set, this bit will force the virtual memory logic to fetch the required map entry from storage on every new virtual access. If a virtual memory map is updated by the software for any reason, this bit should be toggled on and off after the map is updated and before any virtual access happens to ensure that the virtual memory logic is not using stale cached map segments. There is no hardware provided to make sure that the map entry required by the virtual memory logic is not contained in one of the BCACH lines. If is the responsibility of the software to ensure that all modified lines are flushed from the cache before the virtual memory logic needs them. |
| 13 | When set, this bit will force all accesses to packet memory to be serialized. |
| 12 | When set, this bit will cause control accesses to always have priority over packet accesses in a single memory bank configuration. When reset, priority will toggle every time an access is initiated. |
| 11-0 | Reserved. |

## 6.9: VIMEM Access Status Register

This register contains information regarding the current status of the virtual memory logic mainly with respect to detected error access conditions. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D60 and 64 |
| **Power On Value** | X'0000' |
| **Restrictions** | None |

Reserved

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

| Bit(s) | Description |
|---|---|
| 31-17 | Reserved. |
| 16 | When set, this bit indicates that the required conditions for the control, packet, and virtual base registers has not been satisfied. The required conditions are: control base < packet base < virtual base |
| 15 | When set, this bit indicates that the virtual memory logic has detected a page fault error when attempting to read memory. This indicates that no real buffer was available to map into the virtual address space when required. All virtual reads that fail during a page fault regardless of the requesting entity will cause this bit to be set. If the corresponding bit is reset in the lock register, the read operation will complete, but with invalid data. |
| 14 | When set, this bit indicates that a control memory access was detected that was above the value contained in the Packet memory offset register for single bank configurations, or in a multiple bank configuration that the high address bits 31 - 27 were not zero. |
| 13 | When set, this bit indicates that a packet memory access of address zero was detected in single bank mode, or that a packet address was detected that contained an address out of range (high five bits non-zero). |
| 12 | When set, this bit indicates that the virtual memory logic has detected a virtual memory operation that attempted to access a map that was not marked as valid. A virtual buffer map is marked valid by the POOLS entity when the buffer is originally acquired, and is marked as invalid when the buffer is freed back to POOLS. Receiving this error indication, typically means that the software is trying to use a buffer that has not been acquired through the normal means, or trying to use a buffer that has already be freed, or that memory has been corrupted. The valid indication that is checked by the hardware is the value X'?656' in the first 16 bits of the eight-byte map entry being accessed. To determine the failing address, the memory control entity can be locked on this type of failure, and the information saved by the memory controller, along with the base registers in this entity can be used to determine which map was being accessed at the time of failure. |
| 11 | When set, this bit indicates that the virtual memory logic has detected a non-recoverable page fault error when attempting to write memory. This indicates that no real buffer was available to map into the virtual address space when required. All virtual writes that fail during a page fault with the exception of BCACH and RAALL operations will cause this bit to be set. |
| 10 | When set, this bit indicates that the virtual memory logic has detected a recoverable page fault error when attempting to write memory. This indicates that no real buffer was available to map into the virtual address space when required. Operations from BCACH and RAALL will cause this bit to be set instead of the non-recoverable bit because the software can recover from these failures. If a BCACH write to virtual memory fails in this manner, the packet header of the frame being updated will be updated to indicate the failure. Software can check the field in the packet header to ensure that the DMA operation completed successfully. If such a packet is enqueued to CSKED, the packet header is checked and will prevent the frame from being passed on to the segmentation logic. The frame will generate a BAD transmit event in RXQUE instead. If a failure is indicated, the software must perform any required recovery actions. If a RAALL write to virtual memory fails in this manner, the packet currently being received is dropped, it is up to the software to perform any recovery operations that are required. |

| Bit(s) | Description |
|---|---|
| 09 | When set, this bit indicates that the virtual memory logic has detected a read operation that caused a page fault. This is an invalid condition because the data required for a read operation should have been previously initialized by a write operation, so no page fault should ever occur on a read operation. If the corresponding bit in the lock register is reset, a page will be mapped into the current virtual buffer segment and the data that previously was written in that page will be returned. This bit can come on in several situations that are not really errors. In these cases, the associated interrupt and lock bits can be reset so that this error does not cause the adapter to halt normal operation. Several of these conditions are: When predictive fill is enabled, a read from the end of a buffer may cause a predictive read that crosses a virtual segment boundary and causes this bit to be set. If a small buffer (fits entirely in the cache) is copied from one IBM2520L8767 buffer to another IBM2520L8767 buffer, a subsequent read of the last bytes written will cause this bit to be set if the cache hasn't been flushed between the write and the read, and the last write cycle did not write all 4 bytes, and the address that is being written/read is within the first 0x20 bytes of a virtual segment. |
| 08 | When set, this bit indicates that the virtual memory logic has detected an access of a virtual buffer that falls above the limit set by the buffer maximum size register. |
| 07 | When set, this bit indicates that the virtual memory logic has detected an access that does not fall in one of the currently mapped buffer segments based upon the currently-configured virtual buffer map size. |
| 06 | When set, this bit indicates that a virtual access has been detected that used a base register that had an invalid associated buffer size configured in the low order bits. |
| 05 | When set, this bit indicates that a virtual access has been detected that used a base register that was not on the correct memory boundary. For example, if a base register is set up to use 2-KB buffers, then the base register must be set up on a 2-KB boundary. |
| 04 | When set, this bit indicates that a virtual access has been detected that used a base register that contained a value of 0. |
| 03 | Reserved. |
| 02 | When set, this bit indicates that the virtual memory logic has detected a memory access that resulted in the generation of a buffer index that was greater than the currently configured maximum derived from the VIMEM virtual memory total bytes register. |
| 01 | When set, this bit indicates that the currently configured size of buffers is invalid. |
| 00 | When set, this bit indicates that the map base register contains an invalid value. Two possible causes are that bits 5-2 are not zero or bits 31-6 are zero. |

### 6.10: VIMEM Access Status Interrupt Enable Register

This register allows the user to enable interrupts for each of the conditions reported in the VIMEM Access Status register. Each bit corresponds to the same bit in the Status register and when set to a '1' will generate an interrupt to the processor if the condition is detected. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 17 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D68 and 6C |
| **Power On Value** | X'1FFFF' |
| **Restrictions** | None |

### 6.11: VIMEM Memory Lock Enable Register

This register allows the user to selectively allow each of the conditions reported in the VIMEM Access Status register to force a memory lock condition in the memory controller. Each bit corresponds to the same bit in the Status register and when set to a '1' will cause a memory lock if the condition is detected. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 17 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D70 and 74 |
| **Power On Value** | X'1FFFF' |
| **Restrictions** | None |

### 6.12: VIMEM State Machine Current State

This register provides feedback to the user regarding the current status of the state machines in VIMEM. One use of this register is to make sure that the required initialization time has expired after loading the segment size register. This is accomplished by reading this register repeatedly until the initialization state machine is in the idle state.

| | |
|---|---|
| **Length** | 9 bits |
| **Type** | Read Only |
| **Address** | XXXX 0D78 |
| **Power On Value** | X'1?0' |
| **Restrictions** | None |

State of Initialization State Machine    Current State
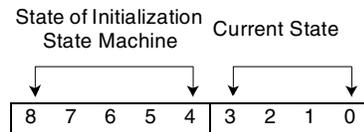
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 8-4 | These bits contain the current state of the initialization state machine. A value of "1----" indicates that the state machine is in the idle state. |
| 3-0 | These bits contain the current state of the VIMEM main state machine. A value of "0000" indicates that the state machine is in the idle state. |

### 6.13: VIMEM Last Processor Read Real Address

This register provides information to the user about the last read access of virtual packet memory by the processor. If a virtual address was accessed, this register will contain the real address generated by the virtual memory logic that can be used to access the same location. This register is intended mainly as an aid in debugging to make virtual address translation easier. To perform the translation, the processor must read from the desired virtual address, after the read is complete, this register will contain the real address that was accessed. The address contained in this register is an offset from the beginning of physical packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0D7C |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

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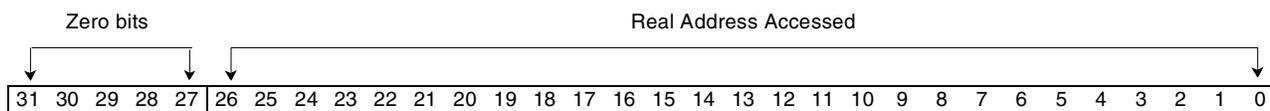
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|---|---|
| 31-27 | These bits will always read as zero. |
| 26-0 | After any read operation from the processor to packet memory, these bits will contain the real address that was accessed. |

## 6.14: VIMEM Virtual Buffer Segment Size Register

This register, along with the lower four bits of the real buffer base registers, defines the size of the second through 16th real buffers that are concatenated to make up a virtual buffer. Two bits of this register are associated with each real buffer segment and indicate one out of four possible associations. The associative possibilities are shown in the Bit table below. Every two bit defines the connection between a particular buffer segment and the real buffer base registers.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D00 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | Care must be used when setting up this register to ensure that only values that correspond to real buffer sizes that POOLS has also been set up to provide are loaded. A write to this register causes the virtual memory logic to calculate the different real buffer boundaries within a virtual buffer. This calculation requires information from the real buffer base registers to determine the size of the different segments making up the virtual buffer. For this reason it is required that this register be written after the real buffer base registers have been initialized. After writing this register, the software must wait at least 2 microseconds before accessing virtual memory. |

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

| Bit(s) | Description | Bit Association |
|--------|-------------|-----------------|
| 31-30 | Defines the 16th buffer segment's connection. | |
| 29-28 | Defines the 15th buffer segment's connection. | |
| 27-26 | Defines the 14th buffer segment's connection. | |
| 25-24 | Defines the 13th buffer segment's connection. | |
| 23-22 | Defines the 12th buffer segment's connection. | |
| 21-20 | Defines the 11th buffer segment's connection. | |
| 19-18 | Defines the 10th buffer segment's connection. | 00    Associates this real buffer segment with Real Buffer Base Register 0 |
| 17-16 | Defines the 9th buffer segment's connection. | 01    Associates this real buffer segment with Real Buffer Base Register 1 |
| 15-14 | Defines the 8th buffer segment's connection. | 10    Associates this real buffer segment with Real Buffer Base Register 2 |
| 13-12 | Defines the 7th buffer segment's connection. | 11    Associates this real buffer segment with Real Buffer Base Register 3 |
| 11-10 | Defines the 6th buffer segment's connection. | |
| 09-08 | Defines the 5th buffer segment's connection. | |
| 07-06 | Defines the 4th buffer segment's connection. | |
| 05-04 | Defines the 3rd buffer segment's connection. | |
| 03-02 | Defines the 2nd buffer segment's connection. | |
| 01-00 | Reserved. The first real buffer is implicitly associated with the virtual buffer, these bits will always be read as zero. | |

### 6.15: VIMEM Buffer Map Base Address

This register contains the address in packet memory at which the buffer map table starts. The buffer map table consists of a variable number of 8 byte entries for each buffer that will be allocated in the system. The first 16 bits of each 8 byte entry contains the POOL ID and various status flags associated with this buffer, thus this base register is used in both real and virtual memory modes. In virtual memory mode, each of the three subsequent 16 bits contains an index which is associated with a buffer size base register using the buffer segment limit register. The index and buffer size base register are used to determine a real buffer address. If the map size is set to 8 bytes, only one 8 byte entry is used for each buffer. If the map size is set to 16 bytes, two 8 byte entries are used for each buffer. If the map size is set to 32 bytes, four 8 byte entries are used for each buffer. If the map size is set to 64 bytes, five 8 byte entries are used for each buffer, the remaining 24 bytes of the map are unused by the hardware.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0D08 |
| **Power On Value** | X'0020 0000' This value is actually the power up contents of the packet memory real base register added to the power up contents of this register (X'00000000') due to the automatic address adjustment explained below. |
| **Restrictions** | The base address for the buffer map must begin on a 64-byte boundary. When a base register is written, the hardware performs an automatic adjustment to the address using the contents of the packet memory real base register, and the packet memory offset register. This results in the actual value being stored, not being the value that is written by the program. This is done to make the virtual accesses that use the base register execute quicker. The reverse adjustment is made when the read operation is performed, so that it appears to the program no different than a normal operation. Care must be taken however to ensure that both the packet memory real base register and the packet memory offset register are set-up before any of the base registers are written. If the packet memory base register or the packet memory offset register are changed, packet memory should not be accessed until all the base registers have been written again. |

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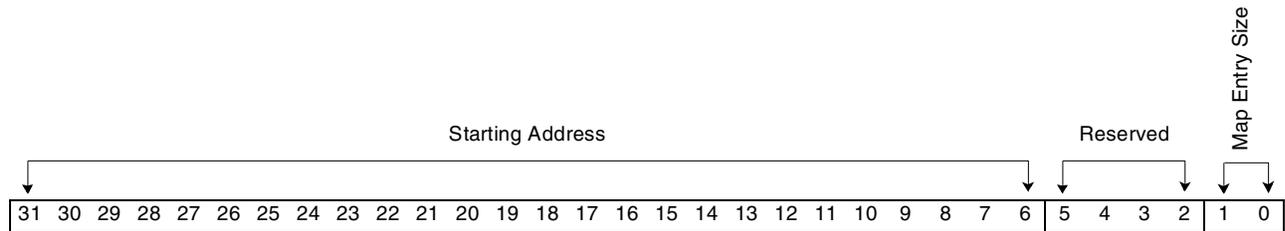
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 31-06 | Defines the starting address of the buffer map |
| 05-02 | Reserved, should be written with 0 |
| 01-00 | Defines the size of each map entry<br>00   8 bytes<br>01   16 bytes<br>10   32 bytes<br>11   64 bytes |

**6.16: VIMEM Real Buffer Base Addresses**

These registers contain the address in packet memory at which a block of memory begins that is used to pro-
vide a given size buffer. In general, the block allocated must be large enough to contain as many buffers as
will be freed to POOLs on initialization, however for Real Buffer Base 4, the size of the block reserved must
be large enough so that one buffer is available for each of the virtual buffers freed to POOLs. These buffers
must not be freed to POOLs because they are implicitly used as the first real buffer segment for each of the
virtual buffers. If a given base register and associated buffer size is not used, the low four bits of the register
should be set to X'F' to ensure that accesses of this buffer size will be detected and flagged as an error.
When using real memory mode (controlled in POOLS), all of these base registers are unused with the excep-
tion of base register zero, which contains the base address for all real memory buffers. In real mode, the low
four bits of base register zero are of no significance. The size of the real buffers is controlled through the
Buffer size register.

| Buffer Size | 0 | 1 | 2 | 3 | 4 (implicit) |
|---|---|---|---|---|---|
| **Length** | 32 bits | 32 bits | 32 bits | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write | Read/Write | Read/Write | Read/Write |
| **Address** | XXXX 0D20 | XXXX 0D24 | XXXX 0D28 | XXXX 0D2C | XXXX 0D30 |
| **Power on Value** | X'0020 000F' | X'0020 000F' | X'0020 000F' | X'0020 000F' | X'0020 000F' |

**Restrictions**  The base address for any given buffer size must begin on a boundary that is equal to the
buffer size. For example the base address for 128-byte buffers must be on a 128-byte
boundary and the base address for 4096-byte buffers must be on a 4096-byte boundary.

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

| Bit(s) | Description |
|---|---|
| 31-6 | Defines the starting address in packet memory of the memory block used to provide real buffers of defined size |
| 5-4 | Reserved (User should write zeros and ignore read value) |
| 3-0 | Defines the size of the real buffers in this block of memory with the following encoding:<br>0000　　　 64 bytes<br>0001　　　 128 bytes<br>0010　　　 256 bytes<br>0011　　　 512 bytes<br>0100　　　 1024 bytes<br>0101　　　 2048 bytes<br>0110　　　 4096 bytes<br>0111　　　 8192 bytes<br>1000　　　 16384 bytes<br>1001　　　 32768 bytes<br>1010　　　 65536 bytes<br>1011　　　 131072 bytes<br>1100 -1111　 Reserved |

# Entity 7: ATM Packet/Control Memory Arbitration Logic (ARBIT)

This section contains descriptions of the registers used by the arbiter logic.

### 7.1: ARBIT Control Priority Resolution Register High

This register, which consists of five 4-bit fields, defines the priority of requesting entities to packet memory.

**Length**            20 bits

**Type**              Read/Write

**Address**           XXXX 0E00

**Restrictions**      None

**Power On Value**    X'C BA98'



| Bit(s) | Description |
|---|---|
| 19-16 | The value loaded into these bits define which entity will be requesting at priority level C (lowest priority)<br>Value C    CHKSM<br>Value B    BCACH LO<br>Value A    POOLS LO<br>Value 9    CSKED<br>Value 8    REASM<br>Value 7    RXQUE<br>Value 6    PCORE<br>Value 5    SEGBF<br>Value 4    RAALL<br>Value 3    GPDMA<br>Value 2    DMAQS<br>Value 1    POOLS HI<br>Value 0    BCACH HI |
| 15-12 | The value loaded into these bits define which entity will be requesting at priority level B. See description of bits 19-16 for entity values. |
| 11-08 | The value loaded into these bits define which entity will be requesting at priority level A. See description of bits 19-16 for entity values. |
| 07-04 | The value loaded into these bits define which entity will be requesting at priority level 9. See description of bits 19-16 for entity values. |
| 03-00 | The value loaded into these bits define which entity will be requesting at priority level 8. See description of bits 19-16 for entity values. |

**7.2: ARBIT Control Priority Resolution Register Low**

This register, which consists of eight 4-bit fields, defines the priority of requesting entities to packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0E04 |
| **Restrictions** | None |
| **Power On Value** | X'7654 3201' |

| Bit(s) | Description |
|---|---|
| 31-28 | The value loaded into these bits define which entity will be requesting at priority level 7.<br>Value C  CHKSM<br>Value B  BCACH LO<br>Value A  POOLS LO<br>Value 9  CSKED<br>Value 8  REASM<br>Value 7  RXQUE<br>Value 6  PCORE<br>Value 5  SEGBF<br>Value 4  RAALL<br>Value 3  GPDMA<br>Value 2  DMAQS<br>Value 1  POOLS HI<br>Value 0  BCACH HI |
| 27-24 | The value loaded into these bits define which entity will be requesting at priority level 6. See description of bits 31-28 for entity values. |
| 23-20 | The value loaded into these bits define which entity will be requesting at priority level 5. See description of bits 31-28 for entity values. See description of bits 31-28 for entity values. |
| 19-16 | The value loaded into these bits define which entity will be requesting at priority level 4. See description of bits 31-28 for entity values. |
| 15-12 | The value loaded into these bits define which entity will be requesting at priority level 3. See description of bits 31-28 for entity values. |
| 11-08 | The value loaded into these bits define which entity will be requesting at priority level 2. See description of bits 31-28 for entity values. |
| 07-04 | The value loaded into these bits define which entity will be requesting at priority level 1. See description of bits 31-28 for entity values. |
| 03-00 | The value loaded into these bits define which entity will be requesting at priority level 0 (highest priority). See description of bits 31-28 for entity values. |

### 7.3: ARBIT Control Error Mask Register

The bits in this register control if ARBIT detected error conditions on an entities interface will lock the control memory subsystem. Bits in this register also control the locking of the control memory subsystem based on control memory, packet memory, virtual memory, and BCACH detected error conditions. Resetting the appropriate bit will force errors from that source to be ignored.

**Length**            18 bits

**Type**              Clear/Set

**Address**           XXXX 0E08 and 0C

**Power On Value**    X'3FFFF'

**Restrictions**      None

| Bit(s) | Bit Name/Function |
|--------|-------------------|
| 17 | ARBIT detected packet errors |
| 16 | POOLs errors |
| 15 | Reserved |
| 14 | BCACH collision |
| 13 | Virtual errors |
| 12 | Packet errors |
| 11 | Control errors |
| 10 | CHKSM |
| 09 | CSKED |
| 08 | REASM |
| 07 | RXQUE |
| 06 | PCORE |
| 05 | SEGBF |
| 04 | RAALL |
| 03 | GPDMA |
| 02 | DMAQS |
| 01 | POOLS |
| 00 | BCACH |

**7.4: ARBIT Control Error Source Register**

The bits in this register provide feedback to indicate the source of errors that have been detected by the memory subsystem.

| | |
|---|---|
| **Length** | 18 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0E18 and 1C |
| **Power On Value** | X'00000' |
| **Restrictions** | Bits 16, 17, and 11 through 14 are driven from external entities and can not be set/reset in this register. They must be set/reset in the entity of origin. |

| Bit(s) | Bit Name/Function |
|--------|-------------------|
| 17 | ARBIT detected packet errors |
| 16 | POOLS errors |
| 15 | Reserved |
| 14 | BCACH collision |
| 13 | Virtual errors |
| 12 | Packet errors |
| 11 | Control errors |
| 10 | CHKSM |
| 09 | CSKED |
| 08 | REASM |
| 07 | RXQUE |
| 06 | PCORE |
| 05 | SEGBF |
| 04 | RAALL |
| 03 | GPDMA |
| 02 | DMAQS |
| 01 | POOLS |
| 00 | BCACH |

## 7.5: ARBIT Control Winner Register

The bits in this register indicate which entity currently owns control memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0E2C |
| **Power On Value** | X'F' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 31 | For performance reasons, two sets of operational latches (bank A and bank B) exist in the arbiter for control memory. When set, this bit indicates that the B latches are active, and when reset it indicates that the A latches are active. When this bit is set and memory is locked, bits 7-4 of this register contain a value that indicates the entity that most recently was accessing memory. If this bit is reset and memory is locked, bits 3-0 of this register contain a value that indicates the entity that most recently was accessing memory |
| 30 | Reserved. |
| 29-8 | Reserved. will read zero. |
| 7-4 | Control winner B |
| 3-0 | Control winner A<br>Value C     CHKSM<br>Value B     BCACH LO<br>Value A     POOLS LO<br>Value 9     CSKED<br>Value 8     REASM<br>Value 7     RXQUE<br>Value 6     PCORE<br>Value 5     SEGBF<br>Value 4     RAALL<br>Value 3     GPDMA<br>Value 2     DMAQS<br>Value 1     POOLS HI<br>Value 0     BCACH HI |

### 7.6: ARBIT Control Address Register A

If latch bank A is active, the bits in this register indicate the last address that was used to access control memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0E10 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |
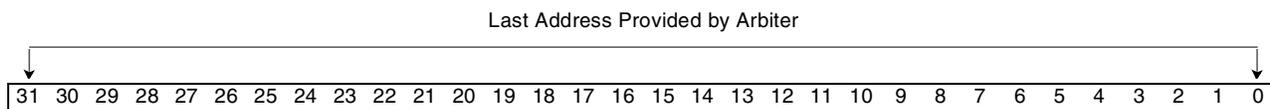
Last Address Provided by Arbiter

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain the last address provided by the arbiter to the control memory controller. |

### 7.7: ARBIT Control Address Register B

If latch bank B is active, the bits in this register indicate the last address that was used to access control memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0E20 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |
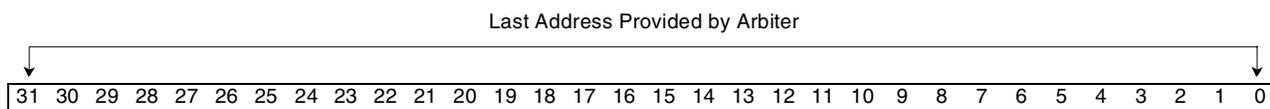
Last Address Provided by Arbiter

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain the last address provided by the arbiter to the control memory controller. |

### 7.8: ARBIT Control Length Register

The bits in this register indicate the last length that was used to access control memory.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read |
| **Address** | XXXX 0E14 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |
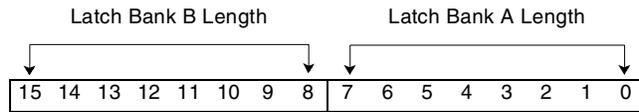
Latch Bank B Length          Latch Bank A Length

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|---|---|
| 15-8 | These bits contain the length used to access control memory through latch bank B. |
| 7-0 | These bits contain the length used to access control memory through latch bank A. |

### 7.9: ARBIT Control Lock Entity Enable Register

The value programmed in this register controls what entity, if any, has access to packet memory immediately after memory has locked. This register powers up to a value that will not allow any entity to access memory after a lock condition until the lock condition has been properly cleared.

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0E28 |
| **Power On Value** | X'F' |
| **Restrictions** | None |

Bit Map Value

| 3 | 2 | 1 | 0 |
|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 3-0 | The value in these bits map to the following entities:<br>Value C    CHKSM<br>Value B    BCACH LO<br>Value A    POOLS LO<br>Value 9    CSKED<br>Value 8    REASM<br>Value 7    RXQUE<br>Value 6    PCORE<br>Value 5    SEGBF<br>Value 4    RAALL<br>Value 3    GPDMA<br>Value 2    DMAQS<br>Value 1    POOLS HI<br>Value 0    BCACH HI |

### 7.10: ARBIT Control Config Register

The bits in this register control the operation of the control memory arbiter.

| | |
|---|---|
| **Length** | 2 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0E38 and 3C |
| **Power On Value** | X'0' |
| **Restrictions** | None |

Reserved  Set/Reset

| 1 | 0 |
|---|---|

| Bit(s) | Description |
|---|---|
| 1 | Reserved |
| 0 | When set, this bit forces all operations to control memory to be serialized. An operation from one entity must be entirely complete before an operation from another entity will be started. When reset, if the memory operation in process can be overlapped, a second operation will be started before the first operation is complete. |

## 7.11: ARBIT Packet Priority Resolution Register High

This register, which consists of five 4-bit fields, defines the priority of requesting entities to packet memory.

**Length**          20 bits

**Type**            Read/Write

**Address**         XXXX 0E80

**Power On Value**  X'C BA98'

**Restrictions**    None

| Priority Level C | Priority Level B | Priority Level A | Priority Level 9 | Priority Level 8 |
|---|---|---|---|---|

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

| Bit(s) | Description |
|---|---|
| 19-16 | The value loaded into these bits defines which entity will be requesting at priority level C (lowest priority).<br>Value C    CHKSM<br>Value B    BCACH LO<br>Value A    POOLS LO<br>Value 9    CSKED<br>Value 8    REASM<br>Value 7    RXQUE<br>Value 6    PCORE<br>Value 5    SEGBF<br>Value 4    RAALL<br>Value 3    GPDMA<br>Value 2    DMAQS<br>Value 1    POOLS HI<br>Value 0    BCACH HI |
| 15-12 | The value loaded into these bits define which entity will be requesting at priority level B. See description of bits 19-16 for entity values. |
| 11-08 | The value loaded into these bits define which entity will be requesting at priority level A. See description of bits 19-16 for entity values. |
| 07-04 | The value loaded into these bits define which entity will be requesting at priority level 9. See description of bits 19-16 for entity values. |
| 03-00 | The value loaded into these bits define which entity will be requesting at priority level 8. See description of bits 19-16 for entity values. |

### 7.12: ARBIT Packet Priority Resolution Register Low

This register, which consists of eight 4-bit fields, defines the priority of requesting entities to packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0E84 |
| **Restrictions** | None |
| **Power On Value** | X'7654 3201' |

| Bit(s) | Description |
|---|---|
| 31-28 | The value loaded into these bits defines which entity will be requesting at priority level 7.<br>Value C    CHKSM<br>Value B    BCACH LO<br>Value A    POOLS LO<br>Value 9    CSKED<br>Value 8    REASM<br>Value 7    RXQUE<br>Value 6    PCORE<br>Value 5    SEGBF<br>Value 4    RAALL<br>Value 3    GPDMA<br>Value 2    DMAQS<br>Value 1    POOLS HI<br>Value 0    BCACH HI |
| 27-24 | The value loaded into these bits define which entity will be requesting at priority level 6. See description of bits 31-28 for entity values. |
| 23-20 | The value loaded into these bits define which entity will be requesting at priority level 5. See description of bits 31-28 for entity values. |
| 19-16 | The value loaded into these bits define which entity will be requesting at priority level 4. See description of bits 31-28 for entity values. |
| 15-12 | The value loaded into these bits define which entity will be requesting at priority level 3. See description of bits 31-28 for value definitions. |
| 11-08 | The value loaded into these bits define which entity will be requesting at priority level 2. See description of bits 31-28 for entity values. |
| 07-04 | The value loaded into these bits define which entity will be requesting at priority level 1. See description of bits 31-28 for entity values. |
| 03-00 | The value loaded into these bits define which entity will be requesting at priority level 0 (highest priority). See description of bits 31-28 for entity values. |

**7.13: ARBIT Packet Entity Error Mask Register**

The bits in this register control if ARBIT detected error conditions on an entities interface will lock the packet memory subsystem. Bits in this register also control the locking of the packet memory subsystem based on control memory, packet memory, virtual memory, and BCACHE detected error conditions. Resetting the appropriate bit will force errors from that source to be ignored.

| | |
|---|---|
| **Length** | 18 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0E88 and 8C |
| **Power On Value** | X'3FFFF' |
| **Restrictions** | None |

| Bit(s) | Bit Name/Function |
|---|---|
| 17 | ARBIT detected control errors |
| 16 | POOLS errors |
| 15 | Re-arbitration failure |
| 14 | BCACH collision |
| 13 | Virtual errors |
| 12 | Packet errors |
| 11 | Control errors |
| 10 | CHKSM |
| 09 | CSKED |
| 08 | REASM |
| 07 | RXQUE |
| 06 | PCORE |
| 05 | SEGBF |
| 04 | RAALL |
| 03 | GPDMA |
| 02 | DMAQS |
| 01 | POOL |
| 00 | BCACH |

### 7.14: ARBIT Packet Error Source Register

The bits in this register provide feedback to indicate the source of errors that have been detected by the memory subsystem.

| | |
|---|---|
| **Length** | 18 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0E98 and 9C |
| **Power On Value** | X'00000' |
| **Restrictions** | Bits 17, 16, and 11 through 14 are driven from external entities and can not be set/reset in this register. They must be set/reset in the entity of origin. |

| Bit(s) | Bit Name/Function |
|---|---|
| 17 | ARBIT detected control errors |
| 16 | POOLS errors |
| 15 | Rearbitration detected while already handling a rearbitration condition. This condition would indicate that the priorities programmed in the priority resolution logic were incorrectly programmed and POOLS HIGH was not given the highest priority. |
| 14 | BCACH collision |
| 13 | Virtual errors |
| 12 | Packet errors |
| 11 | Control errors |
| 10 | CHKSM |
| 09 | CSKED |
| 08 | REASM |
| 07 | RXQUE |
| 06 | PCORE |
| 05 | SEGBF |
| 04 | RAALL |
| 03 | GPDMA |
| 02 | DMAQS |
| 01 | POOLS |
| 00 | BCACH |

IBM

### 7.15: ARBIT Packet Winner Register

The bits in this register indicate which entity currently owns packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0EAC |
| **Power On Value** | X'F' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 31 | For performance reasons, two sets of operational latches (bank A and bank B) exist in the arbiter for packet memory. When set, this bit indicates that the B latches are active, and when reset indicates that the A latches are active. When this bit is set and memory is locked, bits 7-4 of this register contain a value that indicates the entity that most recently was accessing memory. If this bit is reset and memory is locked, bits 3-0 of this register contain a value that indicates the entity that most recently was accessing memory |
| 30 | Reserved |
| 29-8 | Reserved, will read zero. |
| 7-4 | Packet winner B |
| 3-0 | Packet winner A<br>Value C    CHKSM<br>Value B    BCACH LO<br>Value A    POOLS LO<br>Value 9    CSKED<br>Value 8    REASM<br>Value 7    RXQUE<br>Value 6    PCORE<br>Value 5    SEGBF<br>Value 4    RAALL<br>Value 3    GPDMA<br>Value 2    DMAQS<br>Value 1    POOLS HI<br>Value 0    BCACH HI |

**7.16: ARBIT Packet Address Register A**

If latch bank A is active, the bits in this register indicate the last address that was used to access packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0E90 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |
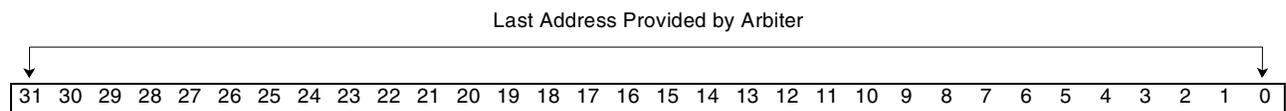
Last Address Provided by Arbiter

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain the last address provided by the arbiter to the packet memory controller. |

**7.17: ARBIT Packet Address Register B**

If latch bank B is active, the bits in this register indicate the last address that was used to access packet memory.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 0EA0 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |
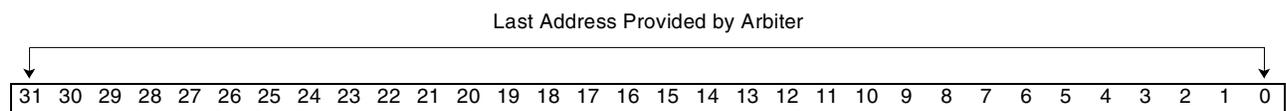
Last Address Provided by Arbiter

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain the last address provided by the arbiter to the packet memory controller. |

### 7.18: ARBIT Packet Length Register

The bits in this register indicate the last length that was used to access packet memory.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read |
| **Address** | XXXX 0E94 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 15-8 | These bits contain the length used to access packet memory through latch bank B |
| 7-0 | These bits contain the length used to access packet memory through latch bank A |
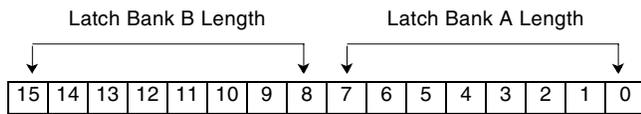
### 7.19: ARBIT Packet Lock Entity Enable Register

The value programmed in this register controls what entity, if any, has access to packet memory immediately after memory has locked. This register powers up to a value that will not allow any entity to access memory after a lock condition until the lock condition has been properly cleared.

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0EA8 |
| **Power On Value** | X'F' |
| **Restrictions** | None |

Bit Map Value

| 3 | 2 | 1 | 0 |
|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 3-0 | The value in these bits map to the following entities:<br>Value C      CHKSM<br>Value B      BCACH LO<br>Value A      POOLS LO<br>Value 9      CSKED<br>Value 8      REASM<br>Value 7      RXQUE<br>Value 6      PCORE<br>Value 5      SEGBF<br>Value 4      RAALL<br>Value 3      GPDMA<br>Value 2      DMAQS<br>Value 1      POOLS HI<br>Value 0      BCACH HI |

### 7.20: ARBIT Packet Config Register

The bits in this register control the operation of the packet memory arbiter.

| | |
|---|---|
| **Length** | 2 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0EB8 and BC |
| **Power On Value** | X'0' |
| **Restrictions** | None |

| Bit(s) | Description |
|:---:|---|
| 1 | Reserved |
| 0 | When set, this bit forces all operations to packet memory to be serialized. An operation from one entity must be entirely complete before an operation from another entity will be started. When reset, if the memory operation in process can be overlapped, a second operation will be started before the first operation is complete. |

## Entity 8: The Bus DRAM Cache Controller (BCACH)

This entity provides the caching function for data transfers on the Control Processor bus. The array is organized in four logically separate cache lines, any of which can be used for processor accesses or master/slave DMA accesses. The cache is accessible on byte boundaries on the Control Processor side; access of this entity to COMET is performed on 64-bit (word) boundaries. The address tags of each of the four 32-byte cache lines are used to compare on the requesting address to select the bank to be used to satisfy the Control Processor bus operation.

Streaming accesses of the cache use a predictive look-ahead scheme to fill the cache for read operations from packet memory. Under normal conditions, a single cache miss will be expected at the start of each DMA read operation. This cache miss will initiate a read operation from packet memory to fetch the requested data and enough additional data to fill the remainder of the cache line. If the requested data is in the last **N** bytes (**N** is programmable via the BCACH control register) of the cache line, the read operation to COMET will be extended to fill the next cache line with sequential data as well. This same programmable value is used to determine when to initiate the next sequential cache line fill operation during a DMA read operation. During non-aligned write operations to packet memory, BCACH will perform read/modify/write cycles to PAKIT.

Processor accesses operate without predictive caching. When a cache miss occurs, a COMET read operation will be initiated to fetch the 32-byte block of data that contains the requested data. The data read from COMET will be loaded into the 'Least Recently Used' cache line.

This section contains descriptions of the registers used by the Bus Cache logic.

## 8.1: BCACH Control Register

The bits in this register control the various functions provided by the cache logic. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1000 and 004 |
| **Power On Value** | X'2000 0000' |
| **Restrictions** | None |



| Bit(s) | Function | Description |
|---|---|---|
| 31 | Enable Caching Packet Memory Reads | When set, reads of packet memory will be cached. |
| 30 | Enable Caching Packet Memory Writes | When set, writes to packet memory will be cached. |
| 29 | Retry Delayed Accesses | When set, and the cache is enabled, any access of packet memory that cannot be satisfied within one cycle will be terminated by the cache with a retry indication. The accessing device is expected to allow other competing devices a chance to gain access to the bus and then retry the same operation again. |
| 28 | BCACH Diagnostic Mode | When set, diagnostic mode is enabled and reads and writes of the BCACH array from the processor are enabled. When reset, reads from the processor will return X'BADD-BADD' and writes will have no affect. Care must be taken when performing writes from the processor, if a cache line fill operation is in process and a write is performed from the processor that writes to the same address in the array as is being written from the fill operation, results are indeterminate. |
| 27 | Flush Cache Line 3 | Setting this bit will force a flush of cache line 3 if it is dirty. This bit will be reset by the hardware when the flush completes. |
| 26 | Flush Cache Line 2 | Setting this bit will force a flush of cache line 2 if it is dirty. This bit will be reset by the hardware when the flush completes. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 25 | Flush Cache Line 1 | Setting this bit will force a flush of cache line 1 if it is dirty. This bit will be reset by the hardware when the flush completes. |
| 24 | Flush Cache Line 0 | Setting this bit will force a flush of cache line 0 if it is dirty. This bit will be reset by the hardware when the flush completes. |
| 23 | Use RXQUE Advice | When set, advice from the receive queue entity will cause the cache logic to fill a line with the data from the start of the buffer that was just dequeued by the software. This should improve performance by having the receive data available when the processor accesses the buffer after the dequeue. To make best use of this feature, the code should access the receive data shortly after the dequeue to avoid the data in the cache line from becoming stale and being invalidated due to other cache functions. When reset, advice from the receive queue entity will be ignored. |
| 22 | Ack RXQUE Immediately | When reset, advice from the receive queue entity will be acknowledged immediately even if the cache is not able to perform the requested data fetch. In this case, the advice will be lost, and the cache will not fetch the data until the processor requests it again. When set, the advice from the receive queue entity will not be acknowledged until the cache has actually latched the advice information. This guarantees that the advice will be used, but may cause delays in the receive queue entities processing. |
| 21 | Enable Ping Pong Buffer Support | When reset, this bit will disable the two-line ping pong feature associated with consistent sequential cache accesses. When set, a series of sequential accesses to packet memory that would normally require more than two cache lines to be satisfied will be limited to only two cache lines, regardless of the length of the transfer. This feature is intended to improve cache performance by preventing cache lines that contain the most recently used processor data from being flushed due to a long streaming access. |
| 20 | Disable Locking on Collisions | When set, this bit will prevent detected collisions from locking up the memory control entity. |
| 19 | Enable automatic flush | When set, this bit will enable the automatic flush feature of the cache. The auto flush feature will force a flush of a cache line to be performed if a sequential write of the last 2 locations in the cache line is detected. |
| 18 | Force predictive fill/flush for non-streaming accesses | When set, this bit will force the predictive fill/flush logic to operate on all accesses of the cache and not just streaming accesses. When reset, the predictive fill logic will only be activated for streaming accesses in the cache. |
| 17-16 | Reserved | Reserved |
| 15-8 | Predictive Fill Threshold | These bits set the threshold at which a predictive fill will be initiated. If all of these bits are set to '1', a predictive fill will be initiated on the first streaming access of a cache line, regardless of which byte in the line is accessed. If this field is set to X'3F' a predictive fill will be initiated on any streaming access of bytes at offset X'2' through X'7' in the cache line. If this field is set to X'03' a predictive fill will be initiated on any streaming access of bytes at offset X'6' or X'7' in the cache line. Setting the field to all zeros will disable predictive fills. |
| 7-0 | Timed Flush Time Out Value | These bits control the time-out value used to monitor dirty cache lines for inactivity. The value loaded into these eight bits is the number of 240ns ticks that can occur without any activity in a dirty cache line before the cache logic will force a flush of the line to main memory. Setting these bits to all zeros disables the timed flush feature. |

**IBM**

## 8.2: BCACH Status Register

The bits in this register reflect the current status of the cache. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1008 and 00C |
| **Power On Value** | X'00' |
| **Restrictions** | None |



| Bit(s) | Function | Description |
|---|---|---|
| 7 | POOLS invalidation of dirty lines | When set, this bit indicates that POOLS requested that the cache logic invalidate a line that was dirty. This is usually an indication that a buffer was freed by the software before data written out to the buffer had been flushed to memory. This may or may not be an error condition |
| 6 | Write Hit on Multiple Lines | When set, the cache logic has detected a write hit to multiple lines. This indicates an internal logic error in the cache. |
| 5 | Read Hit on Multiple Lines | When set, the cache logic has detected a read hit to multiple lines. This indicates an internal logic error in the cache. |
| 4 | Negative Ack from VIMEM | When set, the cache logic has detected a negative acknowledgment from the virtual memory logic entity. This indicates that a virtual buffer boundary was crossed and a new real buffer was needed to map the requested address space into, but no real buffer was available. In addition to setting this status bit, the cache logic will write the pattern X'zzzzzBAD' into the header of the packet at offset X'C' where zzzzz is the offset of the failing write into the packet. |
| 3 | Collision on Cache Line 3 | When set, the cache logic has detected a collision in cache line 3. This is a situation where another entity in the chip was accessing an area of memory that was contained in one of the cache lines that was dirty. Further information for problem diagnosis is latched in the memory controller logic when this condition is detected. |
| 2 | Collision on Cache Line 2 | When set, the cache logic has detected a collision in cache line 2 |
| 1 | Collision on Cache Line 1 | When set, the cache logic has detected a collision in cache line 1 |
| 0 | Collision on Cache Line 0 | When set, the cache logic has detected a collision in cache line 0 |

### 8.3: BCACH Interrupt Enable Register

The low eight bits in this register allow the user to selectively determine which bits in the BCACH status register will cause processor interrupts. A zero in a bit position masks interrupts from the corresponding bit location in the BCACH status register. A one in a bit position allows interrupts for the corresponding bit in the BCACH status register. The high eight bits in this register allow the user to selectively determine which bits in the BCACH status register will lock the cache. A one in any bit position will force the cache to lock if the corresponding bit is set in the BCACH status register. If the cache locks, all status regarding the cache lines is maintained until the cache enable bits in the control register are turned off.

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1010 and 014 |
| **Power On Value** | X'FFFF' |
| **Restrictions** | None |

### 8.4: BCACH High Priority Timer Value

The contents of this register define the number of 30ns cycles that will pass from the time that a valid PCI bus request is raised to BCACH until BCACH will raise it's high priority request to the memory controllers. A value of zero in this register will disable this function completely.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1040 |
| **Power On Value** | X'40' |
| **Restrictions** | None |

Number of 30ns Cycles
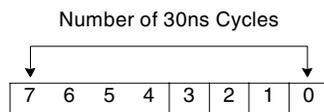
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 7-0 | Determines the number of 30ns cycles before a high priority request |

**8.5: BCACH Line Tag Registers**

These registers are useful only in diagnostic testing of the cache logic. Each register will contain the tag value for the data contained in that particular cache line.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read Only | |
| **Address** | Tag Number 0 | XXXX 1080 |
| | Tag Number 1 | XXXX 10A0 |
| | Tag Number 2 | XXXX 10C0 |
| | Tag Number 3 | XXXX 10C0 |
| **Power on Value** | X'0000 0000' | |
| **Restrictions** | None | |



| Bit(s) | Description |
|---|---|
| 31-05 | These bits contain the address of the 32-byte line of memory contained in the cache line. |
| 04-03 | In an attempt to provide the fastest possible access to data in memory, the 8 byte word in memory that contains the requested read data is accessed first and all other entries in the cache line are filled by wrapping back to the beginning of the cache line if required. These two bits contain the starting address for the modulo read fill operation. They will also contain the least significant address bits when a cache line is initially written to. |
| 02-00 | Will always be returned as zeros. |

### 8.6: BCACH Line Valid Bytes Register

These registers are useful only in diagnostic testing of the cache logic. Each register will contain a bit significant flag indicating which bytes in the 32-byte cache line are valid. All of these bits will be active after a cache line fill operation has happened, but any combination of these bits can be valid after the processor has performed a write operation to memory.

**Length**          32 bits

**Type**            Read Only

**Address**         Tag Number 0      XXXX 1084

                    Tag Number 1      XXXX 10A4

                    Tag Number 2      XXXX 10C4

                    Tag Number 3      XXXX 10E4

**Power on Value**  X'0000 0000'

**Restrictions**    None

Cache Line Fill Operation Set/Reset

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

| Bit(s) | Description |
|---|---|
| 31-00 | Each bit indicates if the associated byte in the cache line contains valid data or not. If the bit is set, the cache line contains valid data and a fetch from main storage is not required to fulfill a request for a read from this location. If the bit is reset, a read of the associated location will require a cache line fill operation before the request can complete. |

## 8.7: BCACH Line Status Register

These registers are useful only in diagnostic testing of the cache logic. Each register will contain a bit significant flag indicating the current status of the associated cache line.

**Length**        32 bits

**Type**        Read Only

**Address**        Tag Number 0        XXXX 1088

                    Tag Number 1        XXXX 10A8

                    Tag Number 2        XXXX 10C8

                    Tag Number 3        XXXX 10E8

**Power on Value**        X'0000 0000'

**Restrictions**        None



| Bit(s) | Function | Description |
|--------|----------|-------------|
| 31 | Valid Tag | When set, this indicates that the associated tag register contains a valid tag. |
| 30-12 | Reserved | Reserved |
| 11 | Loaded by PCI Read | When set, this bit indicates that the associated tag register was loaded due to a read request from the PCI bus |
| 10 | Loaded by Predictive Fill | When set, this bit indicates that the associated tag register was loaded due to a predictive fill request |
| 09 | Loaded by RXQUE Advice | When set, this bit indicates that the associated tag register was loaded due to advice from the receive queue entity |
| 08 | Loaded by PCI Write | When set, this bit indicates that the associated tag register was loaded due to a write request from the PCI bus |
| 07-06 | Reserved | Reserved |
| 05-04 | LRU bits | These bits indicate the cache lines current position with respect to the least recently used algorithm. A value of 0 indicates it is the most recently used while a value of 3 indicates the least recently used. |
| 03-00 | Dirty bits | These bits, when set, indicate that the associated 8 byte word of the cache line is dirty. This information is used on cache line flushes, to lower memory utilization, by eliminating non-dirty word flushes from the cache line flush operation. For example if these bits contain a X'1', only the 8 byte word at offset zero in the cache line is dirty, so the flush operation will only write this one word to memory, saving 3 memory access cycles. If these bits contain a X'C', only the two 8 byte words starting at offset X'10' in the cache line are dirty. |

**8.8: BCACH Cache Line Array**

This array is divided into four 32-byte buffers used as cache lines 0,1,2 and 3. The four cache lines start at the following offsets into the array:

| | |
|---|---|
| **Line 0** | Offset X'00' |
| **Line 1** | Offset X'20' |
| **Line 2** | Offset X'40' |
| **Line 3** | Offset X'60' |

| | |
|---|---|
| **Length** | 16 Words X 64 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1100 - 17F |
| **Restrictions** | This array can only be accessed when the diagnostic mode bit in the control register is set. |

# Entity 9: Buffer Pool Management (POOLS)

POOLS acts as a memory manager for the IBM2520L8767. Memory buffers are checked out and checked in via two operations (primitives) supported by POOLS: the get pointer primitive and the free pointer primitive. These primitives can be performed explicitly by accessing specified addresses within the POOLS entity, and they may also be done by hardware. CSKED can free a buffer upon transmission if specified by the corresponding packet header (see *ECC Syndrome Bits* on page 151), and RAALL gets buffers to store received data. In addition POOLS contains mechanisms to control resource utilization and supports a Real Memory Mode and a Virtual Memory Mode.

## Basic Operation in Real Memory Mode

If memory is viewed as a series of buffers, POOLS maintains a circular list of available buffers. There are pointers (the head and tail) to the start and the end of the list. When a get pointer primitive is executed, the buffer at the head of the list is checked out, the head pointer is advanced and the correct resource group(s) is debited. When a free pointer primitive is executed, the freed buffer is checked in at the end of the list, the tail pointer is advanced and the correct resource group credited.

## Basic Operation in Virtual Memory Mode

With the addition of virtual memory, POOLS must maintain five sets of head and tail pointers, thresholds, and active counts; one for the virtual buffers themselves and the rest for the four regions of real buffers that constitute the virtual buffers. In this case the base virtual address is the item returned from a get pointer operation and returned during a free pointer operation. When the get buffer primitive is executed POOLS creates an active buffer map (page table) for the virtual address. As the virtual address is used and buffer(page) boundaries are crossed VIMEM will request buffers from POOLS when a buffer(page) fault occurs. VIMEM will then place the buffer index in the buffer map. When the virtual buffer is no longer needed and a free pointer primitive is issued with the starting virtual address, POOLS takes the contents of the buffer map and frees the resources that were assigned to the buffer map.

## Resource Controls

POOLS adds another layer of service by creating "POOLs" of buffers (currently a maximum of 16 POOLs). For each POOL, a maximum number of allowable buffers may be specified. The intent is to make it possible for several applications to use the IBM2520L8767 at once without one or more applications starving the remaining applications for memory buffers. A particular POOL's buffers are divided into "guaranteed" and "common" buffers. All the guaranteed buffers are considered to be dedicated to their respective POOL and are therefore not available for general use. The common buffers are all the memory buffers remaining after the guaranteed buffers are subtracted from the total buffers. To maintain the buffer limits on each POOL, every POOL has a guaranteed threshold, total threshold, and an active count. When a request is made for a buffer from a particular POOL, the guaranteed threshold is first checked. If the active count of the POOL is less than the guaranteed threshold, the buffer is provided. If the guaranteed threshold has been reached, then the total threshold is checked. If the active count is equal to the total threshold, no buffer is provided. If the active count is less than the total threshold, and a common buffer is available, a buffer is provided. If there are no common buffers available, a buffer cannot be provided and a null index is returned. To determine if a common buffer is available a count is maintained for each size of buffer.

**Virtual Memory Overview**

Each virtual buffer consists of a number of real buffers. For each virtual buffer there is a buffer map that defines the size and number of real buffers that may be allocated to the virtual buffer. Each map is built from a common template (the VIMEM Virtual Buffer Segment Size Register) that associates 1 to n buffer indexes in the map to a real buffer in one of the four real buffer regions defined in VIMEM. In VIMEM, the Buffer Map Base Address Register defines the size of the map and therefore also the number of buffer indexes in the virtual buffer map. Each eight-byte entry of the map contains the POOL ID of the POOL to which the buffer is allocated plus space for three real buffer segment indexes. This implies the smallest map yields a virtual buffer of one to four real buffer segments (three real buffer segments plus the implicit real buffer that all virtual buffers are allocated). The biggest map defines a virtual buffer of 1-16 real buffer segments (15 plus the implicit).

The intention of this structure is to allow the user to customize the value in the Virtual Buffer Segment Size Register to utilize memory in an efficient manner relative to network data traffic. For example, if network traffic contained 50% packets of < 512 bytes, 35% packets of < 1K bytes, and the rest was < 5K bytes, the user could set up virtual memory to use three real segments of 512 bytes, 512 bytes, and 4K bytes. The incoming data would neatly fit into the segments and minimize wasted memory.

POOLS and VIMEM maintain the maps for the virtual buffers. On a write that crosses a real buffer boundary into an as yet an unresolved region of a virtual buffer, a page fault occurs. When a page fault occurs, POOLS determines whether or not a real buffer can be assigned. If it can be assigned, the index of the real buffer relative to the base address of the particular buffer size is placed by VIMEM into the buffer map. The first buffer is implicitly associated with the virtual memory address for a particular virtual buffer and enough real memory must be available to support the first real buffer of each virtual buffer at initialization time. There is not necessarily enough real storage for all the possible real buffers associated with a virtual buffer.

## Virtual Address Buffer Map

**Virtual Address Buffer Map**

| 16 Bit | 16 Bit | 16 Bit | 16 Bit | 16 Bit | 16 Bit | 16 Bit | | 16 Bit |

| 2 Bit | 2 Bit | 2 Bit | 2 Bit | 2 Bit | | 2 Bit |

**Buffer Limit Segment Register (32 bits)**

Note: The Buffer Index of the first buffer and
the Virtual Address Index are the same.
Also note that they are right justified.

| Buffer Index of the first buffer in the Virtual Address (16 bits max) | Number of bits for largest Packet Size (17 bits max) |

**32 Bit Virtual Address**

All real buffers of a particular size are stored in a contiguous region of memory. The buffer index, in conjunction with the base address for this real buffer size, will point to a particular real buffer. The implicit buffers are also stored in a data structure of this type.

## Buffer/Virtual Memory Allocation Structure in Memory

The single box represents the storage for the whole buffer, whatever its size.

| |
|---|
| Buffer Size One Base Index Zero |
| Buffer Size One Index One |
| Buffer Size One Index Two |
| ...... |
| Buffer Size One Index N |
| Skip to next buffer size bound |
| Buffer Size Two Base Index Zero |
| Buffer Size Two Index One |
| Buffer Size Two Index Two |
| ...... |
| Buffer Size Two Index N |
| Skip to next buffer size bound |
| Buffer Size Three Base Index Zero |
| Buffer Size Three Index One |
| Buffer Size Three Index Two |
| ...... |
| Buffer Size Three Index H |
| Skip to next buffer size bound |
| Buffer Size Four Base Index Zero |
| Buffer Size Four Index One |
| Buffer Size Four Index Two |
| ...... |
| Buffer Size Four Index N |
| Skip to implicit buffer base address |
| Implicit Buffer Index Zero |
| Implicit Buffer Index One |
| Implicit Buffer Index Two |
| ...... |
| Implicit Buffer Index N |

These indexes are determined by the virtual buffer's virtual address

The following example illustrates these concepts.

## Virtual Address Buffer Map
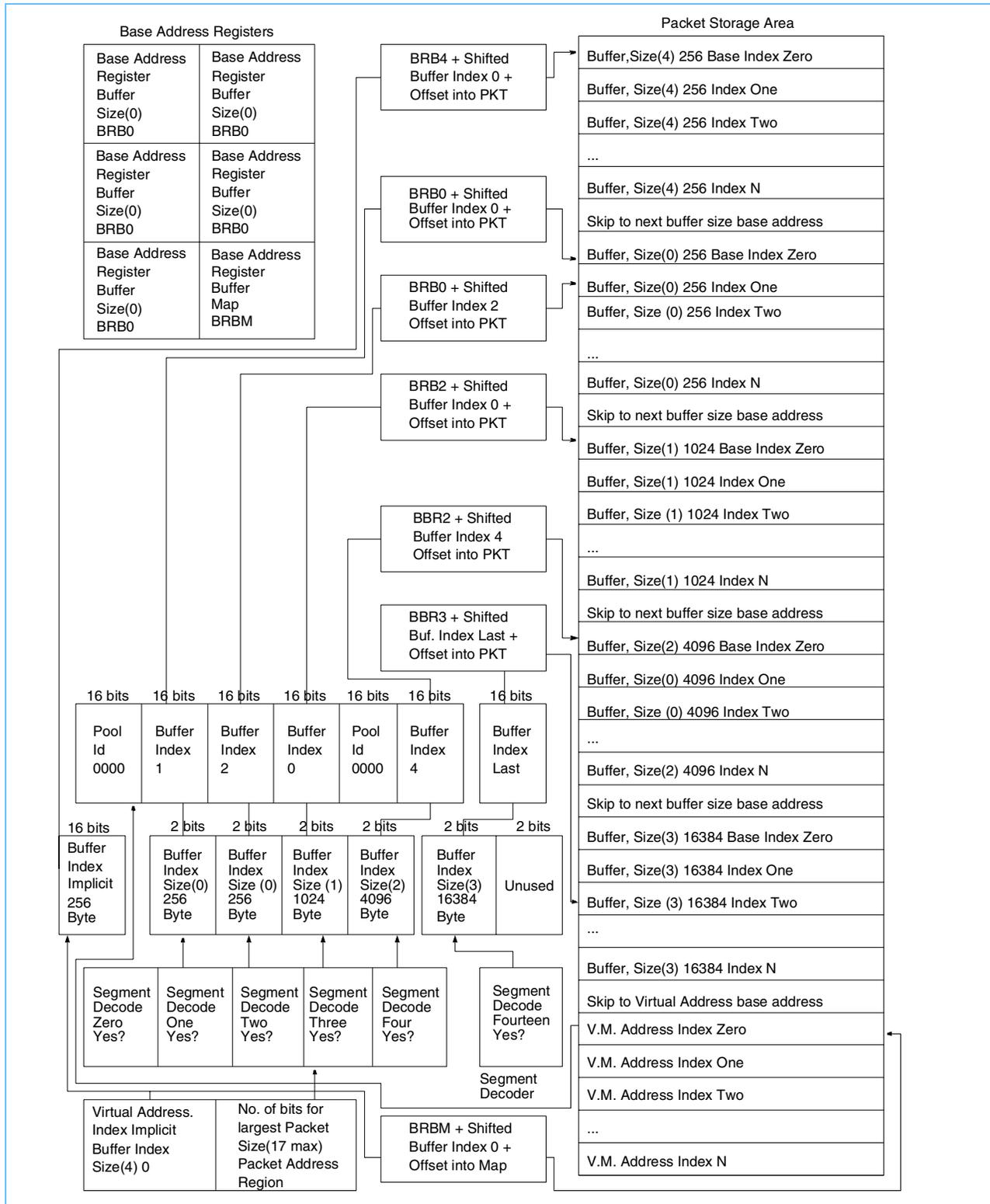
The lower seventeen bits of the virtual address are used in conjunction with the segment template in the VIMEM Virtual Buffer Segment Size Register to determine from which portion of the buffer map the buffer index is retrieved. Once the buffer index is retrieved, it is combined with the appropriate base address for that particular buffer size. The offset into the buffer is then added to get the real 32-bit address that is used in physical memory.

POOLs uses the data structures above to manage packet memory resources. Each LCD is associated with a particular POOL and multiple different LCD's may be associated with that same POOL. Within a POOL, there are five different resource categories and two variables to go with each resource.

## Resources and Variables Example

| Resource Type POOL 0000 | Virtual Memory Addresses | Buffer Type One | Buffer Type Two | Buffer Type Three | Buffer Type Four |
|---|---|---|---|---|---|
| **Guaranteed No.** | 100 | 200 | 50 | 10 | 0 |
| **Total No.** | 150 | 300 | 100 | 5 | 10 |

### 9.1: POOLS Get Pointer Primitive

The POOLS Get Pointer Primitive returns a pointer to the requester. The request to the virtual packet/buffer size 4 address will always return a memory address. If in virtual mode, the address will be virtual. Requests made for buffer sizes 0 to 3 will not return an address but rather a buffer index in bits 15-0. The real address associated with this index can be generated by shifting the index by the buffer size (e.g. six bit positions for a 64-byte buffer) and adding the result to the base address for this size buffer. Access to buffer sizes 0 - 3 is not permitted in operational mode. The address of the primitive also selects the POOL ID. The POOL ID is contained in address bits 5-2; the POOL ID selects which POOL will be charged for the pointer. The buffer size is selected with address bits 8-6.

If there are no more pointers available in the specified POOL, a null pointer is returned. The active pointer count for that POOL is incremented if a non-null pointer is returned. If the guaranteed threshold has been exceeded and a buffer from the common POOL is returned, the common POOLs count for that size is decremented by 1.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read Only | |
| **Address** | Buffer Size 0 | XXXX 3200 |
| | Buffer Size 1 | XXXX 3240 |
| | Buffer Size 2 | XXXX 3280 |
| | Buffer Size 3 | XXXX 32C0 |
| | Virtual Packets / Buffer Size 4 | XXXX 3300 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations this register is to be used as a read only register. Writes to this address will be ignored. | |

### 9.2: POOLS Free Pointer Primitive

The POOLS Free Pointer Primitive returns the pointer to the proper free list. If it is a virtual memory address, the Virtual Memory Buffer Map is traversed to free the indexes associated with the virtual memory address. In the case where it is a real memory buffer, the single index is freed. This primitive uses address mapping to select the size of the object to be freed. The size is contained in address bits 4-2. During normal operation only frees to buffer size four are relevant. During initialization mode, buffer sizes 0 - 3 can be used to load indexes. The indexes are loaded into bits 31-16.

In normal operations it is not necessary to read this "register".

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Write Only | |
| **Address** | Buffer Size 0 | XXXX 3350 |
| | Buffer Size 1 | XXXX 3354 |
| | Buffer Size 2 | XXXX 3358 |
| | Buffer Size 3 | XXXX 335C |
| | Virtual Packets / Buffer Size 4 | XXXX 3360 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | During normal operations this register is to be used as a Write only register. Reads from this address will return zeros. | |

**9.3: POOLS Common Pools Count Registers**

The POOLS Common POOLS Count Registers indicates the number of pointers in the particular common POOL. The bits are a 16-bit count. The GTPTRs that exceed the guaranteed allocation decrement this count by one assuming that the count is non-zero. When the count is zero, the GTBUF will fail. The FRPTRs that operate beyond the guaranteed threshold for a particular client and free the pointer(s) increment this count by one. The microcode should initialize these registers to the value of the respective common POOL that it desires to have.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3000 |
| | Buffer Size 1 | XXXX 3004 |
| | Buffer Size 2 | XXXX 3008 |
| | Buffer Size 3 | XXXX 300C |
| | Virtual Packets / Buffer Size 4 | XXXX 3010 |
| **Power on Value** | X'0000' | |
| **Restrictions** | During normal operations these registers are to be used as a read only. Writing to these registers during operation could create a data loss situation. This register should be set up by the microcode at initialization time. | |

**9.4: POOLS Client Thresholds Array**

The POOLS Client Thresholds Array holds the guaranteed and total threshold values for the 16 POOLs and the four(five) pointer sizes. When a GTPTR primitive is processed, the values in this array are used to determine if a primitive can return a pointer. The active count from the Active Packet Count Array is used with these registers to determine if a threshold has been exceeded. If the guaranteed threshold has been exceeded and the total not exceeded and there is a common pointer available, then the common count will be incremented. If there are no common buffers available or the request will cause the total threshold to be exceeded, the request will be rejected. During a FRPTR primitive processing, the pointer is returned to the free list and these thresholds are used to determine if a common count should be credited. This array contains the guaranteed and total thresholds for the managed POOLs.

| | |
|---|---|
| **Length** | 32 bits X 16 Words |
| **Type** | Read/Write |
| **Address** | Buffer Size 0      XXXX 3400 |
| | Buffer Size 1      XXXX 3440 |
| | Buffer Size 2      XXXX 3480 |
| | Buffer Size 3      XXXX 34C0 |
| | Virtual Packets / Buffer Size 4      XXXX 3500 |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

```
        Guaranteed Threshold                              Total Threshold
  ┌──────────────────────────────────────┐   ┌──────────────────────────────────────┐
  │                                       ▼   ▼                                       ▼
┌────┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┐
│ 31 │30│29│28│27│26│25│24│23│22│21│20│19│18│17│16│15│14│13│12│11│10│ 9│ 8│ 7│ 6│ 5│ 4│ 3│ 2│ 1│ 0│
└────┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┘
```
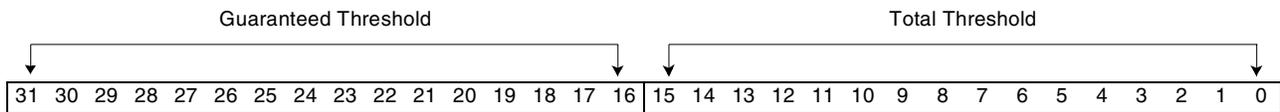
| Bit(s) | Description |
|---|---|
| 31-16 | Guaranteed Threshold |
| 15-0 | Total Threshold. |

**9.5: POOLS User Threshold and Client Active Packet Count Array**

The POOLS User Threshold and Client Active Packet Count Array holds the user thresholds and active pointer counts for each of the 16 managed POOLs and four (five) pointer sizes.

| | | |
|---|---|---|
| **Length** | 32 bits X 16 Words | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3600 |
| | Buffer Size 1 | XXXX 3640 |
| | Buffer Size 2 | XXXX 3680 |
| | Buffer Size 3 | XXXX 36C0 |
| | Virtual Packets / Buffer Size 4 | XXXX 3700 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | None | |

This array contains the user thresholds and Active buffer counts for the managed POOLs and pointer sizes.
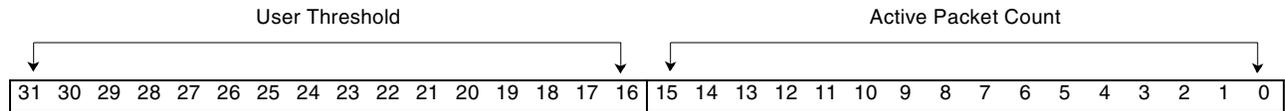
| User Threshold | Active Packet Count |
|---|---|
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |

| Bit(s) | Description |
|---|---|
| 31-16 | User Threshold |
| 15-0 | Active Packet Count |

When a GTPTR primitive is processed, the active count is retrieved and compared with the threshold counts. If it falls within bounds and a pointer is available, the active count will be incremented by one reflecting the additional buffer charged to that queue.

When a FRPTR primitive is processed, the active count is retrieved, and, when the pointer is returned to the free list, the active buffer count is decremented by one.

The user threshold may be used to check on resource utilization as opposed to resource allocation. The Guaranteed and Total Thresholds are used when allocating resources to make decisions. The User Threshold is not used to govern resource allocation directly. One such use is for high water mark indication. When a FRPTR primitive is processed or a GTPTR is processed The active packet count is compared to the user threshold. If the event interface is enabled and a boundary condition is crossed an event is issued to the event interface.

**9.6: POOLS Pointer Queues DRAM Head Pointer Offset Address Register**

The POOLS Pointer Queues DRAM Head Pointer Offset Address Register indicates the address in DRAM where the head of the queue starts. NOTE however, that this address is only relative to the DRAM portion of the queue. Unless the head of the queue portion of the cache is locked out and needs two frames, the actual head of the queue is in the cache.

These 19 bits on write represent the offset to the address in DRAM of the head of the queue relative to the DRAM base address. On a read the address in DRAM of the pointer is returned. This pointer is adjusted every time a cache frame boundary is crossed and a cache update cycle is completed to write through the additional queue elements. Since each memory reference contains four indices, this allows for 128K index locations possible in the queue.

| | | |
|---|---|---|
| **Length** | 32 bits Read/19 bits Write | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3014 |
| | Buffer Size 1 | XXXX 3018 |
| | Buffer Size 2 | XXXX 301C |
| | Buffer Size 3 | XXXX 3020 |
| | Virtual Packets / Buffer Size 4 | XXXX 3024 |
| **Power on Value** | Buffer Size 0 | X'00 01 00 00' |
| | Buffer Size 1 | X'00 02 00 00' |
| | Buffer Size 2 | X'00 02 40 00' |
| | Buffer Size 3 | X'00 02 60 00' |
| | Virtual Packets / Buffer Size 4 | X'00 02 70 00' |
| **Restrictions** | During normal operations this register is to be used as a read only register. This register defaults to zero at initialization. It is assumed that the queues start on a maximum size queue boundary. These registers should be setup at initialization time. This register is cleared when the POOLS Pointer Queues DRAM Lower Bound Address Register is written to. | |

**9.7: POOLS Pointer Queues DRAM Tail Pointer Offset Address Register**

The POOLS Pointer Queues DRAM Tail Pointer Offset Address Register indicates the offset address in DRAM where the tail of the queue starts. NOTE however that this address is only relative to the DRAM portion of the queue. Unless a 'no cache frames to be written through' state is in effect the actual tail of the queue is in the cache.

These 19 bits on write represent the offset to the address in DRAM of the tail of the queue relative to the DRAM base address. On a read the address in DRAM of the pointer is returned. This pointer is adjusted every time a cache frame boundary is crossed and a cache update cycle is completed to write through the additional queue elements. Since each memory reference contains four indices this allows for 128K index locations possible in the queue.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3028 |
| | Buffer Size 1 | XXXX 302C |
| | Buffer Size 2 | XXXX 3030 |
| | Buffer Size 3 | XXXX 3034 |
| | Virtual Packets / Buffer Size 4 | XXXX 3038 |
| **Power on Value** | Buffer Size 0 | X'00 01 C0 00' |
| | Buffer Size 1 | X'00 02 00 00' |
| | Buffer Size 2 | X'00 02 40 00' |
| | Buffer Size 3 | X'00 02 60 00' |
| | Virtual Packets / Buffer Size 4 | X'00 02 70 00' |
| **Restrictions** | During normal operations this register is to be used as a read only register. This register defaults to zero at initialization. It is assumed that the queues start on the maximum size queue boundary. These registers should be setup at initialization time. This register is cleared when the POOLS Pointer Queues DRAM Lower Bound Address Register is written to. | |

### 9.8: POOLS Pointer Queues DRAM Lower Bound Address Register

The POOLS Pointer Queues DRAM Lower Bound Address Register indicates the address in DRAM where the queue data structure is initially started. When the queue reaches the maximum address allowed for in the upper bound register it will wrap back around to the address specified in this register. This implements the queue in a circular buffer.

These thirty-two bits represent the address in DRAM where the queue will begin at and eventually wrap to. At initialization this register and the POOLS Pointer Queues DRAM Tail Pointer Offset Address Register and POOLS Pointer Queues DRAM Head Pointer Offset Address Register must be equal.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 303C |
| | Buffer Size 1 | XXXX 3040 |
| | Buffer Size 2 | XXXX 3044 |
| | Buffer Size 3 | XXXX 3048 |
| | Virtual Packets / Buffer Size 4 | XXXX 304C |
| **Power on Value** | Buffer Size 0 | X'00 01 C0 00' |
| | Buffer Size 1 | X'00 02 00 00' |
| | Buffer Size 2 | X'00 02 40 00' |
| | Buffer Size 3 | X'00 02 60 00' |
| | Virtual Packets / Buffer Size 4 | X'00 02 70 00' |
| **Restrictions** | During normal operations this register is to be used as a read only register. This register should be setup at initialization time. The size of the DRAM queue storage which is formed with the lower and upper bounds is constrained in its size. It can be written when the diagnostic mode bit is set, otherwise the write is ignored. Note if the maximum queue length exceeds the space available in the circular buffer, data corruption will occur when the actual queue length exceeds the maximum queue space available. | |

### 9.9: POOLS Pointer Queues DRAM Upper Bound Register

The POOLS Pointer Queues DRAM Upper Bound Register indicates the max queue length in DRAM of the queue data structure. When the queue reaches this address, it wraps back to the address specified by the lower bound register. This implements the queue in a circular buffer. This upper bound is to be provided as an encoded field. The encoded field represents the number of 8-byte addresses that can be contained by the queue. These four bits represent the encoded maximum queue length in DRAM where, when matched, trigger the queue to wrap back to the address contained in the DRAM Lower Bound Address register.

| | | |
|---|---|---|
| **Length** | 4 bits | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3050 |
| | Buffer Size 1 | XXXX 3054 |
| | Buffer Size 2 | XXXX 3058 |
| | Buffer Size 3 | XXXX 305C |
| | Virtual Packets / Buffer Size 4 | XXXX 3060 |
| **Power on Value** | Buffer Size 0 | X'B' |
| | Buffer Size 1 | X'A' |
| | Buffer Size 2 | X'9' |
| | Buffer Size 3 | X'9' |
| | Virtual Packets / Buffer Size 4 | X'B' |
| **Restrictions** | During normal operations this register is to be used as a read only register. This register should be setup at initialization time. The size of the DRAM queue storage which is formed with the lower and upper bounds is constrained in its size. It can be written when the diagnostic mode bit is set, otherwise the write is ignored. Note if the maximum queue length exceeds the space available in the circular buffer, data corruption will occur when the actual queue length exceeds the maximum queue space available. | |

**Pointer Queues DRAM Upper Bound Address (Page 1 of 2)**

| Encoded Value | Number of 32 Bit Words | Number of Indexes |
|---|---|---|
| X'0' | 8 | 16 |
| X'1' | 16 | 32 |
| X'2' | 32 | 64 |
| X'3' | 64 | 128 |
| X'4' | 128 | 256 |
| X'5' | 256 | 512 |
| X'6' | 512 | 1024 |
| X'7' | 1024 | 2048 |
| X'8' | 2048 | 4096 |
| X'9' | 4096 | 8192 |
| X'A' | 8192 | 16384 |

## Pointer Queues DRAM Upper Bound Address (Page 2 of 2)

| Encoded Value | Number of 32 Bit Words | Number of Indexes |
|---|---|---|
| X'B' | 16384 | 32768 |
| X'C' | 32768 | 65536 |
| X'D' | 65536 | 131072 |
| X'E' | 65536 | 131072 |
| X'F' | 65536 | 131072 |

### 9.10: POOLS Pointer Queues Length Registers

The POOLS Pointer Queues Length Registers indicates the length of the queue. The bits are a 16-bit count. A primitive that adds to the queue increments this counter and primitives that remove items from the queue decrement this counter.

Restrictions: During normal operations this register is to be used as a read only register. It can be written when the diagnostic mode bit is set, otherwise the write is ignored. This register is cleared when the POOLS Pointer Queues DRAM Lower Bound Address Register is written to.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Buffer Size 0 | XXXX 3064 |
| | Buffer Size 1 | XXXX 3068 |
| | Buffer Size 2 | XXXX 306C |
| | Buffer Size 3 | XXXX 3070 |
| | Virtual Packets / Buffer Size 4 | XXXX 3074 |
| **Power on Value** | X'00 00' | |
| **Restrictions** | During normal operations this register is to be used as a read only register. This register should be setup at initialization time. The size of the DRAM queue storage which is formed with the lower and upper bounds is constrained in its size. It can be written when the diagnostic mode bit is set, otherwise the write is ignored. Note if the maximum queue length exceeds the space available in the circular buffer, data corruption will occur when the actual queue length exceeds the maximum queue space available. | |

### 9.11: POOLS Interrupt Enable Register

This register is used to enable bits from the POOLS Status Register and potentially generate interrupts to the control processor. When both a bit in this register and the corresponding bit(s) in the POOLS Status Register are set, the POOLS interrupt to PCINT will be enabled. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *POOLS Status Register on page 215* for the bit descriptions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3078 and 07C |
| **Power On Value** | X'00 03 F8 00' |
| **Restrictions** | None |

**9.12: POOLS Event Enables**

This register is used to enable an event based on bits from the corresponding primitive transaction. If the bits are set in the enable and a transaction occurs that matches the event, an event will be sent to the RXQUE. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Clear/Set |
| **Address** | GTD Event Enables   XXXX 3A00 and A04 |
| | Total Event Enables   XXXX 3A08 and A0C |
| | User Event Enables   XXXX 3A10 and A14 |
| **Power on Value** | X'0000' |
| **Restrictions** | None |

**9.13: POOLS Event Hysteresis Register**

The POOLS Event Hysteresis Register provide the capability for hysteresis on threshold checking.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3A18 |
| **Power on Reset value** | X'0000' |
| **Restrictions** | None |

Hysteresis Value
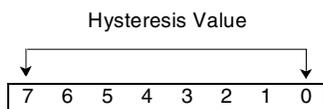
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Function | Description |
|---|---|---|
| 7-0 | Hysteresis Value | When a free occurs the value in this register will be added to the next Active Packet Count. This value will be then tested against the threshold value. If it is Equal to the threshold an event if events are enabled and the event associated with this transaction is enabled will be issued. |

### 9.14: POOLS Event Data Register

The POOLS Event Data Register provides the data that was sent on the last event.

**Length**            32 bits

**Type**              Read

**Address**           XXXX 3A1C

**Power on Reset value**   X'0000003E'

**Restrictions**      None

| Bit position | Field |
|---|---|
| 31 | Free Processed GTD Threshold Match with Hysteresis |
| 30 | Free Processed Total Threshold Match with Hysteresis |
| 29 | Free Processed User Threshold Match with Hysteresis |
| 28 | Get Processed GTD Threshold Match |
| 27 | Get Processed Total Threshold Match |
| 26 | Get Processed User Threshold Match |
| 25-20 | Reserved |
| 19-16 | POOL ID of the Event |
| 15-11 | Reserved |
| 10-8 | Buffer size of the Event |
| 7-6 | Reserved |
| 5-0 | Event Source ID |

| Bit(s) | Function | Description |
|---|---|---|
| 31 | Free Processed GTD Threshold Match with Hysteresis | This event occurs when a free is processed and the threshold is matched. The threshold is modified by the value in the hysteresis register. The event is issued when the actual value of the Active Packet count plus the hysteresis equals the threshold. |
| 30 | Free Processed Total Threshold Match with Hysteresis | This event occurs when a free is processed and the threshold is matched. The threshold is modified by the value in the hysteresis register. The event is issued when the actual value of the Active Packet count plus the hysteresis equals the threshold. |
| 29 | Free Processed User Threshold Match with Hysteresis | This event occurs when a free is processed and the threshold is matched. The threshold is modified by the value in the hysteresis register. The event is issued when the actual value of the Active Packet count plus the hysteresis equals the threshold. |
| 28 | Get Processed GTD Threshold Match | This event occurs when a get is processed and the threshold is matched. The event is issued when the new Active Packet count equals the threshold. |
| 27 | Get Processed Total Threshold Match | This event occurs when a get is processed and the threshold is matched. The event is issued when the new Active Packet count equals the threshold. |
| 26 | Get Processed User Threshold Match | This event occurs when a get is processed and the threshold is matched. The event is issued when the new Active Packet count equals the threshold. |
| 25-20 | Reserved | Reserved |

| Bit(s) | Function | Description |
|---|---|---|
| 19-16 | POOL ID of the Event | This indicates which POOL is associated with this event. |
| 15-11 | Reserved | Reserved |
| 10-8 | Buffer size of the Event | This indicates which size is associated with this event. |
| 7-6 | Reserved | Reserved |
| 5-0 | Event Source ID | This indicates that POOLS is associated with this event. |

### 9.15: POOLS Status Register

The POOLS Status Register provides status information about pools operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

**Length**          32 bits

**Type**            Clear/Set

**Address**         XXXX 3080 and 084

**Power on Reset value**    X'00 00 00 00'

**Restrictions**    During normal operations, if a status bit is cleared, it will be reset if the condition that is causing it is still present.

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

Reserved: bits 31-21.
Bit 20: POOLs Event Issued
Bit 19: Packet Memory Access Timer Expired
Bit 18: Control Memory Access Timer Expired
Bit 17: POOLs Locked
Bit 16: Initialization Error Detected
Bit 15: Unused Buffer Freed Error
Bit 14: Bad Map Detected
Bit 13: Active Count Error Detected
Bit 12: Bad Index Detected
Bit 11: Pointer
Bit 10: Common Buffers Size 4 Exhausted
Bit 9: Common Buffers Size 3 Exhausted
Bit 8: Common Buffers Size 2 Exhausted
Bit 7: Common Buffers Size 1 Exhausted
Bit 6: Common Buffers Size 0 Exhausted
Bit 5: Buffer Size 4 Threshold Crossed
Bit 4: Buffer Size 3 Threshold Crossed
Bit 3: Buffer Size 2 Threshold Crossed
Bit 2: Buffer Size 1 Threshold Crossed
Bit 1: Buffer Size 0 Threshold Crossed
Bit 0: Get Pointer Primitive Failed

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 31-21 | Reserved | Reserved |
| 20 | POOLs Event Issued | This bit is set when a POOLs event is issued. |
| 19 | Packet Memory Access Timer Expired | This bit is set when the Packet Memory access timer hits the Packet Memory access threshold and the control bit in the control register is set to enable this function. |
| 18 | Control Memory Access Timer Expired | This bit is set when the Control Memory access timer hits the Control Memory access threshold and the control bit in the control register is set to enable this function. |
| 17 | POOLs Locked | This bit is set when a lock enable bit is set and the corresponding status bit is set. This causes all state machines to be held in idle once this bit is set. It is the functional equivalent to POOLS Control Register bit 0. |
| 16 | Initialization Error Detected | This bit is set when too many indexes are freed to a queue. |
| 15 | Unused Buffer Freed Error | This bit is set when a previously freed buffer is detected during a free operation. This typically would occur when the buffer was freed two or more times. |
| 14 | Bad Map Detected | This bit is set when a bad map is detected during a free operation. |
| 13 | Active Count Error Detected | This bit is set when an active packet count is decremented from zero to X'FFFF'. This is most likely the result of a subtle map corruption where a POOL ID has been changed. |
| 12 | Bad Index Detected | This bit is set when an Index Threshold is crossed. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 11 | Free Pointer Primitive Null Detected/Max Pointer Queue Length Exceeded. | This bit is set as a result of one of two detectable errors:<br>    Null index detected within free address.<br>    Total allowed storage for a particular queue has been exceeded. |
| 10 | Common Buffers Size 4 Exhausted | Common buffer count for size 4 is zero. |
| 9 | Common Buffers Size 3 Exhausted | Common buffer count for size 3 is zero. |
| 8 | Common Buffers Size 2 Exhausted | Common buffer count for size 2 is zero. |
| 7 | Common Buffers Size 1 Exhausted | Common buffer count for size 1 is zero. |
| 6 | Common Buffers Size 0 Exhausted | Common buffer count for size 0 is zero. |
| 5 | Buffer Size 4 Threshold Crossed | The number of size 4 buffers is equal to or less than the threshold that was set for size 4 buffers. |
| 4 | Buffer Size 3 Threshold Crossed | The number of size 3 buffers is equal to or less than the threshold that was set for size 3 buffers. |
| 3 | Buffer Size 2 Threshold Crossed | The number of size 2 buffers is equal to or less than the threshold that was set for size 2 buffers. |
| 2 | Buffer Size 1 Threshold Crossed | The number of size 1 buffers is equal to or less than the threshold that was set for size 1 buffers. |
| 1 | Buffer Size 0 Threshold Crossed | The number of size 0 buffers is equal to or less than the threshold that was set for size 0 buffers. |
| 0 | Get Pointer Primitive Failed | This bit is set when a null address is returned on a get. |

### 9.16: POOLS Control Register

The POOLS Control Register provide status information about POOLs operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 30C8 and 0CC |
| **Power on Reset value** | X'00 00 20 01' |
| **Restrictions** | Caution must be used when asserting some of the bits during operation. |



| Bit(s) | Function | Description |
|---|---|---|
| 31-18 | Reserved | Reserved |
| 17-14 | Primitive Trap Source Selector | These bits will select the source of the last primitive trapped register.<br>0000　　Free from the PCI bus<br>0001　　Free from RAALL<br>0010　　Free from RXQUE<br>0011　　Free from CSKED<br>0100　　Free from SEGBF<br>0101　　Free from DMAQS<br>0110　　Get from PCI Bus<br>0111　　Get from RAALL<br>1000　　Get from DMAQS<br>1001　　Get from VIMEM (POOL ID(4 bits), Size(2 bits),blank(10 bits),index(16 bits)) |
| 13 | Lock POOLs on Error | When set, this bit in conjunction with the lock mask will hold POOLs State machines in an idle state until cleared. |
| 12 | Packet High Priority on Get or Free FIFOs Full | When set, this bit will cause POOLs to turn on its high priority request to packet memory when either the free or get FIFO is full. |
| 11 | Packet High Priority Always | When set, this bit will cause POOLs to always use its high priority request to packet memory. |
| 10 | Control High Priority Always | When set, this bit will cause POOLs to always use its high priority request to control memory. |

| Bit(s) | Function | Description |
|--------|----------|-------------|
| 9 | Packet High Priority with Request Timer | When set, this bit will cause POOLs to time the wait for packet memory service and when the timer expires move to high priority. |
| 8 | Control High Priority with Request Timer | When set, this bit will cause POOLs to time the wait for control memory service and when the timer expires move to high priority. |
| 7 | Fast Free Mode | When this bit is set, Fast Free Mode is enabled. When POOLs is in Fast Free Mode it does not write out the buffer map with the modified control information that indicates that the map is unused. When in this mode unused buffer free error checking is disabled. |
| 6 | Initialization Mode | When the value of the bit is '0', initialization mode is set. When the value is 1, operational mode is set. During initialization mode indexes are in the upper 16 bits of the data word. It is assumed that when initialization mode is on other normal operations are not active such as transmit or receive. During operational mode packet addresses assumed to be on the data bus. |
| 5 | Virtual Memory Mode | When set to '0', virtual memory mode is enabled. When set to '1', real memory mode is enabled. |
| 4 | Limit Event Generation | When set, this bit will cause POOLs to limit the issuance of events to RXQUE when a GTD threshold, Total Threshold or POOL Threshold is reached. It will issue the first event and disable the related event enable bit. Software must then reset the bit if it wishes to see another such event. However, it is possible that events may be lost when this bit is set on. |
| 3 | Enable Event Interface | When set, this bit will cause POOLs to issue resource events to RXQUE when a GTD threshold, Total Threshold or POOL Threshold is reached. |
| 2 | Enable Out of Range Index Checking | When set, this bit will cause POOLs to check the indexes that are streaming by to be checked against a maximum value for that size index. If the normal initialization sequence is used, these maximum values will auto set. |
| 1 | Force All Queue Transactions to Memory | When set, this will disable the internal tail to head transfer path within the queue. All indexes will proceed into memory before being brought to the head of the queue. This effectively preserves the operational history in memory. However, some caution is warranted since four full entries are required for a write to memory. This could cause indexes to get "stuck" at the back of the queue. When this residue occurs, a zero pointer will be returned even though the operation might have otherwise returned a valid pointer. |
| 0 | Diagnostic Mode | When set, POOLs is in diagnostic mode. When cleared, POOLs is in normal mode. When in diagnostic mode, state machines are held in idle. If they are already active, when they next go to idle they will hold there. |

### 9.17: POOLS Buffer Threshold Registers 0-4

The POOLS Buffer Threshold Registers 0-4 is the threshold set by the software to set the threshold crossed bit in the POOLS Status Register. This register is used to compare with the queue length register. This register consists of a 16-bit count match. The threshold count is compared to the queue length count. If the queue length is less than the value in this register, the appropriate bit is set in the status register respective to this queue.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Virtual Packets/ Buffer Size 0 | XXXX 3088 |
| | Buffer Size 1 | XXXX 308C |
| | Buffer Size 2 | XXXX 3090 |
| | Buffer Size 3 | XXXX 3094 |
| | Buffer Size 4 | XXXX 3098 |
| **Power on Value** | X'0000' | |
| **Restrictions** | None | |

### 9.18: POOLS Index Threshold Registers 0-4

The POOLS Index Threshold Registers 0-4 provide error checking.

The POOLS Index Threshold Registers 0-4 are the thresholds set by the software or hardware to set the index threshold crossed bit in the POOLS Status Register. This register is used to check indexes during free operations to look for an out of bounds index. Each register consists of a 16-bit compare value. The threshold count is compared to the index while being processed. If an index is greater than the value in this register, the appropriate bit is set in the status register.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Virtual Packets/ Buffer Size 0 | XXXX 30F0 |
| | Buffer Size 1 | XXXX 30F4 |
| | Buffer Size 2 | XXXX 30F8 |
| | Buffer Size 3 | XXXX 30FC |
| | Buffer Size 4 | XXXX 3100 |
| **Power on Value** | X'0000' | |
| **Restrictions** | None | |

### 9.19: POOLS Last Primitive Trap Register

The POOLS Last Primitive Trap Register provide debug assistance. This register contains the 32 bit last primitive address.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 30E8 |
| **Use** | The POOLS Last Primitive Trap Register is the last primitive address to POOLS, as selected in the POOLS control register, while in operational mode. |
| **Power on Reset values** | X'00 00 00 00' |
| **Restrictions** | None |

### 9.20: POOLS Last Buffer Map Read on Free Register

The POOLS Last Buffer Map Read on Free Register provide debug assistance. This register contains the 32 bit address of the buffer map used in the last free operation.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 30EC |
| **Use** | The POOLS Last Buffer Map Read on Free Register is the address of the last buffer map read on a free. |
| **Power on Reset values** | X'00 00 00 00' |
| **Restrictions** | None |

### 9.21: POOLS Error Lock Enable Register

The POOLS Error Lock Enable Register provides the ability to halt POOLS when the corresponding status bit in the status register are set. When a bit in this register that corresponds to a bit that is set in the status register, the state machines in POOLS will be held in idle state until the lock is disabled.

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 21 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 30D8 and DC |
| **Power on Reset value** | X'00 F8 00' |
| **Restrictions** | None |

### 9.22: POOLS Packet and Control Memory Access Threshold

The POOLS Packet and Control Memory Access Threshold timers are used to help limit the amount of time that POOLS can be held off from its respective memory. The bits are a 12-bit count. When the proper bit in

the POOLS Control Register is set and a request is made to the requisite memory, a counter is loaded with this value. The counter will then count down to zero in 30ns ticks. When it hits zero, it will force the request to high priority.

| | | |
|---|---|---|
| **Length** | 12 bits | |
| **Type** | Read/Write | |
| **Address** | Packet Memory Timer Threshold | XXXX 30E4 |
| | Control Memory Timer Threshold | XXXX 30E0 |
| **Power on Value** | X'080' | |
| **Restrictions** | None | |

### 9.23: POOLS Buffer Map Group

The POOLS Buffer Map Group holds the buffer map of the packet that is in the process of being freed. From this map the POOL ID and the indexes that have been used will be returned to their correct queue. This register consists of a 16-bit flag fields and 16-bit indices. The flag field contains the POOL ID and the valid bit. When a packet is freed the valid bit is set to zero. When a get operation occurs the valid bit is then set. This helps to find address duplicates and other address related problems that software can generate but are hard to find.

| | | | |
|---|---|---|---|
| **Length** | 32 bits | | |
| **Type** | Read/Write | | |
| **Address** | **Upper 16 bits** | **Lower 16 bits** | |
| | Flag Field 0 | Index 0 | XXXX 309C |
| | Index 1 | Index 2 | XXXX 30A0 |
| | Flag Field 1 | Index 3 | XXXX 30A4 |
| | Index 4 | Index 5 | XXXX 30A8 |
| | Flag Field 2 | Index 6 | XXXX 30AC |
| | Index 7 | Index 8 | XXXX 30B0 |
| | Flag Field 3 | Index 9 | XXXX 30B4 |
| | Index 10 | Index 11 | XXXX 30B8 |
| | Flag Field 4 | Index 12 | XXXX 30BC |
| | Index 13 | Index 14 | XXXX 30C0 |
| **Power on Value** | X'FFFFFFFF' | | |
| **Restrictions** | None | | |
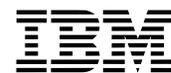
# Transmit Data Path Entities

## Entity 10: Transmit Cell Scheduler (CSKED)

The transmit cell scheduler entity is responsible for receiving a packet from the processor, determining when cells from the packets need to be transmitted, and passing this information to the segmentation buffer entity.

The logic consists of timers and counters for determining transmit opportunities, a register for determining where the timing data is located, a status register, and interfaces to ARBIT (for accessing the timing data and descriptors), PCINT (for register accesses), RXQUE (for queuing events), POOLS (for returning buffers when finished transmitting), and SEGBF (which gets the cells from memory to transmit).

### Operational Description

A logical channel descriptor (LCD), and optionally a logical path descriptor (LPD), containing scheduling parameters for the circuit must be initialized before segmentation can be started. The parameters that are important to the operation of this entity are the average interval, peak interval, transmission priority, and maximum credits that can be accumulated. See *Transmit Descriptor Data Structures* on page 39 for further information on these descriptors.

Packets to be segmented are written to packet memory that has been allocated by POOLS. The address of the LCD describing the channel that this packet is to be transmitted on, must be written to the header of the packet. Packet segmentation is started by issuing the transmit enqueue primitive to this entity. This entity will schedule segmentation of the packet according to the parameters set up in the LCD or VPD.

### Scheduling Options

CSKED has logic to assist in the processing of ABR connections. If the connection is ABR, the LCD will have a different configuration as specified in *Transmit Descriptor Data Structures* on page 39. The following fields need to be initialized before the packets are sent on the connection.

- Scheduling parameters

  This field must be set to the value specifying an ABR connection.
- Nrm

  This field should specify the maximum number of cells a source may send for each forward RM-cell. Number of cells = (2∗∗Nrm)+1.
- Trm

  This field provides an upper bound on the time between forward RM-cells for an active source. Time = 100∗(2∗∗-Trm) msec.
- ADTF

  The ACR Decrease Time Factor is the time permitted between sending RM-cells before the rate is decreased to ICR. Time = ADTF∗.01msec.
- All other ABR fields should be initialized to zero.

This section contains descriptions of the registers used by the Cell Scheduler.

### 10.1: Transmit Enqueue Primitive

Enqueues a buffer for transmission.

**Length**          32 bits

**Type**            Write Only

**Address**         XXXX 1200

**Power On Value**  X'0000 0000'

**Restrictions**    None
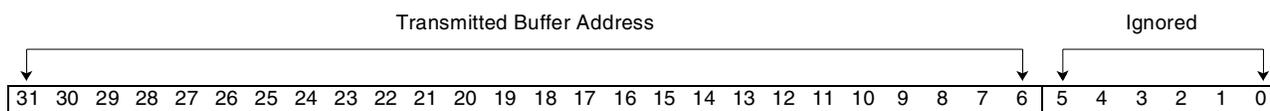
|  | Transmitted Buffer Address |  |  | Ignored |
|---|---|---|---|---|

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

| Bit(s) | Description |
|---|---|
| 31-6 | This must contain the address of the buffer to be transmitted. Buffers must be aligned on at least 64-byte boundaries. The lower six bits are ignored. |

### 10.2: Resume Transmission Primitive

Resumes transmission on a connection specified by an LCD address. On an ABR connection, certain events can cause the transmission to be suspended until a rate conversion is completed. This primitive will resume transmission on those connections. This is normally done by the internal processor.

**Length**          32 bits

**Type**            Read/Write

**Address**         XXXX 1204

**Power On Value**  X'0000 0000'

**Restrictions**    This address should be written with care. This primitive should only be used on connections that have been suspended.
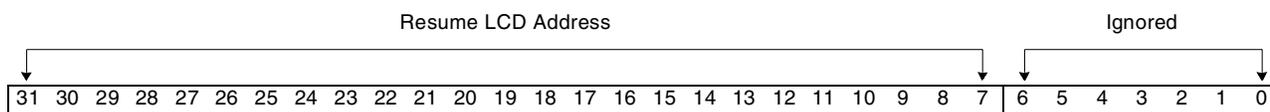
|  | Resume LCD Address |  |  | Ignored |
|---|---|---|---|---|

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

| Bit(s) | Description |
|---|---|
| 31-7 | This must contain the address of the LCD that is to resume transmission. The lower seven bits are ignored. |

### 10.3: Close Connection Primitive

Transmission is complete on a connection specified by an LCD address. When no more traffic is to be sent on a connection, this primitive can be executed to cause an event to be generated when segmentation has stopped on this connection. Segmentation will be stopped immediately, or stopped after all packets on this connection have been transmitted as specified in the CSKED Control Register.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 120C |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

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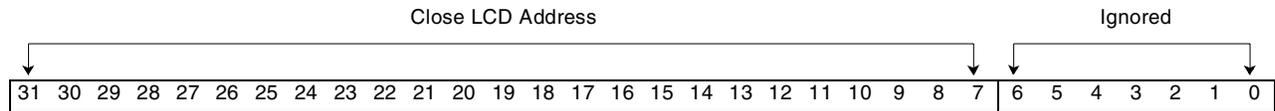
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-7 | This must contain the address of the LCD that is to be closed. The lower seven bits are ignored. |

### 10.4: Start/Stop Timer Primitive

Start or stop a timer with the parameters in the specified LCD address. When this primitive is executed, a timer, with parameters contained in the specified LCD, is started or stopped. Bit 0 specifies whether to start (0) or stop(1) the timer. When the timer pops, a DMA descriptor specified in the LCD will be executed.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 125C |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

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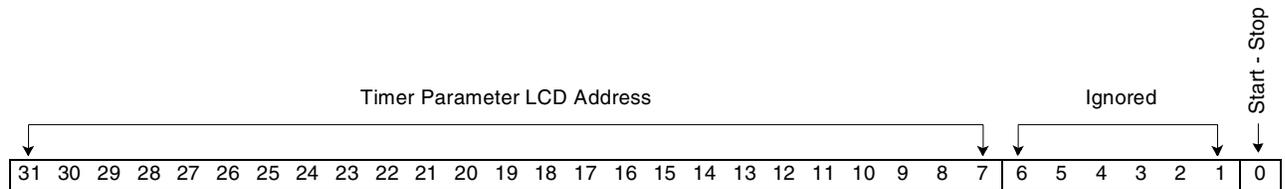
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-7 | This must contain the address of the LCD that contains the timer parameters. The lower seven bits are ignored. |
| 6-1 | These bit are not used. |
| 0 | This bit specifies whether the timer is to be started(0) or stopped(1). |

## 10.5: Timeslot Prescaler Register

This register determines the length of time for 1 timeslot. This controls the rate that the cell scheduling counters are incremented. Each clock cycle the value in this register is added to a 24-bit counter. When the upper bit of the counter changes state the Current Timeslot Counter is incremented. This should normally be set to the time it takes to transmit one cell. It will be initialized to the cell time for a 155 Mb/s SONET connection (149.76 Mb/s payload). The following formula should be used to determine the value to load in this register: Timeslot Prescaler = (clock interval/timeslot interval) x $2**23$.

| | |
|---|---|
| **Length** | 24 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1210 |
| **Power On Value** | X'015B3E' |
| **Restrictions** | This register should be written only at initialization time. |

Timeslot Counter Rate

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

| Bit(s) | Description |
|--------|-------------|
| 23-0 | This value will determine the rate at which the current timeslot counter is advanced. |

## 10.6: Current Timeslot Counter

This counter contains a count of how many prescaled intervals have elapsed. It is used to determine if scheduling needs to be done or credits exist.

| | |
|---|---|
| **Length** | 32 Bits |
| **Type** | Read/Write |
| **Address** | XXXX 1218 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | This register is meant to be read only. It is writable for diagnostic purposes only. |

Elapsed Interval Count

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

| Bit(s) | Description |
|--------|-------------|
| 31-0 | This value represents how many expirations have occurred since the counter rolled over. |

## 10.7: Timing Data Base Address

32KB of control memory must be reserved for timing data. This register contains the base address for the timing data. The lower 15 bits will be taken from the appropriate counter.

**Length**            32 bits

**Type**              Read/Write

**Address**           XXXX 121C

**Power On Value**    X'0001 0000'

**Restrictions**      The timing data base address must be on 32-KB boundary.
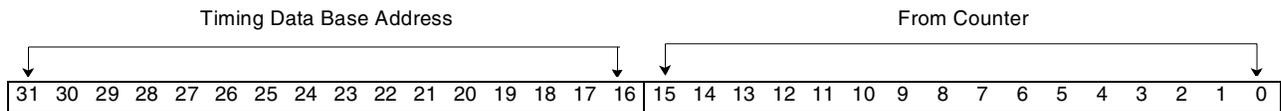
| Bit(s) | Description |
|--------|-------------|
| 31-16 | This value represents the upper 17 bits of the timing data base address. |
| 15-0 | Taken from the appropriate counter. |

## 10.8: CSKED Control Register

This register is used to control the actions of CSKED. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

**Length**            22 bits

**Type**              Read/Write

**Address**           XXXX 1220 and 024

**Power On Value**    X'0759'

**Restrictions**      None

| Bit(s) | Name | Description |
|--------|------|-------------|
| 21 | Disable overlapping transmit requests | This bit is meant for debug purposes only. It will disable the ability of CSKED to overlap requests to SEGBF. |
| 20 | Close connection immediately | Setting this bit will cause segmentation to stop immediately on any connection that has been issued a close connection primitive. If this bit is not set an event will be generated after all traffic queued to this connection has been sent. |
| 19 | Queue LCD based memory management events | Setting this bit will cause LCD based memory management events to be queued to the Transmit Complete Queue, if enabled by bit 18 of this control register. If LCD based memory management is enabled and this bit is off, the receive pool ID associated with this connection will be updated when a threshold is crossed. |
| 18 | LCD based memory management | Setting this bit will enable LCD based memory management for received packets. See *Logical Channel Data Structure* on page 38 and *Definition of LCD-Based Memory Management of Transmit LCD* on page 45 for further information on this function. |
| 17 | Use bandwidth scheduling for low priority | Setting this bit will cause low priority traffic to be bandwidth scheduled. The peak interval in the LCD is used to provide a relative weight in determining the amount of bandwidth the connection will use. For example a peak interval of one will use twice the bandwidth as a connection with a peak interval of two. The average interval specifies the maximum rate that the connection can use. For example if the average interval is set to '2', the maximum rate at which it can send a cell is every two timeslot times (as defined in the Timeslot Prescaler Register). |
| 16 | Control scheduling of medium priority traffic with GFC reset for group A | Setting this bit will cause medium priority traffic to be controlled by the group A GFC bits in the ATM header of received cells. |
| 15 | Control scheduling of low priority traffic with GFC reset for group B | Setting this bit will cause low priority traffic to be controlled by the group B GFC bits in the ATM header of received cells. |
| 14 | Queue the LCD address if freeing and queueing | Setting this bit will cause the LCD address, instead of the packet address, to be queued if both freeing and queueing on transmit complete. |
| 13 | Disable virtual buffer error detection | Setting this bit will cause the buffer enqueue logic to ignore virtual buffer errors. |
| 12 | Disable queuing virtual buffer errors | If virtual buffer error detection is not disabled, detected errors will be queued. If this bit is set, this queuing is disabled and the buffer will be freed. |
| 11 | Reserved | Reserved. |
| 10-8 | SEGBF Queue Length Threshold | Cells can be queued in SEGBF up to the number specified in this register. The default is seven which is above the limit for pass two. Writing these bits to zero will also disable this function. |
| 7 | Disable SEGBF Queue Length in Scheduling | CSKED will normally include SEGBFs queue length in the calculations when rescheduling a cell. If this bit is on it will disable this function and the cell will be scheduled as if the cells were transferred when SEGBF accepted the cells. |
| 6 | Priority of buffer requests | If this bit is not set, scheduling requests have a higher priority than buffer requests. If this bit is set this priority is reversed. It should be set if a significant percentage of packets are only a few cells in length. |
| 5 | Enable timers | Timer descriptors can be enqueued to this entity that will cause a DMA descriptor to be executed on expiration. If these timers are used, this bit must be set. If they are not used, this bit should be reset. |
| 4 | Reschedule Packets in the Slow Queue to the Fast Queue | This function is not implemented in pass one. It is implemented in pass two. If the average or peak interval is greater than 255, the cells will be scheduled in the slow queue. The slow queues will be serviced every 64 prescaler time units. This means that a jitter of up to 64 prescaler time units should be expected for slow traffic. If this bit is set, packets in the slow queue will be rescheduled at the appropriate time to the fast queue. This will decrease the variation in the scheduling but may cause some performance degradation if traffic is heavy. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 3 | Enable High Priority Traffic | For each priority enabled 16KB of control memory must be reserved for timing data. If only one or two priorities are to be used, bits corresponding to unused priorities should be cleared to improve performance. |
| 2 | Enable Medium Priority Traffic | Enable Medium Priority Traffic. |
| 1 | Enable Low Priority Traffic | Enable Low Priority Traffic. |
| 0 | Enable Cell Scheduling | If this bit is off no primitives will be handled or cells scheduled. |

## 10.9: Transmit Segmentation Throttle Register

This register contains the number of cycles to wait between successive requests to transmit a cell. Its purpose is to slow segmentation on all VCIs if software determines that the network cannot handle the generated load. The value in this register will be loaded into the Transmit Segmentation Counter each time a cell is accepted for transmission. For normal operation the value in this register should be zero.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1230 |
| **Power On Value** | X'0000 ' |
| **Restrictions** | None |

Queue Length
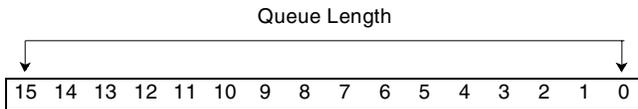
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-0 | When the transmit complete queue length reaches this value an interrupt will be generated. |

## 10.10: Transmit Segmentation Throttle Counter

This register is loaded with the value in the Transmit Segmentation Throttle Register after each cell is accepted for transmission and counts down until it reaches zero. A new cell transmission will not be requested until this counter reaches zero.

**Length**          16 bits

**Type**            Read Only

**Address**         XXXX 1234

**Power On Value**  X'0000'

**Restrictions**    Read Only

Transmit Segmentation Throttle Counter
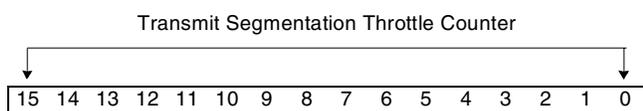
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-0 | When this counter reaches zero a new cell can be transmitted. |

## 10.11: MPEG Conversion Register

This register is used to convert MPEG time units into Timeslot time units. If MPEG traffic is configured in the LCD, the data stream will be monitored for PCRs. If a PCR is detected, it will be scheduled at the time specified in the PCR. A conversion factor needs to be written into this register to convert the MPEG time units into timeslot units. It will be initialized to a value that converts the MPEG time units (90 KHz) into the timeslot units (353.2 KHz, assuming one timeslot is the time it takes to send one cell over a SONET connection). The lower 12 bits of this register contain the fractional portion of this conversion factor.

Example: 353.2076 KHz / 90 KHz = 3.924528 = 3.ECB hex

**Length**          15 bits

**Type**            Read/Write

**Address**         XXXX 1208

**Power On Value**  X'3ECB'

**Restrictions**    None

MPEG Time Conversion Factor
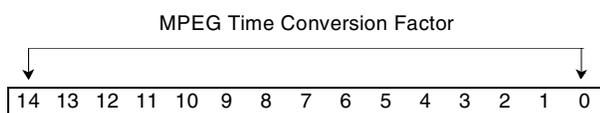
| 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 14-0 | Contains the conversion factor. |

**10.12: GFC Reset Values**

This register is used to load values into the GFC counters upon receiving a cell with the GFC reset bit on in the received ATM header. The lower eight bits will be the value loaded into the Group A counter, and the upper eight bits will be the value loaded into the Group B counter.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1214 |
| **Power On Value** | X'0101' |
| **Restrictions** | None |

Group B Counter          Group A Counter
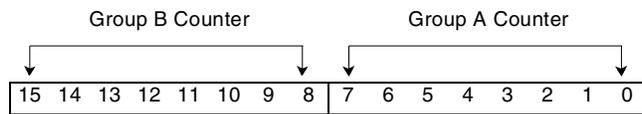
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|---|---|
| 15-8 | Value loaded into the Group B counter. |
| 7-0 | Value loaded into the Group A counter. |

## 10.13: ABR Timer Prescaler Register

This register determines the length of time for a tick of the RM Cell Timer. This controls the rate that the cell scheduling counters are incremented. Each clock cycle the value in this register is added to a 24-bit counter. When the upper bit of the counter changes state, the RM Cell Timer is incremented. This should be set to a value of .78 ms. It will be initialized to .78 ms assuming a 30-ns clock (as set up in SCLOCK). The following formula should be used to determine the value to load in this register: ABR Timer Prescaler = (clock interval/.78 ms) x $2**23$.

| | |
|---|---|
| **Length** | 24 bits |
| **Type** | Read/Write |
| **Address** | XXXX 127C |
| **Power On Value** | X'000143' |
| **Restrictions** | This register should be written only at initialization time. |

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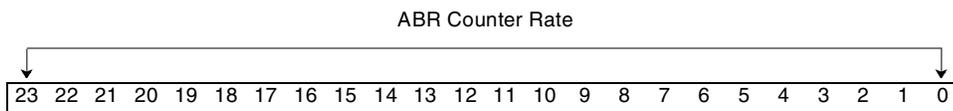
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 23-0 | This value will determine the rate at which the ABR counter is advanced. |

## 10.14: RM Cell Timer

This register is used to keep track of the last time that an ABR RM cell was sent. Its period should be .78 ms.

| | |
|---|---|
| **Length** | 24 bits |
| **Type** | Read/Write |
| **Address** | XXXX 126C |
| **Power On Value** | X'000000' |
| **Restrictions** | None |

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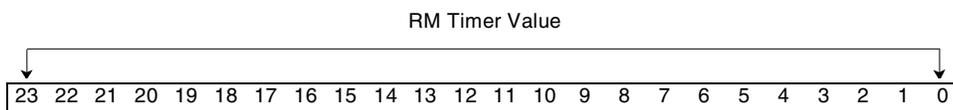
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 23-0 | Timer Value. |

**Performance Registers**

This section contains registers that are for performance purposes.

### 10.15: High Priority Bandwidth Limit Register

This register can be used to limit the bandwidth used by high priority connections. The upper eight bits of this register specifies the number of high priority cells that can be sent in a window specified by the lower eight bits of this register.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1270 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
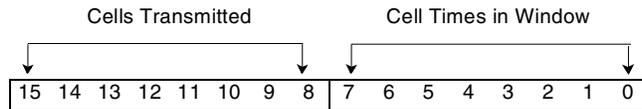
Cells Transmitted      Cell Times in Window

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-8 | This value specifies the number of cells that can be transmitted from high priority connections in one time window. |
| 7-0 | This value specifies the number of cell times in the window. |

**10.16: Medium Priority Bandwidth Limit Register**

This register can be used to limit the bandwidth used by medium priority connections. The upper eight bits of this register specifies the number of medium priority cells that can be sent in a window specified by the lower eight bits of this register.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1274 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
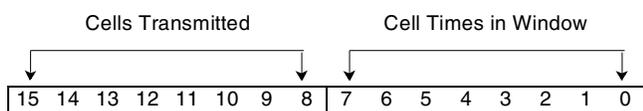
Cells Transmitted          Cell Times in Window

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|---|---|
| 15-8 | This value specifies the number of cells that can be transmitted from medium priority connections in one time window. |
| 7-0 | This value specifies the number of cell times in the window. |

**10.17: Low Priority Bandwidth Limit Register**

This register can be used to limit the bandwidth used by low priority connections. The upper eight bits of this register specifies the number of low priority cells that can be sent in a window specified by the lower eight bits of this register.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1278 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
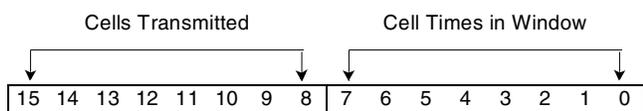
Cells Transmitted          Cell Times in Window

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|---|---|
| 15-8 | This value specifies the number of cells that can be transmitted from low priority connections in one time window. |
| 7-0 | This value specifies the number of cell times in the window. |

## 10.18: High Priority Cells Transmitted Counter

This register contains the number of cells transmitted from high priority connections.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1260 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
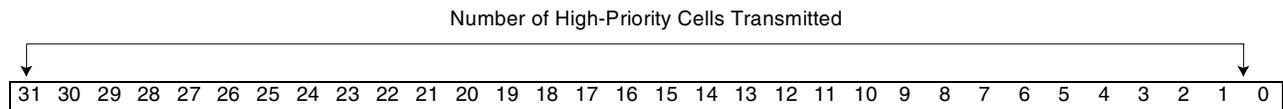
Number of High-Priority Cells Transmitted

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

| Bit(s) | Description |
|---|---|
| 31-0 | This value represents the number of cells transmitted from high priority connections. |

## 10.19: Medium Priority Cells Transmitted Counter

This register contains the number of cells transmitted from high priority connections.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1264 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
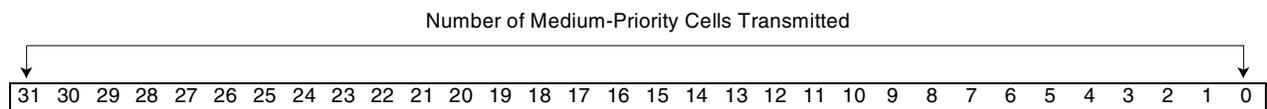
Number of Medium-Priority Cells Transmitted

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

| Bit(s) | Description |
|---|---|
| 31-0 | This value represents the number of cells transmitted from high priority connections. |

**10.20: Low Priority Cells Transmitted Counter**

This register contains the number of cells transmitted from high priority connections.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1268 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

Number of Low-Priority Cells Transmitted

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

| Bit(s) | Description |
|---|---|
| 31-0 | This value represents the number of cells transmitted from high priority connections. |

## Debugging Register Access

This section contains registers that are for debug purposes only. These registers need not be written or read during normal operations.

### 10.21: High Priority Serviced Counter

This register contains the value of the last high priority slot that has been serviced. When this count differs from the Current timeslot count, at least one high priority slot needs servicing. Each time the high priority slot is serviced, this counter will increment.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1240 |
| **Power On Value** | X'0000' |
| **Restrictions** | This register is meant to be read only. It is writable for diagnostic purposes only. |

High-Priority Service Counter

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-0 | This value represents how many times this time wheel has been serviced since the counter rolled over. |

### 10.22: Medium Priority Serviced Counter

This register contains the value of the last medium priority slot that has been serviced. When this count differs from the Current timeslot count, at least one medium priority slot needs servicing. Each time the medium priority slot is serviced, this counter will increment.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1244 |
| **Power On Value** | X'0000' |
| **Restrictions** | This register is meant to be read only. It is writable for diagnostic purposes only. |

Not Used                                                    Medium-Priority Service Counter

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

| Bit(s) | Description |
|--------|-------------|
| 15-0 | This value represents how many times this time wheel has been serviced since the counter rolled over. |

**10.23: Low Priority Serviced Counter**

This register contains the value of the last low priority slot that has been serviced. When this count differs from the Current timeslot count, at least one low priority slot needs servicing. Each time the low priority slot is serviced, this counter will increment.

**Length**            16 bits

**Type**              Read/Write

**Address**           XXXX 1248

**Power On Value**    X'0000'

**Restrictions**      This register is meant to be read only. It is writable for diagnostic purposes only.

Low-Priority Service Counter
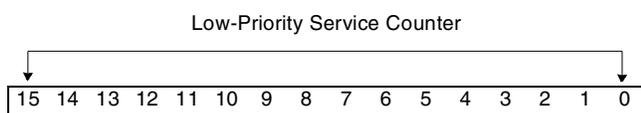
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-0 | This value represents how many times this time wheel has been serviced since the counter rolled over. |

**10.24: Slow Serviced Counters**

There are three slow serviced counters, one for each transmit priority. These registers contain the value of the last slow time slot that has been  serviced. When this count differs from the Current timeslot count, at least one slow slot needs servicing. Each time the slow slot is serviced, this counter will increment.

**Length**            10 bits

**Type**              Read/Write

**Address**           XXXX 124C XXXX 1250 XXXX 1254

**Power On Value**    X'000'

**Restrictions**      This register is meant to be read only. It is writable for diagnostic purposes only.

Slow Service Counter
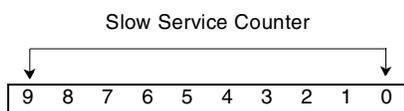
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 9-0 | This value represents how many times this slow wheel has been serviced since the counter rolled over. |

### 10.25: Timer Serviced Counters

In addition to the counters above, there is an additional counter for processing timer requests. This register contains the value of the last timer slot that has been serviced. When this count differs from the Current timeslot count (bits 22-15), at least one slow slot needs servicing. Each time a timer slot is serviced, this counter will increment.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1280 |
| **Power On Value** | X'00' |
| **Restrictions** | This register is meant to be read only. It is writable for diagnostic purposes only. |

Not Used                  Timer Service Counter
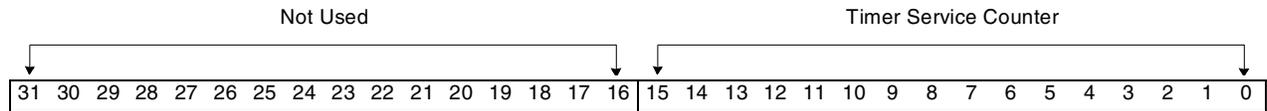
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 15-0 | This value represents how many times this timer wheel has been serviced since the counter rolled over. |

## 10.26: CSKED Status Register

This register is used to control the actions of CSKED. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

**Length**        5 bits

**Type**          Clear/Set

**Address**       XXXX 1228 and 22C

**Restrictions**  None

**Power On Value**  X'0'
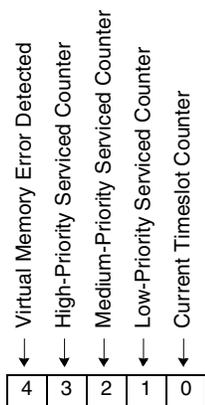
| Bit(s) | Name | Description |
|--------|------|-------------|
| 4 | Virtual Memory Error detected | If a virtual memory write operation could not complete because a real buffer was not available, a signature is written to the packet header. If this signature is detected when the buffer is enqueued for transmission, this bit will be set and an event will be posted to RXQUE. |
| 3 | High priority serviced counter | High priority serviced counter has overrun. |
| 2 | Medium priority serviced counter | Medium priority serviced counter has overrun. |
| 1 | Low priority serviced counter | Low priority serviced counter has overrun. |
| 0 | Current Timeslot Counter | Current Timeslot Counter has wrapped<br>If this bit is on the timeslot counter has wrapped. |

## 10.27: CSKED Interrupt Enable Register

This register is used to enable interrupts from CSKED. If a bit is on in the status register and the corresponding enable bit is on in this register, an interrupt will be generated, if enabled in INTST.See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1238 and 23C |
| **Power On Value** | X'0' |
| **Restrictions** | None |

## 10.28: Timing Data Array

This array contains data relevant to scheduling cells. It should only be written for diagnostic purposes to test the array. It will power up to all zeros and should be rewritten to zeros after the array has been tested.

| | |
|---|---|
| **Length** | 24 words x 32 bits |
| **Type** | Read/write |
| **Address** | XXXX 1300 |
| **Power On Value** | X'000000' |
| **Restrictions** | Should be written for diagnostic use only. Initialize back to zeros when through testing. |

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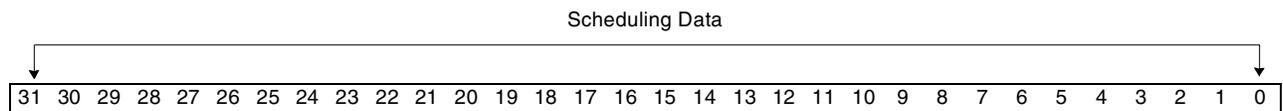
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-0 | Scheduling data. |

**10.29: State Machine Variables**

This register contains the current state of the three main state machines in this entity.

| | |
|---|---|
| **Length** | 14 bits |
| **Type** | Read |
| **Address** | XXXX 1258 |
| **Power On Value** | X'0000' |
| **Restrictions** | None |

State Machine Variables
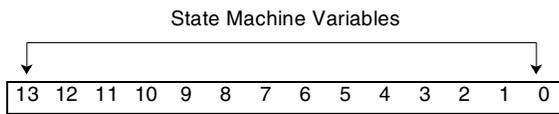
| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 13-0 | Value of state machine variables. |

## Entity 11: ATM Transmit Buffer Segmentation (SEGBF)

The segmentation buffer entity (SEGBF) accepts frames from the cell scheduler (CSKED) or software, and then generates ATM cells to send out over the external physical interface. This entity knows or cares nothing about scheduling cells over time, it will simply construct a cell when it is provided an address of a logical channel descriptor on which to operate. All rate and scheduling concerns must be addressed by the logic or software prior to queueing a frame to SEGBF. The SEGBF logic consists of a four-deep input FIFO for 24-bit LCD addresses, a general purpose CRC generator to handle data from one to eight bytes wide, a special purpose CRC generator to calculate the HEC, a four-deep output FIFO for cells waiting to be transmitted, and various pieces of control logic. A simplified block diagram is shown in the "SEGBF Block Diagram."

## SEGBF Block Diagram
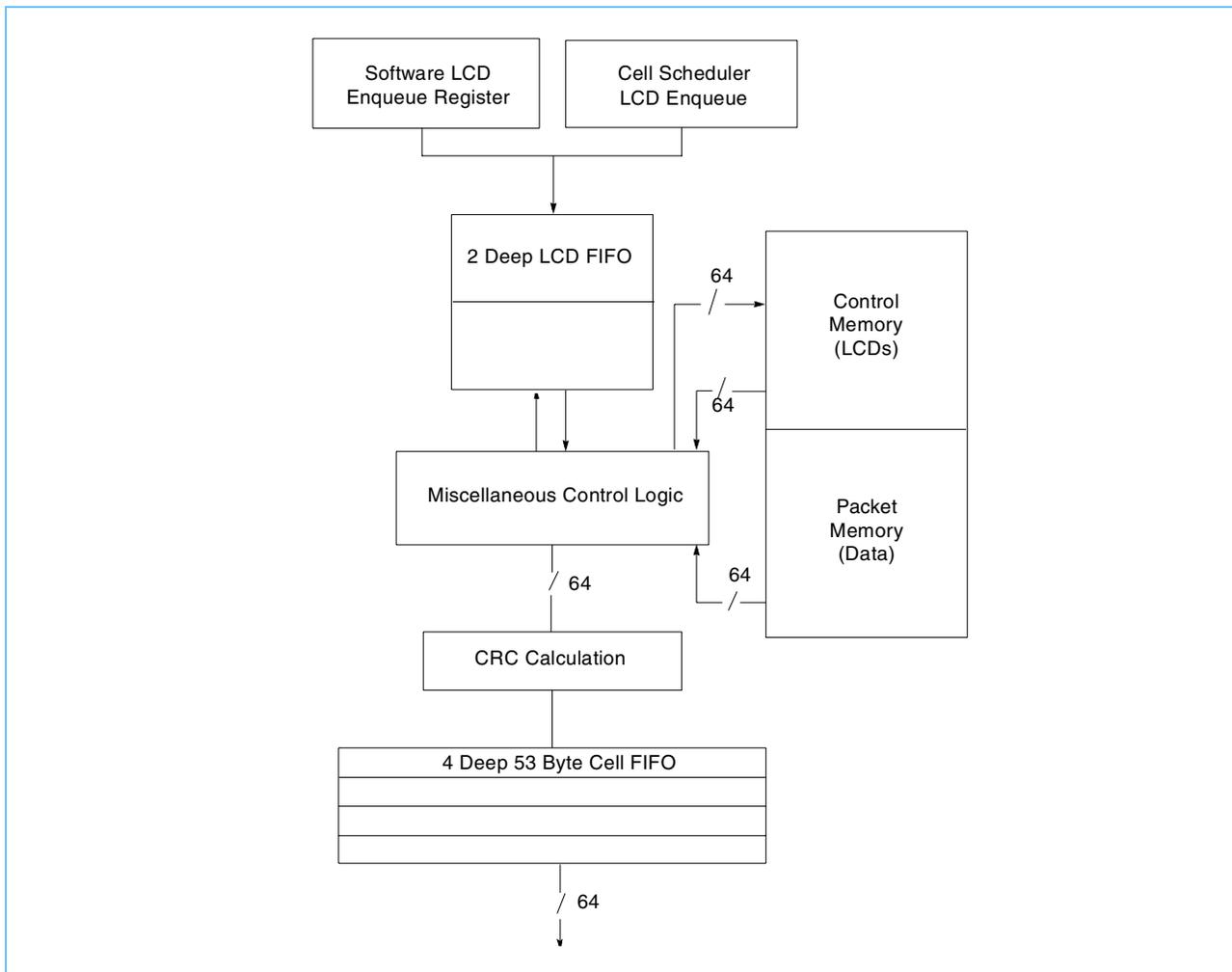
The sequence of events that happens when an AAL5 frame is enqueued to SEGBF is as follows:

1. The required information associated with the enqueued frame is fetched form the logical circuit descriptor (LCD). This data consists of but is not limited to: current data pointer, current CRC, AAL type, four bytes of ATM header, data length and various flag bits.

2. The ATM header fetched above, is used to calculate the single byte HEC. These five bytes are the first bytes of the cell buffer. Alternatively, mainly for diagnostic purposes, the capability exists to force the fifth byte of the cell to be any value the user wishes. An eight-bit register and a control bit are provided to implement this function.

3. The current data pointer and data length from step 1 are used to determine how many data bytes to fetch from data storage. As many as 48 bytes can be fetched or if the previous cell from this frame contained the last byte of data but did not have space left over for the AAL5 trailer, 0 bytes can be fetched.

4. If the internal logic determines that the current cell being assembled will be the last cell of this frame, the AAL5 trailer is appended to the cell buffer, with any unused bytes being padded with zeros. If the frame was provided by CSKED, then an indication is returned to CSKED to allow the appropriate data structures to be updated.

5. The updated currRegisters

This section contains descriptions of the registers used by the Buffer Segmentation logic.

## 11.1: SEGBF Software LCD Enqueue

This register provides a mechanism for software to transmit a single cell or a group of cells making up a buffer that can contain any user defined data at any time. To cause a cell/buffer to be transmitted, the software must write the address of a valid VCI control block to this register. The segmentation hardware will then construct a cell to match the AAL type defined in the LCD control block, using the segmentation pointer contained in the LCD to fetch data and present this cell to the next lower level of hardware to transmit. This method of cell transmission bypasses the cell scheduler completely so it is the responsibility of the software to ensure that peak and average rates are not violated. When the segmentation logic has completed building the cell and queued it for transmission, the LCD address will be loaded into the software LCD complete register. This method of cell transmission is not designed for high performance and as such, there is only a single level of queueing underneath the complete register. It is recommended that only a single software LCD be queued to the segmentation logic at any one time to prevent hanging the segmentation logic as it attempts to queue a complete software LCD to the complete queue.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1400 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | Before enqueuing a VCI, software must ensure that the previous software enqueue has been handled by the hardware. This is accomplished by reading this register before an enqueue is attempted. If a value of zero is returned, the segmentation hardware is ready to accept an enqueue operation. If a non-zero value is returned, it will be the address of the previous VCI that was enqueued and this indicates that the segmentation hardware has not been able to enqueue the VCI to it's internal VCI buffer segmentation queue. If this mechanism shows that this interface is busy and unable to accept new VCI addresses for any appreciable amount of time (tens of μs), it is likely that a condition exists which is preventing the hardware below the segmentation logic from accepting cells for transmission, and the segmentation logics input buffer is full. This mechanism also adds the restriction that a VCI control block should never exist at address zero. |

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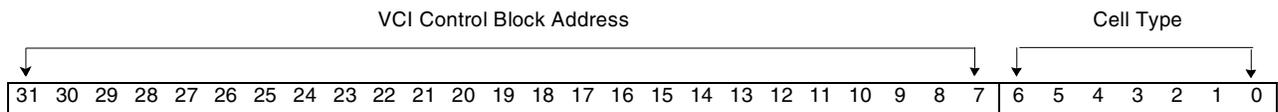
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 31-7 | These bits contain the upper 25 bits of the address of the VCI control block. |
| 6-0 | These bits control what type of cell will be built by the segmentation logic. There are currently only two valid values for these bits. If these bits are all zero, a normal cell as defined by the LCD will be built. If these bits have a value of 0x7F, an ABR cell will be built using fields defined in the LCD. |

**11.2: SEGBF Force HEC Value**

This register provides a mechanism for software to force the fifth byte of a cell to contain any value when it is provided to the externally connected physical interface chip. This value will only be used if the SEGBF control register indicates that the HEC should be loaded and not calculated.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1404 |
| **Power On Value** | X'00' |
| **Restrictions** | None |

5th Byte Insert

```
  7  6  5  4  3  2  1  0
```
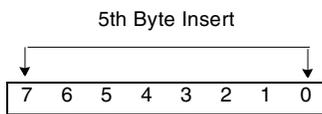
| Bit(s) | Description |
|---|---|
| 7-0 | These bits contain the eight bits that will be inserted in the fifth byte of the cell being assembled |

### 11.3: SEGBF Control Register

This register provides a mechanism to control the various programmable features of SEGBF. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 22 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1408 and 40C |
| **Power On Value** | X'00000' |
| **Restrictions** | None |

'U

| Bit(s) | Description |
|---|---|
| 21 | When set, this bit will force all NUD cells with the PTI field of 111 to be excluded from the NUD cell counter. |
| 20 | When set, this bit will force all NUD cells with the PTI field of 110 to be excluded from the NUD cell counter. |
| 19 | When set, this bit will force all NUD cells with the PTI field of 101 to be excluded from the NUD cell counter. |
| 18 | When set, this bit will force all NUD cells with the PTI field of 100 to be excluded from the NUD cell counter. |
| 17 | When set, this bit will force all LCD address to be shifted by eight bytes when accessed by the segmentation logic. It is meant to be used for diagnostic testing of the segmentation hardware. |
| 16 | When set, this bit will force all LCDs to be fetched from memory and not buffered in SEGBF. |
| 15 | When set, this bit will enable a function in the segmentation logic that replaces the HEC in the cell with a fourth byte of user data from the LCD. The byte that replaces the HEC will be retrieved from a location in the LCD that is specified by the Pre-pended header steering register. |
| 14 | This bit when set will enable the segmentation function which allows user data from the LCD to be pre-pended on the cells as they are built. When enabled, the segmentation logic will always place three bytes of user-defined data from a location in the LCD that is specified in the Pre-pended header steering register, into the cell buffer preceding the ATM header before passing the cell to the next lower level. |
| 13 | This bit when set will cause the segmentation logic to pause when it reaches the idle state. Segmentation will not be continued until this bit has been reset. Care must be taken to leave this bit set for a very short duration so that segmentation throughput will not be adversely affected. |
| 12 | When set, this bit will cause the segmentation logic to delay the cell immediately following the cell that contained the PCR until the correct time. If reset, the last cell of the AAL5 frame that contained a PCR will be delayed. |
| 11 | When set, this bit will cause the segmentation logic to count the number of frames that have been sent instead of the number of cells that have been sent with CLP = 0. Both the register that counts these events across all LCDs as well as the VCI specific field in a particular LCD are affected by this bit. |

| Bit(s) | Description |
|---|---|
| 10 | When set, this bit will prevent PCR containing packets from terminating an AAL5 frame only when the PCR containing packet is the first transport stream packet in the AAL5 frame. The AAL5 frame will be terminated on the second transport stream packet even if that packet does not contain a PCR. |
| 9 | When set, this bit will prevent PCR containing packets from terminating an AAL5 frame. |
| 8 | When set, this bit will enable the MPEG-2 PID screening logic. In this mode, all MPEG-2 PCR related logic will only be active if the 188-byte transport stream packet being processed has the correct PID value. The correct value is determined as the transport stream packets are being segmented. The first 188-byte packet that contains a PID that satisfies the PID limit register, and has a PCR, will cause the segmentation logic to "lock" onto that PID value. Once locked, the PCR rate matching logic will only be active on packets that contain that PID. Any other packets that have a different PID will be sent out at the normal scheduled rate regardless of whether the packet contains a PCR or not. The PID logic will "unlock" from a given PID if a packet with that PID is not processed by the segmentation logic in a programmable amount of time defined in the &regsgpidd. When reset, the PID field in all transport stream packets will be ignored. |
| 7 | When set, this bit will reset all control logic in the entity. After being set, this bit must be reset before the segmentation logic will function properly. This bit must remain set for at least 1 µs to reset the segmentation logic properly. |
| 6 | When set, this bit will force statistics to be kept on all LCDs processed by the segmentation logic. |
| 5 | When set, this bit will prevent statistics from being kept on any LCDs processed by the segmentation logic. This bit will have precedence over the force statistics bit. |
| 4 | When set, this bit will force OAM blocking to occur an all LCDs processed by the segmentation logic. The OAM blocking logic requires that statistics are being kept for the LCD being monitored, so if this bit is set, statistics will be forced on all LCDs. |
| 3 | When set, this bit will prevent OAM blocking for happening on any LCDs processed by the segmentation logic. This bit will have precedence over the force blocking bit. |
| 2 | When set, this bit will cause all requests from the cell scheduler to be ignored. This allows complete program control of all cells being sent out on the external interface |
| 1 | When set, this bit causes the fifth byte of all cells being assembled by SEGBF to contain the value that is contained in the Force HEC register. When reset, the fifth byte will contain a value calculated over the first four ATM header bytes retrieved from the LCD. |
| 0 | This bit when set causes the SEGBF entity to enter diagnostic mode. This bit must be set in order to access the internal array. When accessing the array, care must be taken that normal entity reads and writes of the array are not happening at the same time or the results will be indeterminant. |

## 11.4: SEGBF Status Register

This register provides feedback to the user on the current status of SEGBF. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1410 and 414 |
| **Power On Value** | X'00' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 7-5 | Reserved |
| 4 | When set, this bit indicates that the total OAM cells transmitted register has wrapped. |
| 3 | When set, this bit indicates that the total user cells transmitted with CLP=0 register has wrapped. |
| 2 | When set, this bit indicates that the total user cells transmitted register has wrapped. |
| 1 | When set, this bit indicates that the segmentation logic has completed cell generation for a LCD that was enqueued by the software to the software LCD enqueue register. |
| 0 | When set, this bit indicates that the segmentation logic has detected an invalid condition in one of the LCDs that it was processing. The address of the LCD in error is contained in the Invalid LCD register. Any invalid LCDs detected, are not processed further by the segmentation logic, so the program must do something to clear this condition. The following are detected sources of the Invalid LCD condition:<br>**1** 128-byte LCDs are configured and an LCD on a 64-byte bound is detected<br>**2** An invalid AAL was specified in the LCD<br>**3** The packet length plus the buffer offset is greater than the configured buffer size. |

**11.5: SEGBF Invalid LCD Register**

This register provides feedback to the program when the segmentation logic detects an invalid LCD. If multiple invalid LCDs are being processed, this register will contain the address of the last one that was processed by the segmentation logic. To clear this condition for LCDs being processed by the cell scheduler it is suggested that the program write X'FFFFFFFF' to the segmentation pointer in the indicated LCD. This will cause the segmentation logic to terminate cell transmission on the next opportunity and cause the cell scheduler to clean up and go on to the next buffer indicated in the LCD. There are several invalid LCD situations that the segmentation logic checks for: The first is the LCD address not being on the correct boundary. For example, if the chip is configured to have all LCDs on 128-byte boundaries and a LCD is encountered that is not on a 128-byte boundary. Another invalid condition is when the transmit length configured in the LCD plus the offset in the LCD when added together exceed the maximum overall packet size configured in the chip. It is up to the program to determine which of the possible conditions caused the error to be reported.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1418 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 31-7 | These bits contain the 32-bit address of the LCD detected to be in error. |
| 6-4 | Always read as zeros. |
| 3 | When set, this bit indicates that an LCD has been encountered that requires a larger number of eight-byte words from the VCI than the current value configured in the SEGBF maximum LCD size register. |
| 2 | When set, this bit indicates an LCD containing an invalid AAL-type field was detected. |
| 1 | When set, this bit indicates a transmit length error was detected. |
| 0 | When set, this bit indicates that a bad sync byte was detected in an MPEG-2 transport stream packet. |
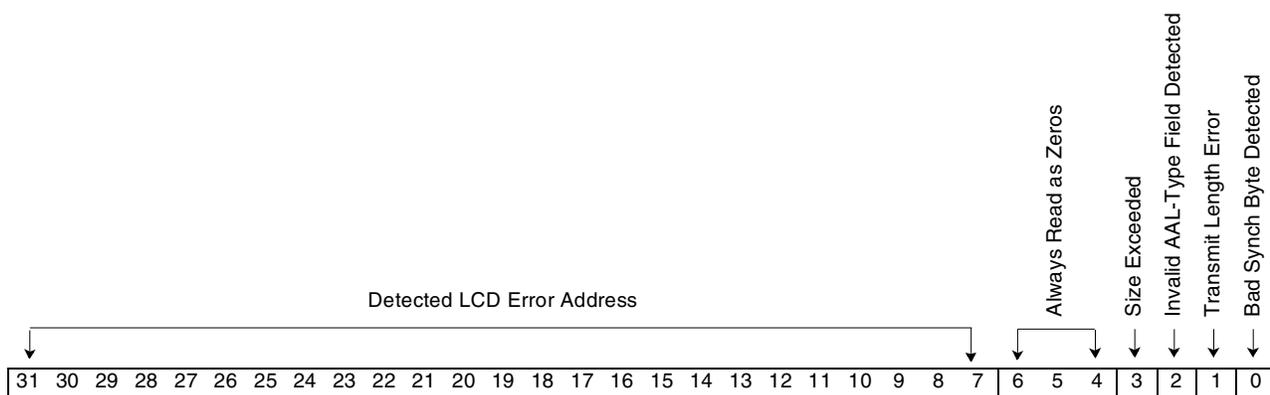
### 11.6: SEGBF Software LCD Complete

This register provides feedback to the program when the segmentation logic completes cell generation for a LCD that was enqueued by the software. After the segmentation logic has updated the LCD, the address of the LCD is copied into this register providing any previous LCD addresses written to this register have been read by the software. If multiple software queued LCDs are outstanding to the segmentation logic at any time, the segmentation process can be delayed when multiple software enqueued LCDs complete without the software getting a chance to read the LCD addresses from this register. To guarantee that the segmentation logic never has to wait for the software to read this register, it is recommended that only one software LCD be enqueued at any one time.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 141C |
| **Power On Value** | X'00000000' |
| **Restrictions** | Bits 5 - 0 are not implemented and will always return zero. To maintain future compatibility, zeros should be written to these bits. |

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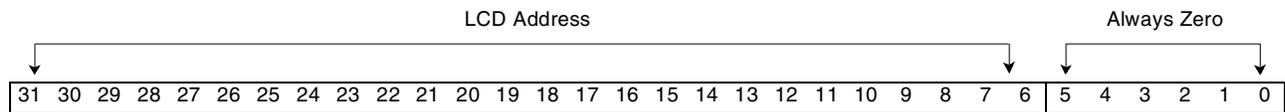
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-6 | These bits contain the upper 26 bits of the LCD address that the segmentation logic has finished processing. |
| 5-0 | These bits will always be read as zero. |

### 11.7: SEGBF Interrupt Enable Register

This register allows the user to selectively determine which bits in the SEGBF status register will cause processor interrupts. A zero in a bit position masks interrupts from the corresponding bit location in the SEGBF status register. A one in a bit position allows interrupts for the corresponding bit in the SEGBF status register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1420 and 424 |
| **Power On Value** | X'01' |
| **Restrictions** | None |

**11.8: SEGBF Total User Cells Transmitted**

This register provides a count of the total number of user cells that have been sent out regardless of VCI or VPI. Any cell assembled and sent with the most significant bit of the payload-type field reset, will cause this counter to be incremented. When the counter wraps, a status bit will be set, and an interrupt can be generated if desired.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1430 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
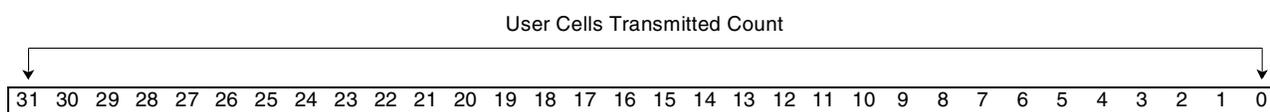
User Cells Transmitted Count

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain a count of the total number of user cells sent by this station. |

**11.9: SEGBF Total User Cells Transmitted with CLP=0**

This register provides a count of the total number of user cells that have been sent out regardless of VCI or VPI with the cell loss priority bit reset. Any cell assembled and sent with the most significant bit of the payload type field reset and the cell loss priority bit reset will be cause this counter to be incremented. When the counter wraps a status bit will be set, and an interrupt can be generated if desired. Alternatively, if bit 11 of the control register is set, this register will count the total number of operations that have been enqueued to CSKED that have been completely processed by SEGBF. By counting each operation enqueued to CSKED and monitoring this register, the software can determine how many operations are still queued for segmentation.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1434 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
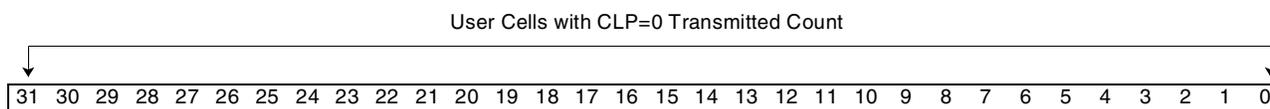
User Cells with CLP=0 Transmitted Count

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

| Bit(s) | Description |
|---|---|
| 31-0 | These bits contain a count of the total number of user cells sent by this station with CLP=0, or the total number of CSKED enqueue operations that have been completed by the segmentation logic. |

### 11.10: SEGBF Total NUD Cells Transmitted

This register provides a count of the total number of NUD cells that have been sent out regardless of VCI or VPI. The SEGBF control register can be used to exclude any of the four possible PTI values from this count. When the counter wraps, a status bit will be set, and an interrupt can be generated if desired.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1438 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

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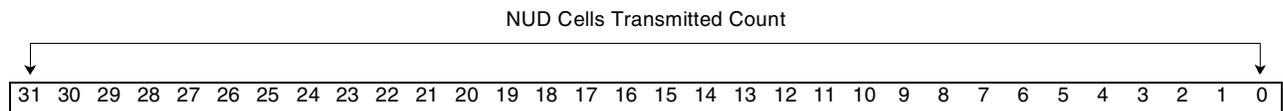
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 31-0 | These bits contain a count of the total number of NUD cells sent by this station. |

### 11.11: SEGBF Cell Queue Status

This register provides status about the number of cells queued up for transmission over the media. SEGBF can have a maximum of six cells queued up for transmission. When an LCD is scheduled for transmission, either by CSKED or software, the address of the LCD is stored in a two deep queue. As space becomes available in the four-deep cell buffer, addresses are removed from the address queue and cells are built in the cell buffers. This register provides status about the LCD address input queue as well as the cell buffers.

| | |
|---|---|
| **Length** | 11 |
| **Type** | Read |
| **Address** | XXXX 143C |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 10-8 | These three bits contain the encoded number of cells queued up for transmission in SEGBF. This number should normally correspond directly to the status defined in the low six bits of this register, with the possibility of a one cycle lag when SEGBF either passes a cell to LINKC or eliminates one based on some detected error condition. |
| 7-6 | Reserved, will always read zeros. |
| 5 | When set, this bit indicates that the second LCD address latch contains an LCD address for segmentation. |
| 4 | When set, this bit indicates that the first LCD address latch contains an LCD address for segmentation. |
| 3 | When set, this bit indicates that the fourth cell buffer contains a valid cell for transmission on the media. |
| 2 | When set, this bit indicates that the third cell buffer contains a valid cell for transmission on the media. |
| 1 | When set, this bit indicates that the second cell buffer contains a valid cell for transmission on the media. |
| 0 | When set, this bit indicates that the first cell buffer contains a valid cell for transmission on the media. |

### 11.12: SEGBF Last Active LCD Data Registers

These registers are useful in determining the contents of the last LCD that the segmentation logic has accessed.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read Only | |
| **Address** | Word Number 0 | XXXX 1440 |
| | Word Number 1 | XXXX 1444 |
| | Word Number 2 | XXXX 1448 |
| | Word Number 3 | XXXX 144C |
| | Word Number 4 | XXXX 1450 |
| | Word Number 5 | XXXX 1454 |
| | Word Number 6 | XXXX 1458 |
| | Word Number 7 | XXXX 145C |
| | Word Number 8 | XXXX 1474 |
| | Word Number 9 | XXXX 1478 |
| **Power on Value** | X'0000 0000' | |
| **Restrictions** | None | |

LCD Fetched Memory Value

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

| Bit(s) | Description |
|---|---|
| 31-00 | These bits contain the value that was fetched from memory the last time an LCD was fetched. The 32 bytes contained in these eight registers contain the LCD data significant to the segmentation logic. |

### 11.13: SEGBF PID High and Low Limit Register

This register provides a mechanism for software to specify what PID values in an MPEG-2 transport stream should be considered significant during PCR rate matching.

| | |
|---|---|
| **Length** | 29 bits |
| **Type** | Read/Write |
| **Address** | XXXX 147C |
| **Power On Value** | X'1FFF0000' |
| **Restrictions** | None |

High-Limit PID Values      Reserved      Low-Limit PID Values
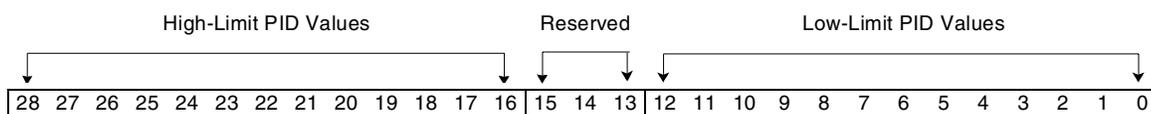
| 28 27 26 25 24 23 22 21 20 19 18 17 16 | 15 14 13 | 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|

| Bit(s) | Description |
|---|---|
| 28-16 | These bits contain the high limit for PID values during MPEG-2 PCR rate matching. If the PID in a 188-byte transport packet is greater than this value, it will not be checked for a PCR, and it will be segmented at the normal scheduled rate. |
| 15-13 | Reserved, should be written with zero. Zeros will always be read. |
| 12-0 | These bits contain the low limit for PID values during MPEG-2 PCR rate matching. If the PID in a 188-byte transport packet is less than this value, it will not be checked for a PCR, and it will be segmented at the normal scheduled rate. |

### 11.14: SEGBF Last Active LCD Address 0

This register provides feedback to the user that can be useful during initial program debug. This register and the register following this one contain the address of either the next two LCDs that SEGBF will operate on or the last two LCDs that SEGBF did operate on.

| | |
|---|---|
| **Length** | 32 |
| **Type** | Read |
| **Address** | XXXX 1460 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

| Bit(s) | Description |
|---|---|
| 31-6 | These bits contain an LCD address. |
| 5-1 | Reserved, will always read zeros. |
| 0 | When set, this bit indicates that the LCD was queued to SEGBF by software else it was queued by CSKED. |
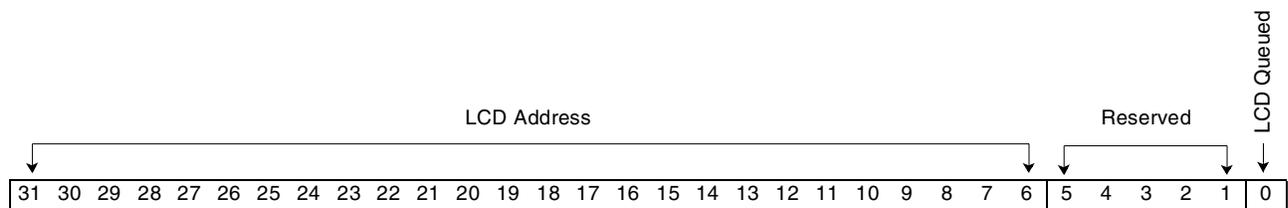
**11.15: SEGBF Last Active LCD Address 1**

This register provides feedback to the user that can be useful during initial program debug. This register and the register following this one contain the address of either the next two LCDs that SEGBF will operate on or the last two LCDs that SEGBF did operate on.

| | |
|---|---|
| **Length** | 32 |
| **Type** | Read |
| **Address** | XXXX 1464 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 31-6 | These bits contain an LCD address. |
| 5-1 | Reserved, will always read zeros. |
| 0 | When set, this bit indicates that the LCD was queued to SEGBF by software else it was queued by CSKED. |
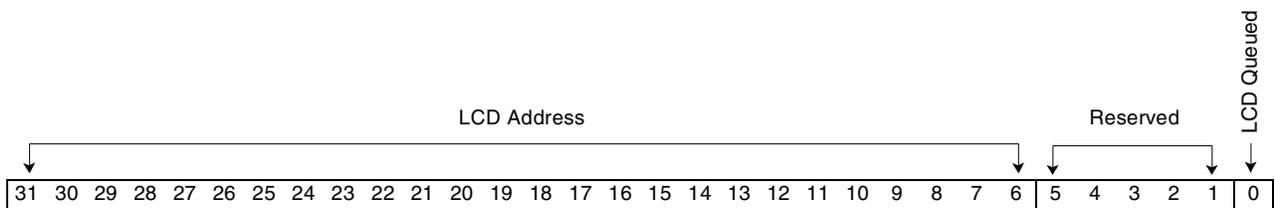
### 11.16: MPEG-2 PCR Increment Register

Each tick of the time base will add the contents of this register to the MPEG PCR Reference Register. This register contains a fixed point number with 27 bits of fraction and five bits of units. This means that the external reference clock can range in speed from 22.5KHz to the maximum speed of this entity which is TBD. Assuming that the entity will run with a 50-MHz clock, the conversion to 720KHz can be done with an accuracy of 1.1 parts in 2 million. (a clock of 19.4MHz will give a conversion accuracy of 1 part in 4.9 million). If the input clock is 19.4MHz, then the value to put in the increment register is (720,000 / 19,400,000) $* 2**27$ or 4,981,277. If the input clock is 33MHz, then the value to put in the increment register is (720,000 / 16,666,666) $* 2**27$ or 5,798,206.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1468 |
| **Power On Value** | X'002C3C9F' (33MHz) |
| **Restrictions** | None |

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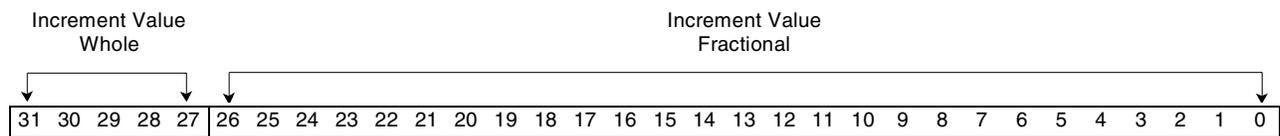
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 31-27 | These bits contain the whole part of the increment value. |
| 26-0 | These bits contain the fractional part of the increment value. |

**11.17: MPEG-2 Local PCR High**

This register contains the high 32 bits of the locally maintained PCR. Accessing this register will provide the high 32 bits of the free running local timebase used by the segmentation logic to deliver MPEG-2 traffic in a timely fashion. Although the PCR field in an MPEG-2 transport stream packet is really 42 bits, only 36 bits are significant and used by the segmentation logic. For correct operation in MPEG-2 mode, the MPEG-2 PCR increment register must be set up with the correct value so that the local PCR counts up at a rate of 720KHz. Owing to the free running nature of this value, when this register is read, the low four bits of the value are latched and can be read at a later time.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 146C |
| **Power On Value** | free running |
| **Restrictions** | None |

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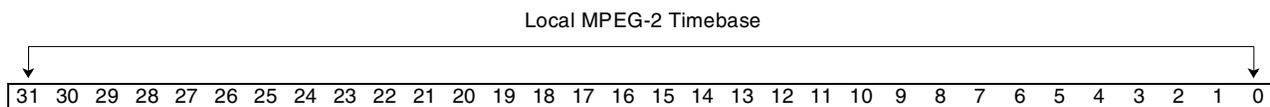
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|--------|-------------|
| 31-0 | These bits contain the high 32 bits of the local MPEG-2 timebase. |

### 11.18: MPEG-2 Local PCR Low

This register contains the low four bits of the locally maintained PCR. Accessing this register will provide the previously latched low four bits of the free running local timebase used by the segmentation logic to deliver MPEG-2 traffic in a timely fashion. The value is latched when the high 32 bits of the timebase are read. Although the PCR field in an MPEG-2 transport stream packet is really 42 bits, only 36 bits are significant to, and used by the segmentation logic. For correct operation in MPEG-2 mode, the MPEG-2 PCR increment register must be set up with the correct value so that the local PCR counts up at a rate of 720KHz.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 1470 |
| **Power On Value** | X'0' |
| **Restrictions** | None |

Low 4 bits
MPEG-2
Timebase

```
 3   2   1   0
```
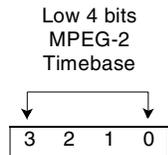
| Bit(s) | Description |
|---|---|
| 31-4 | These bits will always return zero. |
| 3-0 | These bits contain the low four bits of the local MPEG-2 timebase. |

### 11.19: MPEG-2 PID Invalidation Time

This register should be loaded with a value that indicates how long the segmentation hardware should allow a PID that has been locked on, to be absent from a data stream before forcing the hardware to unlock from that PID and begin searching for the next available PID that contains a PCR to lock on. The time base for this register is 720KHz.

| | |
|---|---|
| **Length** | 18 bits |
| **Type** | Write/Read |
| **Address** | XXXX 1480 |
| **Power On Value** | X'9AB0'(55ms) |
| **Restrictions** | None |

Unlock Inactive PIDs

```
17  16  15  14  13  12  11  10  9  8  7  6  5  4  3  2  1  0
```
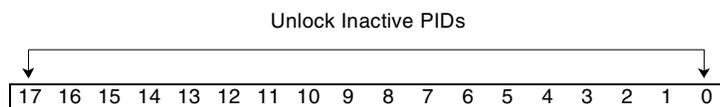
| Bit(s) | Description |
|---|---|
| 17-0 | These bits contain the value used by the segmentation logic to unlock inactive PIDs from a stream. |

**11.20: Pre-Pended Header Byte Steering Register**

This register is used to specify the location in the LCD of the bytes that are to be used in the pre-pended cell header. Four bits are used for each of the eight possible segmentation modes that can be set in the transmit portion of the LCD. The most significant two bits specify which of the words in the LCD following the CSKED/SEGBF shared region contain the bytes to prepend to the cell header. The least significant two bits specify the specific location in the eight-byte word that should be used. 720KHz.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Write/Read |
| **Address** | XXXX 1428 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

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

| Bit(s) | Description |
|---|---|
| 31-28 | These bits contain the steering bits for segmentation type 7. |
| 27-24 | These bits contain the steering bits for segmentation type 6. |
| 23-20 | These bits contain the steering bits for segmentation type 5. |
| 19-16 | These bits contain the steering bits for segmentation type 4. |
| 15-12 | These bits contain the steering bits for segmentation type 3. |
| 11-8 | These bits contain the steering bits for segmentation type 2. |
| 7-4 | These bits contain the steering bits for segmentation type 1. |
| 3-0 | These bits contain the steering bits for segmentation type 0. |
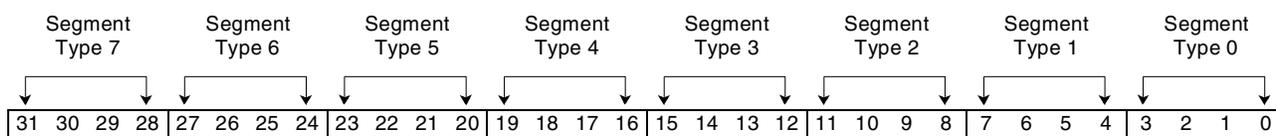
### 11.21: SEGBF Maximum LCD Size

This register should be loaded with a value that indicates the maximum number of eight-byte words that are being used by the segmentation logic. The minimum value is two. This would be the correct value if running only AAL5 with no statistics being kept on each LCD. The maximum value is seven. Setting this register to a value that is too small, will likely cause the chip to function improperly. Setting this register to a value that is too large, will adversely affect performance.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Write/Read |
| **Address** | XXXX 1488 |
| **Power On Value** | X'3'(This is the correct value when running AAL5 with statistics) |
| **Restrictions** | None |

Number of
8-Byte
Words

```
┌───┬───┬───┐
│ 2 │ 1 │ 0 │
└───┴───┴───┘
```
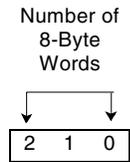
| Bit(s) | Description |
|---|---|
| 2-0 | These bits contain the number of eight-byte words in the LCD to be used by SEGBF during segmentation |

**11.22: SEGBF Internal Status**

This register provides status regarding the current state of the internal SEGBF state machines and various other status bits. It is intended for to assist in debug.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read |
| **Address** | XXXX 1484 |
| **Power On Value** | X'0000' |
| **Restrictions** | None |

| Bit(s) | Description |
|---|---|
| 15 | When set, this bit indicates that the first word of an LCD is being latched |
| 14-12 | These bits provide the current state of the main state machine |
| 11-8 | These bits provide the current state of the control memory access state machine |
| 7 | Reserved |
| 6-5 | These bits provide the current state of the packet memory access state machine. |
| 4 | Reserved |
| 3 | When set, this bit indicates that the current MPEG-2 frame is early and will not be sent. |
| 2-0 | These bits provide the current state of the internal multiply state machine used to generate packet length for fixed-size block mode. |
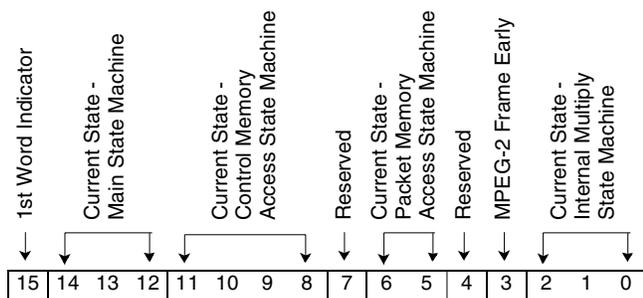
**11.23: SEGBF Cell Staging Array**

This array is divided into four 64-byte buffers used to assemble cells that are ready for transmission on the line.

| | |
|---|---|
| **Length** | 32 Words X 64 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1500 - 5FF |
| **Restrictions** | This array can only be accessed when the diagnostic mode bit in the control register is set. |

# Receive Data Path Entities

## Entity 12: Cell Re-Assembly (REASM)

REASM is the entity which:

- Collects the bytes from LINKC into cells.
- Discards null cells and idle symbols.
- Looks up the VPI/VCI control block in Control store.
- Provides the barrel shifter data interface to packet memory.
- Check and correct the HEC.

REASM does these tasks with a 256-byte FIFO receive buffer which can buffer up to four cells worth of data.

The figure below is a block diagram of the REASM entity.

## REASM Block Diagram
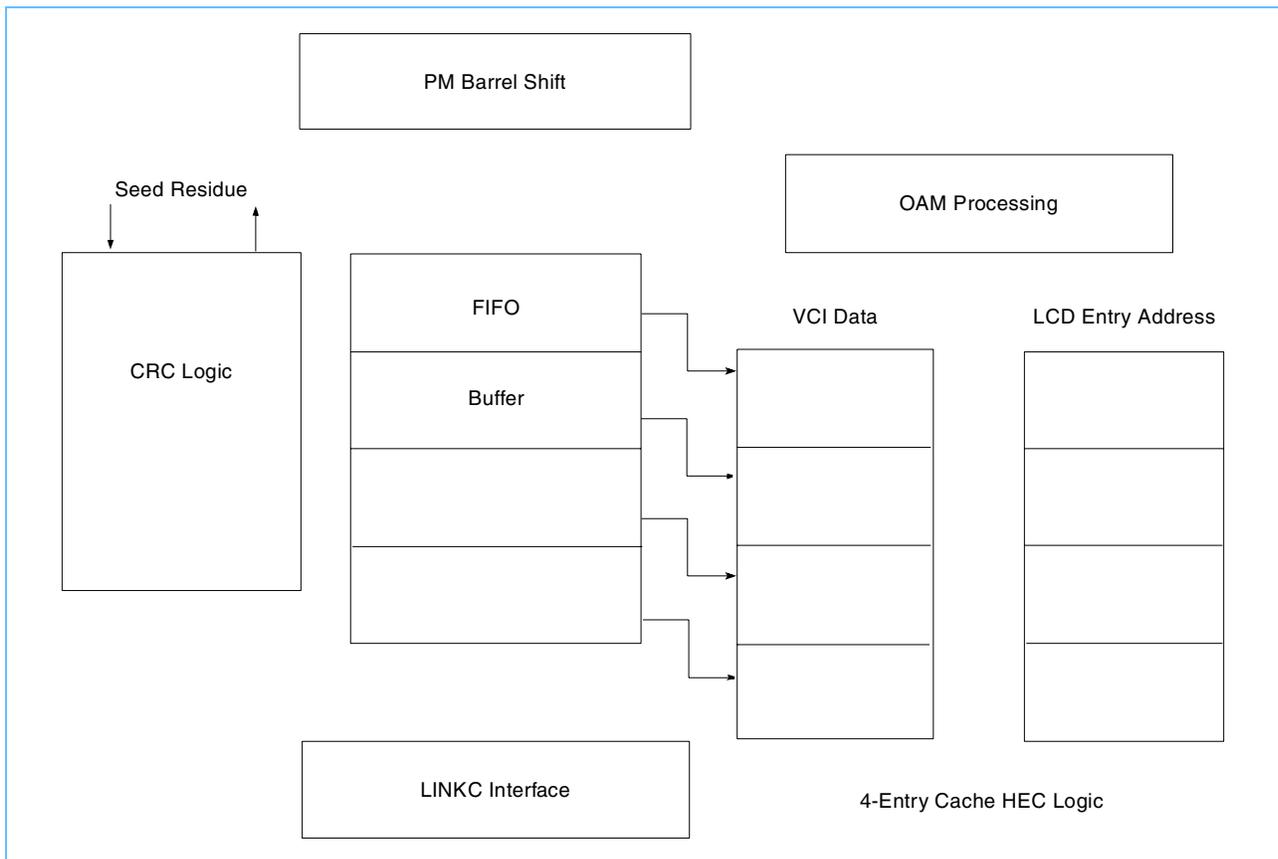


REASM buffers the incoming cells and looks up the proper Logical Channel Descriptor (LCD) address for the cell. It then passes the LCD pointer to RAALL and the cell data to Packet Memory. The lookup of the LCD from the cell data has just one level of indirection. The Logical Channel Descriptor Translate Table is a data structure stored in Control memory and maintained by software.

## VPI/VCI -ž LCT Entry Mapping Function
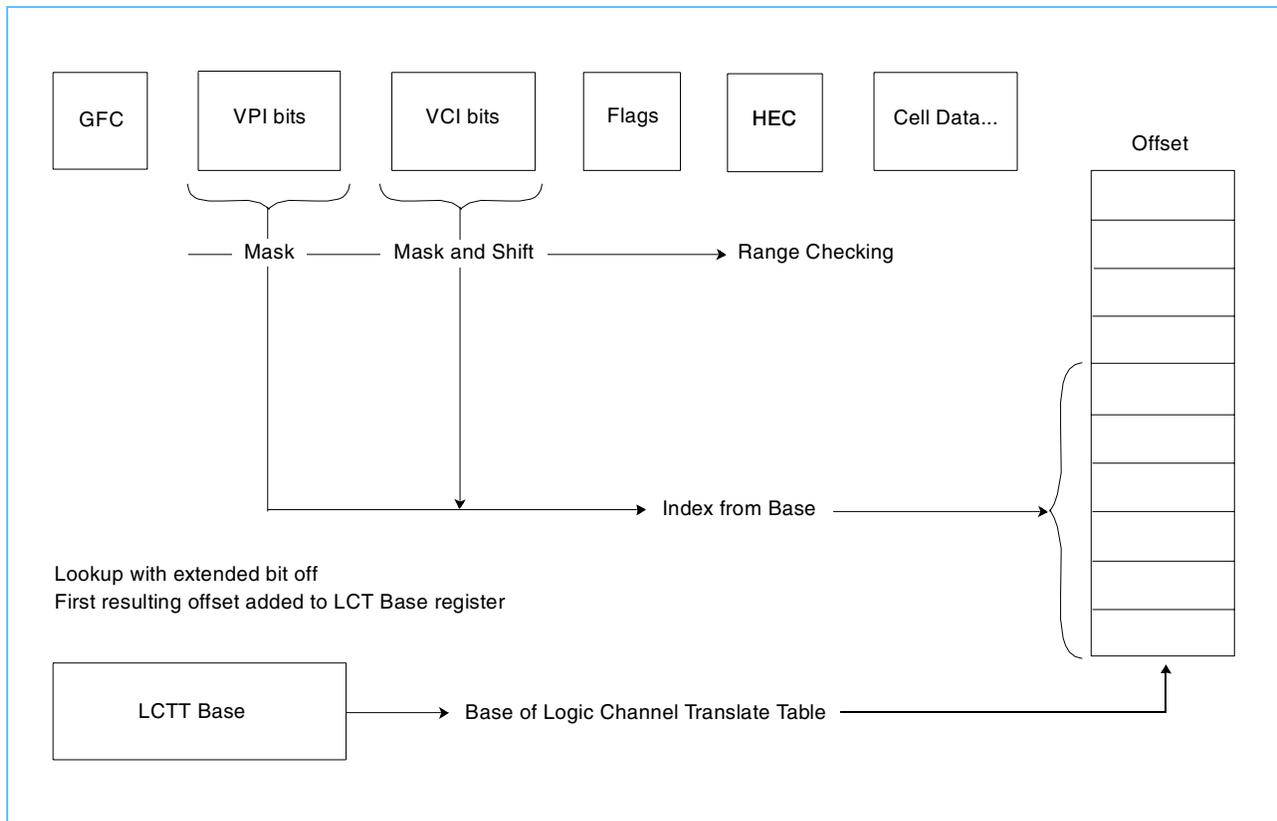
## 12.1: REASM Control Register

Used to control REASM options. Used to determine which bits of the VPI, VCP and MID will be used. Also used to determine the acceptability of cells with fixable and non-fixable HEC errors. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 22 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1600 and 604 |
| **Power On Value** | X' 0010' |
| **Restrictions** | None |



| Bit(s) | Description |
|---|---|
| 21 | Receive cells that have bits in the VPI-VCI that are out of range into the LCD specified in the REASM Out Of Range LCD Register. RAALL will generate an event that contains the cell address on a queue specified in RAALL. |
| 20 | Receive cells that have an index of zero in the VCI to LCD translation table. No index for the LCD is generated. RAALL will generate an event that contains the cell address on a queue specified in RAALL. |
| 19 | Receive cells that have bits in the VPI-VCI that are out of range. No index for the LCD is generated. RAALL will generate an event that contains the cell address on a queue specified in RAALL. |
| 18 | This bit will disable the reassembly VCI to LCD translate cache when set. |
| 17 | This bit will flush the reassembly VCI to LCD translate cache when set. It will reset when the flush is complete. |
| 16 | When set, this bit will reset all control logic in the entity. After being set, this bit must be reset before the logic will function properly. This bit must remain set for at least. 1 μs to reset the reassembly logic properly. |
| 15 | Count Non-user cells instead of HEC errors. |
| 14 | Diagnostic bit. Must be set to enable PCI bus access to the array or write to the internal state machine register. Setting this bit will prevent cells from being processed. |
| 13 | Receive cells with VPI/VCI out of range using the valid bits for generating the index for the LCD. |
| 12 | Ignore HEC errors. HEC errors will not be reported if this bit is set. |
| 11 | Receive cells with invalid HEC. Cells with HEC errors will be received on a queue specified in RAALL. |
| 10 | Disable HEC correction. |
| 9-5 | Number of bits of the VPI (0-12) that make up the table lookup address. |
| 4-0 | Number of bits of the VCI (0-16) that make up the table lookup address. |

## 12.2: REASM Status Register

Used to relay REASM status information. This register contains status for the following conditions.

| | |
|---|---|
| **Length** | 11 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1608 and 60C |
| **Power On Value** | X'00' |
| **Restrictions** | None |

Bit fields (left to right, bits 10–0):
Out-of-Range Counter Overflow (10), Correctable Error Counter Overflow (9), Uncorrectable Error Counter Overflow (8), Received Counter Overflow (7), Received w/ CLP=0 Counter Overflow (6), Non-Null Cell Received (5), Reserved (4), Cell VPI/VCI Out of Range (3), Correctable HEC Error Detected (2), Uncorrectable HEC Error Detected (1), Overrun Detected (0)

| Bit(s) | Description |
|---|---|
| 10 | Cell Address Out Of Range Counter overflow. |
| 9 | Cell HEC Correctable Error Counter overflow. |
| 8 | Cell HEC Uncorrectable Error Counter overflow. |
| 7 | Total User Cells Received Counter overflow. |
| 6 | Total User Cells Received with CLP=0 Counter overflow. |
| 5 | Non-Null Cell received. |
| 4 | Reserved. |
| 3 | Cell VPI/VCI out of range or index of zero in VPI/VCI to LCD translate table detected. |
| 2 | Correctable HEC error detected. |
| 1 | Uncorrectable HEC error detected. |
| 0 | Overrun detected. |

## 12.3: REASM Interrupt Enable Register

Used to enable interrupts for REASM status conditions. If set, the specified condition will generate an interrupt from REASM.

| | |
|---|---|
| **Length** | 11 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1610 and 614 |
| **Power On Value** | X'08' |
| **Restrictions** | None |

Bit layout (bits 10 down to 0):

- 10 — Out-of-Range Counter Overflow
- 9 — Correctable Error Counter Overflow
- 8 — Uncorrectable Error Counter Overflow
- 7 — Received Counter Overflow
- 6 — Received w/ CLP=0 Counter Overflow
- 5 — Non-Null Cell Received
- 4 — Reserved
- 3 — Cell VPI/VCI Out of Range
- 2 — Correctable HEC Error Detected
- 1 — Uncorrectable HEC Error Detected
- 0 — Overrun Detected

| Bit(s) | Description |
|---|---|
| 10 | Cell Address Out Of Range Counter overflow. |
| 9 | Cell HEC Correctable Error Counter overflow. |
| 8 | Cell HEC Uncorrectable Error Counter overflow. |
| 7 | Total User Cells Received Counter overflow. |
| 6 | Total User Cells Received with CLP=0 Counter overflow. |
| 5 | Non-Null Cell received. |
| 4 | Reserved. |
| 3 | Cell VPI/VCI out of range or index of zero in VPI/VCI to LCD translate table detected. |
| 2 | Correctable HEC error detected. |
| 1 | Uncorrectable HEC error detected. |
| 0 | Overrun detected. |

## 12.4: REASM Logical Channel Table Base Register

The REASM Logical Channel Table Base Register indicates the starting address of the logical channel table. This register defines where the Logical Channel Descriptors are located.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1618 |
| **Power On Value** | X'0000 8000' |
| **Restrictions** | The value must be in the range of the physical memory allocated for control memory. |

## 12.5: REASM Logical Channel Translate Table Base Register

This register defines where the VCI to LCD translate table is located. When a cell is received, the VPI/VCI fields are used to generate an index into this table. Each entry of the translate table contains a 16-bit index which specifies which LCD in the Logical Channel Table corresponds to the received VPI/VCI. All unused entries should be initialized to zero, or to the index of an LCD that is intended to receive all unexpected VPI/VCI combinations.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 161C |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

## 12.6: REASM Cell Address Out of Range Counter

Counts cells with invalid VPI/VCI fields. The counter will increment if the VPI/VCI received is not in the range specified in the REASM control Register bits (9-0), or the LCD index corresponding to the received VCI in the REASM Logical Channel Table Table Base Register is zero.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1628 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

### 12.7: REASM Cell HEC Correctable Error Counter/Non-user Cell Counter

Counts cells with correctable HEC errors or non-user cells selectable by bit 15 in the REASM Control Regis-
ter. The counter will increment when a cell with a correctable HEC is received.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 162C |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

### 12.8: REASM Cell HEC Uncorrectable Error Counter/RM Cell Counter

Counts cells with uncorrectable HEC errors or RM cells selectable by bit 15 in the REASM Control Register.
The counter will increment when a cell with an uncorrectable HEC is received.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1630 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

### 12.9: REASM Total User Cells Received Counter

Counts total user cells received. This counter will increment each time a user cell is processed. This count
includes cells with HEC errors and cells with VPI/VCIs that are not valid.

.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1634 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

**12.10: REASM Total User Cells Received with CLP=0 Counter**

Counts total user cells received with CLP bit equal to zero. The counter will increment whenever a user cell is received with the Cell Loss Priority bit equal to zero.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1638 |
| **Power On Value** | X'0000 0000' |
| **Restrictions** | None |

**12.11: REASM Out Of Range LCD Register**

Defines the LCD that all out of range VPI-VCIs will be received on. This register will be used to determine the LCD that cells that have VPI-VCI values that are out of range. The high 26 bits are used to select the LCD, the low six bits are ignored.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 164C |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 12.12: REASM State Machine Register

Returns the state of REASMs internal state machines. This register contains the current state of REASMs state machine variables. It is to be used for debug purposes only. It should not be written during normal operation.

| | |
|---|---|
| **Length** | 10 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1648 |
| **Power On Value** | X'000' |
| **Restrictions** | Diagnostic mode must be set to write this register. |

### 12.13: REASM Cell Staging Array

Provides access to receive buffer. This array is divided into four 64-byte buffers used to assemble cells received from the line.

| | |
|---|---|
| **Length** | 64 Words x 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1700-7FF |
| **Power On Value** | X'00000000' |
| **Restrictions** | This array can only be accessed when the diagnostic mode bit in the control register is set. |

## Entity 13: Receive AAL Processing (RAAL)

**Functional Description**

RAALL implements the receive functions of the ATM Adaptation Layers (AAL). The following functions are supported:

- Raw Cell Mode
    - Routed/Spliced
    - Cut-Through
    - FIFO Mode
- AAL5 - High Speed Data Transfer w/ Variable bit Rate
    - Routed/Spliced
    - Mode 6/7 cut-through
    - Scatter Mode

The AAL functions include:

- Protocol Verification
- Cell Reassembly into Packets
- Cell CRC verification
- Packet Reassembly Timeouts and Errors

RAALL also performs:

- LC based statistics
- OAM F5 Blocking support
- Packet Thresholding for cut-through support

**Reassembly Timeout (RTO) Processing**

Reassembly timeout processing is supported for AAL 5/6/7 LCs. It can be enabled on an LC basis by turning on the RTO enable bit in the LC. The following registers also need to be properly set up to run RTO processing:

- RAALL LC Table Bound Registers
- RAALL Reassembly Timeout value Register
- RAALL Reassembly Timeout Pre-Scaler Register

See the register descriptions for more register details.

The LC table registers define the LC table that the RTO processor examines. The value register is used as a compare value against a counter that counts based on a pre-scaler. Each time the registers compare, RTO processing is started for a single LC and the time base is reset. RTO processing checks the RTO bit. If it is reset, it sets it and continues. If it is set, then a timeout occurs and the LC is placed in error state and the current packet is either freed or surfaced to the user via an event. The RTO bit is reset with each inbound cell received or when a packet either completes or goes into error state.

**Note:** If the RTO bit is set when RTO processing is disabled, it will remain set unless the LC goes into error state. An LC needs to be touched twice to cause a timeout (once to set it and once to detect that it is already set).

The time base starts running as soon as the RTO processing is complete. Currently, the RTO processing takes the back seat if there are cells to be processed. So, RTO processing can be held off indefinitely.

**Note:** For AAL0 FIFO RTO, it is assumed that the software will reset all counts and/or remove all cells from the FIFO before resetting the state to idle. This needs to be done in order to maintain the reassembly count!

**LC Statistics**

If enabled, RAALL maintains LC level statistics. The following statistics are kept:

- Total User Cells Received
- Total User Cells Received with CLP=0

Using these numbers, the Total User Cells Received with CLP=1 can be calculated.

Both are 32-bit counters that wrap on overflow. Using the RAALL LC Statistics Overflow Register, the overflow behavior of the counters can be changed to overflow on a value other than 0xffffffff. If enabled in the mode reg, software is notified of overflow events via the counter overflow event queue specified in RXQUE.

Statistics can be enabled on a LC basis by turning on the statistics enable bit in the LC. Statistics can be globally enabled/disabled across all LCs by setting the appropriate bit in the RAALL mode register. The global enable/disable overrides all lc enables, and the global disable overrides the global enable.

If OAM blocking is enabled, then you might as well turn on statistics because you get them for free.

### OAM F5 Blocking Support

If enabled, RAALL maintains OAM F5 blocking statistics on an LC basis. The following statistics are kept:
- Total User Cells Received - TUC
- 16-bit Interleaved Parity - BIP-16

Both are 16-bit values.

Blocking can be enabled on a LC basis by turning on the blocking enable bit in the LC. Blocking can be globally enabled/disabled across all LC's by setting the appropriate bit in the RAALL mode register. The global enable/disable overrides all LC enables, and the global disable overrides the global enable.

When an OAM F5 flow cell is received, the current receive TUC and BIP-16 are appended in the first full eight-byte word after the cell data (depends on what offset is set to in the LC). This information can then be used along with the values in the cell to complete the reporting/monitoring process. The BIP-16 is reset to zero and the TUC takes the value from each PM cell. By doing this, the software does not need to maintain any temporary copies of these fields. The PM processing does not affect the LC statistics fields.

**Note:** If OAM blocking is enabled, you might as well turn on statistics because you get them for free.

### Bad Cell Support (Bad HEC, VP/VC Out of Range, and VC Index Equal Zero)

If reception of bad cells is enabled in REASM, then bad cells are received as 53-byte cells using the non-user data registers to select the receive queue, offset, and POOL ID. The LC pointer in the packet header is zeroed. These cells are only received if receive bad frame mode has been turned on in RXQUE. This function may be desirable for network tracing tools.

### Raw Cell Routing Support

Raw Cells (53, 52, or 48 bytes) can be routed back out the transmit interface. Normally, when a cell is received, the receive LCD is written into the packet header and the buffer is surfaced to the user. When routing is enabled, by turning on the MSB of the LCD state field, the second word or the receive LCD is used as the data to fill in the packet header LCD field. The buffer is then sent to the scheduler and rescheduled for transmission. This allows cells to be routed out the transmit interface with the same or different VP/VC. When routing cells, the default is to surface non-user data cells to the user. By turning on bit two of the LCD state field, all non-user data cells will be routed the same way as user data cells for that LCD. The user should be sure to turn on the Free On Transmit bit in the second word of the LCD so the cell buffers are freed when they are re-transmitted. The original CLP bit from the inbound cell is always written to the LSB of LCD field in the packet header.

### General Packet/Cell Buffer Layout

When not using FIFOs, all cells and packets are assembled in packet buffers. These buffers have a packet header followed by the data section. The following figure shows the general layout of AAL5 packets and AAL0 cells:

## Packet/Cell Layout in Packet Buffer

AAL 0 and NUD Cell

| Packet Header (16 Bytes) |
| --- |
| Pad (0-7 fBytes) |
| Cell Data (48 - 53 Bytes) |

OAM Blocking Cell

| Packet Header (16 Bytes) |
| --- |
| Pad (0-7 fBytes) |
| Cell Data (48 - 53 Bytes) |
| Pad (0 - 7 Bytes) |
| OAM Data (8 Bytes) |

AAL 5 Packets

| Packet Header (16 Bytes) |
| --- |
| Pad (0-7 fBytes) |
| Packet Data (0 - 64 Bytes) |
| AAL 5 Trailer (8 -48 Bytes) |
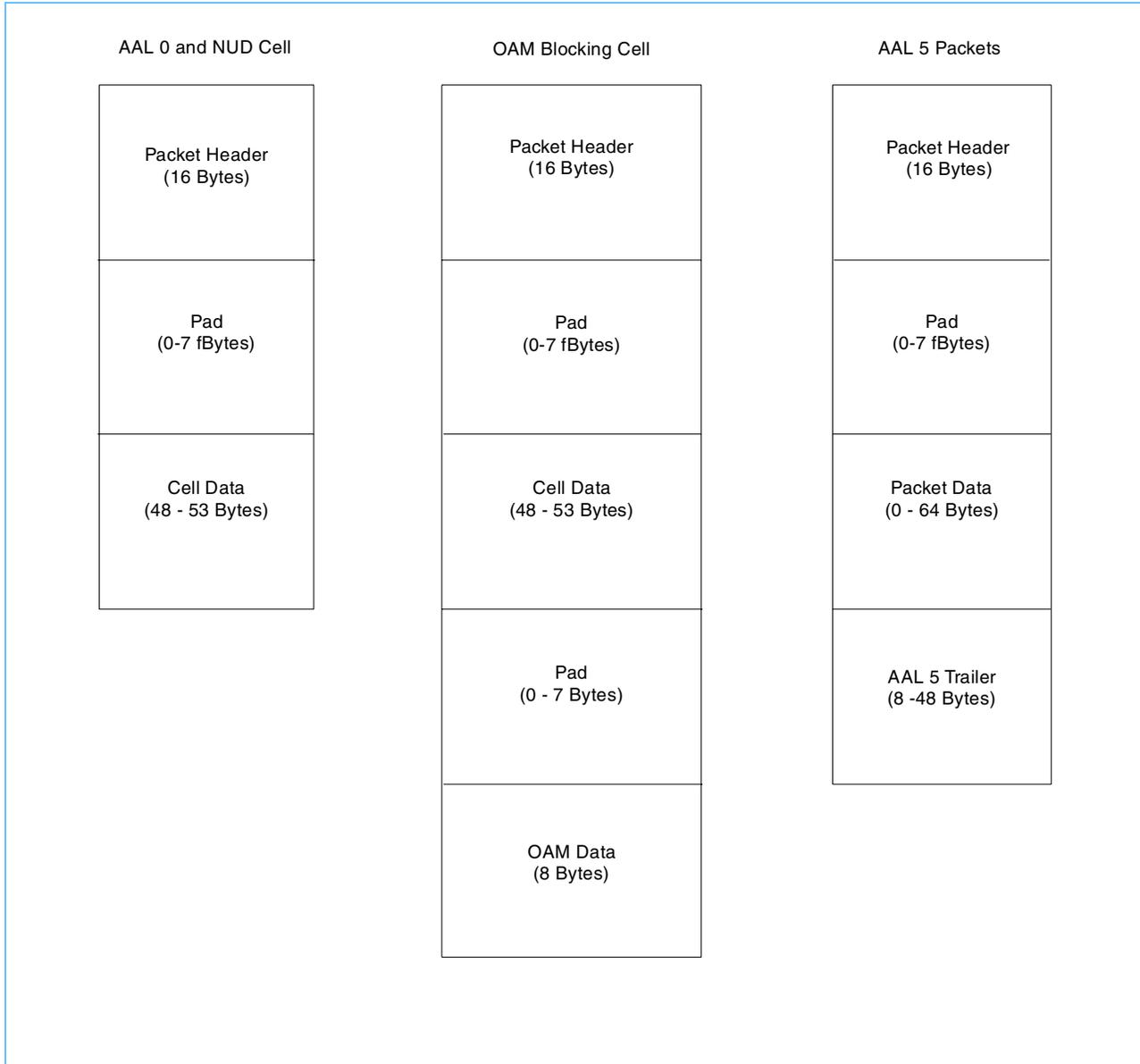
The first pad is determined by the receive offset field in the LC or the NUD receive offset register. In the case of a blocking cell, the second pad is generated to align the current statistics on an eight-byte boundary. In the AAL5 packet, the AAL5 trailer is received into the packet buffer as it was received. So, all pad bytes are also received.

It would be a good idea to have a minimum buffer size of 128 bytes. 64-byte buffers will work only if receiving raw cells in 48-byte mode with zero offset. Also, no OAM cells would be able to be received.

### Shutting Down an LCD

To shut down an LCD for receive, the following steps should be followed:

1. Clear the entry for this LCD in the LCD table table (to stop receiving cells for this LCD).

2. Do an LCD update operation that sets LCD state to down.

3. Read the LCD reassembly pointer.

4. If the reassembly pointer is non-zero and LCD is set up to do cut-through, be sure to free any DMA descriptor that was added with the cut-through operation.

5. If reassembly pointer is non-zero, free it to POOLs.

6. If reassembly pointer is non-zero, decrease the reassembly count by writing the reassembly count register.

If reassembly timeout is being used, the following sequence can be used:

1. Clear the entry for this LCD in the LCD table table (to stop receiving cells for this LCD).

2. Wait for reassembly timeout event.

3. If set up for cut-through, a DMA canceled event will also flow.

4. At this point, do an LCD update operation that sets LCD state to down.

5. If set up to receive the packet address in the event, free it to POOLS.

### Performing an LCD Shutdown of a Cut-Through LCD

There are two possibilities here. The first uses the timeout feature and is immeasurably preferable since it makes the best use of the on-chip mechanisms for resource allocation and control. To shut down the cut-through LCD, use the following procedures.

### Using the Timeout Feature

The key here is to use the Reassembly Timeout feature. One can shut down 100s or 1000s of LCDs and then wait for a short time (approximately 100 microseconds) and everything will return to normal.

You may use the second set of Reassembly Timeout registers allowing you to leave the first set alone and use the second set for the shutdown mechanism.

1. Clear the LCDs entry in the LCD table table (the VCI to LCD translate table in REASM, offset 0x161C). This halts cell reception for this LCD.

2. Flush the REASM cache.

3. Wait for the Reassembly Timeout event (see Note below). IBM2520L8767 now cleans up resources used for this LCD.

4. If the LCD is set for cut-through AND it has a DMA descriptor enqueued to the packet, awaiting packet completion, a DMA Cancelled event will follow the Reassembly Timeout.

5. Now the state of the LCD can be checked to determine if it is 'Error' or 'Idle'. Either way, the LCD is now shut down and any other administrative processing may be done.

6. Using the LCD Update Operation Register (RAALL), set the state of the LCD to 'DOWN'. This is the 'state' field of the Receive LCD and it is set to 0x00.

**Note:** The reassembly timeout process can be considerably sped up by temporarily changing the LC Table Lower Bound Register and the LC Table Upper Bound Register in RAALL. Because it is known which LCD(s) is (are) being shut down, the lower and upper bounds can both be set to the LCD entry you are shutting down. That way, only one LCD is examined at the timeout rather than an entire range of LCDs (as defined by the lower/upper bound registers). Once this is completed, the bounds can be reset to their original values and everything is back to normal.

### Manual Shutdown

This is much more complicated since resources must be checked and freed as required while shutting down an LCD. There are three specific mechanisms depending on whether the LCD is Mode 6, Mode 7, or (DMA) scatter mode. You will have to adapt the mechanism to your specific system. This explanation is intended to give a general understanding as to which issues must be considered.

Because the LCDs are being shut down manually, you must be particularly aware of which resources are in use and free those that will not be used. Resources to keep in mind are packet buffers, DMA descriptors, and caches. Remember also to free the packet (if needed) at the finish of a shutdown.

1. Clear the LCDs entry in the LCD table (the VCI to LCD Translate Table in REASM, base address is offset 0x161C). This halts cell reception for this LCD. (Same as in I.)

2. Flush the REASM Cache. This guarantees that the previous step is utterly complete.

3. Read the state of the LCD state variable in the LCD and perform the following steps:

| Step | If ... | Then ... |
|------|--------|----------|
| 1 | state == DOWN | Done. |
| 2 | state == IDLE | Using LCD Update Op, set state to DOWN. Done. |
| 3 | state == REASM | Decrement the REASM Counter. This is very important! |
| 4 | mode == mode 6 | Read the REASM pointer and free the pointer. Set LCD state to DOWN. Done. |
| 5 | mode == mode 7 (not using scatter/gather) | Read the LCD descriptor state variable. |
| | descriptor state var == FALSE | Set LCD state to DOWN. Free the packet. Done. |
| | descriptor state var == TRUE | Free the descriptor. Set LCD state to DOWN. Free the packet. Done. |
| 6 | scatter/gather MODE | Check the LCD NumDesc field. |
| | NumDesc == 0 (no descriptors) | Set LCD state to DOWN. Free the packet. Done. |
| | NumDesc != 0) (free descriptors) | Either step through descriptor list in packet and free each one, or use DMA engine to do this. NOTE: freeing the descriptor will put the associated host page back on the available list so the host pages do not need special treatment. Set LCD state to DOWN. Free the packet. Done. |
| 7 | state == ERROR | Set LCD state to DOWN. Done. |

### AAL5 Packet Thresholding for Cut-Through Support

RAALL supports packet header thresholding for AAL5. The threshold is set in the LCD. Once a number of bytes $\geq$ the threshold value has been received, a packet threshold event is raised to the user. At this point, the packet header contains valid offset and length values, and the ATM header field in the packet header is from the first cell received for the current packet. This allows the user to inspect the packet header and initial packet data before the entire packet has been reassembled.

The user has two options for determining when the packet has been reassembled. The two options are controlled by the appropriate bit in the control register. First, the user can choose to have a packet complete event raised when the packet completes. Second, the user can poll the packet header to watch for the packet to complete. The polling method utilizes the done and error bits in the packet header.

The method used depends on how the user is doing the cut-through processing. For example, if correlating two events is a problem, the polling method should be used.

**Note:** In both methods error events are always raised. When using the single event mode, it is required that the receive queues are not allowed to reach a full condition or that there is enough receive queue room for all threshold events. If this is not the case, buffers can be lost if threshold events are flushed due to queue full conditions.

### Rx AAL 5/6/7 Cut-Through Support

Cut-through support allows software to be written that minimizes latency and the number of interrupts (events) that occur. Cut-through support uses two mechanisms to accomplish this, automatic header DMAs and software-assisted DMAs upon packet completion.

### Automatic Header DMAs

Automatic header DMAs occur as the cells arrive for a packet. As the amount of receive data exceeds the configurable threshold, a DMA is scheduled that DMAs the packet header and data into a system buffer specified by a software provided system DMA descriptor. An event can be placed on a receive queue after the DMA is complete. At this point, software has all of the header data in system memory and can operate on it in an efficient manner. The threshold should be set to a value that is large enough so that software can process the entire header and set up a subsequent DMA to move the protocol payload data into a system buffer when it arrives. For example, given enough of the protocol headers, the length of a TCP/IP packet can be derived, and the session info can be derived. Given this info, a DMA can be setup to do the TCP/IP checksum and move the data into a system/user buffer upon packet completion. This is referred to as Software assisted DMAs on packet completion.

### Setting up an LCD for RX Cut-Through Support

To set up an LCD for receive cut-through support, the all type must be set to 0x7. By doing this, the most significant three protocol enable flags are redefined to be the receive queue number from which header the DMA descriptors are obtained. This receive queue should be set up to run in reverse direction, and should be filled by software with system DMA descriptor addresses. This queue should not be allowed to go to an empty state or headers will be delayed and eventually packets will be thrown away if the packet completes before the header was scheduled to be DMAed.

In addition, the header threshold value in the LCD needs to be set up. The value in this field determines when the header is DMAed. Once the number of bytes of receive data exceed the value in this field, the header is eligible to be DMAed. If a DMA descriptor is available from the specified receive queue, the DMA is scheduled. If no DMA descriptors available, then an attempt to schedule the DMA will be made when the next cell is

received. If the packet is completed and no DMA descriptors are available and the header has not been DMAed, the packet is discarded. If the packet is completed before the threshold is met, the entire packet is DMAed using a header DMA descriptor.

### DMA Descriptors used for Header DMAs

The DMA descriptor placed in the DMA descriptor receive queue, must be valid descriptor. The source address and length do not need to be filled in the first descriptor, as these are filled in by the IBM2520L8767 when the header DMA is scheduled. Either a single descriptor or a compound descriptor can be used (make sure the number of descriptors is filled in least significant bits when they are placed in receive queue just as if they were being enqueued to GPDMA). The user can be notified when the DMA is complete in several ways. Either the notify on completion flag can be set, or the second descriptor can be used to enqueue an event to a receive queue.

### Doing Software Assisted DMAs on Packet Completion

After processing the packet header, an arbitrary system DMA descriptor can be built for the packet. The packet address can be found in the sixth word of the packet header.

**Note:** This implies that the receive offset is at least 24 bytes.

The address of the DMA descriptor (and the number of descriptors in low order bits) is then written into the RAALL Cut-Through Descriptor Address Registers. Once the descriptor address is written, a RAALL Cut-Through Operation is performed by writing the RAALL Cut-Through Operation (CTOP) Registers with the base address of the packet. RAALL then looks at the current state of the packet. If the packet is complete and has no errors, DMA is immediately scheduled. If the packet is complete, but there were errors, a DMA canceled event is placed on the appropriate receive queue, and the event data contains the system DMA descriptor address of the descriptor that would have been used (the packet is not freed). If the packet is still being reassembled, then the LCD is marked for processing when the packet completes or an error occurs. If the LCD was marked for later processing, when a packet completes, the DMA descriptor is scheduled or a DMA canceled event is posted, depending if errors occurred.

### Alternate Header DMA method

Normally the header data DMA is scheduled as soon as the threshold is crossed and a descriptor is available. Alternatively, the AAL type can be set to six instead of seven. When set to six, the header DMA is not scheduled until the packet is complete.

### Receive AAL0 and Non-User Data Cut-Through Support

Cut-through can also be used in non-FIFO AAL0 modes and can be set up for non-user data cells.

For AAL0 LCDs, the RTO enable bit in the LCD can be set to enable cut-through processing. When this bit is set, the rxq_num field is used to get DMA descriptor instead of specifying which queue to place events on. In this mode, for each cell that is received, a DMA descriptor is obtained from the specified receive queue. The DMA descriptor is then scheduled with GPDMA. If there is no DMA descriptor avail, then a no descriptor event is enqueued.

Cut-through for non-user data cells is similar. The receive queue from which DMA descriptors are obtained, is specified in the RAALL Non-User Data Config Register. Cut-through for non-user data cells is enabled by setting bit 15 in the RAALL Control Register.

**AAL5 Scatter Support**

Scatter (as in Scatter/Gather) is now supported for AAL5. The following parameters need to be specified:

- System page size (see *RAALL Scatter Page Size and Queue Register* on page 286)
- Rxque queue number that pages are stored in (see RAALL Scatter Page Size and Queue Register)
- First/Single Page Size if used (see RAALL Scatter Page Size and Queue Register)
- Number of header data bytes (in LCD)
- Rxque queue number that holds the header DMA descriptors (in LCD)

The following figure shows the format of a packet and how it is divided for DMAs:

## Scatter Packet Buffer Layout

| | |
|---|---|
| 0 x 18 | Packet Header |
| Var | DMA List |
| Off | Pad |
| | Num Head Bytes |
| | Page 1 |
| | Page 2 |

| | |
|---|---|
| 0 x 18 | Packet Header |
| Var | DMA List |
| Off | Pad |
| Single Page Size | |

Single Page Mode

| | |
|---|---|
| | Page N |
| | Page N + 1 |
| | Pad & AAL 5 Trailer |
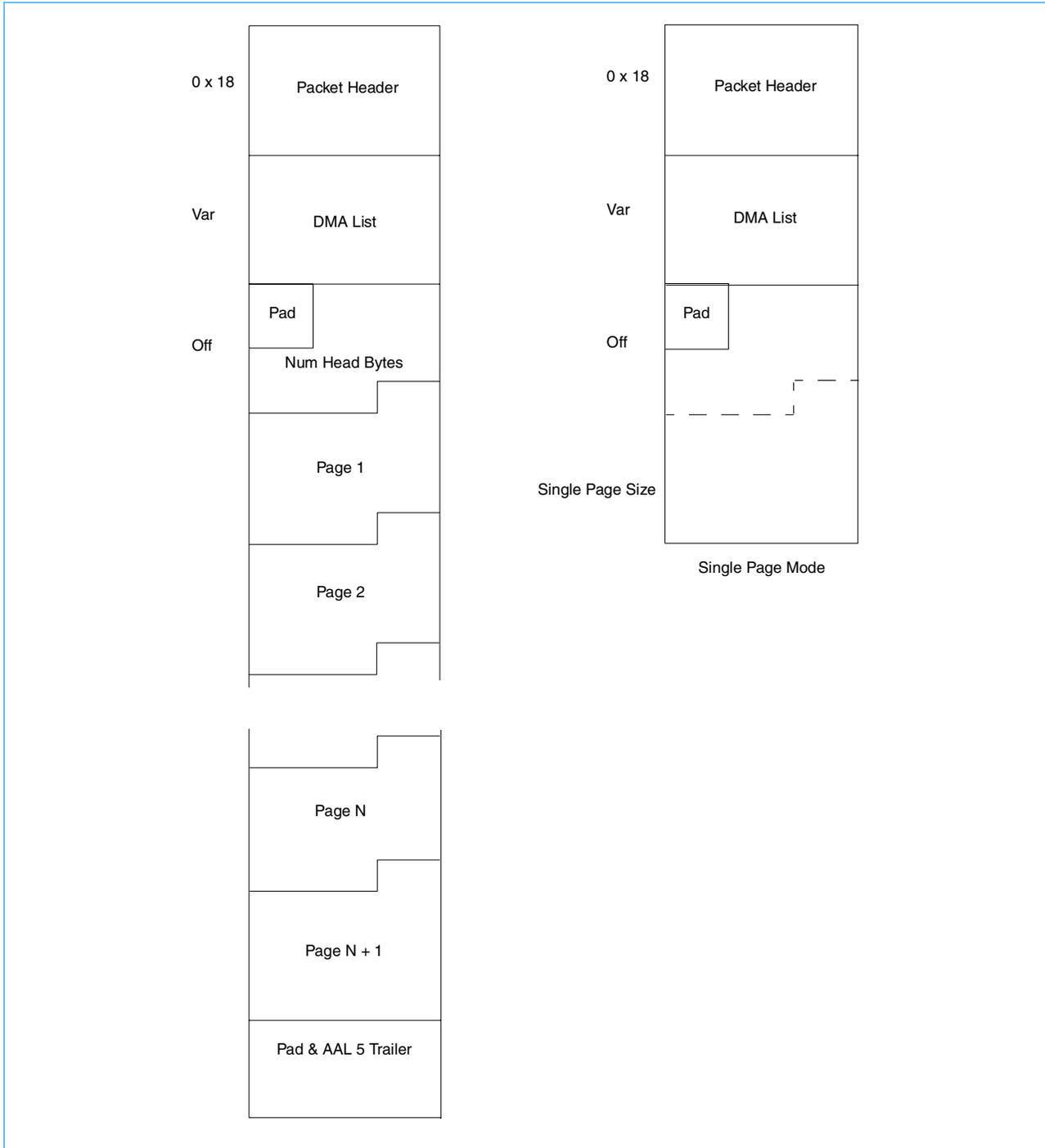
The packet header is 0x18 bytes long when doing scatter mode. The last four bytes of the packet header contains the buffer address in the upper bits, and the lower bits contain how many real buffers are in the DMA list that follows.

The DMA list follows immediately after the packet header and is variable length based on how many pages were needed to DMA the packet into the system. The receive offset specified in the LCD, must be large enough to hold the packet header and the maximum DMA list that will be used.

There may be some pad bytes based on the value of the receive offset specified in the receive LCD. This can be used to align the packet data on some boundary (eight-byte boundary is optimal for IBM2520L8767).

Next is the actual packet received from the network. It is divided into a header area, followed by 0-N fully populated pages, possibly followed by a partial page, followed by the AAL5 pad and trailer bytes. All of these areas are contiguous in the IBM2520L8767. The header area is 0-255 bytes as specified in the LCD. This area is kept with the packet header and DMA list when DMAed to the system. This allows the device driver to split protocol information and user pages before moving to the system.

Complete pages are DMAed to the system as the packet is received. When a complete page has been received, a real system page address is obtained from the specified receive queue number. Using this address, the IBM2520L8767 address, and the page size, a DMA descriptor is built and enqueued to DMAQS, and the address is written to the DMA list in the packet header. When the packet completes (uind=1 in last cell), the final page is DMAed to the system, and the DMA list is updated. The lower bits of this final DMA list entry contains how many bytes were in page N+1 (zero if page was complete). Once the DMA list is updated, a DMA descriptor is obtained from the receive queue number specified in the ctRxqNum in the LCD. This final descriptor is used to DMA the data in packet header through the header bytes into a system specified address. This mechanism is similar to normal cut-through modes, so the IBM2520L8767 buffer can be freed using subsequent DMA descriptors.

**Single Page Mode**

For short packets, a single page size can be specified in RAALL Scatter Page Size and Queue Register. If all of the packet data and the packet header fit in this space, then all of the data is DMAed with the packet header when the packet completes.

**Scatter Error Recovery**

During reception, it is OK to not have a real page address available. As more cells are received, RAALL will attempt to catch up. If a packet completes and there is a lack of pages, then a no page event is posted. If running in receive bad frame mode then the user needs to query and free up any entries in the DMA list. This can be done easily with the new N-to-1 DMA descriptor. Alternately, the user can move the rest of the data and surface the packet because the packet is actually a good packet. If not running in receive bad frame mode, the DMA list and packet are automatically freed for the user and the event contains the LCD address instead of the packet address.

If an error occurs in the packet (eg. CRC), then the corresponding event (eg. bad CRC) is posted. Again, the cleanup depends on bad frames being received.

If a packet is received, and all pages were DMAed, but there was a lack of DMA descriptor to DMA the header, then a no descriptor event is posted. Again, cleanup depends on bad frames being received.

## 13.1: RAALL Max SDU Length Register

Used to verify AAL5 packet lengths will not exceed the size set in this register. This register contains the maximum SDU size that is allowed to be received. If the packet size exceeds this value, a "too big" event is surfaced. This register is only used if maximum SDU size in the LC is set to zero.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C5C |
| **Power On Value** | X'0000ffff' |
| **Restrictions** | None |

## 13.2: RAALL LC Reassembling Count Register

Used to count the number of LCs that are currently reassembling packets. This register contains the count of LCs that are currently reassembling packets. This count is incremented each time a new reassembly is started and decremented each time a reassembly completes. This is a free running counter that can be thresholded using the RAALL LC Reassembling Threshold Register. An interrupt can also be generated when this counter is in zero and non-zero.

Normally, only AAL5 packets count in this calculation by default. If the proper bit in the control register is set, then AAL0 FIFO LCs also count. FIFO LCs are added when any cell is received in the FIFO, and removed when all cells are removed from the FIFO. They are also removed when they timeout.

If this register is written in non-diagnostic mode, the value is ignored, and the value of this register is decremented by one. This function is intended to be used when shutting down an LCD.

| | |
|---|---|
| **Length** | 20 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C38 |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be written with a value in diagnostic mode. |

**13.3: RAALL LC Reassembling Threshold Register**

Used to threshold the number of LCs reassembling. This register contains the threshold count of LCs reassembling. When the value in the RAALL LC Reassembling Count Register is greater or equal to this register, the appropriate status bit is turned on. When this register is written with a value of zero, no thresholding is done.

| | |
|---|---|
| **Length** | 20 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C3C |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

**13.4: RAALL Scatter Page Size and Queue Register**

Used to specify the system physical page size. This register specifies the encoded system page size and the receive queue number that holds the real buffers for scatter mode. The following are the bit descriptions:

| | |
|---|---|
| **Length** | 11 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C20 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 10-8 | Single Page Size | Specifies the buffer size used for single page packets. The following encodings are used:<br>0 Disable single page mode<br>1 128 bytes<br>2 256 bytes<br>3 512 bytes<br>4 1K bytes<br>5 2K bytes<br>6 4K bytes<br>7 8K bytes |
| 7 | Reserved | Reserved. |
| 6-4 | Buffer Queues | Specifies which receive queue holds the real buffer addresses. |
| 3 | Reserved | Reserved. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 2-0 | Page Size | Specifies the real buffer size. The following encodings are used:<br>0      512 bytes<br>1      1K bytes<br>2      2K bytes<br>3      4K bytes<br>4      8K bytes<br>5      16K bytes<br>6      reserved<br>7      reserved |
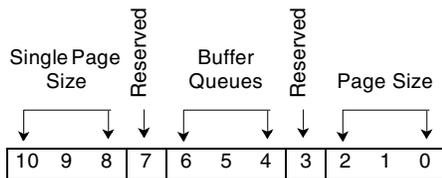
## 13.5: RAALL Scatter DMA List Free Destination Register

Used to specify the destination that is used to free DMA lists for bad packets. This register specifies the destination address that is used to free DMA lists for bad packets. This is only used when receive bad frame mode is set to not receive bad frames. In this mode, this destination should be set the receive queue enqueue address for the queue number that is used to store scatter page addresses. A DMA descriptor is built that frees the DMA list back to the appropriate receive queue by using this destination address.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C24 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

## 13.6: RAALL Non-User Data Config Register

Used to specify how non-user data (NUD) should be processed. This register contains the information on how to process non-user data cells. The information in this register is specified using the same format as the receive LCD for consistency. The NUD config register has the same format as the packed information in word zero of the receive LCD, and bits 27-26,20,18-16,14-0 are valid. If NUD cells are to be routed for a particular LCD, the routed LCD for that LCD is used.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C40 |
| **Power On Value** | X'00000010' |
| **Restrictions** | None |

IBM

### 13.7: RAALL Raw Mode Early Drop Pool-Id Register

Used to specify the POOLS pool ID to use for error state End of Packet (EOP) cells when routing raw cells. This register contains the POOLS pool ID to use for error state EOP cells when routing raw cells. EOP cells have uind=1.

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C18 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 13.8: RAALL Interrupt Enable Register

Used to specify which status register bits should be used to generate interrupts. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *RAALL Status Register on page 289* for the bit descriptions.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0C00 and 04 |
| **Power On Value** | X'00000039' |
| **Restrictions** | None |

### 13.9: RAALL Status Register

Used to set RAALL modes and relay RAALL status information. This register contains the mode bits and the status bits. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The following are the bit descriptions:

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0C08 and 0C |
| **Power On Value** | X'00000002' |
| **Restrictions** | Status bits are driven by RAALL, so even if the software writes these, the value will most likely change. |



| Bit(s) | Name | Description |
|:---:|---|---|
| 5 | Bad Offset in LCD | When this bit is set, a bad offset was detected in the LCD. |
| 4 | Sync Op Failed | When this bit is set, a sync operation has failed because more cells were synchronized than were in the receive FIFO. |
| 3 | LC Reassembling Threshold Exceeded | When this bit is set, the RAALL LC Reassembling Count Register has exceeded the threshold set in the RAALL LC Reassembling Threshold Register. |
| 2 | LC Reassembling Count /= 0. | When this bit is set, the RAALL LC Reassembling Count Register is not equal to zero. |
| 1 | LC Reassembling Count = 0. | When this bit is set, the RAALL LC Reassembling Count Register is equal to zero. |
| 0 | Reserved | Reserved |
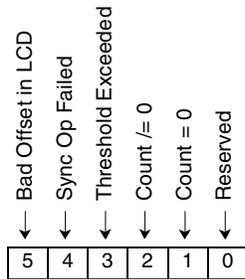
## 13.10: RAALL Control Register

Used to set RAALL modes. This register contains the mode bits that specify how RAALL is to operate. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The following are the bit descriptions:

| | |
|---|---|
| **Length** | 29 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0C10 and 14 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

The bit positions (28 down to 0) are labeled as follows:

- 28 — Disable Cut-Through For MPEG FIFO Mode
- 27 — Enable Packet Header Timestamp
- 26 — Enable Direct CutThrough
- 25 — Disable LCD Caching
- 24 — Include AAL5 trailer in CutThrough DMA
- 23 — Disable AAL5 Length checking
- 22 — Disable AAL5 CRC checking
- 21 — Disable AAL5 CPI checking
- 20 — Use CLP from cell when cell routing
- 19 — Use CLP from packet when packet routing
- 18 — Enable Packet Counts
- 17 — Enable Host Data Word in LCD/Packet Header
- 16 — Enable New AAL5 Error Checking
- 15 — Enable Non-User Data Cut Through
- 14 — Enable AAL 5 Packet trailer debug word
- 13 — AAL 5 Threshold Mode
- 12 — Enable Reassembly Timeout Processing for Raw FIFO Mode
- 11 — Enable reassembly counting for Raw FIFO mode
- 10 — Enable CRC-10 for Cells with PTI=7
- 9 — Enable CRC-10 for Cells with PTI=6
- 8 — Enable CRC-10 for Cells with PTI=5
- 7 — Enable CRC-10 for Cells with PTI=4
- 6 — Rx Bad CRC-10 Cells
- 5 — Enable Counter Overflow Events
- 4 — Enable Statistics
- 3 — Reserved
- 2 — Enable Blocking
- 1 — Diagnostic Mode
- 0 — Graceful Reset

| Bit(s) | Name | Description |
|---|---|---|
| 28 | Disable Cut-Through For MPEG FIFO Mode | When set, all LCDs that use the AAL5 MPEG FIFO mode, will surface packet events instead of doing an AAL5 Mode 6 cut-through operation. |
| 27 | Enable Packet Header Timestamp | When set, a timestamp is generated by the RXQUE timestamp logic and is placed in the packet header at offset 0x0c when the last cell is processed. It is also generated when the first cell is generated. So if using cut-through Mode 7, you can potentially get both timestamps. By using this option, you lose the ATM header field in the packet header. This has some consequences. For example, you should use 52- or 53-byte raw modes for raw mode cells if the ATM header needs to be examined. This function is intended for use by sniffer-like functions. |
| 26 | Enable Direct CutThrough | When set, cut assumes that the cut-through receive queue holds buffer addresses instead of DMA descriptors. The DMA descriptor is built by RAALL and has a length of one. This mode has the advantage of being more efficient, but only a single descriptor is allowed. |
| 25 | Disable LCD Caching | 'When set, the receive LCD information is not cached in RAALL. |
| 24 | Include AAL5 trailer in CutThrough DMA | When set, the AAL5 trailer is DMAed as part of a Mode 6 cut-through DMA. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 23 | Disable AAL5 Length checking | When set, the AAL5 trailer length field is not checked. |
| 22 | Disable AAL5 CRC checking | When set, the AAL5 trailer CRC is not checked. |
| 21 | Disable AAL5 CPI checking | When set, the AAL5 trailer CPI bytes are not checked against zero. |
| 20 | Use CLP from cell when cell routing | When set, the CLP bit from the received cell is used to fill in the CLP bit in the LCD of the packet header when routing cells. When cleared, the LSB of the second word of the LCD is used to fill in the CLP bit of the LCD of the packet header when routing cells. |
| 19 | Use CLP from packet when packet routing | When set, the ORDed CLP bit across an AAL5 packet is used to fill in the CLP bit in the LCD of the packet header when routing packets. When cleared, the LSB of the second word of the LCD is used to fill in the CLP bit of the LCD of the packet header when routing packets. |
| 18 | Enable Packet Counts | When set, the Total User Cells Rx With CLP=0 field in the LCD is used as a Total Packets Rx count. The same event is generated when the counter overflows regardless of how this bit is set. This count is a total of the good and bad packets received. |
| 17 | Enable Host Data Word in LCD/Packet Header | When set, the eighth word in the receive LCD is used as host data. This word is written into the fifth word of the packet header. The low order eight bits of the host data are ignored and should be set to zero. |
| 16 | Enable New AAL5 Error Checking | When set, the new proposed AAL5 error checking is enabled. This is useful for MPEG-2 processing, and allows the new AAL5 event to flow. This new event specifies that an AAL5 packet was received, and the CRC was bad, but the length was good. |
| 15 | Enable Non-User Data Cut Through | When set, non-user data cells are automatically DMAed into system storage using DMA descriptor from the receive queue specified in the RAALL Non-User Data Config Register. When cleared, non-user data events are surfaced as normal. If no descriptors are avail, then a no DMA descriptor event is surfaced. |
| 14 | Enable AAL 5 Packet trailer debug word | When set, 0xbadddeadbeffcafe is written in the word following the word that contains the AAL5 packet trailer (or last data in too big case). When cleared, the word is not written. |
| 13 | AAL 5 Threshold Mode | When set, packet complete events are not generated if a packet threshold event was generated. When cleared, packet complete events are always generated. Error events are always generated regardless of the state of this bit. |
| 12 | Enable Reassembly Timeout Processing for Raw FIFO Mode | When set, the reassembly timeout processing is performed on Raw FIFO lCs. |
| 11 | Enable reassembly counting for Raw FIFO mode | When set, the count of LCs reassembly includes AAL0 FIFO LCs. |
| 10 | Enable CRC-10 for Cells with PTI=7 | When set, CRC-10 checking is done on cells with a PTI=7. |
| 9 | Enable CRC-10 for Cells with PTI=6 | When set, CRC-10 checking is done on cells with a PTI=6. |
| 8 | Enable CRC-10 for Cells with PTI=5 | When set, CRC-10 checking is done on cells with a PTI=5. |
| 7 | Enable CRC-10 for Cells with PTI=4 | When set, CRC-10 checking is done on cells with a PTI=4. |
| 6 | Rx Bad CRC-10 Cells | When set, non-user data cells with bad CRC-10 fields are received as events. The event will be the actual cell buffer if in receive bad frame mode, otherwise only the LC is surfaced in the event.<br>When reset, these cells are flushed. This is only valid for PTI field types that have CRC-10 checking enabled. |
| 5 | Enable Counter Overflow Events | When set, counter overflow events are generated when an LC statistics counter overflows. When reset, the counter will roll over with no notification. |
| 4 | Enable Statistics | When set receive statistics are kept on all LCs. |

| Bit(s) | Name | Description |
|---|---|---|
| 3 | Reserved | Reserved. |
| 2 | Enable Blocking | When set enables OAM blocking on all LCs. Note: the LC enable went away. |
| 1 | Diagnostic Mode | When set, RAALL is placed in diagnostic mode. |
| 0 | Graceful Reset | When set, no new buffers are requested. This has the affect of allowing all currently reassembling buffers to complete, while all new reassembly requests are rejected. Reassembly timeout processing continues while set. During graceful reset, all non-user data cells and AAL0 cells (raw) are discarded. The AAL0 FIFOs are treated like reassembly buffers. If there are cells in the FIFO, new cells are accepted. If no cells are in the FIFO or the FIFO is in error state, then new cells are not accepted. |

## 13.11: RAALL LC Table Bound Registers

Used to specify the lower/upper bounds of the LC table. The lower bound should be initialized to the address of the first LC in the LC table if reassembly timeout processing is to be done. The upper bound should be initialized to the address of the last LC in the LC table if reassembly timeout processing is to be done.

The second set of RTO related registers is meant to be used with MPEG FIFO mode LCDs, but can be used for whatever.

| **Reg** | Lower Bound 1 | Upper Bound 1 | Lower Bound 2 | Upper Bound 2 |
|---|---|---|---|---|
| **Length** | 32 bits | 32 bits | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write | Read/Write | Read/Write |
| **Address** | XXXX 0C2C | XXXX 0C30 | XXXX 0C44 | XXXX 0C48 |
| **Power On Value** | X'00000000' | X'00000000' | X'00000000' | X'00000000' |
| **Restrictions** | Must be 128-byte aligned (low order seven bits not writable). | | | |

### 13.12: RAALL Reassembly Timeout Value Register

Used to specify the time interval used for reassembly timeout processing. This register is the number of pre-scaler intervals between reassembly processing. The pre-scaler interval is determined by RAALL Reassembly Timeout Pre-Scaler Register A single LC is checked for reassembly timeout during each reassembly processing interval.

When this register is set to zero, reassembly timeout processing is disabled.

For more information on how reassembly timeout conditions are processed see *Reassembly Timeout (RTO) Processing *.

The second set of RTO related registers is meant to be used with MPEG FIFO mode LCDs, but can be used for whatever.

| Reg | Timeout 1 | Timeout 2 |
|---|---|---|
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C34 | XXXX 0C4C |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | None | |

### 13.13: RAALL Reassembly Timeout Pre-Scaler Register

Used to specify the time interval of each RTO timer tick. This register determines the number of 30-ns intervals between RTO timer ticks. The value in the register plus one is the number of 30-ns intervals between RTO timer ticks. So, the default value of zero means that the RTO timer ticks every 30ns. If a value of four is placed in this register, the RTO timer ticks every 150ns ($5 \bullet 30$ ns).

For more info on how reassembly timeout conditions are processed see *Reassembly Timeout (RTO) Processing *.

The second set of RTO related registers is meant to be used with MPEG FIFO mode LCDs, but can be used for whatever.

| Reg | Prescale 1 | Prescale 2 |
|---|---|---|
| **Length** | 16 bits | 16 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C54 | XXXX 0C1C |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | None | |

### 13.14: RAALL LC Statistics Overflow Register

Used to specify the overflow value of the LC statistics. This register specifies at what value the LC statistics roll over. By default, this register is set to zero and specifies that the counters roll over at 0xffffffff. If this register was set to 0x00010000, the LC statistic counters would count up to 0x0000ffff and then roll over to zero.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C58 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 13.15: RAALL FIFO Sync Operation Register

Used to synchronize RAALL and the software FIFO counts. This register is written to update the FIFO cell counts. RAALL maintains the total cell count for each FIFO. As software processes these cells, the synchronizing operation needs to be performed to decrement the cell count to make room for incoming cells. It is obviously important to keep the software and RAALL in synchronization.

When this register is written, the most significant bits specify what the LC the operation is for. The low order six bits specify a cell count. The value of zero in the cell count field means use the threshold value. This cell count is subtracted from the RAALL total cell count when the operation is completed. For example, if the cell threshold is set to 128, and the low order bits are zero, then 128 is subtracted from the total cell count. If they are set to five, then five is subtracted from the total cell count.

A read to this register returns the current/last LCD address in the upper 26 bits. bit zero is zero if a synchronization operation is not running, and one if a synchronization operation is running.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0C50 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 13.16: RAALL LCD Update Data Registers

Used to specify data to write into the lcd on the update LC operation. These registers contain the data used in the LC update operation. For more information on its use, see the RAALL LCD Update Op Registers.

This register changes to contain the updated data written to the LC word while the operation is completing.

The second set of LCD update registers is meant for the core to use, but is available for general use.

| Register | Update 1 | Update 2 |
|---|---|---|
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C6C | XXXX 0C80 |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | None | |

### 13.17: RAALL LCD Update Mask Registers

Used to specify which data to write into the LCD on the update LC operation. This register contains the mask used in the LC update operation. For more information on its use, see the RAALL LCD Update Op Registers.

The second set of LCD update registers is meant for the core to use, but is available for general use.

| Register | Update 1 | Update 2 |
|---|---|---|
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C70 | XXXX 0C84 |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | None | |

### 13.18: RAALL LCD Update Op Register

Used to specify what LCD word to update. This operation is used to update a portion of an LCD. If this operation is not used, then software or the IBM2520L8767 updates of the LCD may be lost because the LCD is cached in the IBM2520L8767 while cells are being processed.

This register is written with the address of the LCD word to update. Once this register is written the update operation starts. All subsequent reads or writes to the data, mask, or update registers are held off until the operation completes. A read-modify-write will occur to update the portion specified by the mask with the masked value in the data register.

Normally this register would not be read. However, if it is read, the low order bit is read as zero and the next lowest order bit (bit 1) is read as the busy bit. This signifies whether an operation is still going on. If an operation is still going on, a new write to any of the data, mask, or update operation registers is held off until the original operation is complete.

The second set of LCD update registers is meant for the core to use, but is available for general use.

| Register | Update 1 | Update 2 |
| --- | --- | --- |
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C74 | XXXX 0C88 |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | The low order two bits are not writable. | |

### 13.19: RAALL Cut Through Desc Address Registers

Used to hold the DMA descriptor address for cut-through operations. This register is used to hold the DMA descriptor address for cut-through operations. For more information, see the cut-through discussion.

A second set of CTOP registers was added this pass.

| Register Set | CTOP 1 | CTOP 2 |
| --- | --- | --- |
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C78 | XXXX 0C90 |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | None | |

### 13.20: RAALL Cut Through Op (CTOP) Registers

Used to notify RAALL that a packet has a DMA descriptor address to be scheduled on packet completion. This register is used to notify RAALL that a packet has a DMA descriptor address to be scheduled on packet completion. For more information, see the cut-through discussion.

A second set of CTOP registers was added this pass.

| | | |
|---|---|---|
| **Register Set** | CTOP 1 | CTOP 1 |
| **Length** | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write |
| **Address** | XXXX 0C7c | XXXX 0C94 |
| **Power On Value** | X'00000000' | X'00000000' |
| **Restrictions** | Reads last value used for previous cut-through operation. Low order bit indicates CTOP ongoing if set. Low order six bits are not writable. | |

## 13.21: RAALL Cut Through Hardware FIFO Registers

Used to specify addresses of Hardware FIFO and Hardware Complete registers. These registers specify addresses of Hardware FIFO and Hardware Complete registers. The base registers specify where FIFO and complete register set zero are located. The FIFO and complete registers for each additional channel are calculated using the channel number ∗ the corresponding size register. The maximum size (distance between registers) is 128K. The size registers have the following encoding:

| Register | FIFO Base | FIFO Size | Complete Base | Complete Size |
|---|---|---|---|---|
| **Length** | 32 bits | 32 bits | 32 bits | 32 bits |
| **Type** | Read/Write | Read/Write | Read/Write | Read/Write |
| **Address** | XXXX 0CB0 | XXXX 0CB4 | XXXX 0CB8 | XXXX 0CBC |
| **Power On Value** | X'00000000' | X'00000000' | X'00000000' | X'00000000' |
| **Restrictions** | None | | | |

| Bit(s) | Description |
|---|---|
| 31-4 | Reserved. |
| 0-3 | 0000    4byte offsets<br>0001    8byte offsets<br>0010    16 byte offsets<br>0011    32 byte offsets<br>0100    64 byte offsets<br>0101    128 byte offsets<br>0110    256 byte offsets<br>0111    512 byte offsets<br>1000    1K byte offsets<br>1001    2K byte offsets<br>1010    4K byte offsets<br>1011    8K byte offsets<br>1100    16K byte offsets<br>1101    32K byte offsets<br>1110    64K byte offsets<br>1111    128K byte offsets |

**13.22: RAALL - DMA Flag Registers**

Used to specify the flags field of DMA descriptors that are built by RAALL.

Used For: Cut-Through Descriptor, Scatter Page Descriptor, Free DMA List Descriptor, Hardware FIFO Descriptor 1, and Hardware FIFO Descriptor 2. These registers specify the flags field of DMA descriptors that are built by RAALL.

Flags can be set for normal cut-through descriptor (includes Mode 6 & 7 cut-through and descriptor for scatter header), scatter mode page descriptor, free DMA list descriptor, hardware FIFO DM descriptor, and hardware complete descriptor.

| | | | | | |
|---|---|---|---|---|---|
| **Length** | 18 bits | | | | |
| **Type** | Read/Write | | | | |
| **Address** | XXXX 0CC0 | XXXX 0CC4 | XXXX 0CC8 | XXXX 0CCC | XXXX 0CD0 |
| **Power On Value** | X'00000010' | X'00000010' | X'00002220' | X'00000010' | X'00000000' |
| **Restrictions** | None | | | | |

| Bit(s) | Description |
|---|---|
| 17-16 | Specify how the event data (if any) is formed. Normally, the DMA descriptor is used for DMA events, but when the entire DMA descriptor is built there is no descriptor address to use.<br>00    Use DMA Descriptor address<br>01    Use Source Address<br>10    Use Destination Address |
| 15-0 | DMA flags. |

## Entity 14: Receive Queues (RXQUE)

### Functional Description

RXQUE has a single function: to manage the receive queues for software by providing an easy to use primitive interface. When talking about the receive queues, the term rxq is used to talk about a receive queue, and the term deq is used to refer to a dequeue operation and enq is used to refer to a enqueue operation.

### Receive Queue Interface

A group of eight receive queues are available for software use. The receive queues hold events or user specified data.

Each queue entry (event) is 32 bits, and contains both a event type field as well as some event information.

## RAALL RXQUE Event Structure

| 26 | 6 |
|---|---|
| Event Info | Event Type |
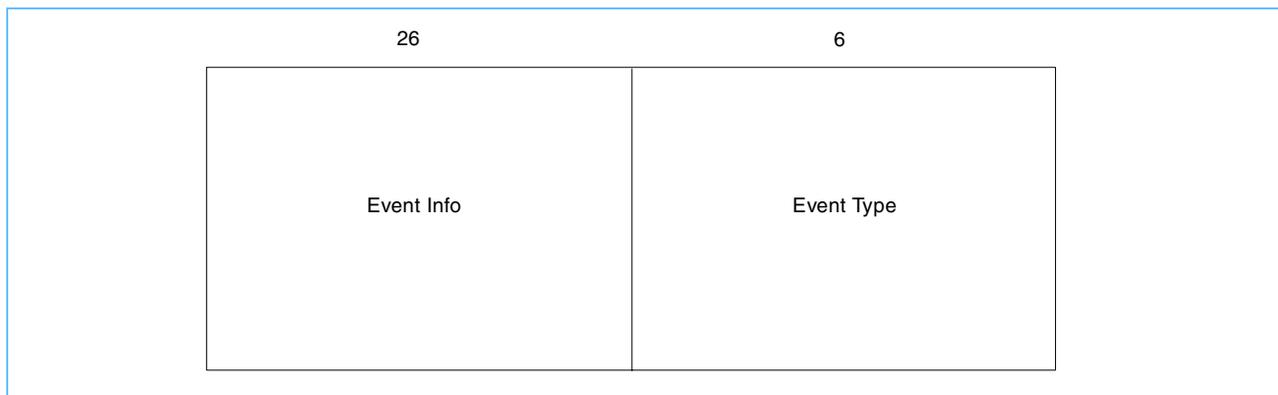
The event information typically contains a pointer (when low order bits are zeroed) to a packet buffer, a cell buffer, or an LCD. It can also contain a DMA descriptor address or user specified data.

The following are the different event types that are defined:

## Event Summary and Routing Info (Page 1 of 2)

| Event Number | Description | Event Information | Error | Count | Tx Comp | Tx DMA | Rx DMA | ABR | POOL |
|---|---|---|---|---|---|---|---|---|---|
| 000000 = 00 | AAL5 packet event (packet complete) | Packet/LCD | | | | | | | |
| 000001 = 01 | AAL5 packet header event (packet start) | Packet | | | | | | | |
| 000010 = 02 | AAL5 packet with bad CRC | Packet/LCD | X | | | | | | |
| 000011 = 03 | AAL5 packet with bad length field | Packet/LCD | X | | | | | | |
| 000100 = 04 | AAL5 packet that exceeds maximum length in LC | Packet/LCD | X | | | | | | |
| 000101 = 05 | AAL5 packet timeout | Packet/LCD | X | | | | | | |
| 000110 = 06 | AAL5 packet forward abort | Packet/LCD | X | | | | | | |
| 000111 = 07 | AAL5 packet CPI field not equal to zero | Packet/LCD | X | | | | | | |
| 001110 = 0e | AAL5 FIFO packet | Packet | | | | | | | |
| 001111 = 0f | AAL5 packet with bad CRC and good length | Packet/LCD | X | | | | | | |
| 001000 = 08 | Cell event (user data) | Packet/LCD | | | | | | | |
| 001001 = 09 | NUD cell event (non-user data) | Packet/LCD | | | | | | | |
| 001010 = 0a | NUD cell with bad CRC-10 | Packet/Lcd | X | | | | | | |
| 001011 = 0b | Bad cell - bad HEC | Packet | X | | | | | | |
| 001100 = 0c | Bad cell - out of range | Packet | X | | | | | | |
| 001101 = 0d | Bad cell - index equal zero | Packet | X | | | | | | |
| 010000 = 10 | AAL0 cell dropped - lack of POOLS buffers | LCD | X | | | | | | |
| 010001 = 11 | AAL5 cell dropped - lack of POOLS buffers | LCD | X | | | | | | |
| 010010 = 12 | NUD cell dropped - lack of POOLS buffers | LCD | X | | | | | | |
| 010011 = 13 | FIFO cell dropped - FIFO full | LCD | X | | | | | | |
| 010100 = 14 | FIFO threshold event | LCD | | | | | | | |
| 010101 = 15 | FIFO full event | LCD | | | | | | | |
| 010110 = 16 | FIFO RTO event | LCD | X | | | | | | |
| 011000 = 18 | Total user cells receive counter overflow | LCD | | X | | | | | |
| 011001 = 19 | Total user cells rx clp=0 counter overflow | LCD | | X | | | | | |
| 011010 = 1a | Total user cells tx counter overflow | LCD | | X | | | | | |
| 011011 = 1b | Total user cells tx clp=0 counter overflow | LCD | | X | | | | | |
| 011110 = 1e | Threshold 1 crossed | LCD & Dir | | | X | | | | |
| 011111 = 1f | Threshold 2 crossed | LCD & Dir | | | X | | | | |
| 100000 = 20 | Transmit complete | Packet | | | X | | | | |
| 100001 = 21 | Transmit complete buffer freed | Packet/LCD | | | X | | | | |
| 100010 = 22 | "bad" found in first word of packet | Packet | | | X | | | | |
| 100011 = 23 | Connection closed | LCD | | | X | | | | |
| 100100 = 24 | Transmit DMA complete | Descriptor | | | | X | | | |

IBM

## Event Summary and Routing Info (Page 2 of 2)

| Event Number | Description | Event Information | Error | Count | Tx Comp | Tx DMA | Rx DMA | ABR | POOL |
|---|---|---|---|---|---|---|---|---|---|
| 100101 = 25 | Receive DMA complete | Descriptor | | | | | X | | |
| 100110 = 26 | Transmit DMA complete with error | Descriptor | | | | X | | | |
| 100111 = 27 | Receive DMA complete with error | Descriptor | | | | | X | | |
| 101000 = 28 | Transmit DMA complete with virtual error | Descriptor | | | | X | | | |
| 101001 = 29 | Zero address in DMA descriptor SRC/DST address | Descriptor | | | | X | X | | |
| 101100 = 2c | ADTF Event | LCD | | | | | | X | |
| 101101 = 2d | CRM Event | LCD | | | | | | X | |
| 101110 = 2e | CCR=0 Event | LCD | | | | | | X | |
| 101111 = 2f | RM Cell Event | Packet | | | | | | X | |
| 110000 = 30 | User event | | | | | | | | |
| 110001 = 31 | User event | | | | | | | | |
| 110010 = 32 | User event | | | | | | | | |
| 110011 = 33 | User event | | | | | | | | |
| 110100 = 34 | User event | | | | | | | | |
| 110101 = 35 | User event | | | | | | | | |
| 110110 = 36 | User event | | | | | | | | |
| 110111 = 37 | User event | | | | | | | | |
| 111000 = 38 | Virtual memory resource event | Packet/LCD | X | | | | | | |
| 111001 = 39 | Buffer overflow event | Packet/LCD | X | | | | | | |
| 111010 = 3a | No DMA descriptor for AAL7 packet | Packet/LCD | X | | | | | | |
| 111011 = 3b | DMA canceled for AAL7 packet due to error | Descriptor | | | | | X | | |
| 111100 = 3c | No scatter pages available and packet complete | Packet/LCD | X | | | | | | |
| 111101 = 3d | Entity counter overflow event | Counter | | X | | | | | |
| 111110 = 3e | POOLS status event | Status | | | | | | | X |
| 111111 = 3f | Timestamp event | Timestamp | | | | | | | |

**AAL5 Packet Events**

For AAL5 packet events, the event specifies the packet buffer address, and the event type field specifies the type of packet event. The following event types are defined

| Bit(s) | Name | Description |
|---|---|---|
| 0x00=000000 | Normal AAL 5 Packet Event (Packet Complete) | This event specifies that an AAL5 packet was received and has passed all AAL5 protocol checks (CRC, length). The event information contains a pointer to the packet. |
| 0x0e=001110 | Normal AAL5 Fifo Packet Event | This event specifies that an AAL5 FIFO packet was received and has passed all AAL5 protocol checks (CRC, length, ...). The event information contains a pointer to the packet. |
| 0x01=000001 | Normal AAL5 Packet Header Threshold Event (Packet Start) | This event specifes that the AAL5 packet header threshold was exceeded as set in the LCD. The event information contains a pointer to the packet header, and the user can access up to the packet header threshold bytes of data. |
| 0x02=000010 | AAL5 Packet with Bad CRC was RX on LC | This event specifies that a AAL5 packet was received and the AAL5 CRC is bad. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x0f=001111 | AAL5 Packet with Bad CRC and good length was RX on LC | This event specifies that a AAL5 packet was received and the AAL5 CRC is bad, but the AAL5 length was good. This event is useful for MPEG-2 operation. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x03=000011 | AAL5 Packet with Bad Length Field was RX on LC | This event specifies that a AAL5 packet was received and the AAL5 length field is bad. For example, there is too much data or not enough, but typically the bad CRC is detected first. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x04=000100 | AAL5 Packet that exceeds Max Len in LC was RX | This event specifies that a AAL5 packet was received but the amount of data has exceeded the maximum length as specified in the LCD or in the MSDU register. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x05=000101 | AAL5 Packet Timeout on this LC | This event specifies that a reassembly timeout has occurred for a AAL5 packet that was being reassembled. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x06=000110 | AAL5 Packet Forward Abort | This event specifies that a AAL5 packet was terminated with a forward abort. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x07=000111 | AAL5 Packet CPI Field not equal to Zero | This event specifies that a AAL5 packet was received and the AAL5 CPI field was not set to zero which is an AAL5 protocol violation. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |

## Cell Events

For AAL0 events, the event specifies a cell buffer address, and the event type field specifes type of AAL0 event. The following event types are defined:

| Bit(s) | Name | Description |
|---|---|---|
| 0x08=001000 | AAL0 Cell Event | This event specifies that a AAL0 (non-FIFO mode) cell was received. The event information contains a pointer to the cell. |
| 0x09=001001 | Non-User Data Cell Event | This event specifies that a non-user data cell was received and the CRC-10 was good if checking was enabled. The event information contains a pointer to the cell. |
| 0x0a=001010 | Non-User Data Cell with Bad CRC-10 Event | This event specifies that a non-user data cell was received and the CRC-10 was bad. The event information contains either a pointer to the cell if receiving bad frames, or a pointer to the LCD on which this cell was received. |
| 0x0b=001011 | Cell with Bad HEC Event | This event specifies that a cell was received with a bad HEC. The event information contains a pointer to the cell. |
| 0x0c=001100 | Cell with VP/VC Out Of Range Event | This event specifies that a cell was received with a VP/VC that was out of range. The event information contains a pointer to the cell. |
| 0x0d=001101 | ***Cell with VC Index Equal Zero*** | This event specifies that a cell was received with a VC index equal zero. The event information contains a pointer to the cell. |

## LC Events

For LC events, the event specifies a LC, and the event type field specifes what happened on the LC. The following event types are defined:

| Bit(s) | Name | Description |
|---|---|---|
| 0x10=010000 | AAL0 Cell was Dropped due to Lack of Pools Buffers | This event specifies that a AAL0 (non-FIFO mode) cell was received, but was discarded because no POOL buffers were available. The event information contains a pointer to the LCD on which this cell was received. |
| 0x11=010001 | AAL5 Cell was Dropped due to Lack of Pools Buffers | This event specifies that the first AAL5 cell for a packet was received, but was discarded because no POOL buffers were available. The event information contains a pointer to the LCD on which this cell was received. |
| 0x12=010010 | Non-user Data Cell was Dropped due to Lack of Pools Buffers | This event specifies that a non-user data cell was received, but was discarded because no POOL buffers were available. The event information contains a pointer to the LCD on which this cell was received. |
| 0x13=010011 | Fifo Cell was Dropped because Fifo Full | This event specifies that a AAL0 (FIFO mode) cell was received, but was discarded because the FIFO was full. The event information contains a pointer to the LCD on which this cell was received. |
| 0x14=010100 | Fifo Threshold Occurred | This event specifies that the FIFO threshold has been crossed. The event information contains a pointer to the LCD on which this threshold occured. |
| 0x15=010101 | Fifo is Full | This event specifies that a FIFO has become full. The event information contains a pointer to the LCD on which this timeout occured. |
| 0x16=010110 | Fifo has Timed Out | This event specifies that a reassembly timeout has occurred for a FIFO. The event information contains a pointer to the LCD on which this timeout occured. |
| 0x17=010111 | Reserved | Reserved |
| 0x18=011000 | LC Total User Cells RX Counter Overflow | This event specifies that the TUC RX counter in the LCD has overflowed. The event information contains a pointer to the LCD. |
| 0x19=011001 | LC Total User Cells CLP=0 RX Counter Overflow | This event specifies that the TUC w/CLP=0 RX counter in the LCD has overflowed. The event information contains a pointer to the LCD. |
| 0x1a=011010 | LC Total User Cells TX Counter Overflow | This event specifies that the TUC TX counter in the LCD has overflowed. The event information contains a pointer to the LCD. |
| 0x1b=011011 | LC Total User Cells CLP=0 TX Counter Overflow | This event specifies that the TUC w/CLP=0 TX counter in the LCD has overflowed. The event information contains a pointer to the LCD. |

## ABR Events

The following events are used for ABR processing, and are routed to the receive queue specified in the ABR event routing register.

| Bit(s) | Name | Description |
|---|---|---|
| 0x2c=101100 | ADTF Event | This event occurs when the TADTF timer (as set in the LCD) has expired. The event information contains a pointer to the LCD. |
| 0x2d=101101 | CRM Event | This event occurs when CRM or more RM cells are sent without receiving a backward RM cell. The event information contains a pointer to the LCD. |
| 0x2e=101110 | CCR = 0 Event | This event specifies that the current cell rate has been set to 0. The event information contains a pointer to the LCD. |

| Bit(s) | Name | Description |
|---|---|---|
| 0x2f=101111 | RM Cell Rx-ed | This event specifies that a RM cell was received. The event information contains a pointer to the receive buffer. |

## Miscellaneous

| Bit(s) | Name | Description |
|---|---|---|
| 0x1e=011110 | Thresh 1 Event | This event specifies that a memory management threshold was crossed. The event information contains the LCD address, and bit six contains the direction (1 if up, 0 if down). |
| 0x1f=011111 | Thresh 2 Event | This event specifies that a memory management threshold was crossed. The event information contains the LCD address, and bit six contains the direction (1 if up, 0 if down). |
| 0x20=100000 | Transmit Complete | This event specifies that a packet/cell was successfully transmitted. The event information contains a pointer to the buffer transmitted. |
| 0x21=100001 | Transmit Complete Buffer Freed | This event specifies that a packet/cell was successfully transmitted and the buffer was freed back to pools. The event information contains a pointer to the buffer transmitted. |
| 0x22=100010 | Transmit Bad | This event specifies that a packet was to be transmitted, but the buffer was marked as bad, so was canceled. This is casued by getting a page fault when DMAing into the transmit packet buffer. The event information contains a pointer to the bad buffer. |
| 0x23=100011 | Connection Closed | This event specifies that all packets for the given LCD have been transmitted. The event information contains the LCD address. |
| 0x24=100100 | TX DMA Complete (into IBM2520L8767) | This event specifies that a TX DMA completed successfully. The event information depends on how the DMA was set up. |
| 0x25=100101 | RX DMA Complete (out of IBM2520L8767) | This event specifies that a RX DMA completed successfully. The event information depends on how the DMA was set up. |
| 0x26=100110 | TX DMA Complete with Error (into IBM2520L8767) | This event specifies that a TX DMA had errors. The event information depends on how the DMA was set up. |
| 0x27=100111 | RX DMA Complete with Error (out of IBM2520L8767) | This event specifies that a RX DMA had errors. The event information depends on how the DMA was set up. |
| 0x28=101000 | TX DMA Complete with Virtual Error | This event specifies that a TX DMA had a virtual error. The reminder of the DMA descriptor was cancelled. The event information contains the DMA descriptor address. |
| 0x29=101001 | DMA Desc has Zero Address | This event specifies that a DMA descriptor source destination address was zero. The remainder of the DMA descriptor was cancelled. The event information contains the DMA descriptor address. |
| 0x30=110000 | User Defined | |
| 0x31=110001 | User Defined | |
| 0x32=110010 | User Defined | |
| 0x33=110011 | User Defined | |
| 0x34=110100 | User Defined | |
| 0x35=110101 | User Defined | |
| 0x36=110110 | User Defined | |
| 0x37=110111 | User Defined | |

| Bit(s) | Name | Description |
|---|---|---|
| 0x38=111000 | Virtual Memory Resource Event | This event specifies that an AAL5 cell was received, but could not be written into the buffer because a virtual memory boundary was crossed and a buffer was not available to fill in the next segment. This can also happen if the cell crosses a boundary that would make the buffer larger than the virtual buffer size. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD that on which this packet was received. |
| 0x39=111001 | Buffer Overflow Event | This event specifies that an AAL5 cell was received, but could not be written into the buffer because it would exceed the real buffer size. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x3a=111010 | No DMA Desc for AAL7 Packet Event | This event specifies that an AAL7 (AAL5 with cut-through) packet was completed, but no DMA descriptors were available to DMA the packet header. The event information contains either a pointer to the packet if receiving bad frames, or a pointer to the LCD on which this packet was received. |
| 0x3b=111011 | DMA Cancelled for AAL7 Packet Due to Error Event | This event specifies that a DMA desc enqueued with a cut-through operation was cancelled because an error condition was detected with the packet (CRC, length, ...). The event information contains the system descriptor address that was cancelled. |
| 0x3c=111100 | No Pages Available for AAL5 Scatter Packet | This event specifies that a AAL5 packet completed with no errors, but there was a lack of scatter buffers to complete the scatter DMA. The user must interrogate the partial DMA list in the packet header to recover the system pages. Once this is done, the packet should be freed. The user can alternately treat this as a good packet and complete the packet processing by setting up additional DMAs to move the remaining data to system pages. The event information contains the IBM2520L8767 buffer address. |
| 0x3d=111101 | IBM2520L8767 Counter Overflow Event | This event specifies that a counter overflow event has been raised. The event info specifies which counter overflowed. Multiple counter overflows can be specified with a single event. The following is the definition of the event information:<br>bit 31  Segbf Total User Cells Tx Overflow<br>bit 30  Segbf Total User Cells with CLP=0 Tx Overflow<br>bit 29  Reasm Total User Cells Rx Overflow<br>bit 28  Reasm Total User Cells with CLP=0 Rx Overflow<br>bit 27  Gpdma Write Dma Byte Count Overflow<br>bit 26  Gpdma Read Dma Byte Count Overflow<br>bit 25  Rxque Timestamp Counter Overflow<br>bit 24  Pcint Performance Counter 1 Overflow<br>bit 23  Pcint Performance Counter 2 Overflow |
| 0x3e=111110 | Pools Status Event | This event specifies that a pools status event has been raised. The event info contains the status. See POOLS for the definition. |
| 0x3f=111111 | Timestamp Event | This event specifies that a timestamp event has been placed ahead of next event. The event info contains the timestamp. |

The receive queues are maintained by RXQUE with the following operations being avail to software:

- dequeue
- Remove the entry at the head of the queue
- enqueue
- Add arbitrary entry at the tail of the queue

The following figure shows how events, in a receive queue, link to other data structures including LC control blocks, packet buffers, and cell buffers.

## General Queue, Event, and Data Structure Linkage
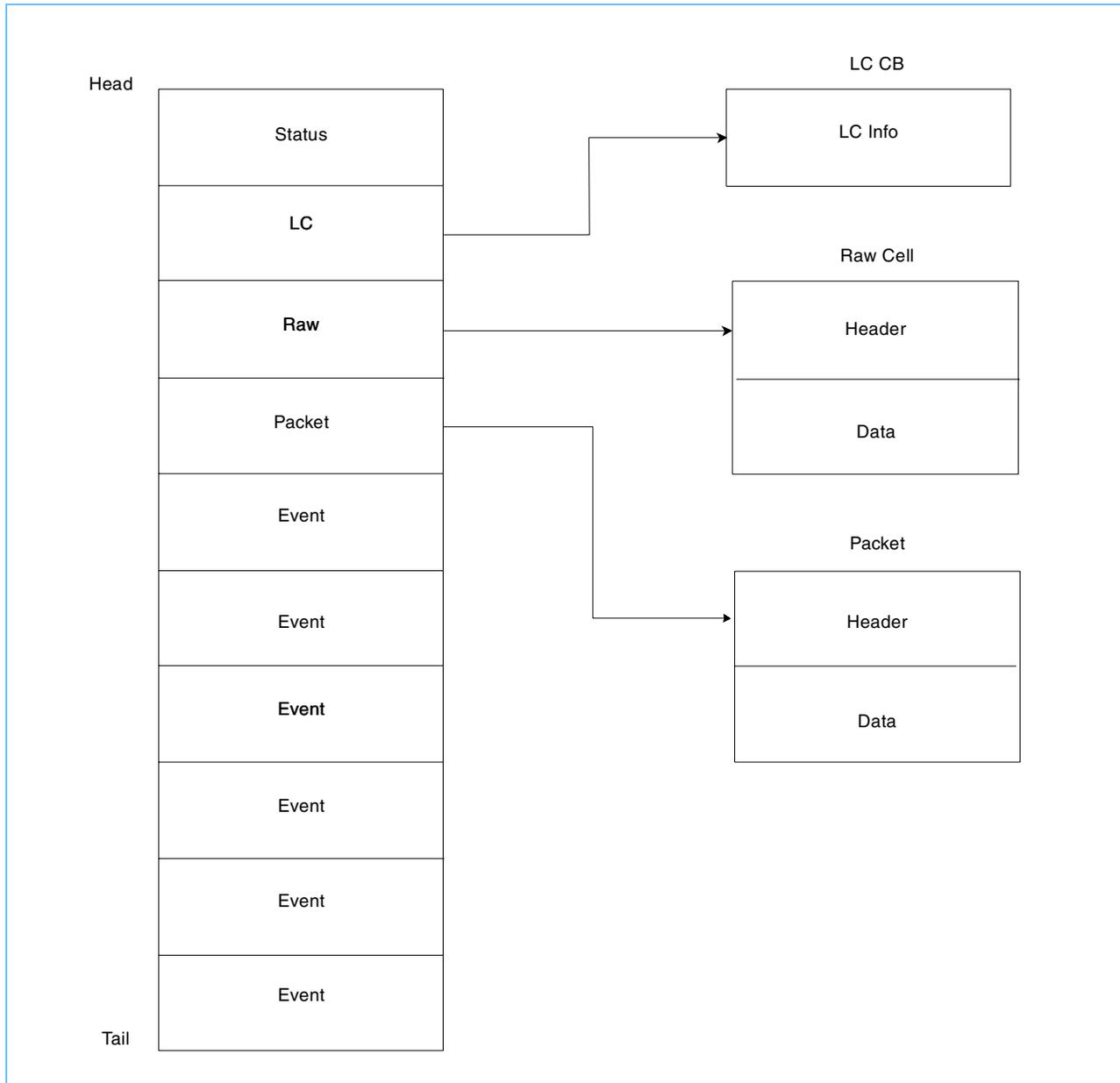
### RXQUE Structure

Each queue is implemented as a wrap around buffer with the head chasing the tail. Each queue entry is a 32-bit event. Each queue has a number of registers that define the queue and its behavior:

| | |
|---|---|
| **lower bound** | pointer to beginning of wrap around buffer |
| **upper bound** | encoded maximum length of wrap around buffer |
| **head pointer** | pointer to head of queue |
| **tail pointer** | pointer to the next free entry in queue. Points to head if queue is full or empty. |
| **queue length** | current length of the queue |
| **maximum queue length** | encoded maximum length of the queue (can be different from upper bound) |
| **threshold** | length threshold used to generate interrupts |
| **high water threshold** | length threshold used to generate interrupts |

The following figure shows the structure of the queue in memory.

## General RXQUE Queue Structure
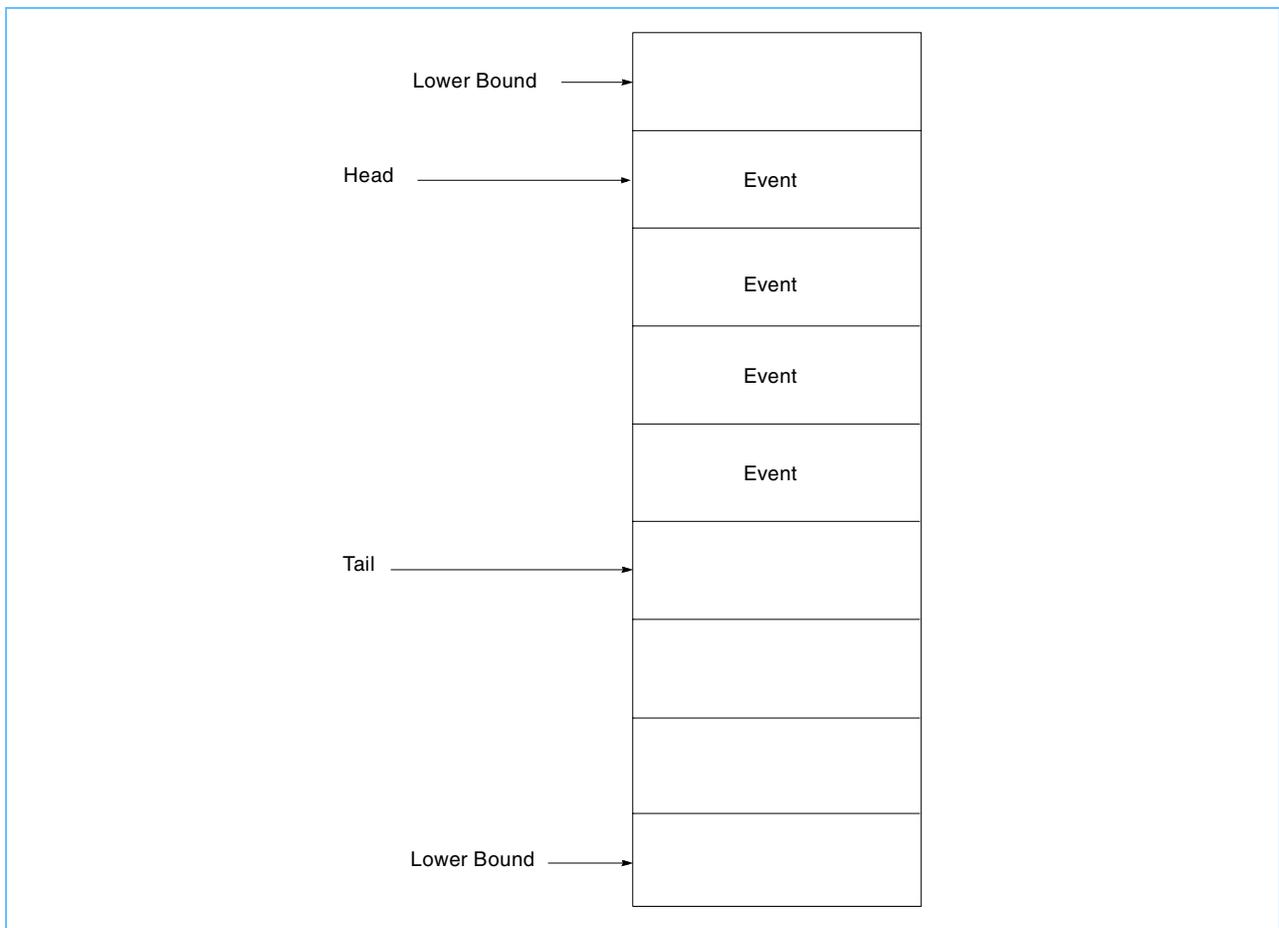


The entire queue is stored in either the control or packet portion of the IBM2520L8767 memory. The head/tail of each queue is cached to allow fast dequeue operations by software.

### RXQUE Initialization

The receive queues power on enabled and ready to run. If different addresses or options are to be used, the receive queues need to be initialized.

To set up a receive queue, at least two pieces of information are needed. First, the base address of the receive queue. Second, the length of the receive queue is needed. The following general pseudo code illustrates how to set up a receive queue for operation:

## RXQUE Initialization Code

```
void InitRXQ(bit32* BaseAddr, bit8 PhySize, bit16 Thresh) {

    // RXQUE should be in diag mode for this routine to succeed

    RXQUE->LowerBound = BaseAddr;
    RXQUE->UpperBound = PhySize;   // PhySize is encoded
    RXQUE->Threshhold = Thresh;
}
```

The example above applies to any of the receive queues. The routine could also take the receive queue number, if the includes for the registers are properly defined.

The following restrictions should be taken into account when setting up a queue:

- The lower bound and upper bound must be at least 512-byte aligned. The low order bits of these registers are not writable. So the minimum physical size of a receive queue is 128 entries (512 bytes). The alignment should correspond to the size specified in the upper bound register (i.e. 4-KB aligned if upper bound is 4KB).
- The head and tail pointers are initialized when the lower bound register is written. These registers are only writable for diagnostic purposes.
- The threshold is level-sensitive. As long as the queue length is greater than or equal to the threshold, the appropriate status bit is driven.
- All registers, except the threshold registers, can only be written in diagnostic mode and are intended to be written only once when they are set up.

**RXQUE Event Routing**

Events are routed to a receive queue based the current event type and mode of the chip. Events fall into these categories:

- Normal Events
- Error Events
- Counter Events
- Transmit Complete Events
- DMA Events (transmit/receive)
- ABR Events
- Pools Status Events

Refer to the table at the beginning of this chapter to see how the different events are categorized.

All events other than normal and error events are routed using the corresponding RXQUE Event Routing Registers.

Normal events are always routed to the receive queue specified in the receive portion of the LCD.

Error events are special; they are routed based on the values of the "receive bad frame" mode bit and the "always route error events" mode bit in RXQUE Control Register. If the always route error events bit is on, then the error events are always routed to the error queue. Otherwise, if the receive bad frame mode is on, the error events are routed to the receive queue specified in the receive portion of the LCD just as a normal event. When receive bad frames is off, the error events are routed to the error queue.

**RXQUE Normal Operation**

This section is meant to help describe how to use the receive queues (rxq) and the receive queue operations.

The receive queue contains events for the end user to process. These events are obtained by the user by executing the REASM dequeue operation. The user can be notified of new events by setting up the threshold and interrupt enable registers appropriately. Otherwise, the receive queues length register can be polled to check for events.

The dequeue operation is executed by reading the dequeue register address for the appropriate receive queue. The event at the head of the queue is returned and the event is removed from the queue. Some events have a packet/cell buffer associated with it. This buffer is owned by the user and it is the users responsibility to free this buffer.

The following pseudo code illustrates how a receive queue could be processed:

## RXQUE Dequeue Event Loop

```
            // rxq was polled or int ocurred to get here

            Event = RXQUE->Deq;              // read an event from rxq

            if (Event neq 0) {               // need to check for null event
                EventType = Event & 0x3f;    // calc event type
                Event = Event & 0xffffffc0;  // calc lc or buffer ptr

                switch (EventType) {
                    case(Event1):
                        ProcessSimpleEvent1(Event);
                        break;
                    case(Event2):
                        ProcessSimpleEvent2(Event);
                        break;


                    case(EventX):
                        ProcessSimpleEventX(Event);
                        break;
                }
            }
```

### RXQUE Queue Full Operation

When a receive queue is full (length is equal to maximum length), the appropriate status bit is set. When a queue is full, all subsequent events are flushed until room is available in the receive queue. If a buffer was associated with the event, it is freed back to POOLs.

When an event is dropped, the event dropped status bit is set and the event data that was dropped can be found in RXQUE Last Event Dropped Register. RXQUE Last Event Dropped Register will not change until the event dropped status bit is cleared.

It is obviously not good to let a receive queue become full!

### RXQUE Event Timestamping

When timestamp mode is set in the RXQUE Control Register, events are timestamped. When timestamping is enabled, a timestamp event is placed in the corresponding receive queue followed by the actual event. The event info of the timestamp event carries the timestamp. The timestamp is determined from the RXQUE Timestamp Register, RXQUE Timestamp Pre-Scaler Register, and the RXQUE Timestamp Shift Register.

If the corresponding receive queue is full, both events are dropped. It is possible to lose only the timestamp event or lose the actual event, depending on the length of the queue and the timing of the dequeue operations.

### 14.1: RXQUE Lower Bound Registers

These registers specify the lower bound of the corresponding receive queue data structure. These registers specify the lower bound of the corresponding receive queue data structure. The head and tail of the receive queue are initialized when this register is written. When the receive queue wraps past the upper bound, it wraps back to the value in the lower bound register, thus implementing the receive queue as a circular buffer.

When this register is written, the corresponding receive queue is essentially reset. This is because the head, tail, and length of the queue are all reset.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 1800 |
| | Queue 1 | XXXX 1840 |
| | Queue 2 | XXXX 1880 |
| | Queue 3 | XXXX 18C0 |
| | Queue 4 | XXXX 1900 |
| | Queue 5 | XXXX 1940 |
| | Queue 6 | XXXX 1980 |
| | Queue 7 | XXXX 19C0 |
| **Power on Value** | Queue 0 | X'0000e000' |
| | Queue 1 | X'0000e400' |
| | Queue 2 | X'0000e800' |
| | Queue 3 | X'0000ec00' |
| | Queue 4 | X'0000f000' |
| | Queue 5 | X'0000f400' |
| | Queue 6 | X'0000f800' |
| | Queue 7 | X'0000fc00' |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the control register. | |
| | The lower bound registers must be at least 512-byte aligned (low order nine bits not writable). The alignment should also correspond to the size specified in the upper bound register. For example, it should be 4-KB aligned if the upper bound specifies 4-KB size. | |

IBM

### 14.2: RXQUE Upper Bound Registers

These registers specify the encoded upper bound of the corresponding receive queue data structure. The actual upper bound is calculated by adding the decoded queue size to the lower bound. When the receive queue wraps past the upper bound, it wraps back to the lower bound register, thus implementing the receive queue as a circular buffer.

| | | |
|---|---|---|
| **Length** | 3 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 1804 |
| | Queue 1 | XXXX 1844 |
| | Queue 2 | XXXX 1884 |
| | Queue 3 | XXXX 18C4 |
| | Queue 4 | XXXX 1904 |
| | Queue 5 | XXXX 1944 |
| | Queue 6 | XXXX 1984 |
| | Queue 7 | XXXX 19C4 |
| **Power on Value** | X'00000001' | |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the control register. | |

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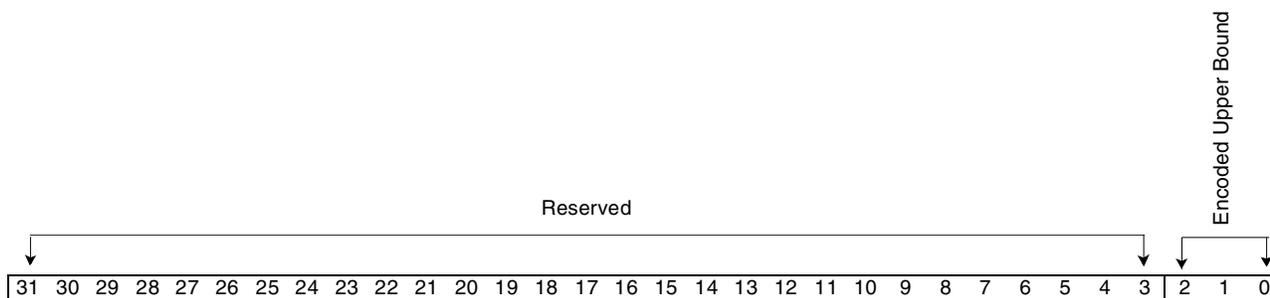
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 | 2 1 0 |

| Bit(s) | Description |
|---|---|
| 31-3 | Reserved. |
| 2 - 0 | 000   128 entries -> 512 bytes of memory<br>001   256 entries -> 1KB of memory<br>010   512 entries -> 2KB of memory<br>011   1K entries -> 4KB of memory<br>100   2K entries -> 8KB of memory<br>101   4K entries -> 16KB of memory<br>110   8K entries -> 32KB of memory<br>111   16K entries -> 64KB of memory |

### 14.3: RXQUE Head Pointer Registers

These registers point to the head element of the corresponding receive queue.

The head pointer registers are four-byte aligned. (low order two bits not writable)

Bits 31-16 are calculated internally, and are not writable.

During normal operations, these registers do not need to be read or written, as they are used by the IBM2520L8767 to implement the receive queues. These registers are initialized when the lower bound register for the corresponding receive queue is written.

**14.4: ᴄRXQUE Tail Pointer Registers**

These registers point to the next free element of the corresponding receive queue. During normal operations, these registers do not need to be read or written, as they are used by the IBM2520L8767 to implement the receive queues. These registers are initialized when the lower bound register for the corresponding receive queue is written.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| **Address** | Queue 0 | XXXX 180C |
| | Queue 1 | XXXX 184C |
| | Queue 2 | XXXX 188C |
| | Queue 3 | XXXX 18CC |
| | Queue 4 | XXXX 190C |
| | Queue 5 | XXXX 194C |
| | Queue 6 | XXXX 198C |
| | Queue 7 | XXXX 19CC |
| **Power on Value** | Queue 0 | X'0000E000' |
| | Queue 1 | X'0000E400' |
| | Queue 2 | X'0000E800' |
| | Queue 3 | X'0000EC00' |
| | Queue 4 | X'0000f000' |
| | Queue 5 | X'0000f400' |
| | Queue 6 | X'0000f800' |
| | Queue 7 | X'0000fC00' |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the control register. | |
| | Bits 31-16 are calculated internally, and are not writable. | |

### 14.5: RXQUE Length Registers

These registers specify the length (number of valid entries) of the corresponding receive queue and are cleared when the corresponding lower bound is written. These registers can be used to query the status of a receive queue.

| | | |
|---|---|---|
| **Length** | 15 bits | |
| **Type** | Read | |
| **Address** | Queue 0 | XXXX 1810 |
| | Queue 1 | XXXX 1850 |
| | Queue 2 | XXXX 1890 |
| | Queue 3 | XXXX 18D0 |
| | Queue 4 | XXXX 1910 |
| | Queue 5 | XXXX 1950 |
| | Queue 6 | XXXX 1990 |
| | Queue 7 | XXXX 19D0 |
| **Power on Value** | X'0000' | |
| **Restrictions** | These registers can only be written in diagnostic mode. | |

### 14.6: RXQUE Threshold Registers

These registers specify a queue length threshold at which the corresponding status bit is generated. These registers should be set equal to the number of queue entries that should cause status to be generated. For example, if the value was set to five, then no interrupt would be generated until five or more events were queued on the corresponding receive queue. The threshold is level sensitive, so as long as the length is greater than or equal to the threshold, the corresponding status bit is set. When this register is set to zero, no thresholding is done.

When the direction bit is set for a receive queue, the threshold has the opposite polarity. For example, as long as there are more events in the queue than specified in the threshold register no status would be raised.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read | |
| **Address** | Queue 0 | XXXX 1818 |
| | Queue 1 | XXXX 1858 |
| | Queue 2 | XXXX 1898 |
| | Queue 3 | XXXX 18D8 |
| | Queue 4 | XXXX 1918 |
| | Queue 5 | XXXX 1958 |
| | Queue 6 | XXXX 1998 |
| | Queue 7 | XXXX 19D8 |
| **Power on Value** | X'0000' | |
| **Restrictions** | During normal operations, these registers are read only. These registers can only be written when the diagnostic bit has been set in the control register. | |
| | Bits 31-16 are calculated internally, and are not writable. | |

**14.7: RXQUE Dequeue Registers**

These registers are used to retrieve the event at the head of the corresponding receive queue.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read | |
| **Address** | Queue 0 | XXXX 181C |
| | Queue 1 | XXXX 185C |
| | Queue 2 | XXXX 189C |
| | Queue 3 | XXXX 18DC |
| | Queue 4 | XXXX 191C |
| | Queue 5 | XXXX 195C |
| | Queue 6 | XXXX 199C |
| | Queue 7 | XXXX 19DC |
| **Power on Value** | X'00000000' | |
| **Restrictions** | This is a read only register, and all writes will be ignored. Events are only returned when the diagnostic bit is reset in the control register, otherwise zero will be returned. | |

**14.8: RXQUE Enqueue Registers**

These registers are used to enqueue user events at the tail of the corresponding receive queue.

| | | |
|---|---|---|
| **Length** | 32 bits | |
| **Type** | Read/Write | |
| | Queue 0 | XXXX 1820 |
| | Queue 1 | XXXX 1860 |
| | Queue 2 | XXXX 18A0 |
| | Queue 3 | XXXX 18E0 |
| | Queue 4 | XXXX 1920 |
| | Queue 5 | XXXX 1960 |
| **Power on Value** | X'00000000' | |
| **Restrictions** | All reads result in zero. RXQUE should be enabled to do this. | |

### 14.9: RXQUE Last Event Dropped Register

Contains the event that was last dropped. This register contains the last event that was dropped. It holds its value until the event dropped status bit is cleared.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1AC0 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.10: RXQUE Timestamp Register

Used to specify the current timestamp measured using the timestamp pre-scaler ticks. This register keeps the current timestamp. It counts based on the value in the RXQUE Time-stamp Pre-Scaler Register. It can be read or written at any time. It is cleared when the pre-scaler register is written.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1AC0 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.11: RXQUE Timestamp Pre-Scaler Register

Used to specify the time interval of each timestamp timer tick. This register determines the number of 30-ns intervals between timestamp timer ticks. The value in the register plus one is the number of 30-ns intervals between timestamp timer ticks. So, the default value of zero means that the timestamp timer ticks every 30ns. If a value of four is placed in this register, the timestamp timer ticks every 150ns (5∗30 ns).

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1A10 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.12: RXQUE Timestamp Shift Register

Used to specify the number of bits to shift the timestamps. This register determines the number of bits that the timestamps are shifted. For example, if a value of zero is placed in this register, then the timestamp is not shifted, and the low order six bits are lost. If a value of two is placed in this register, then the timestamps are shifted two places and only the low order four bits of the timestamp are lost. This allows the user to control what portion of the timestamp is lost due to the low order event bits.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1A14 |
| **Power On Value** | X'00000002' |
| **Restrictions** | None |

### 14.13: RXQUE Event Routing Registers

Used to specify to which receive queue different types of events should be routed. These registers contain the receive queue that different types of events should be routed to. See *Event Summary and Routing Info on page 301* for event type mappings.

| | | |
|---|---|---|
| **Length** | 3 bits | |
| **Type** | Read/Write | |
| **Address** | Tx Complete | XXXX 1A3C |
| | Counter Overflow | XXXX 1A40 |
| | Error | XXXX 1A44 |
| | Tx Dma Comp | XXXX 1A48 |
| | Rx Dma Comp | XXXX 1A6C |
| | Pools Status | XXXX 1A88 |
| | ABR | XXXX 1A8C |
| **Power on Value** | X'00000000" | |
| **Restrictions** | For code compatibility, the Rx Dma Comp Routing register is written when the Tx Dma Comp Routing register is written. If both registers need to be written, Tx Dma Comp Routing register should be written first and then the Rx Dma Comp Routing register. | |

### 14.14: RXQUE Event Latency Timer Register

Used to specify the event latency time interval. This register is specified in 30-ns intervals. When a new event is placed on a receive queue, the event latency timer is started (if not already started). When this timer expires, the event latency timer expired status bit is set, and the timer is stopped. The status bit must be reset before the timer is started again. Every time the status register (or prioritized status) is accessed, the timer is stopped. If this register is written while the timer is running, the new value takes effect immediately. If this register is set to zero, the latency timer does not run.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1A18 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.15: RXQUE Interrupt Enable Registers

Used to specify which status register bits should be used to generate interrupts. Used to specify which status register bits should be used to generate interrupts. Each mask register is used to drive a different rxque status bit in intst as specified below. The different masks and status bits allow a high and low priority rxque interrupt on both the interrupt A and B pins. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *RXQUE Status and Enabled Status Registers on page 323* for the bit descriptions.

| | | |
|---|---|---|
| **Length** | 27 bits | |
| **Type** | Clear/Set | |
| **Address** | Enable 1 | XXXX 1A20 and A24 |
| | Enable 3 | XXXX 1A90 and A94 |
| | Enable 5 | XXXX 1AA0 and AA4 |
| | Enable 7 | XXXX 1AA0 and AA4 |
| **Power on Value** | Enable 1 | X'07ffff00' |
| | Enable 3 | X'07ffff00' |
| | Enable 5 | X'00000000' |
| | Enable 7 | X'00000000' |
| **Restrictions** | None | |

### 14.16: RXQUE Status and Enabled Status Registers

This register contains the status bits. The enabled version of these regs provide a version of the status register that is masked with the corresponding interrupt enable register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The following are the bit descriptions:Used to relay RXQUE status information.

| | | |
|---|---|---|
| **Length** | Masked With None | 27 bits |
| | Masked With Enable 1, 3, 5, or 7 | 32 bits |
| **Type** | Masked With None | Clear/Set |
| | Masked With Enable 1, 3, 5, or 7 | Read Only |
| **Address** | Masked With None | XXXX 1A28 and A2C |
| **Address** | Masked With Enable 1 | XXXX 1AB0 |
| | Masked With Enable 3 | XXXX 1AB4 |
| | Masked With Enable 5 | XXXX 1AB8 |
| | Masked With Enable 7 | XXXX 1ABC |
| **Power on Value** | X'00000000' | |
| **Restrictions** | Only the Event LatenEnable 1cy Timer Expired bit is writable. The enabled status registers are read only. | |



| Bit(s) | Name | Description |
|---|---|---|
| 31 | General Purpose Timer Status | This is a mirror of the general purpose timer status bit in interrupt status. |
| 30 | Int 1 Status or PCORE Normal Int Status | When read from the PCI bus, this bit indicates if there is status other than receive queue status and general purpose timer status in the interrupt source register using interrupt mask register 1 as a mask. When read from the PCORE polling interface, the mask used is the interrupt status 1. |
| 29 | Int 2 Status or PCORE Critical Int Status | When read from the PCI bus, this bit indicates if there is status other than receive queue status and general purpose timer status in the interrupt source register using interrupt mask register 2 as a mask. When read from the PCORE polling interface, the mask used is the PCORE normal interrupt status in the interrupt status register. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 28 | Reserved | Reserved |
| 27 | Reserved | Reserved |
| 26 | Reserved | Reserved |
| 25 | Event Latency Timer Expired | When this bit is set, the event latency timer has expired. This indicates that new events are waiting to be processed on some queue(s) and the queue has not been processed for a period equal to the latency timer. This bit must be reset to re-enable the event latency timer. |
| 24 | Event Dropped | When this bit is set, at least one event has been dropped. RXQUE Last Event Dropped Register contains the event that was dropped. |
| 23-16 | Head Valid | These bits are set when the head is valid for queue 7-0 respectively. If these bits are set, then a dequeue operation should complete successfully. |
| 15-8 | Threshold Exceeded | These bits are set when the queue length register equals or exceeds the value in the queue threshold register for queue 7-0 respectively. |
| 7-0 | Queue Full/Empty | |

## 14.17: RXQUE Control Register

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. This register contains the mode bits that specify how RXQUE is to operate. It is used to set RXQUE modes.

**Length**         32 bits

**Type**          Clear/Set

**Address**       XXXX 1A30 and A34

**Power On Value**  X'00000000'

**Restrictions**    None



| Bit(s) | Name | Description |
|---|---|---|
| 31 | Reset FIFO | When this bit is set, the internal FIFO is flushed, and this bit is reset. The result is this bit will always be read as a zero. This bit can only be set in diagnostic mode. |
| 30-16 | Reserved | Reserved. |
| 15-8 | Queue Direction | When set, the direction of the corresponding queue is assumed to be reversed. This only affects the full condition, the threshold exceeded condition, and the high water threshold exceeded condition. When this bit is set, the polarity of these status signals changes. So, the full condition becomes an empty condition, and the two threshold conditions trigger when the length of the queue is less than the corresponding threshold instead of greater than or equal. This mode is mainly used for queues that relay information from the system to the IBM2520L8767. Also, event enqueues to a queue that is reversed do not start the event latency timer (since no new event for system has arrived). |
| 7 | Always Route Error Events | When this bit is set, all error events are routed to the error queue even if receive bad frames (bit 2) is turned on. When cleared, error events are only routed to the error queue if receive bad frames is turned off. This bit allows the user to keep bad frames in time sequence with good frames or to route them to the error queue. The clear state of this bit is code compatible with previous passes. |
| 6 | Enable Chip Overflow Events | When set, the chip level counter overflow events are surfaced. |
| 5 | Memory Select | When this bit is set, RXQUE will use packet memory instead of control memory to store the event queues. |
| 4 | Inhibit Enqueues | When this bit is set, the enqueue state machine will not accept any new enqueue requests. This should be used in extreme cases as it holds off all enqueues indefinitely. |
| 3 | Timestamp Mode | When this bit is set, timestamp events are inserted before each real event. The timestamps correspond to when the event happened on chip. When this bit is off, timestamps can still be read from the timestamp register. The timestamps would correspond to when the event was dequeued in this scenario. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 2 | Receive Bad Frames | When this bit is set, bad frame events (all error events), will be received in the normal receive queue defined in the LCD. All buffers are not freed, and the packet address is raised in the event data. |
| | | When this bit is reset, bad frame events are routed to the receive queue specified by the Error Event Receive Queue Register. All packet based events will carry the LC address in the event data instead of the packet address. All buffers are freed back to POOLs. |
| | | This bit should only be changed sparingly, because it changes the way packets are freed and what is surfaced in an event (LCD vs frame pointer). It should really only be changed when the receive side is inactive. |
| 1 | BCACH Advice | This bit when set allows RXQUE to give BCACH cache fill advice based on events that are dequeued. |
| 0 | Diagnostic Mode | When this bit is set or when the chip is disabled, the RXQUE entity is in diagnostic mode and primitive execution is disabled. |

## 14.18: RXQUE Control 2 Register

This register contains the mode bits that specify how RXQUE is to operate. It is used to set RXQUE Queue modes. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The following are the bit descriptions:

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 1A80 and A84 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

Disable BCACH Advice for Queue 7-0        Disable Timestamps for Queue 7-0
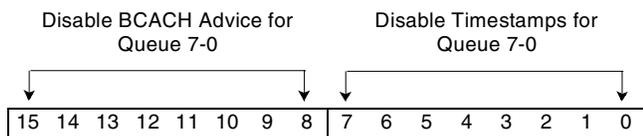
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 15-8 | Disable BCACH Advice for Queue 7-0 | When set, the BCACH advice is disabled for queue 7-0. This is necessary in order to use a queue as a general purpose container for user data. |
| 7-0 | Disable Timestamps for Queue 7-0 | When set, timestamps are disabled for queue 7-0. This is necessary in order to run cut-through modes. |

**Debugging Register Access**

This section is a very brief documentation of access that has been put in for the internal registers of RXQUE. These addresses need not be written or read during normal operations.

### 14.19: RXQUE RXQ State Machine Variable Register

Main state variable for RXQ processing state machine.

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1A38 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.20: RXQUE RXQ ENQ State Machine Variable Register

Main state variable for RXQ processing state machine.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1A4C |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

### 14.21: RXQUE Enq FIFO Head Ptr Register

Points to the head FIFO entry in the FIFO array. The MSB bit, is used to determine if the head is chasing the tail, and is inverted each time the head pointer wraps. It is used to maintain the enqueue FIFO.

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1B78 |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be written in diag mode. |

### 14.22: RXQUE Enq FIFO Tail Ptr Register

Points to the next free FIFO entry in the FIFO array. It is used to maintain the enqueue FIFO. The MSB bit, is used to determine if the head is chasing the tail, and is inverted each time the tail pointer wraps.

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1B7C |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be written in diagnostic mode. |

### 14.23: RXQUE Enq FIFO Array

Holds events waiting to be placed on a receive queue. Array is organized as a 16x36 array. To access the upper four bits of each word (holds the receive queue number for event), the array word number should be used as the address. To access the low order 32 bits (the event portion), the array word number times two plus four should be used. For example, address zero accesses the receive queue portion of array word zero and address four accesses the event portion of the array word zero.

**Note:** The most significant bit is not used. Only the three bits are needed for the receive queue number.

| | |
|---|---|
| **Length** | 16 words X 36 bits |
| **Type** | Read/Write |
| **Address** | XXXX 1B80-BFC |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be read/written in diagnostic mode. When read in non-diagnostic mode, zero is returned. |

# PHY Level Interfaces

## Entity 15: The PHY Interface (LINKC)

### Functional Description

LINKC provides the interface between the IBM2520L8767 and either an ATM PHY device or, when the internal framer is selected, a serializer/deserializer device. LINKC is composed of three pieces. LINKX, which contains all the registers described below, is clocked with the same clock as other parts of the chip. LINKT, the transmit logic, is clocked on the transmit clock, which is selected via the Clock Control Register (described on page 388). LINKR, the receive logic, is clocked on the receive clock, which is also selectable via the Clock Control Register. Transmit and receive transfers are synchronized via their respective interface transfer clock. The data path size is 8 or 16 bits wide and is selectable through bit three of the control register. The PHY devices that the IBM2520L8767 interfaces to are:

- PMC SIERRA PM5346 SUNI LITE FOR SONET STS-3c 155.52 MB/s
- UTOPIA 8 or 16 bit interface.
- IBM 25 MB/s over UTP.

## 15.1: LINKC Control Register

This register contains the information which controls the operation of LINKC. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0B30 AND 34 |
| **Power On Value** | X'00006D' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-30 | Additional header bytes for router function | The value of bits 31-30 indicate the number of additional header bytes that will be read from segbuf and added to the beginning of each cell as each is transmitted to the PHY. The bytes are meant to be used for additional routing information. These control bits have no effect in IBM 25 Mb/s PHY mode, and should be set to zeros when in internal SONET/SDH framer mode. If used in conjunction with 52-byte mode, the byte normally containing the cell HEC will not be transmitted and the total number of cells transmitted will be the value of this field plus 52. If 16-bit PHY mode is selected, by default, the byte alignment will follow that of normal 52- or 53-byte 16-bit mode, with the additional header bytes contiguously prepended. As a result, a mode with three additional header bytes cannot be obtained in 53-byte, 16-bit mode (LSB is normally padded with zeros so MSB gets truncated). Bit three of this register is therefore provided to adjust the alignment in 16-bit, 53-byte mode so all five header bytes will be transmitted with up to three additional router bytes prepended. |

| Bit(s) | Name | Description |
|---|---|---|
| 29 | Disable Limited HEC Checking on Received Idle/Unassigned Cells | If bit 29 is set to '1', the receive logic will ignore the HEC byte of the header of idle and unassigned cells. Idle is defined as a header of X'00000001' and unassigned is defined as a header of X'0000000n' where n is 'xxx0'. If bit nine is set to enable unassigned/idle cell reception, all cells will be passed to REASM regardless of how this bit is set. If bit nine is set to disable unassigned/idle cell reception and this bit is set to '0', the HEC byte of cells with an apparent idle header will be completely checked before deciding whether or not to pass the cell to REASM. If a cell appears to have an unassigned header, HEC bits 7,6, and 0 will be checked because they are a constant regardless of the value of bits 3-1 of the header. If other HEC bits are bad, REASM will detect the HEC error and discard the cell. If there is a correctable HEC error and the cell is indeed unassigned, an out of range error will occur in REASM. |
| 28 | Disable HEC Generation on Transmitted Idle Cells | If bit 28 is set to '1', X'00' will be placed in the HEC byte of idle cells (assuming bit four is set to enable idle cell generation). This bit set to '0' allows HEC to be generated on idle cells. |
| 27 | Transmit PHY Sampling Latch Override | If bit 26 is set, this bit determines whether the transmit PHY interface uses sampling latches on its inputs or not. Set to '1', this bit enables the use of sampling latches. |
| 26 | Enable Transmit PHY Sampling Latch Override | When set, bit 26 will allow bit 27 to determine whether the transmit PHY interface uses sampling latches on its inputs or not regardless of the setting of bit 14. |
| 25 | Ignore GFC in Null/Idle Cell Determination | When set, bit 25 causes the receive logic to ignore the first four bits of the ATM header in determining whether a cell being received is a null or idle cell. |
| 24 | Enable XON/XOFF | When set, bit 24 allows the XON/XOFF bit of the header of a received cell to suspend/continue transmission from the IBM2520L8767's transmit logic. |
| 23-21 | Transmit PHY Device | 23-21 override the settings in bits 7-5 when bit 20 is set to '1'. These bits only effect the PHY transmit interface. They select the transmit PHY interface with the same encoding as bits 7-5. |
| 20 | Enable Transmit PHY Interface Override | This bit, when set to '1' allows bits 23-21 to override the PHY interface setting in bits 7-5. This only alters the transmit PHY interface. |
| 19-17 | Receive PHY Device | 19-17 override the settings in bits 7-5 when bit 16 is set to '1'. These bits only effect the PHY receive interface. They select the receive PHY interface with the same encoding as bits 7-5. |
| 16 | Enable Receive PHY Interface Override | when set to '1', this bit allows bits 19-17 to override the PHY interface setting in bits 7-5. This only alters the receive PHY interface. |
| 15 | Enable Output Latches | when set to '1', this bit causes LINKT and LINKR to use sampling latches on PHY outputs in UTOPIA mode. This should be set to '0' in normal operation and is only provided for a possible backward compatibility situation with the 1.5 release. |
| 14 | Use Sampling Latches on PHY Inputs | when set to '1', this bit causes LINKT and LINKR to use sampling latches on PHY inputs. When set to '0', the inputs are used raw. This should be set to '0' when using the internal SONET Framer. |
| 13 | Disable PHY Bus Drivers | when set to '1', this bit tri-states the drivers of the PHY bus. When set to '0', the drivers are enabled. |
| 12-11 | Reserved | Reserved. |
| 10 | Enable Parity Checking | When set, this bit enables checking of parity on data from the receive path. The default is parity checking is disabled. The upper bit of the transmit parity is not valid when the the internal SONET/SDH Framer has been selected as the receive PHY device. The upper bit of the receive parity is also not valid when the internal SONET/SDH Framer has been selected as the transmit PHY device. This is only a concern if a combination of the internal framer and an external PHY is being used and that external PHY has a 16-bit data interface. In this case, parity cannot be checked/generated on the upper byte. |
| 9 | Unassigned/Idle Cell Reception | When set to '1' , this bit enables unassigned/idle cell reception. This should be set to '0' when using the internal SONET Framer. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 8 | Modify byte alignment in 16-bit PHY mode | When set to '1', this bit changes the default byte alignment in 16-bit PHY mode if this register also contains a non-zero value in bits 31-30. See the description of those bits for further details. |
| 7-5 | PHY Device | 7, 6 and 5 indicate which PHY the IBM2520L8767 will be interfacing.<br>'000' Reserved<br>'001' Reserved<br>'010' IBM 25 Mb/s<br>'011' PMC PM5346 SUNI LITE/UTOPIA interface (STS-3c/STM-1 OR STS-1)<br>'100' Internal SONET(sts-3c)/SDH(STM-1) Framer with external SERDES (Parallel interface)<br>'101' Internal SONET(sts-3c)/SDH(STM-1) Framer with internal SERDES (Serial interface). |
| 4 | Support Null Cell Generation | When set, this bit enables the IBM2520L8767 to send unassigned cells to the assigned cell stream. This bit needs to be set when interfacing with a PHY that doesn't support null cell generation and has synchronous cell time slots(e.g.SONET, DS3). This should be set to '0' when using the internal SONET Framer. |
| 3 | PHY Data Path Size | This bit, when set to '0', selects a 16-bit wide data path to the PHY device. When set to '1', the data path width to the PHY will be eight bits. This bit has no affect on the internal SONET/SDH framer except if the internal framer has been selected as the Rx PHY device but not as the transmit PHY device. In this case, '1' on this bit will allow FYTDAT(15 - 13) to be used for the 16-bit external transmit PHY device, while '0' will allow FYTDAT(15 - 13) to be used for the receive HDLC controller. This implies that it is not possible to use the internal receive framer, the receive HDLC interface, and an external 16-bit transmit framer at the same time. |
| 2 | Loop Back Mode | This bit set to '1' places the IBM2520L8767 in an internal loop back mode. The PHY interface will be disabled. The clocks to LINKT and LINKR should be set to the same source in *Clock Control Register (Nibble Aligned)* (described on page 388). This bit is flushed to '1' after POR. |
| 1 | 52 Byte Cell | When set, the cell sent to the PHY will be 52 bytes. NO HEC byte will be sent or received. This bit does not affect the internal SONET Framer. |
| 0 | Even/Odd Parity Selection | Even parity is selected when this bit is cleared. The default value is for Odd parity. |

### 15.2: LINKC Transmitted HEC Control Byte

The transmitted control byte (HKCS) is an error mask that allows the insertion of one or more errors into the HEC byte (byte five of the ATM cell header). A logic one in a given position causes the inversion of the corresponding HEC bit position.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0B04 |
| **Power On Value** | X'00' |

Initiate Inversion
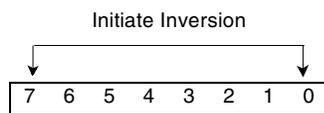
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Description |
|---|---|
| 7-0 | When set to '1', any of these bits will cause the inversion of the corresponding bit in the generated HEC byte. |

### 15.3: LINKC Interrupt/Status Register

This register reports the status of LINKC.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0B10 AND 14 |
| **Power On Value** | X'00' |



| Bit(s) | Description |
|---|---|
| 7 | Reserved. |
| 6 | Indicates that the IBM2520L8767 has received a null cell. |
| 5 | Indicates a parity error on the upper byte of receive data from the PHY. |
| 4 | Indicates a parity error on the lower byte of receive data from the PHY. |
| 3-1 | Reserved. |
| 0 | No carrier detect from the PHY. |
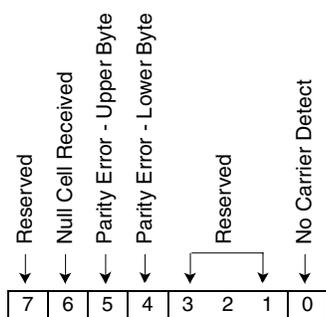
### 15.4: LINKC Interrupt Enable Register

This register enables the LINKC interrupt to INTST. When a bit is set in this register and the corresponding bit is set in the interrupt status register, the LINKC interrupt will be driven. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0B18 AND 1C |
| **Power On Value** | X'00' |
| **Restrictions** | None |

### 15.5: LINKC Prioritized Interrupts

Used to access the prioritized encoding of LINKC interrupts. Reading this location will give a decimal number that is the prioritized encoding of bits 7 - 0 in COMET/PAKIT Status Register (seven being the most significant bit) assuming the corresponding enable bit is on.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | XXXX 0B2C |
| **Power On Value** | X'00' |
| **Restrictions** | None |

### 15.6: LINKC Transmit State Machine Register

This register indicates the state of the transmit sequencer.

| | |
|---|---|
| **Length** | 3 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0B24 |
| **Power On Value** | X'0' |
| **Restrictions** | None |

Transmit
state
machine
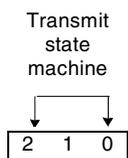
| 2 | 1 | 0 |
|---|---|---|

| Bit(s) | Description |
|---|---|
| 2-0 | Transmit state machine. |

### 15.7: LINKC Receive State Machine Register

This register indicates the state of the receive sequencer.

| | |
|---|---|
| **Length** | 2 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0B28 |
| **Power On Value** | X'0' |
| **Restrictions** | None |

Receive State Machine

| 1 | 0 |
|---|---|

| Bit(s) | Description |
|---|---|
| 1-0 | Receive state machine. |

### 15.8: LINKC Unassigned Cell Payload Data

This register contains the data within the payload of an unassigned cell. The lower address selects bits 63 - 32 and the higher address selects bits 31 - 0.

| | |
|---|---|
| **Length** | 64 bits |
| **Type** | Write/Read |
| **Address** | XXXX 0B38 AND 3C |
| **Power On Value** | X'AAAA55555555AAAA' |
| **Restrictions** | None |

### 15.9: LINKC Unassigned Cell Payload Data -- BIT REVERSED

This register contains the same data as the LINKC Unassigned Cell Payload Data register except each byte is bit-reversed. 31 - 0.

| | |
|---|---|
| **Length** | 64 bits |
| **Type** | Read |
| **Address** | XXXX 0B08 AND 0C |
| **Power On Value** | X'5555AAAAAAAA5555' |
| **Restrictions** | None |

**15.10: LINKC Passed TX Data Register**

The bits in this register are passed over the PHY transmit data I/O pins 15-8 when using an eight-bit wide PHY data bus.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0B40 |
| **Power On Value** | X'00' |
| **Restrictions** | None |

| Bit(s) | Description |
|:---:|---|
| 7 | Passed to IBM2520L8767 FYTDAT(15) |
| 6 | Passed to IBM2520L8767 FYTDAT(14) |
| 5 | Passed to IBM2520L8767 FYTDAT(13) |
| 4 | Passed to IBM2520L8767 FYTDAT(12) |
| 3 | Passed to IBM2520L8767 FYTDAT(11) |
| 2 | Passed to IBM2520L8767 FYTDAT(10) |
| 1 | Passed to IBM2520L8767 FYTDAT(9) |
| 0 | Passed to IBM2520L8767 FYTDAT(8). |

## 15.11: LINKC PDH Interface Register

The bits in this register are used to make adjustments to the Utopia interface for compatibility with the SUNI-PDH PHY.

**Length**          8 bits

**Type**            Read/Write

**Address**         XXXX 0B44

**Power On Value**  X'00'

**Restrictions**    None

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-3 | | Reserved. |
| 2 | Drive RENB Inactive When Not Receiving | When set to '1', this bit forces the receive logic to deactivate RENB when in the idle state. |
| 1 | Gate RSOC with RCA | When set to '1', this bit forces the receive logic to see both RSOC and RCA before considering RSOC valid. |
| 0 | Receive Extra Header Byte | When set to '1', this bit allow an extra header byte to be accepted at the start of a cell by the receive logic. The extra byte is discarded. |

# Entity 16: Nodal Processor Bus Interface (NPBUS)

This entity controls the signals of the NP Bus. The PHY registers are accessible to the processor by way of the address space of the IBM2520L8767. In addition, the operation of the front end is effected by the NPBUS Status Register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

## 16.1: NPBUS Control Register

This register is used to report PHY level hardware errors and interrupts.

| | |
|---|---|
| **Length** | 28 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 2000 and 004 |
| **Power On Value** | X'0002010' |
| **Restrictions** | None |

Bit assignments (bit : name):

| Bit | Name |
|---|---|
| 27 | Status LED 4 Toggle |
| 26 | Status LED 3 Toggle |
| 25 | Status LED 2 Toggle |
| 24 | Status LED 1 Toggle |
| 23 | Status LED 4 Flashing |
| 22 | Status LED 3 Flashing |
| 21 | Status LED 2 Flashing |
| 20 | Status LED 1 Flashing |
| 19 | Status LED 4 On |
| 18 | Status LED 3 On |
| 17 | Status LED 2 On |
| 16 | Status LED 1 On |
| 15 | +UTP/-STP Interface Select |
| 14 | Disable driving the NP address over the ENSTATE(47 - 32) pins |
| 13 | Enable Carrier Detect LED |
| 12 | Enable PHY Data Bus Parity Detection |
| 11 | Invert Interrupt Inputs and PHY reset |
| 10 | Access Internal SONET Framer register space |
| 9 | PHY Bus Interface Type |
| 8 | Enable Hardware Error to Disable PHY |
| 7 | Reboot serial/parallel EPROM |
| 6 | Remove Internal SONET Framer from reset state |
| 5 | Force a PHY logic reset |
| 4 | Enable the front end |
| 3 - 2 | PB0PHY2 Control |
| 1 | External EPROM Type |
| 0 | Reserved |

| Bit(s) | Name | Description |
|---|---|---|
| 27 | Status LED 4 Toggle | When this bit is set, the state of bit 19 of this register will be toggled by repeatedly setting bit 19. |
| 26 | Status LED 3 Toggle | When this bit is set, the state of bit 18 of this register will be toggled by repeatedly setting bit 18. |
| 25 | Status LED 2 Toggle | When this bit is set, the state of bit 17 of this register will be toggled by repeatedly setting bit 17. |
| 24 | Status LED 1 Toggle | When this bit is set, the state of bit 16 of this register will be toggled by repeatedly setting bit 16. |
| 23 | Status LED 4 Flashing | When set to '1', this bit will flash status indicator LED 4. Bit 19 of the register will override this bit. |
| 22 | Status LED 3 Flashing | When set to '1', this bit will flash status indicator LED 3. Bit 18 of the register will override this bit. |
| 21 | Status LED 2 Flashing | When set to '1', this bit will flash status indicator LED 2. Bit 17 of the register will override this bit. |

| Bit(s) | Name | Description |
|---|---|---|
| 20 | Status LED 1 Flashing | When set to '1', this bit will flash status indicator LED 1. Bit 16 of the register will over-ride this bit. |
| 19 | Status LED 4 On | When set to '1', this bit will turn on status indicator LED 4. |
| 18 | Status LED 3 On | When set to '1', this bit will turn on status indicator LED 3. |
| 17 | Status LED 2 On | When set to '1', this bit will turn on status indicator LED 2. |
| 16 | Status LED 1 On | When set to '1', this bit will turn on status indicator LED 1. |
| 15 | +UTP/-STP Interface Select | This bit controls a chip output pin to switch high or low, and can be used to select different PHY interfaces, etc. Where this bit is off, or a logical '0', the chip output is high, or a logical '1'. |
| 14 | Disable driving the NP address over the ENSTATE(47 - 32) pins | For debug reasons, the driven of the address for eprom and phy fetches can be turned off with this bit. |
| 13 | Enable Carrier Detect LED | When set to '1', this bit allow indicator LED 1 to reflect the status of Carrier Detect. This is a chip input. |
| 12 | Enable PHY Data Bus Parity Detection | When set to '1', if a parity error occurs on the PHY Data bus during a PHY register access, bit one of the NPBUS Status Register will be set. |
| 11 | Invert Interrupt Inputs and PHY reset | When set to '0', the PHY interrupt chip inputs will set bits two or three of the NPBUS Status Register when active low. When set to '1', the interrupt inputs will be inverted such that a positive input on the PHY interrupt lines will set bits two or three of the NPBUS Status Register, and the PHY reset out line will be inverted so that a high level will be a reset. This would be used for an IBM TRAC or compatible chip. |
| 10 | Access Internal SONET Framer register space | When this bit is zero, the external PHY register space can be accessed through PHY 1 Registers or PHY 2 Registers. When this bit is set to '1', the internal SONET framer registers can be accessed (see *Sonet Framer Core* on page 397). The full offset range for this access is X'2100' to X'2FFF'. |
| 9 | PHY Bus Interface Type | When this bit is '0', PHY access speed is 200ns (SUNI-like interface). When '1', access requires an acknowledge input response. This is to support a UTOPIA-like micro-processor interface. |
| 8 | Enable Hardware Error to Disable PHY | Allows bit four (Master enable) of the INTST Control Register register to reset bit four of this register. (disables Front End logic). This function assumes that bit four of the INTST Control Register register has already been enabled and that either a hardware or software event has turned the bit off. |
| 7 | Reboot serial/parallel EPROM | This bit will restart the external serial or parallel EPROM initialization code. The access time expected for the serial EPROM is the number of crisco instructions x the instruction length in bytes (typically seven bytes) + three overhead bytes to address the serial EPROM plus one ending instruction. This will give the total number of bytes. There are nine clocks per byte. Multiply the total number of clocks times the serial EPROM clock period (either 10μs or 20μs depending on the FFCFG bit settings and assuming a PCI 33-MHz clock). Example: 13 intrs x 7 + 3 + 1 = 95 bytes. $95 * 9$ clocks $* 20$μs = 17.1ms. |
| 6 | Remove Internal SONET Framer from reset state | This bit powers up to a zero and keeps the internal SONET Framer in reset mode. Setting this bit to a '1' will enable normal operation. |
| 5 | Force a PHY logic reset | Before any software reset, turn this bit on and off for the PHY specified amount of time. If the IBM ATM-TC (25 Mb/s ENDEC) is used, this bit will power-up to an active reset (since the input to the ENDEC is positive reset). This bit must then be turned off for normal operation. |
| 4 | Enable the front end | When this bit is '0', no data will be transmitted to or received from the PHYs or IBM2520L8767. See bit eight for more information on control of this bit. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 3-2 | PB0PHY2 Control | Encoded control for the PB0PHY2 output pin. The enabled of PIBSELO overrides these bits and is controlled by PCINT Cascade Control Register.<br>X'0'   Enable PB0PHY2 pin<br>X'1'   Enable PBDATAP pin and its detection of valid parity.<br>X'2'   Enable MPMDSEL pin<br>X'3'   Reserved |
| 1 | External EPROM Type | This bit will set at reset time as to what type of EPROM is detected. When set, a serial EPROM has been detected. When '0', parallel EPROM is assumed (or none at all). This will also indicate from what type of device a PCI ROM access will retrieve VPD data. |
| 0 | Reserved | Reserved |

## 16.2: NPBUS Status Register

This register is used to report PHY level hardware errors and interrupts. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 7 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 2028 and 02C |
| **Power On Value** | X'x1', where x is determined by which type of Crisco IPL EPROM is used |
| **Restrictions** | None |

| Parallel EPROM Access Complete | Serial EPROM Access Failed | Serial EPROM Access Complete | Internal SONET Framer interrupt | Interrupt PHY1 | PHY Data Bus Parity Error | Crisco Execution Complete |
|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 6 | Parallel EPROM Access Complete | The requested action to the parallel eprom has been completed. See NPBUS EPROM Address/Command Register. |
| 5 | Serial EPROM Access Failed | The requested action to the serial EPROM has missed an acknowledge sequence while trying to complete the action. See NPBUS EPROM Address/Command Register. |
| 4 | Serial EPROM Access Complete | The requested action to the serial EPROM has been completed. See NPBUS EPROM Address/Command Register. |
| 3 | Internal SONET Framer interrupt | The internal Framer has signaled an interrupt. |
| 2 | Interrupt PHY1 | This bit indicates that an interrupt occurred on PHY 1. |
| 1 | PHY Data Bus Parity Error | When set to '1', a data parity was detected over the PHY Data eight-bit bus. Parity checked is odd. |
| 0 | Crisco Execution Complete | External serial/parallel EPROM initialization has run and is completed. |
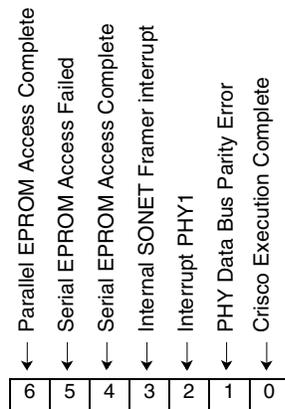
**16.3: NPBUS Interrupt Enable Register**

This register is used to mask bits from the NPBUS Status Register and potentially generate interrupts to the control processor. When both a bit in this register and the corresponding bit in the NPBUS Status Register are set, the NPBUS status bit will be set in NPBUS Status Register. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See NPBUS Status Register for the bit descriptions.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 2008 and 00C |
| **Power On Value** | X'00' |
| **Restrictions** | None |

## 16.4: NPBUS EPROM Address/Command Register

Used to accessed a maximum of 2K external serial EPROM or 16Mg of parallel EPROM.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 2010 |
| **Power On Value** | X'00000100' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-28 | Reserved | Reserved |
| 27 | Execute Parallel EPROM Access | This bit will start a read or write function to the parallel EPROM. This bit will auto reset after the command is issued. |
| 26 | More Bytes to Read/Write Serial EPROM | This bit set to '1' will help speed up sequential acesses to the serial EPROM. If writing, there is a limit as to how many bytes can be written before the serial EPROM write buffer is full. Typical range is from two to eight bytes, depending on the device. |
| 25: | Not read/Write | This bit set to '1' will cause a write function to the serial/parallel EPROM. This bit set to '0' will cause a read function to the serial/parallel EPROM. |
| 24 | Execute Serial EPROM Access | This bit will start a read or write function to the serial EPROM. This bit will auto reset after the command is issued. |
| 23-0 | EPROM/Extended PHY Address | This holds the address field that will be used address the serial/parallel EPROM. It is also where the 15 - 8 address bits will be held if addressing a PHY with more address ability than eight bits. |
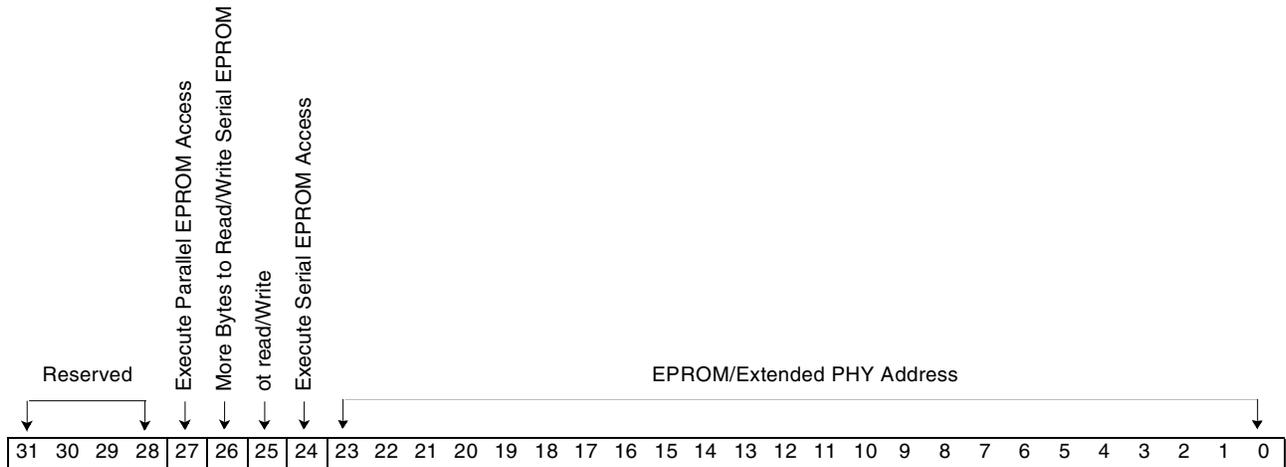
## 16.5: NPBUS EPROM Data Register

Used to accessed a maximum of 2K external serial EPROM or 16 Meg of parallel EPROM.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 2018 |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

| Reserved | Read Data | Reserved | Write Data |
|---|---|---|---|

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

| Bit(s) | Name | Description |
|---|---|---|
| 31-24 | | Reserved |
| 23-16 | Read Data | The holds the data that was read back from the serial/parallel EPROM. |
| 15-8 | | Reserved |
| 7-0 | Write Data | The holds the data that is destined to be written to the serial/parallel EPROM. |
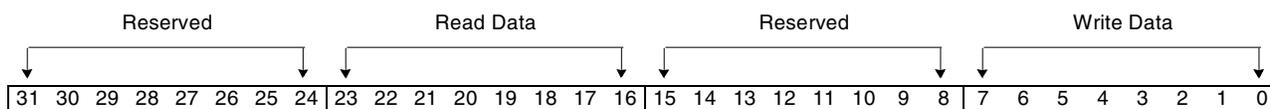
## 16.6: PHY 1 Registers

This address range provides access to the PHY 1 hardware. The details of the registers can be found in the specific publication for the PHY hardware.

| | |
|---|---|
| **Length** | 256 Doublewords (Only lowest eight bits valid) |
| **Type** | Read/Write |
| **Address** | XXXX 2400 - 7FF |
| **Power On Value** | Reference the PHY-specific publication. |
| **Restrictions** | Reference the PHY-specific publication. |

### PHY 2 Registers

This address range provides access to the PHY 2 hardware. It should be noted that not all applications of IBM2520L8767 will use this access port. The details of the registers can be found in the specific publication for the PHY hardware.

| | |
|---|---|
| **Length** | 256 Doublewords (Only lowest 8 bits valid) |
| **Type** | Read/Write |
| **Address** | XXXX 2800 - BFF |
| **Power On Value** | Reference the PHY-specific publication. |
| **Restrictions** | Reference the PHY-specific publication. |

# Hardware Protocol Assist Entities

## Entity 17: On-chip Checksum and DRAM Test Support (CHKSM)

**Functional Description**

The CHKSM entity has two functions. First, it is capable of initializing and/or testing packet and control memory. Second, it can perform TCP checksums (2s complement, 16-bit sum with "end-around-carry").

### 17.1: CHKSM Base Address Register

The CHKSM Base Address Register indicates the starting address of a test operation or checksum calculation. The base address can be set up with any alignment and valid address.

This register increments with each read or write to memory. On a test mode error, the register holds the address of the 64 bit word which was read in error.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0A00 |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be written when CHKSM is not enabled (see EE bit in CHKSM Control Register). |

**17.2: CHKSM Read/Write Count Register**

The CHKSM Read/Write Count Register indicates the count of remaining bytes of a checksum operation. This register keeps track of how many bytes remain in the current checksum operation and is decremented with each read or write operation.

Any length can be set in the 30 lower significant bits.

The upper two bits of this register can be written when starting a checksum operation instead of writing the control register. The following are the meanings of the upper two bits:

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0A04 |
| **Power On Value** | X'00000000' |
| **Restrictions** | Can only be written when CHKSM is not enabled (see EE bit in CHKSM Control Register). |

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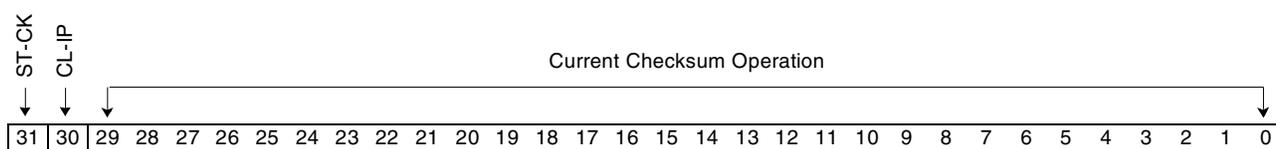
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 31 | ST-CK | Start a checksum operation. When this bit is written, bits 0 and 1 in the control register are set, and a checksum operation is started. This should only be done when the rest of the control register bits are already set up. (i.e. memory select, invert checksum, ...) |
| 30 | CL-IP | When this bit is written it will clear the CHKSM TCP/IP Checksum Data Register. This is the same as writing '1' to bit six of the CHKSM Control Register. |

### 17.3: CHKSM TCP/IP Checksum Data Register

The CHKSM TCP/IP Checksum Data Register collects the 16-bit, 2s complement, end-around-carry sum of the bytes. The source data is zero padded if it starts/ends on an odd byte boundary. The CHKSM TCP/IP Checksum Data Register collects the 16-bit, 2s complement, end-around-carry sum of the bytes. It can be seeded with an initial value. If it is not cleared before running a checksum, the previous value will act as a seed. This register can be cleared when starting a checksum operation by writing the CLIP bit in the CHKSM Control Register or by writing upper bits of CHKSM Read/Write Count Register.

See description of CHKSM Control Register for description of how to get/set current checksum alignment.

| | | |
|---|---|---|
| **Length** | 16 bits | |
| **Type** | Read/Write | |
| **Address** | Normal sum | XXXX 0A08 |
| | Inverted sum | XXXX 0A0c |
| | Swapped sum | XXXX 0A34 |
| | Inverted swapped sum | XXXX 0A38 |
| **Power On Value** | X'0000' | |
| **Restrictions** | Can only be written when CHKSM is not enabled (see EE bit in CHKSM Control Register) | |

### 17.4: CHKSM Ripple Base Register

This register is used as base of a ripple pattern when in test mode. This register forms the base for a ripple pattern. Each consecutive byte will be incremented by 1 or 8 in the pattern. The ripple mode must be set in the Control Register to use this feature. The value of this register will change during the test operation. So if a write and compare operation are being performed, this register needs to be setup again for the second operation.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0A14 |
| **Power On Value** | X'00' |
| **Restrictions** | Can only be written when CHKSM is not enabled (see EE bit in CHKSM Control Register). |

**17.5: CHKSM Ripple Limit Register**

This register is used to determine when the ripple base register overflows to zero. This register forms the compare value for the ripple base register. When the value of the ripple base reg is greater than or equal to this register, the base register will overflow to zero. For example, when this register is set to four, the ripple base register would count from zero through four if counting by one.

When set to 0x00, no overflows to zero occur. For example, when the ripple value is 0xff, and you are counting by eight, the next value would still be seven. If counting by one, then the next value would be zero.

This register should be written before the ripple base.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0A10 |
| **Power On Value** | X'ff' |
| **Restrictions** | Can only be written when CHKSM is not enabled (see EE bit in CHKSM Control Register). |

**17.6: CHKSM Interrupt Enable Register**

Used to specify which status register bits should be used to generate interrupts. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See "CHKSM Status Register" for the bit descriptions.

| | |
|---|---|
| **Length** | 12 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0A20 and 24 |
| **Power On Value** | X'ffe' |
| **Restrictions** | None |

### 17.7: CHKSM Status Register

Used to relay CHKSM status information. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The following are the bit descriptions:

| | |
|---|---|
| **Length** | 12 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0A18 and 1c |
| **Power On Value** | X'01' |
| **Restrictions** | The count zero bit is not writable. |



| Bit(s) | Name | Description |
|---|---|---|
| 11-4 | TEX -- Test Error in Byte 7-0 | When these bits are set, the checksum has determined that a read comparison error was encountered in the corresponding byte. Byte zero is bits 63 - 56 in a 64-bit long word. |
| 3 | LCK -- Comet Lock | When this bit is set, the checksum has determined that COMIT has ceased operation for some reason. |
| 2 | TE1 -- Test Error MSW | When this bit is set, checksum has determined that a read comparison error was encountered in the most significant 32-bit word. |
| 1 | TE0 -- Test Error LSW | When this bit is set, the checksum has determined that a read comparison error was encountered in the least significant 32-bit word. |
| 0 | CZ -- Count Zero | This bit is set when the terminal count of an operation is reached |
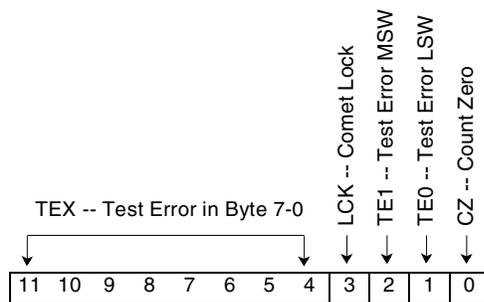
## 17.8: CHKSM Control Register

The various bits in this register control the mode in which the checksum entity operates. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. The various bits are described below:

| | |
|---|---|
| **Length** | 13 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 0A28 and 2c |
| **Power On Value** | X'00' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 12 | CL-FF -- Clear to All Ones | When this bit is set, the CHKSM TCP/IP Checksum Data Register is set to 0xffff when it is cleared. When this bit is cleared, it is the CHKSM TCP/IP Checksum Data Register is set to zero when it is cleared. This option should be used if the TCP/IP checksum should never be set to zero (0xffff is zero also). |
| 11 | EX-AL -- Expose Alignment | When this bit is set, the internal checksum alignment is exposed for reading/writing. For writes, bit 16 of the write data is used to set the internal alignment. For reads, the alignment is exposed in bit 16 or bit zero depending on the value of the HI-LO bit in this register. This can be useful if doing non-consecutive multiple part check sums (need to preserve alignment between chunks). When this bit is cleared, the internal checksum alignment is not exposed. It is always cleared when the CL-IP bit in this register is set. Normally the internal alignment is calculated and maintained across consecutive check sums. |
| 10 | HI-LO -- Hi Lo Word | When this bit is set, the checksum data register data is placed in the most sig 16 bits of the 32 bit value read. When this bit is cleared, the checksum data register data is placed in the least significant 16 bits of the 32 bit value read. This bit does not affect how writes to the checksum data register occur, the data from the least significant 16 bits is always used. |
| 9 | SW-SUM -- Swap CheckSum | When this bit is set, the checksum data register data is byte-swapped when read. When this bit is cleared, the checksum data register data is read normally. |
| 8 | IN-SUM -- Invert CheckSum | When this bit is set, the checksum data register data is inverted when read. When this bit is cleared, the checksum data register data is read normally.<br>There are also new checksum data register addresses that can be read that do the same thing as this control bit. This bit is depreciated. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | RP-ADD -- Ripple Addend | When this bit is set, the ripple base register counts up by 1. When this bit is cleared, the ripple base register counts up by eight. |
| 6 | CL-IP -- Clear IP | When this bit is written it will clear the CHKSM TCP/IP Checksum Data Register and itself. The result of this will be that this bit will never be read as a '1'. The internal alignment is also cleared. |
| 5 | RP -- Ripple | When this bit is set, a ripple pattern will be used in both the read and write test modes. The ripple pattern is used instead of the constant test pattern. When this bit is reset, the constant test pattern is used for the test mode data. |
| 4 | MS -- Memory Select | When this bit is set, all CHKSM memory accesses are to the control memory. When this bit is cleared, all CHKSM memory accesses are to the packet memory. |
| 3 | RW -- R/-W Test Mode | When this bit is set, the entity will take the data that is read, and compare it to the test/ripple pattern. <br> When this bit is reset, the checksum entity will write data using the test/ripple pattern to the DRAM. |
| 2 | TM -- Test Mode | When this bit is set, the entity will take the data that is read, and compare it to the test/ripple pattern, or will write data using the test/ripple pattern to the DRAM depending on the setting of the RW bit. In both cases, the reading or writing will continue until either an error is encountered, or the CHKSM Read/Write Count Register counts down to zero. When this bit is reset, the checksum entity will operate as described by the other bits. Test and CHKSM modes are mutually exclusive, and test mode takes precedence. |
| 1 | ET -- Enable TCP Checksum Updates | When this bit is set, the entity will collect the TCP checksum in the CHKSM TCP/IP Checksum Data Register. <br> When this bit is reset, the CHKSM TCP/IP Checksum Data Register will not be changed by data that is read from the DRAM. <br> Test and CHKSM modes are mutually exclusive, and test mode takes precedence. |
| 0 | EE -- Enable Entity CHKSM | When this bit is set, the entity will run as specified. <br> When this bit is reset, the entity will not run. |

### 17.9: CHKSM Internal State

Internal state of checksum. **Note: This register should not be written unless you know what you are doing!**

**Length**          3 bits

**Type**            Read/Write

**Address**         XXXX 0A3c

**Power On Value**  X'00000000'

**Restrictions**    None

**Software Use of CHKSM**

This section outlines some ways CHKSM can be set up and used.

**Test Mode Possible Patterns**

In test mode, a 64-bit pattern is written/compared to/with memory. There are several different patterns that can be used:

**Constant Test Pattern** When in test mode, and the RP bit is cleared, the CHKSM Ripple Base Register is replicated eight times to form a 64-bit pattern.

**Ripple Pattern w/ Increment of 1** When in test mode and the RP bit is set and RP-ADD is set and CHKSM Ripple Limit Register is set to zero, a 64-bit pattern is generated using the CHKSM Ripple Base Register as a base. For example, if the CHKSM Ripple Base Register is set to one, the following pattern is generated:

```
0102030405060708
0203040506070809
030405060708090a
0405060708090a0b
```
…

**Ripple Pattern w/ Increment of 8** When in test mode and the RP bit is set and RP-ADD is cleared and CHKSM Ripple Limit Register is set to zero, a 64-bit pattern is generated using the CHKSM Ripple Base Register as a base. For example, if the CHKSM Ripple Base Register is set to one, the following pattern is generated:

```
0102030405060708
090a0b0c0d0e0f10
1112131415161718
191a1b1c1d1e1f20
```
…

**Ripple Pattern with Ripple Limit** Each of the above ripple patterns can also make use of the CHKSM Ripple Limit Register. By setting this register, the user can control when the ripple pattern rolls over to zero. For example, when the CHKSM Ripple Limit Register is set to three in increment by one mode the ripple pattern looks like:

```
0102030405060708
0203040506070809
030405060708090a
0001020304050607
0102030405060708
0203040506070809
030405060708090a
```
…

Similarly, when the CHKSM Ripple Limit Register is set to ten, in increment-by-8 mode, the ripple pattern looks like:

```
0102030405060708
090a0b0c0d0e0f10
1112131415161718
0001020304050607
```
…

**Initializing Packet/Control Memory**

The following list shows the steps to use CHKSM to initialize packet or control memory:
- make sure CHKSM is in diagnostic mode, and other mode bits are reset
- set the start address by writing the base addr
- set up the read/write count with number of bytes to initialize
- set up the test pattern register (ripple pattern register) with pattern to use
- set up the Control Register to enable test mode, enable checksum entity, and set the memory select bit correctly based which memory is to be initialized
- now busy wait until operation is done (or set up interrupt Enable register and wait for interrupt)

**Testing Packet/Control Memory**

The following list shows the steps to use CHKSM to test packet or control memory:
- first initialize memory with a pattern using above sequence
- make sure CHKSM is in diagnostic mode, and other mode bits are reset
- set the start address by writing the base addr
- set up the read/write count with number of bytes to test (same as initialization value)
- the test pattern register (ripple pattern register) already contains the pattern
- set up the Control Register to enable test mode, turn on RW bit, enable checksum entity, and set the memory select bit correctly based which memory is to be initialized
- now busy wait until operation is done (or set up interrupt Enable register and wait for interrupt)
- When done, check the status register for any errors

**Using Ripple Pattern Generation/Checking in Packet/Control Memory**

The procedures to use the ripple pattern generation and checking, are the same as using test write/read modes. The only difference is that the use ripple pattern mode bit must be set and the ripple pattern base register must be set up.

**Running a TCP/IP Checksum in Packet/Control Memory**

The following list shows the steps to use CHKSM to generate/verify a TCP/IP checksum:
- make sure CHKSM is in diagnostic mode (not enabled)
- set the start address by writing the base address
- set up the read/write count with number of bytes to run checksum over, and set the upper two bits of the read/write count register. Writing these upper two bits assumes other mode bits are set correctly (i.e. memory bank select).
- now busy wait until operation is done (or set up interrupt Enable register and wait for interrupt)

# Entity 18: Processor Core (PCORE)

PCORE contains the on-board processor and its local subsystems. The primary intent is to run Available Bit Rate (ABR) control software. The reason that this is done in a processor is that the standards for this function are in a state of flux at the time of this design. In addition a number of customers have expressed a desire to run additional code specific to their applications such as protocol termination code. Below is a diagram of the PCORE entity:

## PCORE Block Diagram

```
                                    ┌──────────┐
                                    │  Bridge  │◄──────────────────────────┐
                                    │          │        Control            │
                                    └────┬─────┘                           │
                         Control         │         sl    ┌──────────────┐  │
                                         │          ┌───►│     SRAM     │◄─┤
                                         │          │    └──────────────┘  │
  ┌────────────────────┐  Mst 0  ┌──────┴──┐  Data  │    ┌──────────────┐  │
  │ TMM                │◄───────►│         │◄───────┘    │    Packet     │◄─┤
  │   ┌────────────┐   │  Mst 1  │  PLB    │       sl    │  Memory Int   │  │
  │   │  401 Core  │   │◄───────►│         │    ┌───────►└──────────────┘  │
  │   └────────────┘   │         │         │    │         ┌──────────────┐ │
  │                    │         └─────────┘    │    sl   │   Control     │◄┤
  └──┬───────┬──────┬──┘                        │ ┌──────►│  Memory Int   │ │
     │       │      │                       sl  │ │       └──────────────┘ │
     ▼       ▼      ▼                           │ │       ┌──────────────┐ │
  ┌──────┐ ┌─────────────┐                      │ └──────►│     PCI       │◄┤
  │ DCR  │ │  Interrupt  │◄───┐                 └────────►│  Master Int   │ │
  │Interf│ │ Controller  │    │                           └──────────────┘ │
  └──────┘ └─────────────┘    │                           ┌──────────────┐ │
     ▲        ▲               │                           │     PCI       │◄┘
     │        │               └──────────────────────────┤   Slave Int   │
     │     ┌──────────────┐                               └──────────────┘
     └────►│   Clock &    │◄──────────────────────────────┘
           │  Power Mgmt  │
           └──────────────┘
```
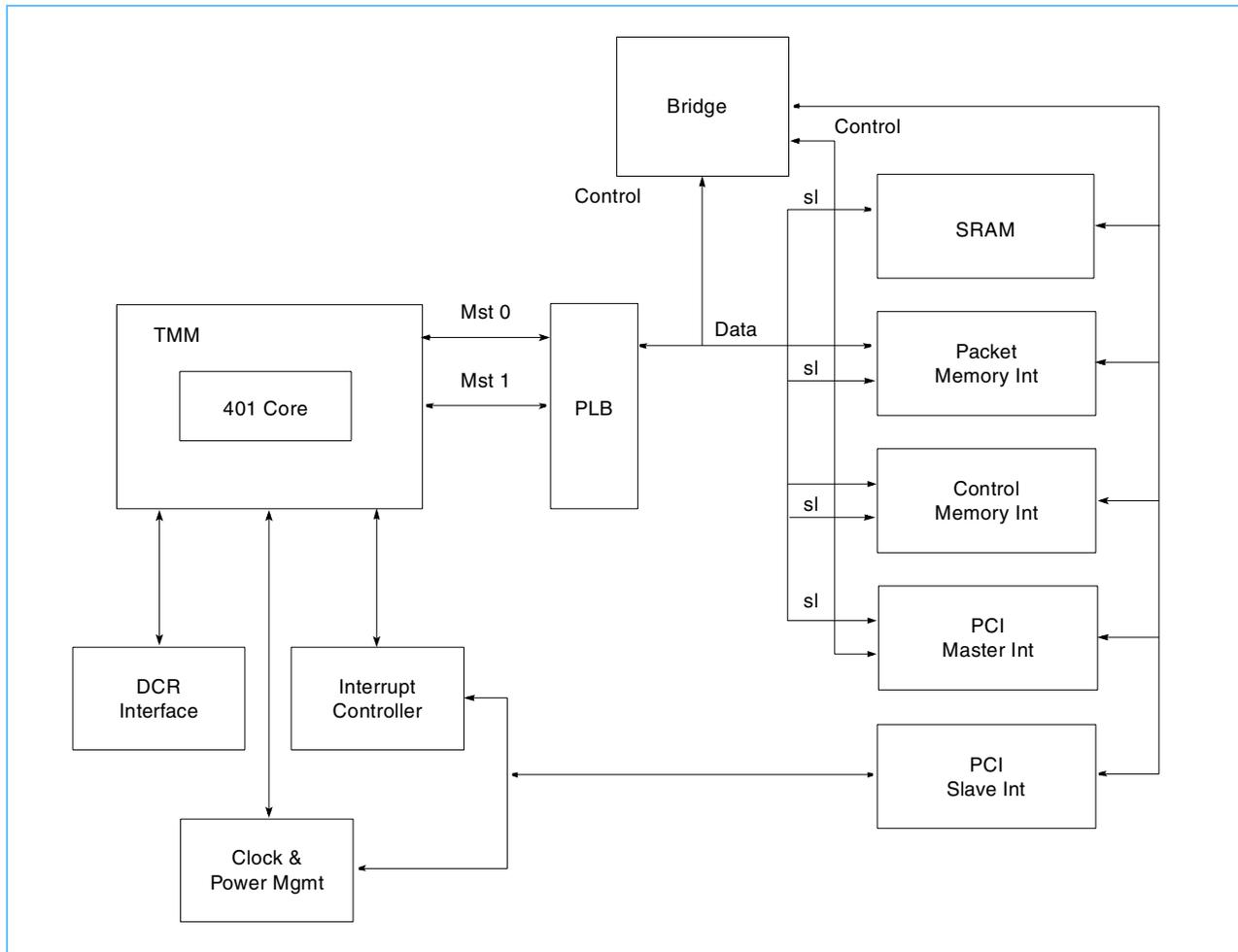
### DCR Interface

The Device Control Register(DCR) interface is a special processor bus to access local registers. These include registers in the Processor Local Bus(PLB) logic, Serial Port and various IBM2520L8767 registers.

## Interrupt Controller

This logic manages the interrupts that are passed on to the 401 core. There are two levels of interrupt for the core, Critical Interrupts and Normal Interrupts. Interrupts can be taken from both on-chip and off-chip sources. PCORE has a variety of interrupt source and enable registers.

## Clock & Power Management

This logic controls the various sleep and wakeup options for the 401 core.

## Processor Local Bus(PLB)

The PLB is used as an interface between the 401 core and its variety of slave devices. The 401 core Instruction bus and the 401 core Data bus are each connected as masters to this bus. The Instruction bus is connected as Master 0. The Data bus is connected as Master 1.

## Bridge

The bridge translates processor space addresses to slave subsystem addresses. When a PLB read or write transaction is issued from the PLB, the bridge function translates the address from the processor address to the slave subsystem address and starts a slave system access.

## SRAM

There is an on-chip SRAM for the use of the processor. This SRAM is typically used only by the processor, therefore, it has a generally predictable access time. This SRAM would typically be mapped into the processor's address space. There are a number of different ways that this can be done.

## Control Memory

Control Memory can be accessed by the processor. This memory may be mapped into the processor space in a number of different ways.

## Packet Memory

Packet Memory can be accessed by the processor. This memory may be mapped into the processor space in a number of different ways. Packet memory space also includes the virtual memory space of the IBM2520L8767.

## PCI Master Interface-External

The processor can access the PCI bus through this interface. Parts of PCI space are mapped into processor space. There are a number of different ways that this can be mapped into processor space.

## IBM2520L8767 Register Space

This access mode of the PCI Master Interface allows access to the internal IBM2520L8767 registers. This access is handled internally and does not affect the external PCI bus.

### PCI Slave Interface

The PCORE registers are accessible via this interface.

### Address Translation Examples

### 18.1: PCORE Control Register

The PCORE Control Register provides control information about PCORE operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C00 and C04 |
| **Power on Reset value** | X'00 00 00 87' |
| **Restrictions** | Caution must be used when asserting some of the bits during operation. |



| Bit(s) | Name | Description |
|---|---|---|
| 31-11 | Reserved | Reserved |
| 12-11 | State Mux Select | This selects what data is sent to the state multiplexor.<br>'00' ICU Conections<br>'01' DCU Connections<br>'10' DCR Connections/Pcore State<br>'11' PLB Slave Side Connections |
| 10 | Disable PLB 16 cycle timer | When this bit is set the PLB 16-cycle timer will be disabled. |
| 9 | Memory Lock Normal/Critical Interrupt | When this bit is written to zero, PCORE will treat memory locked as a critical interrupt. When it is one this condition will be treated as a normal interrupt. |
| 8 | Little Endian Mode | When this bit is written to zero, PCORE Master PCI access will operate in big endian mode. When one, will operate in little endian mode. When in little endian mode, both the source and destination must be aligned on four-byte boundaries. |
| 7 | Slave Access Dead Man Timer Enabled | When set, this bit enables the slave-access, dead-man timer to operate. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 6 | Serial Port Receive Interrupt Priority | When set, this bit causes the receive interrupt to be a Critical Interrupt. When not set, it is a regular interrupt. |
| 5 | Serial Port Transmit Interrupt Priority | When set, this bit causes the transmit interrupt to be a Critical Interrupt. When not set, it is a regular interrupt. |
| 4 | 401 Timer Sleep Mode Allowed | When set, this bit allows the core timer logic to be put to sleep by the core. |
| 3 | 401 Sleep Mode Allowed | When set, this bit allows the core to put itself into sleep mode. |
| 2 | Lock PCORE on Error | When this bit is set, an error occurs, and the corresponding lock enable bit is set, PCORE will lock. This state is equivalent to being in Diagnostic Mode. |
| 1 | 401 Core Reset | When set, this bit places the 401 Core in Reset State. The output of this register is used in addition to the master reset in CRSET that holds the Core in Reset during Power Up. This bit must be cleared in order to have the core leave Reset state. |
| 0 | Diagnostic Mode | When set, PCORE is in diagnostic mode. When cleared, PCORE is in normal mode. When in diagnostic mode, state machines are held in idle. If they are already active, when they next go to idle, they will hold there. |

**18.2: PCORE Status Register**

The PCORE Status Register provides status information about PCORE operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C08 and C0C |
| **Power on Reset value** | X'00 00 00 00' |
| **Restrictions** | During normal operations, if a status bit is cleared, it will be reset if the condition that is causing it is still present. |



| Bit(s) | Name | Description |
|---|---|---|
| 31-14 | Reserved | Reserved |
| 14 | Instruction Cache Slave Cycle Aborted. | The current cycle associated with the Instruction Cache has been aborted. |
| 13 | Data Cache Slave Cycle Aborted | The current cycle associated with the Data Cache has been aborted. |
| 12 | 401 MSR Wait Enabled | The 401 is in a wait state. |
| 11 | 401 Machine Check Occurred | The 401 has encountered a machine check error. |
| 10 | Slave Dead Man Timer Expired during a Write Cycle | The long transaction timer post address acknowledge has expired. |
| 9 | Slave Dead Man Timer Expired during a Read Cycle | The long transaction timer post address acknowledge has expired. |
| 8 | 401 Processor Stopped | This bit is set when the 401 is in stop mode. |
| 7 | 401 Processor Sleeping | This bit is set when the 401 is in sleep mode. |
| 6 | 401 Timers Sleeping | This bit is set when the timers of the 401 are in sleep mode. |

| Bit(s) | Name | Description |
|---|---|---|
| 5 | Bridge Translation Type Target Error | This occurs when the Bridge has an unsupported transaction target type issued. |
| 4 | PLB Transaction Size Error | This occurs when the PLB has an unsupported transaction size issued. |
| 3 | PLB Transaction Type Error | This occurs when the PLB has an unsupported transaction type issued. |
| 2 | PLB Transaction Timeout | This occurs when the address is not acknowledged in 16 cycles. |
| 1 | PLB Address Out of Range | This happens when an address on the PLB bus cannot be translated since it does not fall into a range covered by the address translation array. |
| 0 | PCORE Locked | This bit is set when Locking is enabled, an error has occurred and the lock mask bit is set that matches the error. |

### 18.3: PCORE User Status Register

The PCORE User Status Register provides user defined status information about PCORE software operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

**Length**               32 bits

**Type**                 Clear/Set

**Address**              XXXX 3C10 and C14

**DCR Address**          X'200 and 201'

**Power on Reset value**  X'00 00 00 00'

**Restrictions**         During normal operations, if a status bit is cleared, it will be reset if the condition that is causing it is still present.

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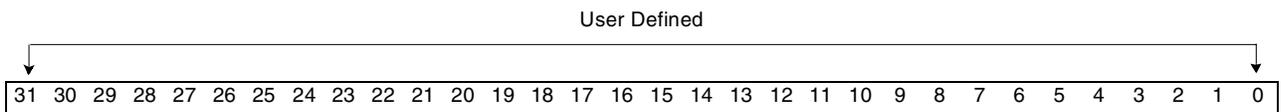
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 31-0 | User Defined | Reserved |

**18.4: PCORE 401 External Status Register**

The PCORE 401 External Status Register provides user defined status information about PCORE software operations. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C6C and C70 |
| **DCR Address** | X'202 and 203' |
| **Power on Reset value** | X'00 00 00 00' |
| **Restrictions** | During normal operations, if a status bit is cleared, it will be reset if the condition that is causing it is still present. |



| Bit(s) | Name | Description |
|---|---|---|
| 31-10 | User Defined | Reserved |
| 9 | Memory Controller Locked Critical Interrupt | This occurs when the memory controller is locked and this condition is set as critical. |
| 8 | Memory Controller Locked Non-Critical Interrupt | This occurs when the memory controller is locked and this condition is set as non-critical. |
| 7 | Serial Port Receive Critical Interrupt | This occurs when the serial controller has a Transmit Interrupt and the corresponding critical-interrupt enable is on in the control register. |
| 6 | Serial Port Receive Non-Critical Interrupt | This occurs when the serial controller has a Transmit Interrupt and the corresponding critical-interrupt enable is on in the control register. |
| 5 | Serial Port Transmit Critical Interrupt | This occurs when the serial controller has a Transmit Interrupt and the corresponding critical-interrupt enable is on in the control register. |
| 4 | Serial Port Transmit Non-Critical Interrupt | This occurs when the serial controller has a Transmit Interrupt and the corresponding critical-interrupt enable is on in the control register. |
| 3 | Critical Interrupt | This occurs when a bit in the IBM2520L8767 primary status register is set and the corresponding critical interrupt enable is on. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 2 | Non-Critical Interrupt | This occurs when a bit in the IBM2520L8767 primary status register is set and the corresponding non-critical interrupt enable is on. |
| 1 | External Critical Interrupt | This occurs when an Off Chip Interrupt is received and the non-critical enable for Off Chip Interrupts is set. |
| 0 | External Non-Critical Interrupt | This occurs when an Off Chip Interrupt is received and the non-critical enable for Off Chip Interrupts is set. |

### 18.5: PCORE IBM2520L8767 Shadow Status Register

This register is used to shadow the INTST Interrupt Source. The purpose of this register is to allow polling for IBM2520L8767 interrupts without having to use the PCI bus.

| | |
|--|--|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 3C7C |
| **DCR Address** | X'208' |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.6: PCORE IBM2520L8767 Shadow Rxque Status Register

This register is used to shadow the RXQUE Status and Enabled Status Registers. The purpose of this register is to allow polling for IBM2520L8767 interrupts without having to use the PCI bus.

| | |
|--|--|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 3C88 |
| **DCR Address** | X'20F' |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.7: PCORE Interrupt Enable Register

This register is used to enable bits from the PCORE Status Register and potentially generate interrupts to the control processor. When both a bit in this register and the corresponding bit(s) in the PCORE Status Register are set, the PCORE interrupt to PCINT will be enabled. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *PCORE Status Register on page 360* for the bit descriptions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C18 and C1C |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.8: PCORE User Interrupt Enable

This register is used to enable an interrupt based on bits from the corresponding PCORE User Status Register and potentially generate interrupts to the control processor. When both a bit in this register and the corresponding bit(s) in the status register are set, the PCORE status bit(s) will be set in the corresponding *PCORE User Status Register* (described on page 361). See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *PCORE User Status Register on page 361* for the bit descriptions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C20 and C24 |
| **DCR Address** | X'204 and 205' |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.9: PCORE 401 Interrupt Enable Register

This register is used to enable bits from the PCORE 401 External Status Register and generate interrupts to the 401 processor. When both a bit in this register and the corresponding bit(s) in the PCORE 401 External Status Register are set, the 401 interrupt to the 401 core will be enabled. See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing. See *PCORE Status Register on page 360* for the bit descriptions.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C74 and C78 |
| **DCR Address** | X'206 and 207' |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.10: PCORE Error Lock Enable Register

The PCORE Error Lock Enable Register provides the ability to halt PCORE when the corresponding status bit in the status register are set and locking is enabled. When a bit in this register corresponds to a bit that is set in the status register, the state machines in PCORE will be held in idle state until the lock is disabled.

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C28 and C2C |
| **Power on Reset value** | X'00 00 7F FF' |
| **Restrictions** | None |

### 18.11: PCORE User Error Lock Enable Register

The PCORE User Error Lock Enable Register provides the ability to halt PCORE when the corresponding status bit in the user status register are set and locking is enabled. When a bit in this register corresponds to a bit that is set in the status register, the state machines in PCORE will be held in idle state until the lock is disabled.

See *Note on Set/Clear/Read Type Registers* on page 71 for more details on addressing.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Clear/Set |
| **Address** | XXXX 3C30 and C34 |
| **Power on Reset value** | X'FF FF FF FF' |
| **Restrictions** | None |

### 18.12: PCORE Transaction Dead Man Timer Value Register

This register is used to load a timer that counts to zero from the value loaded in this register. The maximum wait for an I/O transaction is about 2ms when this is set to X'FFFF'. The value of this register is written into the PCORE Transaction Dead Man Timer Value Register after the slave address is validated during a processor access to a slave device.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3C80 |
| **Power on Reset value** | X'FF FF' |
| **Restrictions** | None |

**18.13: PCORE Address Translation Base Address Array**

The PCORE Address Translation Base Address Array provides the base address of the region to which the 128MB 401 address range corresponds. When an address is issued from the 401 core it will fall within 1 of 32, 128-MB address ranges. The ranges may have special attributes associated with them in registers in the 401 core (cachable/noncachable etc.). The first address corresponds to the first 128-MB region of core memory. The second address corresponds to the second 128-MB region and so on. The address contained in this array is the base address of the target memory space.

| | |
|---|---|
| **Length** | 32x32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3E00 to E7C |
| **Power on Reset value** | X'00 00 00 00' |
| **Restrictions** | None |

**18.14: PCORE Address Transaction Type and Range Array**

The PCORE Address Transaction Type and Range Array provides the transaction type (SRAM, Packet, Control...) information and valid range information. When an address is issued from the 401 core it will fall within one of thirty-two 128MB address ranges. The ranges may have special attributes associated with them in registers in the 401 core (cachable/noncachable etc.). The first address corresponds to the first 128MB region of core memory. The second address corresponds to the second 128MB region and so on.

| | |
|---|---|
| **Length** | 32x32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3E80 to EFF |
| **Power on Reset value** | X'00 00 00 00' |
| **Restrictions** | None |

| Bit(s) | Description |
|---|---|
| | The transaction type information is as follows: |
| | '00000'      On Chip SRAM |
| | '00001'      Packet Memory |
| | '00010'      Control Memory |
| | '00011'      Control/Packet |
| | '11000'      On Chip SRAM/Control Memory |
| | '11001'      On Chip SRAM/Packet Memory |
| | '00100'      IBM2520L8767 Registers |
| | '00101'      PCI Memory Access (Non-IBM2520L8767) |
| | '01100'      PCI I/O Access |
| | '10100'      PCI Config Access |

Single range memory targets have address range checking as described below. However, dual target memory locations behave differently. The 27-bit range field is used to mark the processor address range crossover point. In the case of shared SRAM, the SRAM base address is assumed to be zero. The DRAM target address is the new base address plus the offset from the crossover point. In the case of shared control and packet memory access, the 27-bit range is the crossover point and the offset from the PLB address is added to the new base address. However, when the crossover point is reached, the memory target is changed from control to packet memory.

The 27-bit range field is used to check for invalid addresses. If the Address Translation Base Address plus the Range are less than the translated address, an address out-of-range error is specified.

**18.15: PCORE Last PLB Address Register**

The PCORE Last PLB Address Register is the address from the PLB bus at the time of the Hang Condition. When a PLB long timeout occurs, this register will hold the address of the failed access.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 3C38 |
| **DCR Address** | X'209' |
| **Power on Reset value** | X'FF FF FF FF' |
| **Restrictions** | None |

### 18.16: PCORE Last PLB Error Register

The PCORE Last PLB Error Register is the information associated with the last error on the PLB bus. This register contains an error syndrome for a slave bus timeout.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | XXXX 3C84 |
| **DCR Address** | X'20D' |
| **Power on Reset value** | X'00 01 FF FF' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-18 | Reserved | Reserved |
| 17 | Read/Not Write | When set, a read cycle; when not a write cycle. |
| 16 | One/Four Word | This bit is set when one word is transferred. When not set, four words are transferred. |
| 15 | SRAM Memory Access | This bit is set when SRAM is accessed. |
| 14 | Packet Memory Access | This bit is set when Packet memory is accessed. |
| 13 | Control Memory Access | This bit is set when Control memory is accessed. |
| 12 | Combination Control/Packet | This bit is set when the cycle is to the one combined IBM2520L8767 Memory Access region. |
| 11 | PCI Master Access Off Chip | This bit is set when an offchip PCI Master access is underway. |
| 10 | Hang State Bit | Indicates the Hang State. |
| 9 | IBM2520L8767 Register Access | This bit is set when the PLB access was to the IBM2520L8767 Registers. |
| 8-5 | Byte Enables | These are the byte enables for the access. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 4-0 | Access Type | These bits indicate the transaction type<br>1  '00000' On Chip SRAM<br>2  '00001' Packet Memory<br>3  '00010' Control Memory<br>4  '00011' Control/Packet<br>5  '11000' On Chip SRAM/Control Memory<br>6  '11001' On Chip SRAM/Packet Memory<br>7  '00100' IBM2520L8767 Registers<br>8  '00101' PCI Memory Access (Non IBM2520L8767)<br>9  '01100' PCI I/O Access<br>10  '10100' PCI Configuration Access |
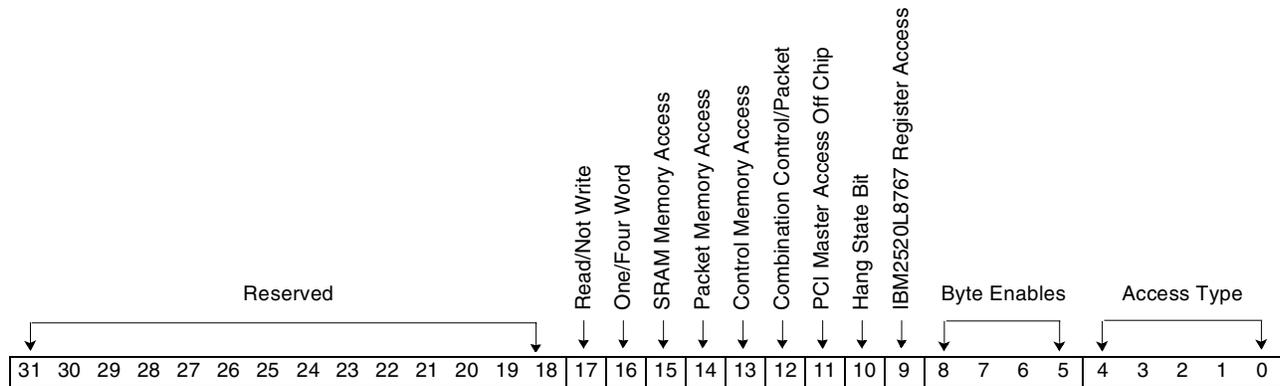
## 18.17: PCORE SRAM

This SRAM is used by the 401 processor to hold on chip code and data. This SRAM is accessed via a window starting at the address starting at the SRAMBase register. The window allows access to 64 locations in the SRAM at any given time.

| | |
|---|---|
| **Length** | 8Kx32 |
| **Type** | Read/Write(with window) |
| **Address** | XXXX 3F00 to FFF |
| **Power On Value** | X'UU UU UU UU' |
| **Restrictions** | None |

## 18.18: PCORE SRAM Base Address

The SRAM Base Address register is used to select the base address of the window to access the SRAM.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3C5C |
| **Description** | The SRAM Base Address used to point into a 64-entry region of the SRAM. |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.19: PCORE Read Data Transfer Registers

The PCORE Read Data Transfer Registers holds the data that is being transferred between the core and one of the DRAM slave subsystems. On single word transfers Data Transfer Register 0 will hold the data. On cache line transfers all four registers are used. These registers store the data that is being transferred between Packet or Control Memory and the core.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | Read Data Transfer Register 0    XXXX 3C3C |
| | Read Data Transfer Register 1    XXXX 3C40 |
| | Read Data Transfer Register 2    XXXX 3C44 |
| | Read Data Transfer Register 3    XXXX 3C48 |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

### 18.20: PCORE Write Data Transfer Registers

The PCORE Write Data Transfer Registers holds the data that is being transferred between the core and Control memory. On single word transfers Data Transfer Register 0 will hold the data. On cache line transfers all four registers are used. These registers store the data that is being transferred between Control Memory and the core.

**Note:** CC0-CFC is used by the serial port.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read |
| **Address** | Write Data Transfer Register 0    XXXX 3C4C |
| | Write Data Transfer Register 1    XXXX 3C50 |
| | Write Data Transfer Register 2    XXXX 3C54 |
| | Write Data Transfer Register 3    XXXX 3C58 |
| **Power on Value** | X'00000000' |
| **Restrictions** | None |

### 18.21: PCORE IBM2520L8767 Polling Register

The PCORE IBM2520L8767 Polling Register provides status information to PCORE about IBM2520L8767 operations. This allows PCORE to poll specific IBM2520L8767 status without using PCI bus bandwidth.

| | |
|---|---|
| **Length** | 5 bits |
| **Type** | Read |
| **Address** | XXXX 3C60 |
| **DCR Address** | X'20E' |
| **Power on Reset value** | X'00 00 00 00' |
| **Restrictions** | During normal operations, if a status bit is cleared, it will be reset if the condition that is causing it is still present. |



| Bit(s) | Name | Description |
|---|---|---|
| 31-5 | Reserved | Reserved |
| 4 | Memory Locked | Memory is locked. |
| 3 | Control Memory Locked | Control Memory is locked. |
| 2 | Packet Memory Locked | Packet Memory is locked. |
| 1 | Virtual Lock | VIMEM is the locker of memory. |
| 0 | Arbit Lock | Arbit is the locker of memory. |

### 18.22: PCORE Integer Input Rate Conversion Register

This register is the integer input port for the rate conversion logic. It is used as input to the rate conversion logic. An integer rate is placed in this register. The on-board logic converts it to an ABR rate format.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 3C64 |
| **DCR Address** | X'20B' |
| **Power On Value** | X'00 00 00 00' |
| **Restrictions** | None |

### 18.23: PCORE ABR Output Rate Register

This register is the ABR Output port for the rate conversion logic.An integer rate was placed in the Integer Input register. The logic converts it to an ABR rate and places the result in this register.

| | |
|---|---|
| **Length** | 16 bits |
| **Type** | Read |
| **Address** | XXXX 3C68 |
| **DCR Address** | X'20C' |
| **Power On Value** | X'00 00' |
| **Restrictions** | None |

### 18.24: PLB PACR Register

This is the PLB arbitration Control Register. It contains information regarding the state of the transmit and receive interfaces.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | DCR Only |
| **DCR Address** | X'0F7' |
| **Power On Value** | X'00000000' |
| **Restrictions** | Accessible Only through the DCR interface. |



| Bit(s) | Name | Description |
|---|---|---|
| 31 | PPMOD - PLB Priority Mode | 0 - Fixed<br>1 - Fair |
| 30-29 | PRIOO - Priority Order | Note this looks strange because there are only two masters in this implementation.<br>00 - Instruction - Data<br>01 - Data - Instruction<br>10 - Instruction - Data<br>11 - Instruction - Data |
| 28-0 | Reserved | Reserved |
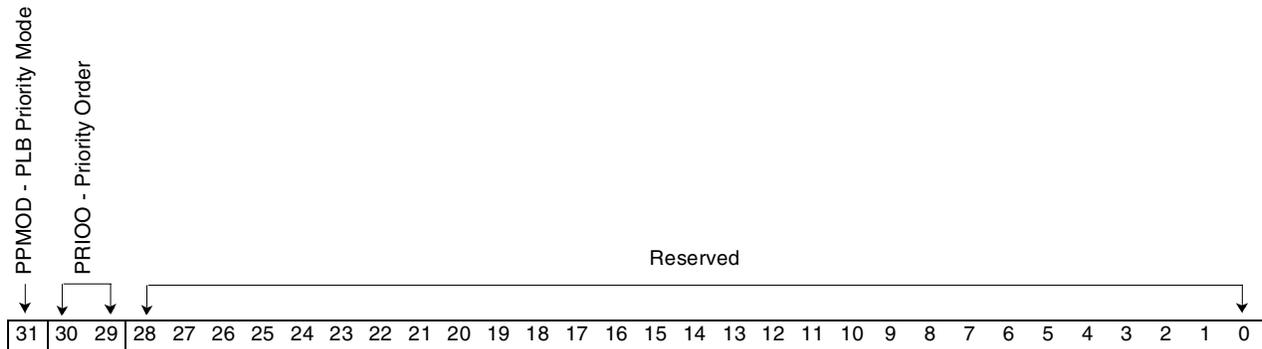
## Entity 19: RS-232 Interface Logic (RS-232)

The RS232 entity provides a means by which an external debugger and the processor core can communicate. The base RS-232 core operates on a byte transmit and receive basis. Using the RS-232 Mode Register, however, the entity can be configured to operate on a four byte wide basis.

### 19.1: RS-232 Line Status Register

This register contains information regarding the state of the transmit and receive interfaces. Bits in this register are cleared by writing '1' to them. Writing '0' has no effect. When in single byte transmit or receive mode, reading the receive buffer will clear bit seven and writing the transmit buffer will clear bit two.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CC0 |
| **DCR Address** | X'210' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-8 | Reserved | Reserved |
| 7 | Receive Buffer Ready | This bit indicates that a byte has been received on the interface. |
| 6 | Framing Error | This bit indicates that a stop bit was not detected when expected. |
| 5 | Overrun Error | This bit indicates that an overrun condition occurred. Owing to internal buffering, this bit will be set when the SPU core begins receiving a third byte when two unprocessed bytes have been received. |
| 4 | Parity Error | This bit indicates a parity error was detected on a received byte. |
| 3 | Line Break | This bit indicates a line break was received. |
| 2 | Transmit Buffer Ready | This bit indicates the transmit buffer is ready to accept data. |
| 1 | Transmit Shift Register Ready | This bit indicates that the transmit logic serializer is ready to accept data. |
| 0 | Reserved | Reserved. |

### 19.2: RS-232 Handshake Status Register

This register contains information regarding the state of the handshaking signals. If either bit in this register is set, it must be cleared before the port will transmit. Bits in this register are cleared by writing '1' to them. Writing '0' has no effect. When in single byte transmit or receive mode, read the receive buffer will clear bit seven and writing the transmit buffer will clear bit two.

| | |
|---|---|
| **Length** | 2 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CC8 |
| **DCR Address** | X'212' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

DSR Inactive
CTS Inactive

| 7 | 6 |
|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 31-8 | Reserved | Reserved |
| 7 | DSR Inactive | This bit indicates that the data set ready signal is inactive. |
| 6 | CTS Inactive | This bit indicates that the clear to send signal is inactive. |
| 5-0 | Reserved | Reserved |

### 19.3: RS-232 Baud Rate Divisor High Register

This register contains the upper portion of the value used to determine the baud rate. The value to place in this register can be determined by this formula: BaudRate = 33MHz/(16*((Baud Rate High) + 1)).

| | |
|---|---|
| **Length** | 4 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CD0 |
| **DCR Address** | X'214' |
| **Power On Value:** | X'00000000' |
| **Restrictions** | None |

Baud Rate
Divisor

```
3  2  1  0
```
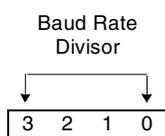
| Bit(s) | Name | Description |
|---|---|---|
| 31-4 | Reserved | Reserved |
| 3-0 | | Upper byte of Baud Rate Divisor. |

### 19.4: RS-232 Baud Rate Divisor Low Register

This register contains the lower portion of the value used to determine the baud rate. The value to place in this register can be determined by this formula:
BaudRate = 33MHz/(16*((Baud Rate Low) + 1)).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CD4 |
| **DCR Address** | X'215' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

Baud Rate Divisor

```
7  6  5  4  3  2  1  0
```
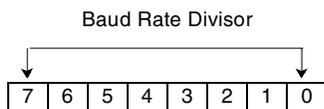
| Bit(s) | Name | Description |
|---|---|---|
| 31-8 | Reserved | Reserved |
| 7-0 | | Upper byte of Baud Rate Divisor. |

**19.5: RS-232 Serial Port Control Register**

This register controls how the port operates.

**Length**             8 bits

**Type**               Read/Write

**Address**            XXXX 4CD8

**DCR Address**        X'216'

**Power On Value**     X'00000000'

**Restrictions**       None



| Bit(s) | Name | Description |
|--------|------|-------------|
| 31-8 | Reserved | Reserved |
| 7-6 | Loopback Mode | These bits control how the incoming and outgoing data streams are processed. The bits are encoded as follows:<br>'00' Normal transmit and receive operation.<br>'01' Internal loopback - transmit stream is connected to the receive stream.<br>'10' Automatic Echo - received stream is connected to the transmit stream.<br>'11' Reserved. |
| 5 | Data Terminal Ready | If DTR is under software control, this bit determines the state of DTR. If DTR is under hardware control, setting this bit to '0' forces DTR inactive. |
| 4 | Request to Send | If RTS is under software control, this bit determines the state of RTS. If RTS is under hardware control, setting this bit to '0' forces RTS inactive. |
| 3 | Data Size | This bit set to '1' selects eight data bits. This bit set to '0' selects seven data bits. |
| 2 | Parity Enable | set to '1', this bit selects odd parity. Set to '0', this bit selects even parity. |
| 1 | Parity Type | set to '1', this bit selects odd parity. Set to '0', this bit selects even parity. |
| 0 | Stop bits | set to '1', this bit selects 2 stop bits. Set to '0', this bit selects 1 stop bit. |

IBM

**19.6: RS-232 Receive Command Register**

This register controls how the receive logic operates.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CDC |
| **DCR Address** | X'217' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-8 | Reserved | Reserved |
| 7 | Enable Receiver | Set to '1', this bit enables the receiver logic. |
| 6-5 | DMA/Interrupt Mode | These bits control how the receive logic transfers received bytes. They are encoded as follows:<br>'00'    No DMA or interrupts (polled operation).<br>'01'    DMA disabled, Receive Interrupt Driven on byte reception.<br>'10'    DMA enabled, Receive Interrupt on byte reception disabled - mode to select when using 4-byte mode. The logic external to the RS-232 core will drive an interrupt when four bytes have been received.<br>'11'    Reserved. |
| 4 | Receive Error Interrupt Enable. | If set to '1' an interrupt will be driven when a receive error is detected. The value of bits 6-5 have no effect on this bit. |
| 3 | Pause Mode Enable. | Set to '0', this bit puts RTS under software control. If this bit is '1', RTS is under hardware control. If under hardware control, RTS may be dropped if:<br>The receive buffer internal to the RS-232 core is full and there are only six bits left to receive of another byte.<br>A receive error is active. |
| 2-0 | Reserved. | Reserved |

### 19.7: RS-232 Transmit Command Register

This register controls how the transmit logic operates.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CE0 |
| **DCR Address** | X'218' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |

| Bit(s) | Name | Description |
|---|---|---|
| 31-8 | Reserved | Reserved |
| 7 | Enable Transmitter | Set to '1', this bit enables the transmitter logic. |
| 6-5 | DMA/Interrupt Mode | These bits control how the transmit logic transfers bytes to be transmitted. They are encoded as follows:<br>'00' No DMA or interrupts (polled operation).<br>'01' DMA disabled, Transmit Interrupt Driven when RS-232 core can accept more data.<br>'10' Reserved<br>'11' DMA enabled, Transmit Interrupt on empty buffer disabled-mode to select when using four-byte mode. The logic external to the RS-232 core will drive an interrupt when the four bytes transmit buffer is empty. |
| 4 | Transmit Shift Register Empty Interrupt Enable. | If set to '1' an interrupt will be driven when the transmit shift register is empty. The value of bits 6-5 have no effect on this bit. |
| 3 | Transmit Shift Register Error Interrupt Enable. | If set to '1' an interrupt will be driven when a transmit shift register error is detected. The value of bits 6-5 has no effect on this bit. |
| 2 | Pause/Stop on CTS Inactive. | Set to '0', this bit causes the hardware to pause transmitting when CTS is inactive. Set to '0', this bit causes the hardware to stop transmitting and discard the byte it was sending. |
| 1 | Line Break | Set to '0', this bit causes a line break to be sent. |
| 0 | Reserved. | Reserved |

**19.8: RS-232 Byte Transmit/Receive Buffer**

This register is both the receive and transmit buffer. Writing this address places data in the transmit buffer and reading this address reads what is in the receive buffer. If using four-byte mode, code should use the RS-232 Four Byte Transmit/Receive Buffer.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CE4 |
| **DCR Address** | X'219' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |
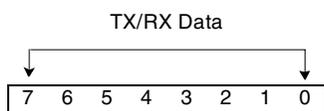
TX/RX Data

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 31-8 | Reserved | Reserved |
| 7-0 | | TX/RX Data. |

## 19.9: RS-232 Mode Register

This register is both the receive and transmit buffer when in four-byte mode. Writing this address places data in the transmit buffer and reading this address reads what is in the receive buffer.

| | |
|---|---|
| **Length** | 12 bits |
| **Type** | Read/Write |
| **Address** | XXXX 4CF0 |
| **DCR Address** | X'21C' |
| **Power On Value** | X'00000000' |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 31-12 | Reserved | Reserved. |
| 11-8 | Receive Valid bits | These bits indicate which parts of the four-byte receive buffer are valid. This is provided to aid software in determining when an error occurred within a message. A bit with a value of '1' represents a valid byte that has been received. Bit 11 corresponds to bits 31-24 of the four-byte receive buffer, bit 10 to bits 23-16, etc. |
| 7-4 | 'Transmit Valid bits | These bits indicate which parts of the four-byte transmit buffer have been sent. This is provided to aid software in determining when an error occurred within a message. A bit with a value of '0' means the corresponding byte has been transmitted. Bit seven corresponds to bits 31-24 of the four-byte transmit buffer, bit six to bits 23-16, etc. |
| 3 | Force CTS Active | Set to '1', this bit forces Clear to Send to the RS-232 core active. |
| 2 | Force DSR Active | Set to '1', this bit forces Data Set Ready to the RS-232 core active. |
| 1 | Reset | Set to '1', bit one will reset the logic that provides the four-byte interface to the RS-232 core logic. This bit does NOT reset the RS-232 core. This bit need only be on for one cycle for the reset to occur. |
| 0 | Enable Four Byte Mode | Set to '1', this bit enables four-byte mode operation. Set to '0', this bit enables single-byte operation. |

**19.10: RS-232 Four Byte Transmit/Receive Buffer**

This register is both the receive and transmit buffer when in four-byte mode. Writing this address places data in the transmit buffer and reading this address reads what is in the receive buffer.

**Length**        32 bits

**Type**        Read/Write

**Address**        XXXX 4CF4

**DCR Address**        X'21D'

**Power On Value**        X'00000000'

**Restrictions**        None

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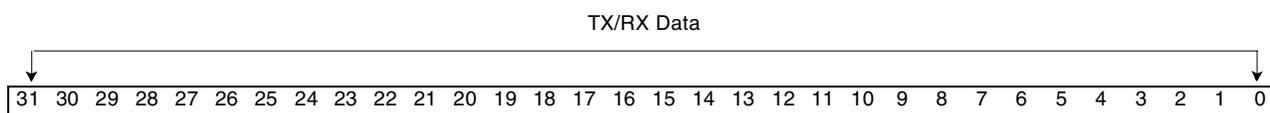
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|

| Bit(s) | Name | Description |
|---|---|---|
| 31-0 | | TX/RX Data. |

## Entity 20: Reset and Power-on Logic (CRSET)

This entity will perform BIST and flush operations. Chip software resets can be control by this entity, as well as the chip clock control.

### 20.1: Reset Status Register

This register is used to reflect what that last type of reset was. A hardware reset will clear software reset status bits, but a software reset will not have an effect on the hardware status bits.

| | |
|---|---|
| **Length** | 7 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0500 |
| **POR Value** | 'DB00001' or 'DB00010', where B is the state of the BIST results, and D is the PLL phase detection. |
| **Software Reset Value** | 'DB001QQ' or 'DB010QQ', where Q is the state of this bit before the software reset and B is the state of the BIST results. |
| **Restrictions** | None |



| Bit(s) | Name | Description |
|---|---|---|
| 7 | PCI clock frequency change | A value of '1' means that the real time PCI frequency calculator has detected a major change in frequency and has calculated new range bits for the PLL. |
| 6 | PLL out-of-phase detect | A value of '1' means that the out-of-phase detector circuit has triggered. This is just an indicator and is normal operation. |
| 5 | BIST results | A value of '1' means that a failure occurred within the BIST checking logic. |
| 4 | PCORE Reset | The PCORE entity has requested a chip reset via its hardware interface. |
| 3 | Software Reset | A Software Reset has occurred, the chip was flushed. |
| 2 | Software Reset/BIST | A Software Reset has occurred, and BIST/flush was run. |
| 1 | POR Reset | A POR Hardware Reset that flushed the chip has occurred. |
| 0 | | Reserved |
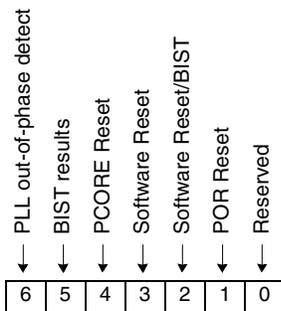
**IBM**

## 20.2: Software Reset Enable Register

This register protects the Software Reset Register If this register is not set, then a reset will not occur. Write a X'B4' to this register to enable software reset. A software reset will clear this register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0504 |
| **POR Value** | X'0' |
| **Restrictions** | None |

## 20.3: Software Reset Register

This register generates a scan path flush reset of the chip, or software initiated run of BIST, with the exception of the registers in the reset entity.

| | |
|---|---|
| **Length** | 6 bits |
| **Type** | Write Only |
| **Address** | XXXX 0508 |
| **POR Value** | '0' |
| **Restrictions** | Writing to this register without first setting the Software Reset Enable Register will have no effect. The register will not be set, so the order of writing the enable and the software reset is important; the enable must be written first. Additionally, all current operations being performed by the IBM2520L8767 must be terminated before doing a reset operation. A minimum number of enable bits to turn off would be bits four, five and six in INTST Control Register and bit two in PCINT Config Word 1. |

PCORE System Reset
PCORE Chip Reset
PCORE Reset
Total Software Reset
Run BIST
Software Reset
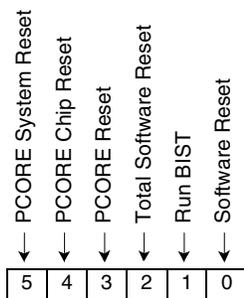
| 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 5 | PCORE System Reset | Writing this bit to a '1' delivers a system reset condition to the internal processor core. |
| 4 | PCORE Chip Reset | Writing this bit to a '1' delivers a chip reset condition to the internal processor core. |
| 3 | PCORE Reset | Writing this bit to a '1' causes the internal processor core to reset. |
| 2 | Total Software Reset | Writing this bit to a '1' causes software reset, and will be cleared after the software reset has occurred. |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 1 | Run BIST | Writing this bit to a '1' causes BIST to run, and will be cleared after the software reset has occurred. This function is primarily for pre-loading the BIST registers to get more test coverage. |
| 0 | Software Reset | Writing this bit to a '1' causes software reset, and will be cleared after the software reset has occurred. |

### 20.4: Memory Type Register

This register indicates the type of memory used for control and packet memory so that reset hardware will know how to properly preserve it during a reset.

**Length**          4 bits

**Type**            Read/Write

**Address**         XXXX 050C

**POR Value**       X'0'

**Restrictions**    None
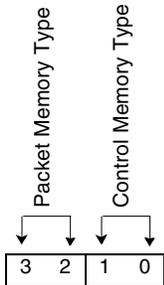


| Bit(s) | Name | Description |
|--------|------|-------------|
| 3-2 | Packet Memory Type | Decodes the same as bits 9-8 of *COMET/PAKIT Control Register* on page 143. |
| 1-0 | Control Memory Type | Decodes the same as bits 9-8 of *COMET/PAKIT Control Register* on page 143. |

## 20.5: CRSET PLL Range Debug

Used to debug the PPL operation.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read Only |
| **Address** | XXXX 0518 |
| **POR Value** | X'xxxxxxxx' |



| Bit(s) | Description |
|---|---|
| 31-4 | TBD |
| 3 | PLL Range bit 1 (A) |
| 2 | PLL Range bit 0 (A) |
| 1 | PLL Range bit 1 (B) |
| 0 | PLL Range bit 0 (B) |
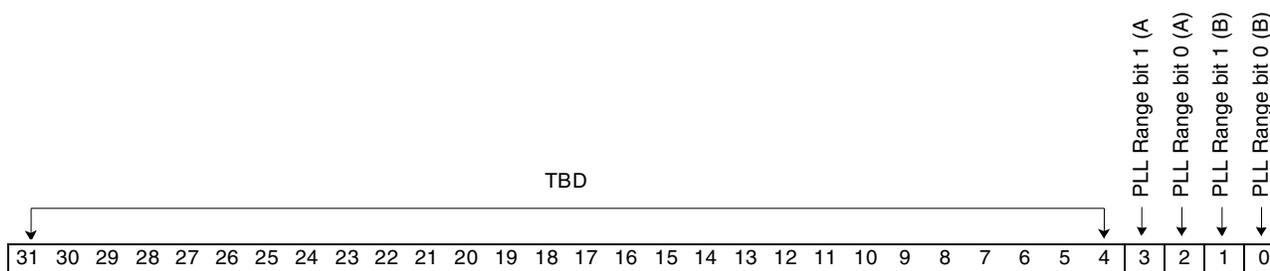
## 20.6: CRSET Control Register

Used to control PCI frequency detection logic.

| | |
|---|---|
| **Length** | 10 bits |
| **Type** | Read/Write |
| **Address** | XXXX 0510 |
| **POR Value** | X'30' |

Freq. Change

Reserved

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 9 | Disable the frequency change detection interrupt (Write Only) | Setting this bit to a one will disable using the frequency change detection bit as an interrupt source to INTST. |
| 8-0 | | Reserved |

## 20.7: Clock Control Register (Nibble Aligned)

Used to disable clocks for power conservation and provide the "Select A Clock" function for MPEG and front end support. To change a nibble field in this register, ALWAYS set it to zero first, and then to the new value.

**Length**            29 bits

**Type**              Read/Write

**Address**           XXXX 0520

**POR Value**         X'6676632'

| Bit(s) | Name | Description |
|--------|------|-------------|
| 28) | Framer Tree Disabled (FRAMR) | When set this bit will disable the clock tree to the Sonet Framer Logic. |
| 27-24 | Encoded Control for BIST Speed | Hardcoded to 6. |
| 23-20 | Reserved (for Encoded Control for PCORE Clock Rate.) | Hardcoded to 6. |
| 19-16 | Encoded Control for Various On Chip TimeStamp Logic (RXQUE,etc.) | Same as bits 3-0. |
| 15-12 | Encoded Control for Cell Opportunity Logic (CELLO) | Same as bits 3-0. |
| 11-8 | Encoded Control for MPEG Clocking Logic (MPEGT) | Same as bits 3-0. |
| 7-4 | Encoded Control for Transmit Logic (LINKT) and Sonet Framer (FRAMR) | Same as bits 3-0. |
| 3-0 | Encoded Control for Receive Logic (LINKR) and Sonet Framer (FRAMR) | Below is the encoded value of the bits that select a given clock. Always refer to *Select A Clock" Selection Matrix on page 389* for inputs supported for each clock out type.<br>X'0'  Turn this clock off.<br>X'1'  Use the external MPEG oscillator.<br>X'2'  Use the external RX clock.<br>X'3'  Use the external TX clock.<br>X'4'  Use the internal 15-ns clock. Assumes 33-MHz PCI clock.<br>X'5'  Use the internal 20-ns clock. Assumes 33-MHz PCI clock.<br>X'6'  Use the internal 30-ns clock. Assumes 33-MHz PCI clock.<br>X'7'  Use the internal 60-ns clock. Assumes 33-MHz PCI clock.<br>X'8'  Use the internal 120-ns clock. Assumes 33-MHz PCI clock.<br>X'9'  Use the internal 240-ns clock. Assumes 33-MHz PCI clock.<br>X'A'  Use the internal 480-ns clock. Assumes 33-MHz PCI clock.<br>X'B'  Use the differential Receiver clock divided by 8.<br>]X'C'  Use the differential Transmit clock divided by 8. |

## Select A Clock" Selection Matrix

| Clock Frequency Base | BIST BBSTNG CBSTNG1 | PCORE CPPUL | CELLO BCO CCO | MPEGT BMT CMT | LINKT(TX) BTX CTX RTX | LINKR(RX) BRX CRX RRX | Selection Encoding Register Bits |
|---|---|---|---|---|---|---|---|
| HTX Osc | | | | | xx | | '1100' |
| HRX Osc | | | | | | xx | '1011' |
| 480 | | | | | xx | xx | '1010' |
| 240 ns | | | | | xx | xx | '1001' |
| 120 ns | | | | | xx | xx | '1000' |
| 60 ns | | | | | xx | xx | '0111' |
| 30 ns | xx | xx | xx | xx | xx | xx | '0110' |
| 20 ns | | xx | | | | | '0101' |
| 15 ns | | xx | | | | | '0100' |
| TX Osc | xx | | xx | xx | xx | xx | '0011' |
| RX Osc | xx | | xx | xx | xx | xx | '0010' |
| MPEG OSC | xx | | xx | xx | xx | xx | '0001' |
| OFF | | | xx | xx | xx | xx | '0000' |
| Control bits | 27-24 | 23-20 | 15-12 | 11-8 | 7-4 | 3-0 | |

### 20.8: CBIST PRPG Results

This is the PRPG results register, updated after BIST has run. It is used by the BIST function for chip test.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 05B0 |
| **POR Value** | X'FFFFFFFF' |

### 20.9: CBIST MISR Results

This is the MISR results register, updated after BIST has run. It is used by the BIST function for chip test.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 05B4 |
| **POR Value** | X'00000000' |

## 20.10: CBIST Bist Rate

This register will hold a counter value that will separate the time between when the A clock and the B clock are launched during bist. This allows finer tuning to how much power BIST uses versus how much testing gets done within the time allowed. It is used by the BIST function for chip test.

**Length**          3 bits

**Type**            Read/Write

**Address**         XXXX 05B8

**POR Value**       X'0'


## 20.11: CBIST PRPG Expected Signature

This is the PRPG signature register, which should be written by Crisco code with the expected value of signature, based on the value in CBIST CYCT Load Value and the clock selected for BIST to run from. It is used by the BIST function for chip test.

**Length**          32 bits

**Type**            Read/Write

**Address**         XXXX 05C0

**POR Value**       X'FFFFFFFF'

### 20.12: CBIST MISR Expected Signature

This is the MISR signature register, which should be written by Crisco code with the expected value of signature, based on the value in CBIST CYCT Load Value and the clock selected for BIST to run from. It is used by the BIST function for chip test.

| | |
|---|---|
| **Length** | 32 bits |
| **Type** | Read/Write |
| **Address** | XXXX 05C4 |
| **POR Value** | X'00000000' |

### 20.13: CBIST CYCT Load Value

This register is the loaded value for the CBIST BIST Rate register. The time for BIST to run can be computed by the following equation: (shift count) $*$ (c30 clock$*$2) $*$ (cycle time). It is used by the BIST function for chip test.

| | |
|---|---|
| **Length** | 18 bits |
| **Type** | Read/Write |
| **Address** | XXXX 05C8 |
| **POR Value** | X'00005800' |

## Entity 21: JTAG Interface Logic (CJTAG)

The CJTAG entity contains logic to support a test access port (TAP) controller compliant with the IEEE 1149.1-1990 standard. The proper operation of these signals and the TAP controller is defined in the IEEE 1149.1-1993 standard. The TAP controller is accessed via the following five pins:

**TCK**        Test Clock. All activity of the JTAG interface is clocked via TCK. Events occur on the rising or falling edge of TCK. TCK should have a maximum frequency of 20MHz.

**TMS**        Test Mode Select. Test Mode Select is used to control state transitions in the TAP controller. These transitions occur on the rising edge of TCK. The BTR selected for TMS should be one with an internal pull-up.

**TDI**        Test Data In. Serial data input to the JTAG logic. The BTR selected for TDI should be one with an internal pull-up.

**TDO**        Test Data Out. Serial data output to the JTAG logic.

**TRST**        Test Reset. Asynchronous, minus active reset to the TAP controller. Assertion of this input causes the TAP controller to reset and the JTAG instruction register to load the IDCODE instruction. It is preferable to have TRST be independent of any chip reset. With an independent reset, the JTAG logic can be reset, allowing the chip's state to be examined without having to reset the core logic. The BTR selected for TRST should be one with an internal pull-up.

### Scanning

The TAP controller supports two types of scans: instruction scans and data scans. Instruction scans control the type of operation and select which (if any) scan chains are involved in the operation. Data scans generally clock the data on TDI into the selected scan chain.

## Instruction Format

The JTAG logic in the IBM2520L8767 supports 32-bit instructions in one of two formats. The first format uses OpCodes compliant with the IEEE standard; the other supports opcodes as defined by the Walnut chip that are compatible but not compliant with the IEEE standard. The general command format is as follows:
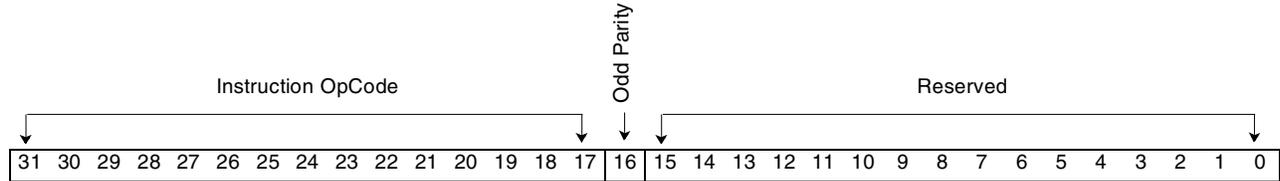


| Bit(s) | Description |
|--------|-------------|
| 31-17 | Instruction OpCode |
| 16 | In compatible mode, this is odd parity over bits 15-0 |
| 15-0 | Reserved |

As an instruction is scanned in, status for the previous instruction is presented on TDO. The 32 status bits have the following format:
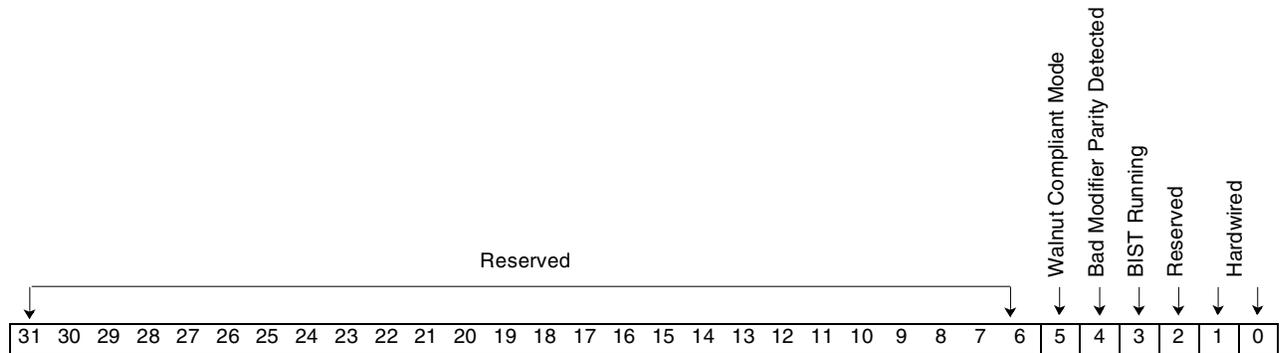


| Bit(s) | Description |
|--------|-------------|
| 31-6 | Reserved |
| 5 | Walnut Compliant Mode. |
| 4 | Bad Modifier Parity Detected. |
| 3 | BIST Running. |
| 2 | Reserved |
| 1-0 | Hardwired to '01' as required by IEEE specification. |

The following instructions are supported:

### 21.1: IDCODE

Returns a 32-bit identification code when a data scan is performed. The IDCODE has the following structure:

**OpCode**                X'0300XXXX'
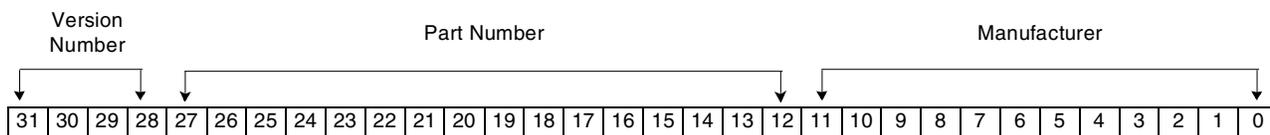
| Version Number | | Part Number | | Manufacturer | |
|---|---|---|---|---|---|

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

| Bit(s) | Name | Description |
|---|---|---|
| 31-28 | Version Number | This is set to X'4' for the IBM2520L8767. |
| 27-12 | Part Number | This is set to X'1D00' for the IBM2520L8767. |
| 11-0 | Manufacturer | This is set to X'049' for IBM. |

### 21.2: SAMPLE/PRELOAD

Captures the state of the boundary scan I/O. As the values captured are scanned out, new values can be loaded into the boundary scan latches. This operation will not effect functional operation.

**OpCode**                X'0402XXXX'

### 21.3: EXTEST

Drives the values in the boundary scan latches onto their respective I/O. This function can be used in conjunction with SAMPLE/PRELOAD to perform card wire tests.

**OpCode (Compliant)**    X'00000000'
**OpCode (Compatible)**   X'0600XXXX'

### 21.4: BYPASS

Selects the single bit bypass register for data scans.

**OpCode (Compliant)**    X'FFFFFFFF'
**OpCode (Compatible)**   X'FFFFXXXX' or X'0000XXXX'

### 21.5: RUNBIST

Causes built in self test (BIST) to execute.

**OpCode**                X'0770XXXX'

### 21.6: BIST_RESULTS

Returns a 64-bit value when a data scan is performed. Bits 63-32 are the PRPG and bits 31-0 are the MISR from the BIST logic.

**Opcode**             X'1F02XXXX'

### 21.7: WALNUT_MODE

This command enables Walnut compatible mode.

**Opcode**             X'3000'

### 21.8: COMPLIANT_MODE

This command enables JTAG compliant mode.

**Opcode**             X'33010000'

### 21.9: STOP

This command halts the functional clocks of IBM2520L8767 in anticipation of a scan. After the STOP command is scanned in, a data scan that takes the TAP controller through the Capture-DR, Exit1-DR, and Update-DR states should be performed.

**Opcode**             X'2002'

### 21.10: SCAN

This command causes TDI to be clocked into the scan chain during a subsequent data scan. The scan out of the scan chain is placed on TDO. This command will not work unless a STOP command is sent down immediately before the SCAN command is issued.

**Opcode**             X'0802'

### 21.11: SCAN_IN

This command causes TDI to be clocked into the scan chain during a subsequent data scan. TDO is forced to B'0'. This command will not work unless a STOP command is sent down immediately before the SCAN_IN command is issued.

**Opcode**             X'0900'

### 21.12: SCAN_OUT

This command causes the scan out of the scan chain to be placed on TDO. Data is circulated through the scan chains. TDI is ignored. This command will not work unless a STOP command is sent down immediately before the SCAN_OUT command is issued.

**Opcode**             X'0A00'

# Sonet Framer Core

Refer to NPBUS Control Register to access registers in this section.

## FRAMR Chiplet Address Mapping

| Chiplet Name | Short Name | Chiplet Base Address | Chiplet Address Range | Number of Bytes |
|---|---|---|---|---|
| Reserved | | X'000' | X'000 - 0FF' | 256 |
| ACH_Tx | HT | X'100' | X'100 - 1FF' | 256 |
| ACH_Rx | HR | X'200 ' | X'200 - 2FF ' | 256 |
| Reserved | | X'300 ' | X'300 - 3FF ' | 256 |
| OFP_Tx | OT | X'400 ' | X'400 - 7FF ' | 1024 |
| OFP_Rx | OR | X'800 ' | X'800 - BFF ' | 1024 |
| GPPINT | GP | X'C00 ' | X'C00 - CFF' | 256 |
| Reserved | | X'D00 ' | X'D00 - FFF ' | 768 |

## GPPINT Architecture

The General Purpose Processor INTerface (GPPINT) provides direct access to registers located in the GPPINT module, but delayed access to registers and counters located in the GppHandler modules of the various chiplets of the SONET core. GPPINT controls the handshaking with the external microprocessor as well as the handshaking with the GppHandlers at the asynchronous chiplet interfaces. Address decoding is done to the chiplet level in GPPINT. In addition, addresses are decoded to the register level for the local GPPINT registers.

### Reset Register

Each chiplet is controlled by one reset bit. At power-on, all reset bits are active and the chiplets disabled. They can be released by the General Purpose Processor (GPP) only after all global configuration parameters have been set and the clocks to the chiplets have been established. In addition, there are reset bits for the chiplets that do not have their own GppHandler.

### Interrupt Registers

The interrupt register is used as a pointer to the chiplet interrupt registers with pending requests, the clock status error register and the handshaking error register. An active bit of the interrupt register is reset by removing the cause for the request in the corresponding chiplet or by masking the active IRQ bit(s) in the chiplet; therefore, the interrupt registers (including the pointer) are read-only. All interrupt and pointer registers have a corresponding MASK register (R/W). Every un-masked, active interrupt bit causes an active pointer bit. Every un-masked, active pointer bit causes activation of the interrupt signal to the microprocessor.

## Handshaking Error Registers

Each bit of the handshaking error registers indicates a locked interface to one of the chiplet GppHandlers. Two additional bits indicate various timeout events. To reset an individual bit of the handshaking error register, the cause for the request must be removed AND a '1' must be written into the bit location of the register (R/W). Reading the register will reset the whole (eight bit) register if the corresponding "clear-register" option is set in the configuration register. The handshaking error indication register has a corresponding MASK register (R/W). Every un-masked, active handshaking error bit causes activation of the pointer bit in the GPPINT interrupt register.

## Clock Monitor Status Registers

The clock monitor status register bits indicate the loss of a specific chiplet's clock. They are set whenever a difference between the clock test signal and the individual chiplet clock acknowledge signal occurs after one clock monitor test period. To reset an individual bit of the clock monitor status registers, the clock of the corresponding chiplet must be restored AND a '1' must be written into the bit location of the register (R/W). Reading one of the registers will reset the whole (eight bit) register if the corresponding "clear-register" option is set in the configuration register. The clock monitor status register has corresponding MASK register (R/W). Every un-masked, active clock monitor status bit causes activation of the pointer bit in the GPPINT register.

## Local GPPINT Configuration Registers

There are registers (R/W) for the Clock Monitor Test Period, the Watchdog Timer Period and the "clear-register" option. A read-only register provides the Vital Product Data (VPD).

## Global Static Configuration Registers

These are configuration parameters that are shared by many chiplets or that are needed by chiplets that have no GppHandler. The initial values can be modified by the microprocessor after power-on, but should not be changed later on. All global static configuration registers are R/W.

## Status Registers

These registers provide status information from chiplets that have no GppHandler and are read-only. Presently, there is only one status register for the SIM chiplet (PLL lock status).

## GPPINT Chiplet Address Mapping Overview: Base address = x'C00

| Register Name | Description (all registers are of 8-bit width) | Address Offset | Type | Initial Value |
|---|---|---|---|---|
| RESGP1 | Reset register | X'00' | R/W | '11111111' |
| ... | Reserved | X'01 - 0F' | | |
| IRQGP1 | Chiplet interrupt request register #1 | X'10' | R | '00000000' |
| ... | Reserved | X'11 - 17' | | |
| IRMGP1 | Chiplet interrupt mask register #1 | X'18' | R/W | '00000000' |
| ... | Reserved | X'19 - 1F' | | |
| HShake1 | Handshaking error register #1 | X'20' | R/W | '00000000' |
| ... | Reserved | X'21 - 27' | | |
| HSMask1 | Handshaking error mask register #1 | X'28' | R/W | '00000000' |
| ... | Reserved | X'29 - 2F' | | |
| ClkStat1 | Clock status register #1 | X'30' | R/W | '00000000' |
| ... | Reserved | X'31 - 37' | | |
| ClkMask1 | Clock status mask register #1 | X'38' | R/W | '00000000' |
| ... | Reserved | X'39 - 47' | | |
| CMonGP1 | Clock monitor test period | X'48' | R/W | '00000000' |
| WDTGP1 | Watchdog timer period | X'49' | R/W | '11111111' |
| ConfGP1 | "Clear-register" option register | X'4A' | R/W | '11111111' |
| ... | Reserved | X'4B - 4F' | | |
| VMD | Vital Macro Data register | X'50' | R | '10000001' |
| ... | Reserved | X'51 - 57' | | |
| GATMCS | Common ATM/CS static configuration register | X'58' | R/W | '00000000' |
| GCasc | Common Cascading static configuration register | X'59' | R/W | '10101010' |
| GLoopTx | Transmit Loopback static configuration register | X'5A' | R/W | '00000000' |
| GLoopRx | Receive Loopback static configuration register | X'5B' | R/W | '00000000' |
| GExtRes | External clock recovery circuit reset register | X'5C' | R/W | '00000000' |
| ... | Reserved | X'5D - 67' | | |
| OFPTXGP | OFP_Tx static configuration register | X'68' | R/W | '00000000' |
| OFPRXGP1 | OFP_Rx static configuration register #1 | X'69' | R/W | '00000000' |
| OFPRXGP2 | OFP_Rx static configuration register #2 | X'6A' | R/W | '00000000' |
| ... | Reserved | X'6B - 71' | | |
| PIMRConf2 | PIM_Rx static configuration register #2 | X'73' | R/W | '00000000' |
| ... | Reserved | X'74 - 7E' | | |
| SIMStat | SIM status register | X'7F' | R | N.A. |
| ... | Reserved | X'80 - FF' | | |

## 22: GPPINT Register Description

**Chiplet Reset Register (RESGP)**

The bits of the chiplet reset register control the resetting (enabling / disabling) of complete chiplets. For each bit position, 0 = Reset inactive for this chiplet, 1 = Reset active (chiplet is disabled; DEFAULT).

**Length**          8 bit

**Type**            Read/Write

**Address**         C00

**Power On Value**  X'FF'
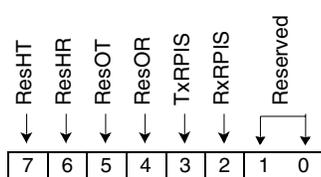
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | ResHT | Reset to chiplet ACH_Tx |
| 6 | ResHR | Reset to chiplet ACH_Rx |
| 5 | ResOT | Reset to chiplet OFP_Tx |
| 4 | ResOR | Reset to chiplet OFP_Rx |
| 3 | TxRPIS | Reset to chiplet PIS_Tx |
| 2 | RxRPIS | Reset to chiplet PIS_Rx |
| 1-0 | Reserved | Reserved |

**Chiplet Interrupt and Mask Registers (IRQGP1, IRMGP1)**

The chiplet interrupt request register indicates pending interrupt requests from individual chiplets. An active bit of this register is reset by removing the cause for the request in the corresponding chiplet or by masking the active IRQ bit(s) in the chiplet; therefore, this register is read-only.

For each bit position for IRQGP1:

  0 = No chiplet interrupt request pending
  1 = Chiplet has pending interrupt request(s).

The chiplet interrupt request mask register bits control the propagation of a chiplet interrupt request to the Sonet macro Interrupt output pin. The mask registers allow read and write access.

For each bit position for IRMGP1:

  0 = The corresponding interrupt request bit is masked (DEFAULT),
  1 = The corresponding interrupt request bit is active (for IRMGP1: the corresponding interrupt request bit activates the Sonet Macro Interrupt).

| **Length** | 8 bits | 8 bits |
|---|---|---|
| **Type** | Read | Read |
| **Address** | IRQGP1 | C10 |
| | IRMGP1 | C18 |
| **Power On Value** | X'00' | X'00' |

| IRQHT | IRQHR | IRQOT | IRQOR | Reserved | | FElocCS | FElocHS |
|---|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | | ↓ | ↓ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7 | IRQHT | IRQ from ACH_Tx |
| 6 | IRQHR | IRQ from ACH_Rx |
| 5 | IRQOT | IRQ from OFP_Tx |
| 4 | IRQOR | IRQ from OFP_Rx |
| 3-2 | Reserved | Reserved |
| 1 | FElocCS | Pending clock status error active |
| 0 | FElocHS | Pending handshaking error active |
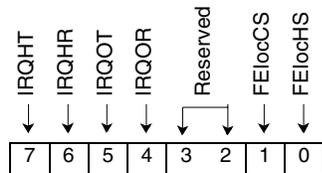
**Handshaking Error Indication and Mask Registers (HShake1, HSMask1)**

The local handshaking error indication register indicates pending handshaking error requests from the GPPINT chiplets.

For each bit position for HShake1:

 0 = Normal operation of the corresponding chiplet
 1 = The corresponding chiplet did not de-assert its DTACK signal.

Exception: The signals TOError and IntError (HShake2(1-0)) have the following meaning:

 00 = Normal operation
 01 = GPP de-asserts Strobes without waiting for DTACK assertion
 10 = Watchdog Timeout in REST state
 11 = Watchdog Timeout in REQ state.

An active bit of the handshaking error indication register is reset by removing the cause for the malfunctioning of the chiplet and by writing a ONE into the corresponding bit position. Reading one register will reset all bits of this register if the "clear-register" option is set in ConfGP1(2).

The handshaking error indication mask register bits control the propagation of the GPPINT handshaking error request of the register HShake1. HSMask1 controls propagation to the signal FElocHS (Bit 0 of IRQGP1 register). The mask registers allow read and write access.

For each bit position in HSMask:

 0 = The corresponding handshaking error indication bit is masked (DEFAULT)
 1 = The corresponding request bit is active (for HSMask1, the corresponding request bit activates signal FElocHS (Bit 0 of IRQGP1 register)). "clear-register" option set in ConfGP1(2).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | HShake1      C20 |
| | HSMask      C28 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|:---:|:---:|:---|
| 7 | | DTACK from ACH_Tx stuck at ONE |
| 6 | | DTACK from ACH_Rx stuck at ONE |
| 5 | | DTACK from OFP_Tx stuck at ONE |
| 4 | | DTACK from OFP_Rx stuck at ONE |
| 3-2 | Reserved | Reserved |
| 1 | TOError | Time Out Error of the GPP interface (see above) |
| 0 | IntError | GPP interface error (see above) |
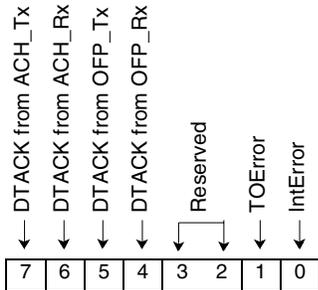
## Clock Monitor Status and Mask Registers (ClkStat1, ClkMask1)

The clock monitor status register bits indicate the loss of a specific island's clock. They are set whenever a difference between the clock test signal and the individual island's clock acknowledge signal occurs after the clock monitor test period.

For each bit position in ClkStat1:

    0 = Normal operation of the corresponding clock island
    1 = The corresponding island clock is lost.

An active bit of this register is reset by restoring the clock of the corresponding clock island and by writing a ONE into the corresponding bit position. Reading one register will reset all bits of this register if the "clear-register" option is set in bit ConfGP1(3).

The clock monitor mask register ClkMask1 controls the propagation of active clock monitor status signals. ClkMask1 controls propagation to the signal FElocCS (Bit 1 of IRQGP1 register). The mask registers allow read and write access.

For each bit position in ClkMask1:

    0 = The corresponding clock status bit is masked (default)
    1 = The corresponding clock status bit is active (for ClkMask1: the corresponding bit activates the signal FElocCS (Bit 6 of IRQGP1 register)).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | ClkStat1    C30 |
| | ClkMask1    C38 |
| **Power On Value** | X'00' |



| Bit(s) | Description |
|---|---|
| 7 | Island ACH_Tx lost clock |
| 6 | Island ACH_Rx lost clock |
| 5 | Island OFP_Tx lost clock |
| 4 | Island OFP_Rx lost clock |
| 3-0 | Reserved |
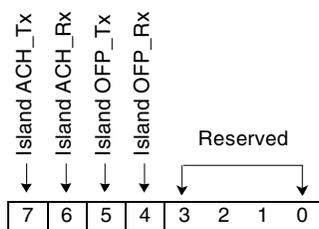
**Clock Monitor Test Period Register (CMonGP1)**

Divider ratio to derive the clock monitor test period from the GPPCLK clock. Clock monitoring is disabled if equal x'00' (DEFAULT).

**Length**              8 bits

**Type**                Read/Write

**Address**             C48

**Power On Value**      X'00'
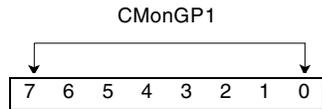
CMonGP1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | CMonGP1(7-0) | Number of GPPCLK cycles/test period |

**Watchdog Timer Period Register (WDTGP1)**

Divider ratio to derive the interface timeout period from the GPPCLK clock. This register is reset to x'FF' whenever a timeout occurs; it has to be reconfigured by a GPP write access.

**Length**              8 bits

**Type**                Read/Write

**Address**             C49

**Power On Value**      X'FF'
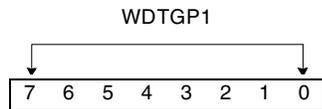
WDTGP1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | WDTGP1(7-0) | Number of GPPCLK clock cycles per timeout period |

## GPPINT Local Configuration Registers (ConfGP1)

The bits of this local configuration register control the resetting of complete registers upon read access ("clear register" option). For each bit position0 = No action upon read access, 1 = The corresponding register is reset upon read access (DEFAULT).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | C4A |
| **Power On Value** | X'FF' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | Reserved | Reserved |
| 3 | | Clear-bit for registers ClkStat1 & ClkStat2 |
| 2 | | Clear-bit for registers HShake1 & HShake2 |
| 1 | | Clear-bit for register SIMStat |
| 0 | Reserved | Reserved |
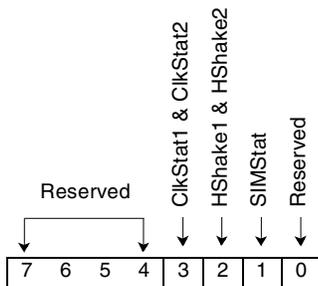
**Vital Macro Data Register (VPD)**

This read-only register displays the macro identification.

**Length**            8 bits

**Type**              Read

**Address**           C50

**Power On Value**    X'01'

Macro Type   Version Number
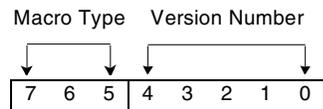
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Description |
|--------|-------------|
| 7-5 | Macro type (000) |
| 4-0 | Version number |

**Static Configuration Register (GATMCS)**

Common static configuration data, providing control signals that are distributed to multiple chiplets. Set once by the GPP before the individual chiplets get enabled and not changing during normal operation.

**Length**            8 bits

**Type**              Read/Write

**Address**           C58

**Power On Value**    X'00'

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Reserved | GATMCSRx | Reserved | GATMCSTx
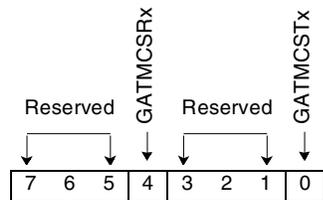
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-5 | Reserved | Reserved |
| 4 | GATMCSRx | ATM cell or CS mode for SDH macro in receive direction:<br>0    SDH macro in ATM mode<br>1    SDH macro in CS mode |
| 3-1 | Reserved | Reserved |
| 0 | GATMCSTx | ATM cell or CS mode for SDH macro in transmit direction:<br>0    SDH macro in ATM mode,<br>1    SDH macro in CS mode |

## GCasc

**Length**            8 bits

**Type**              Read/Write

**Address**           C59

**Power On Value**    X'88'
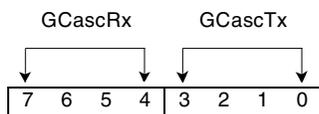


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | GCascRx(7-4) | Defines SDH macros in receive direction<br>0001  STS3c<br>1000  STM1 others => reserved |
| 3-0 | GCascTx(7-4) | Defines SDH macros in transmit direction<br>0001  STS3c<br>1000  STM1 others => reserved |

## GLoopTx

Transmit loopback control. For each bit position, 0 = ACH Loopback disabled (DEFAULT), 1 = ACH Loopback enabled.

**Length**            8 bits

**Type**              Read/Write

**Address**           C5A

**Power On Value**    X'00'
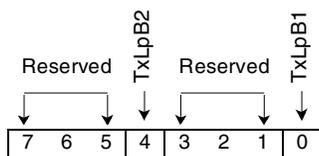


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-5 | Reserved | Reserved |
| 4 | TxLpB2 | Loopback #2 control, Tx macro |
| 3-1 | Reserved | Reserved |
| 0 | TxLpB1 | Loopback #1 control, Tx macro |

**GLoopRx**

Receive Loopback control. For each bit position, 0 = ACH Loopback disabled (DEFAULT),
1 = ACH Loopback enabled.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | C5B |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-1 | Reserved | Reserved |
| 0 | RxLpB2 | Loopback #2 control, Rx macro |

**GExtRes**

External clock recovery circuit reset signal. Delivered to external circuit (deserializer) via device pins. The active level depends on the external circuit used. Default value at power-on-reset is LOW.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | C5C |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-1 | Reserved | Reserved |
| 0 | RSTCRec | External recovery reset |
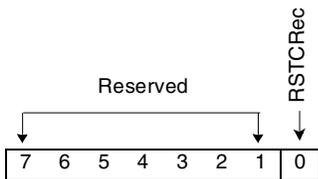
**OFPTXGP**

Static configuration data, providing control signals for chiplet OFP_Tx. Set once by the GPP before the individual chiplets are enabled and not changing during normal operation.

**Length**          8 bits

**Type**            Read/Write

**Address**         C68

**Power On Value**  X'00'

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-5 | Reserved | Reserved |
| 4 | PtrProc | 0   AU pointer processing disabled in ATM mode<br>1   AU pointer processing enabled in ATM mode |
| 3-1 | Reserved | Reserved |
| 0 | SDHC1Tx | 0   C1 byte replaced by section trace J1 byte (ITU-T standard)<br>1   Old numbering scheme is used OFPRXGP1 & 2 : Static configuration data, providing control signals for chiplets OFP_Rx. Set once by the GPP before the individual chiplets are enabled and not changing during normal operation. |

**OFPRXGP1**

**Length**          8 bits

**Type**            Read/Write

**Address**         C69

**Power On Value**  X'00'
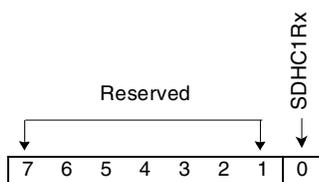
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-1 | Reserved | Reserved |
| 0 | SDHC1Rx | 0   The new (ITU-T standard) numbering scheme is used<br>1   Old numbering scheme is used |

## OFPRXGP2

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | C6A |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1-0 | A2Frm | OFP_Rx RxSoFrm assertion controls<br>00    RxSoFrm asserted during 3rd A2 byte<br>01    RxSoFrm asserted during 1st A2 byte<br>10    RxSoFrm asserted during 2nd A2 byte<br>11    RxSoFrm asserted during 3rd A2 byte |

## PIMRConf2

Static configuration data, providing control signals for chiplets PIM_Tx/PIM_Rx. Set once by the GPP before the individual chiplets are enabled and not changing during normal operation.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | C73 |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1-0 | Algo1(7-6) | Selects frame pattern recognition algorithm:<br>00    All bits checked maximum 4 bad frames<br>01    12 bits checked only maximum 4 bad frames<br>10    All bits checked maximum 5 bad frames<br>11    12 bits checked only. maximum 5 bad frames |

**SIMStat**

Status register, providing the GPP with information from the SIM chiplet via PIM. Either SIM-internal or exter-
nal PLL lock status. "Clear-register" option set in ConfGP1(1).

**Length**            8 bits

**Type**              Read

**Address**           C7F

**Power On Value**    N/A
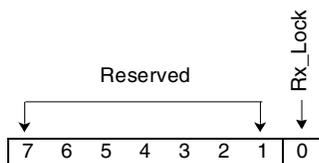


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-1 | Reserved | Reserved |
| 0 | Rx_Lock | 0    Rx PLL is still in phase aquisition process<br>1    Rx PLL is enabled and has locked to the incoming data stream incoming data stream |

## 23: GPPHandler Architecture

All GPP handlers for the various chiplets have the following general register structure.

## GPPHandler Architecture

| Address Range | Register Function |
|---|---|
| X'0 - 1' | Read on the Fly registers |
| X'2 - 3' | Counter enable registers |
| X'4 - 2F' | Counters and counter threshold registers |
| X'30' | Reset register |
| X'31 - 32' | Command registers |
| X'33 - 37' | Event latch registers (was called status) |
| X'38 - 47' | Interrupt registers (addr=int reg, addr-1=int mask reg) |
| X'48 - 57' | Configuration registers |

### Counter Registers

Every counter has an enable bit in the counter enable register (addr 2 or 3), and optionally up to two programmable thresholds. Each counter has an interrupt bit for overflow and up to two interrupt bits for threshold crossing in the counter interrupt registers. For all counters in one handler there is one common read on the fly register', that is used to store the higher order bytes to obtain a correct readback value for counter larger than eight bits. Counters are read only registers, the count enable registers are read / write. Note on COUNTER reading: Independent of the counter length, given that a counter has address n as base, reading address n or address n-1 both yield the least significant byte of the counter. Reading address n has no influence on the counter but reading address n-1 will reset the counter after the read. Reading address n or n-1 will always latch the higher order bytes into the read on the fly register (before the optional automatic reset). Counters can only be read and not written to. For a 16-bit counter the most significant byte should be read from ROFmid (address 0). For a 24-bit counter, the most significant byte is read from ROFhi (address 1), the next byte from ROFmid (address 0). To completely read a 24-bit counter: first read least significant byte from counter address n or n-1, followed by reading ROFmid and ROFhi (address 0; address 1).

### Reset Registers

Each handler has a two bit reset register. Bit zero is the chiplet reset control. This bit is active high after power on reset, causing the chiplet to be disabled. Bit one is the chiplet halt signal, which for selected chiplets freezes the state machines for diagnostic purposes. This is a read / write register.

### Command Registers

The optional command register(s) will generate events to the chiplet. When a bit is written high by the microprocessor, it will remain high for one chiplet clock cycle. Therefore, reading back a command register will always read back zeros. This is a read / write register.

## Event Latch Registers

The optional event latch register(s) remember one ore more occurrences of events that happen in a chiplet. This may be considered as a one-bit saturating counter. Each bit in the register corresponds to an event in the chiplet. Such bits remain high after the event happened until the microprocessor implicitly or explicitly resets the bit. This is configurable: implicit reset is done by writing a high value to the bit that is to be reset. Explicit will reset all bits of one register when the register is read. This is a read / write register.

## Interrupt Registers

When there are counters, user interrupts or fatal bits in a chiplet, a MAIN INTERRUPT register will be present. Bit zero always is the fatal interrupt bit, which is set as soon as any of the fatal interrupt events occur. The other bits refer to counters or user interrupt registers, to allow easy determination of the interrupt cause. Each Interrupt register has an interrupt MASK register to enable or disable interrupt. After power on Reset, interrupts are disabled. The interrupt registers are the same as the event latch registers, with the addition that when an interrupt register bit is set, and the corresponding mask register bit is set, the interrupt signal to the GPPINT chiplet is activated. The same mechanism to reset the interrupt register bits is used as for the event latch registers. The interrupt MASK registers are only changed by the microprocessor. The interrupt and interrupt mask registers are read / write.

## Configuration Registers

These registers are programmed by the microprocessor with setup information, and are read/write. The first configuration register reserves bit one and seven to configure explicit or implicit reset of the event latch registers and interrupt registers respectively (when such registers are present).

**F**     Read-On-The-Fly register (auto-generated)

**N**     counter register

**R**     reset register

**I**     interrupt register (auto-generated)

**C**     configuration register

**X**     control or mask register (auto-generated)

**S**     status (event latch) register

**O**     command register

## ATM Cell Handler Architecture: Transmit Direction

### ACH_Tx GPP Handler Address Mapping  Base Address = x'100

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| ROFmid | read-on-the-fly register | X'0' | F 8 | '00000000' |
| ROFhi | read-on-the-fly register (MSByte) | X'1' | F 8 | '00000000' |
| CntEn1 | COUNT ENABLE register | X'2' | X 3 | '000' |
| ACBC | cell counter (read from external FIFO),no threshold (2) | X'4/5' * | N 24 | 'x'000000'' |
| IUC | idle/unassigned cell counter, no threshold (2) | X'6/7'* | N 24 | 'x'000000'' |
| ACBE | corrupted cell error counter (2) | X'8/9' * | N 8 | '00000000' |
| ACBETh11 | threshold register for counter ACBE | X'A' | X 8 | '10000000' |
| RESET | default RESET register | X'30' | R 2 | '01' |
| STAT1 | status register #1 | X'33' | S 8 | |
| IUCSTAT1 | status register #2 | X'34' | S 2 | |
| MainIRQ | MAIN INTerrupt register | X'38' | I 2 | |
| M_MainIRQ | INT MASK register (for MainIRQ) | X'39' | X 2 | '00' |
| CntrIRQ1 | COUNTER INTerrupt register | X'3A' | I 4 | |
| M_CntrIRQ1 | INT MASK register (for CntrIRQ1) | X'3B' | X 4 | '0000' |
| CELLTENABLE | Chiplet cofiguration register | X'48' | C 6 | '001111' |
| ACBTXTHRPAE | programmable almost empty threshold | X'49' | C 7 | '0001110' |
| HEADERBYTE1 | IU-cell header byte 1 (1) | X'4A' | C 8 | '00000000' |
| HEADERBYTE2 | IU-cell header byte 2 (1) | X'4B' | C 8 | '00000000' |
| HEADERBYTE3 | IU-cell header byte 3 (1) | X'4C' | C 8 | '00000000' |
| HEADERBYTE4 | IU-cell header byte 4 (1) | X'4D' | C 8 | '00000001' |
| HEADERBYTE5 | IU-cell header byte 5 (1) | X'4E' | C 8 | '01010010' |
| PAYLOADBYTE | IU-cell payload byte | X'4F' | C 8 | '01101010' |
| HECENCTRL | HEC processing control | X'50' | C 7 | '0001100' |
| HECOFFSET | HEC offset pattern register | X'51' | C 8 | '01010101' |
| HECMASKAND | HEC error corruption mask (AND) | X'52' | C 8 | '11111111' |
| HECMASKOR | HEC error corruption mask (OR) | X'53' | C 8 | '00000000' |
| SDBTXTHRPAF | programmable almost full threshold | X'54' | C 6 | '110000' |

1. Defaults according ITU I.432
2. Meaning of counter address marked as (*) Independant of the counter width, given that a counter has chiplet address N as a base, reading address N or address N-1 both yield the Least Significant Byte of the counter. Reading address N has no effect on the counter but reading address N-1 resets the counter after read operation

## Counter Registers

### ROFmid

Read-on-the-fly register. Middle Significant Byte

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 100 |
| **Power On Value** | X'00' |

ROFmid

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ROFmid(7-0) | Read-on-the-fly register Middle Significant Byte |

### ROFhi

Read-on-the-fly register. Most Significant Byte

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 101 |
| **Power On Value** | X'00' |
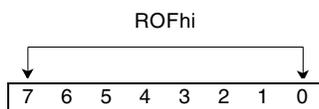
ROFhi

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ROFhi(7-0) | Read-on-the-fly register Most Significant Byte |

**ACBC**

Number of cells read from external FIFO (24-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 104/105 |
| **Power On Value** | X'00' |

ACBC(16:23)

```
7  6  5  4  3  2  1  0
```
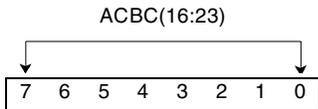
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ACBC(16:23) | External FIFO cell counter Least Significant Byte |

**IUC**

Number of transmitted Idle and Unassigned cells (24-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 106/107 |
| **Power On Value** | X'00' |

IUC(16:23)

```
7  6  5  4  3  2  1  0
```
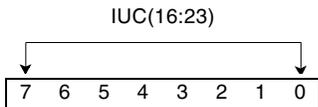
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | IUC(16:23) | Idle/unassigned cell counter Least Significant Byte |

## ACBE

Number of errors (corrupted cell read from external FIFO). Eight-bit Counter overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 108/109 |
| **Power On Value** | X'00' |
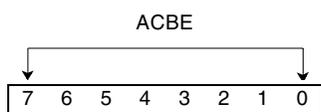
ACBE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ACBE(7-0) | External FIFO error counter |

## ACBETh11-

Threshold for number of errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 10A |
| **Power On Value** | X'80' |
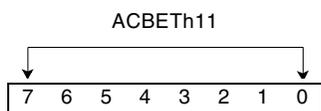
ACBETh11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ACBETh11(7-0) | Threshold for error counter |

**CntEn1**

Counter On/Off control register for ACH_Tx. For each bit position, 0 = Counter is disabled,
1 = Counter is enabled.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 102 |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-3 | Reserved | Reserved |
| 2 | EN-ACBE | Error counter enable |
| 1 | EN-IUC | Idle/unassigned cell counter enable |
| 0 | EN-ACBC | Cell counter enable |
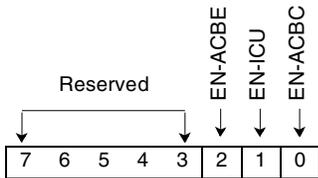
**Reset Register (RESET)**

Reset / Halt chiplet control register. This register is automatically preset to the default value by the reset
signal ResHT from the GPPINT. For each bit position, 0 = Reset/Halt not active, 1 = Reset/Halt active.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 130 |
| **Power On Value** | X'01' |



| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | Reserved | Reserved |
| 1 | | Halt (freeze) ACH_Tx chiplet |
| 0 | | Reset (disable) ACH_Tx chiplet |
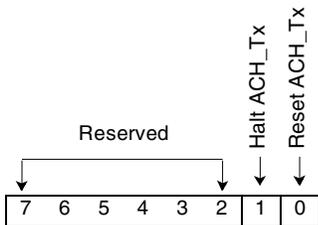
**Status Registers**

**STAT1**

Status register #1 of this chiplet. This is an event latch register.

**Length**          8 bits

**Type**            Read/Write

**Address**         133

**Power On Value**  -
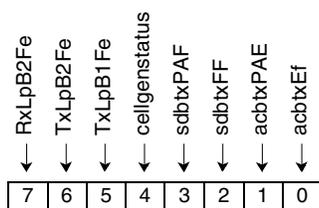
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | RxLpB2Fe | Rx Loopback #2 configuration mismatch |
| 6 | TxLpB2Fe | Tx Loopback #2 configuration mismatch |
| 5 | TxLpB1Fe | Tx Loopback #1 configuration mismatch |
| 4 | cellgenstatus | 0      Idle/unassigned cell is transmitted<br>1      Cell from external FIFO is transmitted |
| 3 | sdbtxPAF | Programmable almost full flag from SDB_Tx |
| 2 | sdbtxFF | FIFO full flag from SDB_Tx |
| 1 | acbtxPAE | Programmable almost empty flag from External Transmit FIFO |
| 0 | acbtxEf | FIFO empty flag from external Transmit FIFO |

## IUCSTAT1

Status register #2 of this chiplet. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 134 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|---|---|---|
| 7-1 | Reserved | Reserved |
| 0 | IUCFErr | Unexpected state transition in FSM |
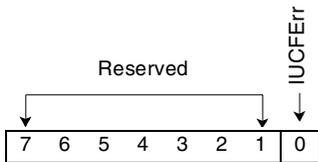
## Interrupt Request and Mask Registers

## MainIRQ

Register to indicate fatal interrupt events and to point to user IRQ registers with active requests. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 138 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |
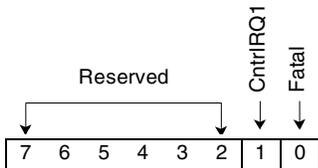
## M_MainIRQ

Register to mask pending interrupt requests. A masked request will not generate an outgoing IRQ to the GPPINT. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates signal IRQHT1 to GPPINT.

**Length**            8 bits

**Type**              Read/Write

**Address**           139

**Power On Value**    X'00'
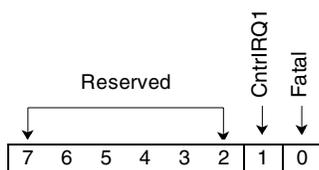


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | Reserved | Reserved |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |

## CntrIRQ1

Register to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

**Length**            8 bits

**Type**              Read/Write

**Address**           13A

**Power On Value**    -
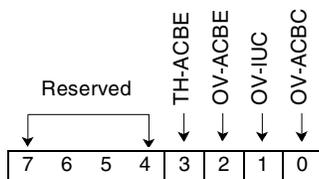


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | Reserved | Reserved |
| 3 | TH-ACBE | Threshold overstep error counter |
| 2 | OV-ACBE | Overflow error counter |
| 1 | OV-IUC | Overflow idle/unassigned cell counter |
| 0 | OV-ACBC | Overflow cell counter |

**M_CntrIRQ1**

Register to mask pending counter interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates the pointer bit in MainIRQ reg.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 13B |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | Reserved | Reserved |
| 3 | TH-ACBE | Threshold overstep error counter |
| 2 | OV-ACBE | Overflow error counter |
| 1 | OV-IUC | Overflow idle/unassigned cell counter |
| 0 | OV-ACBC | Overflow cell counter |
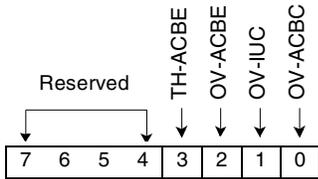
## Configuration Registers

**CELLTENABLE**

Register to control various modes of operation of this chiplet.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 148 |
| **Power On Value** | X'0F' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |

| Bit(s) | Name | Description | |
|--------|----------|---|---|
| 5 | TxLpB1only | 0 | On the fly monitoring (LpB #1) |
| | | 1 | Loopback #1 only |
| 4 | TxLpB2only | 0 | On the fly monitoring (LpB #2) |
| | | 1 | Loopback #2 only |
| 3 | IUCenable | 0 | Generation of IUC disabled |
| | | 1 | Generation of IUC enabled |
| 2 | ACBenable | 0 | External FIFO read disabled |
| | | 1 | External FIFO read enabled |
| 1 | AutRst_Sta | 0 | No action on read access |
| | | 1 | Auto-reset status registers upon read access |
| 0 | AutRst_Int | 0 | No action on read access |
| | | 1 | Auto-reset interrupt request registers upon read access |
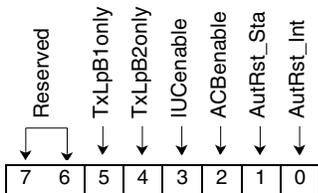
## ACBTXTHRPAE

Threshold for Programmable Almost Empty flag of external FIFO in transmit direction.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 149 |
| **Power On Value** | X'0E' |

Reserved

ACBTXTHRPAE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7 | Reserved | Reserved |
| 6-0 | ACBTXTHRPAE(7-1) | Threshold for PAE flag |

## SDBTXTHRPAF

Threshold for Programmable Almost Full flag (SDB_Tx).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 154 |
| **Power On Value** | X'30' |

Reserved

SDBTXTHRPAF

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5-0 | SDBTXTHRPAF(7-2) | Threshold for PAF flag |

## HEADERBYTE1

Idle/Unassigned cell header byte #1. Default pattern according to ITU I.432.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 14A |
| **Power On Value** | X'00' |

HEADERBYTE1

```
  7  6  5  4  3  2  1  0
```
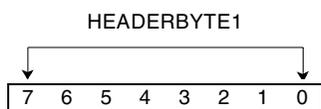
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | HEADERBYTE1(7-0) | IU-cell header byte #1 |

## HEADERBYTE2

Idle/Unassigned cell header byte #2. Default pattern according to ITU I.432.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 14B |
| **Power On Value** | X'00' |

HEADERBYTE2

```
  7  6  5  4  3  2  1  0
```
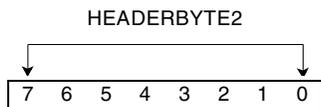
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | HEADERBYTE2(7-0) | IU-cell header byte #2 |

## HEADERBYTE3

Idle/Unassigned cell header byte #3. Default pattern according to ITU I.432.

**Length**            8 bits

**Type**              Read/Write

**Address**           14C

**Power On Value**    X'00'

HEADERBYTE3
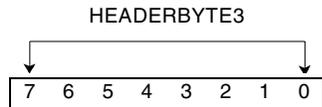
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
| --- | --- | --- |
| 7-0 | HEADERBYTE3(7-0) | IU-cell header byte #3 |

## HEADERBYTE4

Idle/Unassigned cell header byte #4. Default pattern according to ITU I.432.

**Length**            8 bits

**Type**              Read/Write

**Address**           14D

**Power On Value**    X'01'

HEADERBYTE4
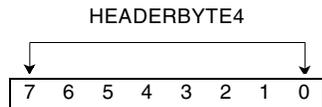
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
| --- | --- | --- |
| 7-0 | HEADERBYTE4(7-0) | IU-cell header byte #4 |

## HEADERBYTE5

Idle/Unassigned cell header byte #5. Default pattern according to ITU I.432.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 14E |
| **Power On Value** | X'52' |

HEADERBYTE5
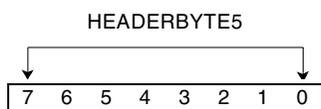
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | HEADERBYTE5(7-0) | IU-cell header byte #5 |

## PAYLOADBYTE

Idle/Unassigned cell payload byte.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 14F |
| **Power On Value** | X'6A' |

PAYLOADBYTE
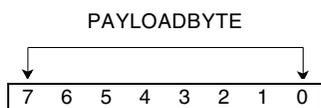
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | PAYLOADBYTE(7-0) | IU-cell payload byte |

**HECENCTRL**

HEC processing control configuration register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 150 |
| **Power On Value** | X'0C' |



| Bit(s) | Name | Description |
|---|---|---|
| 7 | | Reserved |
| 6-4 | Payload byte control | 000  Each payload byte is the same (default)<br>001  Increment payload byte for each ATM cell, start with default after reset<br>010  Increment each payload byte of a cell, start each cell with default byte<br>011  Increment each PL byte of a cell, cross cell boundaries, start first cell after reset with default byte<br>1xx  Each payload byte is the same |
| 3 | SCRe-enable | 0 => ATM cell payload scrambling disabled<br>1 => ATM cell payload scrambling enabled |
| 2 | HECenable | 0 => HEC calculation/manipulation disabled<br>1 => HEC calculation/manipulation enabled |
| 1-0 | HECCntlUDF2 | Mode of final HEC manipulation by UDF1 byte after HECOffset, HECMaskAND, HECMaskOR operations:<br>00    No manipulation<br>01    HEC XOR UDF1<br>10    HEC AND UDF1<br>11    HEC OR UDF1 |

### HECOFFSET

HEC offset pattern register for the byte pattern used in the ATM cell header HEC calculation as base offset according to ITU I.432.

**Length**          8 bits

**Type**            Read/Write

**Address**         151

**Power On Value**  X'55'

HECOFFSET
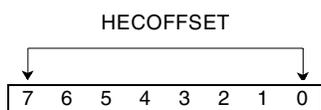
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
| --- | --- | --- |
| 7-0 | HECOFFSET(7-0) | HEC offset pattern |

### HECMASKAND

HEC mask pattern register for the byte pattern used in the ATM cell header HEC calculation as dedicated (ANDing) HEC error corruption mask.

**Length**          8 bits

**Type**            Read/Write

**Address**         152

**Power On Value**  X'FF'

HECMASKAND
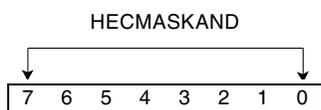
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
| --- | --- | --- |
| 7-0 | HECMASKAND(7-0) | HEC error corruption mask (AND) |

**HECMASKOR**

HEC mask pattern register for the byte pattern used in the ATM cell header HEC calculation as dedicated (ORing) HEC error corruption mask.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 153 |
| **Power On Value** | X'00' |

HECMASKOR

```
  7  6  5  4  3  2  1  0
```
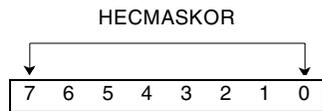
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | HECMASKOR(7-0) | HEC error corruption mask (OR) |

## ATM Cell Handler Architecture: Receive Direction

### ACH_Rx GPP Handler Address Mapping  Base Address = x'200

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| ROFmid | read-on-the-fly register | X'0' | F 8 | '00000000' |
| ROFhi | read-on-the-fly register (MSByte) | X'1' | F 8 | '00000000' |
| CntEn1 | COUNT ENABLE register | X'2' | X 4 | '0000' |
| FHR | Counter, ATM cells written into external FIFO, no threshold | X'4/5' * | N 24 | 'x'000000'' |
| IHR | Counter, received Idle cells from OFP, no threshold | X'6/7' * | N 24 | 'x'000000'' |
| EHR1 | Counter, detected HEC errors with threshold | X'8/9' * | N 16 | 'x'0000'' |
| EHR1Th12 | Threshold reg Byte2 (LSByte) for counter EHR1 | X'A' | X 8 | '00000001' |
| EHR1Th11 | Threshold reg Byte1 for counter EHR1 | X'B' | X 8 | '10000000' |
| BHR | Counter, FIFO full discarded cells: (DiscPAF1=1) AND (TxLpB11=0) with threshold. 2 | X'C/D' * | N 16 | 'x'0000'' |
| BHRTh12 | Threshold reg Byte2 (LSByte) for counter BHR | X'E' | X 8 | '00000001' |
| BHRTh11 | Threshold register Byte1 for counter BHR | X'F' | X 8 | '10000000' |
| RESET | Default RESET register | X'30' | R 2 | '01' |
| CMD1 | Command register (FIFO reset) | X'31' | O 2 | '00' |
| STAT1 | status register | X'33' | S 6 | |
| MainIRQ | MAIN INTerrupt register | X'38' | I 2 | |
| M_MainIRQ | INTerrupt MASK register (for MainIRQ) | X'39' | X 2 | '00' |
| CntrIRQ1 | COUNTER INTerrupt register | X'3A' | I 6 | |
| M_CntrIRQ1 | INTerrupt MASK register (for CntrIRQ1) | X'3B' | X 6 | '000000' |
| CONF5 | Chiplet configuration register | X'48' | C 8 | '00000011' |
| CONF6 | Chiplet configuration register (Alpha/Delta) | X'49' | C 8 | '01100101' |
| H1CONF | 1 | X'4A' | C 8 | '00000000' |
| H2CONF | 1 | X'4B' | C 8 | '00000000' |
| H3CONF | 1 | X'4C' | C 8 | '00000000' |
| H4CONF | 1 | X'4D' | C 8 | '00000001' |
| H5CONF | Dummy byte to align Payload in external FIFO | X'4E' | C 8 | '11010000' |
| CONFC | External FIFO buffer Almost Full threshold | X'4F' | C 7 | '1100000' |

1. Confirmation bytes to identify Idle or Unassigned cells.
2. Meaning of counter address marked as (*) Independant of the counter width, given that a counter has chiplet address N as a base, reading address N or address N-1 both yield the Least Significant Byte of the counter. Reading address N has no effect on the counter but reading address N-1 resets the counter after read operation.

**Counter Registers**

**ROFmid**

Read-on-the-fly registers. Middle Significant Byte.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 200 |
| **Power On Value** | X'00' |
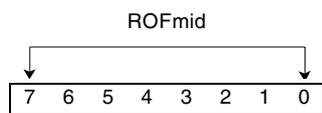
ROFmid

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ROFmid(7-0) | Read-on-the-fly register Middle Significant Byte |

**ROFhi**

Read-on-the-fly registers. Most Significant Byte.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 201 |
| **Power On Value** | X'00' |
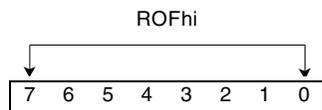
ROFhi

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ROFhi(7-0) | Read-on-the-fly register Most Significant Byte |

### FHR

Number of ATM cells written into external FIFO (24 bit counter). Overflow leads to an interrupt request.

**Length**          8 bits

**Type**            Read

**Address**         204/205

**Power On Value**  X'00'
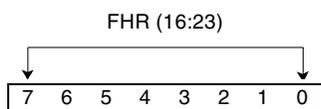
FHR (16:23)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | FHR(16:23) | ATM cell counter Least Significant Byte |

### IHR

Number of Idle cells received from OFP_Rx (24 bit counter). Overflow leads to an interrupt request.

**Length**          8 bits

**Type**            Read

**Address**         206/207

**Power On Value**  X'00'
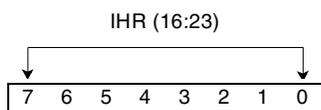
IHR (16:23)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | IHR(16:23) | Idle/unassigned cell counter Least Significant Byte |

**EHR1**

Number of detected HEC errors (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 208/209 |
| **Power On Value** | X'00' |
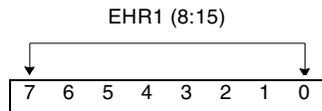
EHR1 (8:15)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | EHR1(8:15) | HEC error counter Least Significant Byte |

**EHR1Th11**

Threshold for number of HEC cells (Most Significant Byte).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 20B |
| **Power On Value** | X'80' |
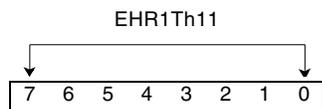
EHR1Th11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | EHR1Th11(7-0) | Threshold for HEC error counter Most Significant Byte |

## EHT1Th12

Threshold for number of HEC errors (Least Significant Byte). Threshold overstep leads to an interrupt request.

**Length**          8 bits

**Type**            Read/Write

**Address**         207

**Power On Value**  X'01'

EHR1Th12
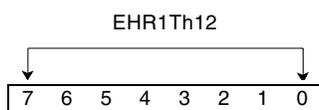
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | EHR1Th12(7-0) | Threshold for HEC error counter Least Significant Byte |

## BHR

Number of discarded cells because of FIFO full condition (16 bit counter). Overflow leads to an interrupt request.

**Length**          8 bits

**Type**            Read

**Address**         20C/20D

**Power On Value**  X'00'
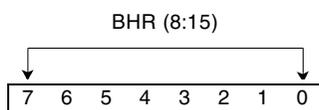
BHR (8:15)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | BHR(8:15) | Discarded cell counter Least Significant Byte |

**BHRTh11**

Threshold for number of discarded cells (Most Significant Byte).

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 20F |
| **Power On Value** | X'80' |

BHRTh11

```
   7  6  5  4  3  2  1  0
```
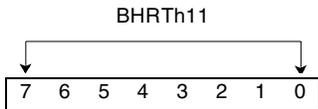
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | BHRTh11(7-0) | Threshold for discarded cell counter, Most Significant Byte |

**BHRTh12**

Threshold for number of discarded cells (Least Significant Byte) Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 20E |
| **Power On Value** | X'01' |

BHRTh12

```
   7  6  5  4  3  2  1  0
```
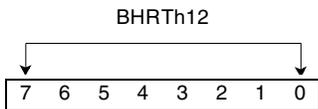
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | BHRTh12(7-0) | Threshold for discarded cell counter, Least Significant Byte |

### CntEn1

Counter On/Off control register for ACH_Rx. For each bit position, 0 =Counter is disabled,
1 = Counter is enabled.

**Length**          8 bits

**Type**            Read/Write

**Address**         202

**Power On Value**  X'00'
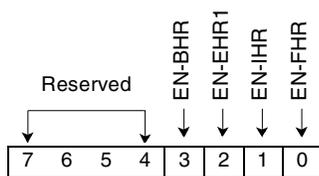
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | Reserved | Reserved |
| 3 | EN-BHR | Discarded cell counter enable |
| 2 | EN-EHR1 | HEC error counter enable |
| 1 | EN-IHR | Idle cell counter enable |
| 0 | EN-FHR | ATM cell counter enable |

### Reset Register (RESET)

Reset / Halt chiplet control register. This register is automatically preset to the default value by the reset
signal ResHR from the GPPINT. For each bit position, 0 = Reset/Halt not active, 1 = Reset/Halt active.

**Length**          8 bits

**Type**            Read/Write

**Address**         230

**Power On Value**  X'01'
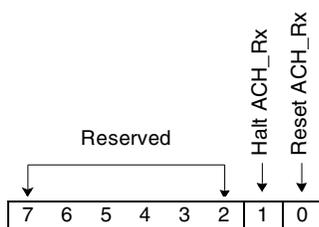
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | Reserved | Reserved |
| 1 | | Halt (freeze) ACH_Rx chiplet |
| 0 | | Reset (disable) ACH_Rx chiplet |

## Command Register (CMD1)

Command register for this chiplet. Single-cycle active if '1' is written into bit position.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 231 |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | | Reserved |
| 1 | ACBfifo | Reset External FIFO |
| 0 | SDBfifo | Reset SDB_Rx fifo |
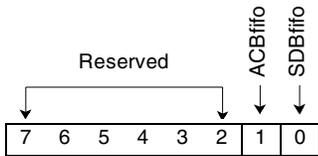
## Status Register (STAT1)

Status register of this chiplet. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 233 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | | Reserved |
| 5-3 | CellDelO | State of the cell delineation process:<br>000  Reset state<br>001  Hunt state<br>010  Presync state<br>100  Sync state |
| 2 | RdFlagEF | SDB FIFO: Read FIFO Empty Flag |
| 1 | WrFlagPAF | External FIFO: Write FIFO programmable almost full flag |
| 0 | WrFlagFF | External FIFO: Write FIFO Full flag |
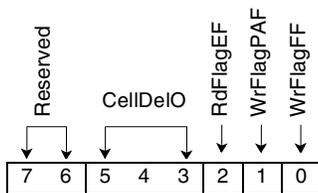
**Interrupt Request And Mask Registers**

**MainIRQ**

Register to indicate fatal interrupt events and to point to user IRQ registers with active requests. For each bit position, 0 = No interrupt request pending,1 = Interrupt request pending

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 238 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |
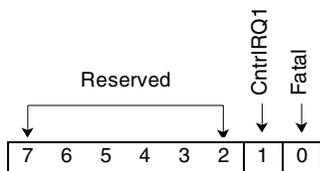
**M_MainIRQ**

Register to mask pending interrupt requests. A masked request will not generate an outgoing IRQ to the GPPINT. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates signal IRQHR1 to GPPINT.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 239 |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |
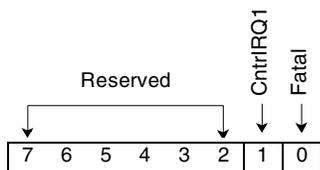
### CntrIRQ1

Register to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 240 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | TH-BHR | Threshold overstep discarded cell counter |
| 4 | OV-BHR | Overflow discarded cell counter |
| 3 | TH-EHR1 | Threshold overstep HEC error counter |
| 2 | OV-EHR1 | Overflow HEC error counter |
| 1 | OV-IHR | Overflow idle cell counter |
| 0 | OV-FHR | Overflow ATM cell counter |

**M_CntrIRQ1**

Register to mask pending counter interrupt requests for each bit position, 0 = The corresponding pending request bit is masked (DEFAULT) 1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| **Length** | 8 bits |
|---|---|
| **Type** | Read/Write |
| **Address** | 241 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | TH-BHR | Threshold overstep discarded cell counter |
| 4 | OV-BHR | Overflow discarded cell counter |
| 3 | TH-EHR1 | Threshold overstep HEC error counter |
| 2 | OV-EHR1 | Overflow HEC error counter |
| 1 | OV-IHR | Overflow idle cell counter |
| 0 | OV-FHR | Overflow ATM cell counter |
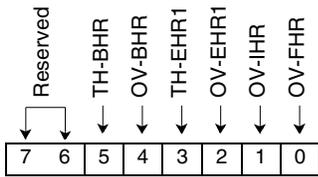
**Configuration Registers**

**CONF5**

Register to control various modes of operation of this chiplet.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 248 |
| **Power On Value** | X'03' |

| No action on read access | Name | Description | |
|---|---|---|---|
| 7 | DetStrtOC | 0 | Do not detect start of cell |
| | | 1 | Detect start of cell |
| 6 | NWrToFifo | 0 | Write into ACB FIFO |
| | | 1 | Do not write into ACB FIFO; all received cells are discarded |
| 5 | NotDetHecEr | 0 | Detect ATM cell with HEC errors |
| | | 1 | Do not detect ATM cell with HEC errors |
| 4 | WrtCHecE | 0 | Do not write ATM cell with HEC errors |
| | | 1 | Write ATM cell with HEC errors |
| 3 | Ndescramb | 0 | De-scramble ATM cell payload |
| | | 1 | Do not de-scramble ATM cell payload |
| 2 | WrIdleC | 0 | Do not write Idle cell into external FIFO |
| | | 1 | Write Idle cell into ACB FIFO |
| 1 | AutRst_Sta | 0 | No action on read access |
| | | 1 | Auto-reset status register upon read access |
| 0 | AutRst_Int | 0 | No action on read access |
| | | 1 | Auto-reset interrupt request registers upon read access |
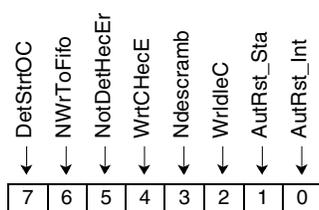
## CONF6

Register to control ATM cell synchronization in this chiplet.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 249 |
| **Power On Value** | X'65' |

```
        Alpha          Delta

      ┌─────────────┐  ┌─────────────┐
      ↓             ↓  ↓             ↓
    ┌───┬───┬───┬───┬───┬───┬───┬───┐
    │ 7 │ 6 │ 5 │ 4 │ 3 │ 2 │ 1 │ 0 │
    └───┴───┴───┴───┴───┴───┴───┴───┘
```
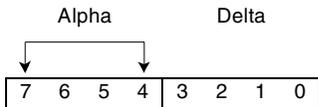
| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | Alpha(7-4) | Required number of consecutive false HEC detected to return from SYNC to HUNT state |
| 3-0 | Delta(7-4) | Required number of consecutive good HEC detected to jump from PRESYNC to SYNC state |

## CONFC

Threshold for Programmable Almost Full flag of the external FIFO.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24F |
| **Power On Value** | X'60' |

```
  Reserved
            CONFC (7-1)
      ┌──────────────────────┐
    ↓ ↓                      ↓
    ┌───┬───┬───┬───┬───┬───┬───┬───┐
    │ 7 │ 6 │ 5 │ 4 │ 3 │ 2 │ 1 │ 0 │
    └───┴───┴───┴───┴───┴───┴───┴───┘
```
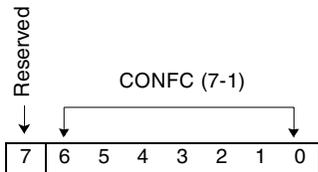
| Bit(s) | Name | Description |
|---|---|---|
| 7 | Reserved | Reserved |
| 6-0 | CONFC(7-1) | Threshold for PAF flag HEADERBYTE1/2/3/4/5 : Idle/Unassigned cell header bytes, default pattern according to ITU I.432. |

## H1CONF

Header pattern #1 to identify Idle/Unassigned cells.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24A |
| **Power On Value** | X'00' |

H1CONF

```
 7   6   5   4   3   2   1   0
```
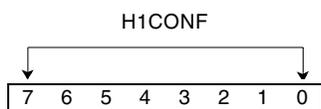
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | H1CONF(7-0) | Header byte #1 |

## H2CONF

Header pattern #2 to identify Idle/Unassigned cells.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24B |
| **Power On Value** | X'00' |

H2CONF

```
 7   6   5   4   3   2   1   0
```
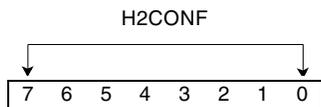
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | H2CONF(7-0) | Header byte #2 |

## H3CONF

Header pattern #3 to identify Idle/Unassigned cells.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24C |
| **Power On Value** | X'00' |

H3CONF

```
  7   6   5   4   3   2   1   0
```
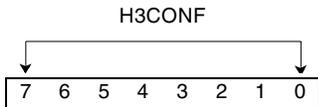
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | H3CONF(7-0) | Header byte #3 |

## H4CONF

Header pattern #4 to identify Idle/Unassigned cells.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24D |
| **Power On Value** | X'01' |

H4CONF

```
  7   6   5   4   3   2   1   0
```
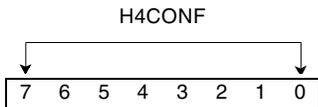
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | H4CONF(7-0) | Header byte #4 |

**H5CONF**

Dummy byte to align the Payload in the ACB_Rx buffer.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 24E |
| **Power On Value** | X'D0' |
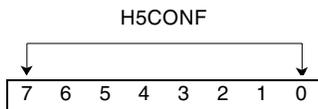
H5CONF

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | H5CONF(7-0) | Payload alignment byte |

## Overhead Frame Processor Architecture: Transmit Direction

### OFP_Tx GPP Handler Address Mapping  Base Address = x'400 (Page 1 of 3)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| CntEn1 | COUNT ENABLE register | X'2' | X 4 | '0000' |
| PTRINC | Pointer increment event counter (1) | X'4/5' * | N 8 | '00000000' |
| | no threshold | | N 8 | |
| PTRDEC | Pointer decrement event counter (1) | X'6/7' * | N 8 | '00000000' |
| | no threshold | | N 8 | |
| ND_EVCNT | New data event counter, no threshold (1) | X'8/9' * | N 8 | '00000000' |
| JUSCNT | Justification error counter (1) | X'A/B' * | N 8 | '00000000' |
| | with no threshold | | N 8 | |
| JUSCNTTh11 | Threshold register for counter JUSCNT | X'C' | X 8 | '10000000' |
| RESET | Default RESET register | X'30' | R 2 | '01' |
| CMD1 | Njus, Pjus, NDF | X'31' | O 3 | '000' |
| STAT1 | Init, hug, mode(7-5) | X'33' | S 6 | |
| STAT2 | Njus, Pjus, NDF | X'34' | S 3 | |
| MainIRQ | MAIN INTerrupt register | X'38' | I 3 | |
| M_MainIRQ | INTerrupt MASK register (for MainIRQ) | X'39' | X 3 | '000' |
| CntrIRQ1 | COUNTER INTerrupt register | X'3A' | I 5 | |
| M_CntrIRQ1 | INTerrupt MASK register (for CntrIRQ1) | X'3B' | X 5 | '00000' |
| IRQ3 | USER INTerrupt register | X'3C' | I 6 | |
| M_IRQ3 | INTerrupt MASK register (for IRQ3) | X'3D' | X 6 | '000000' |
| CONF1 | Configuration register #1 (general A) | X'48' | C 8 | '00000011' |
| CONF2 | Configuration register #2 (general B) | X'49' | C 3 | '000' |
| CONF3 | Configuration register #3 | X'4A' | C 8 | '11111110' |
| | (fscr reload pattern) | | N 8 | |
| CONF4 | Configuration register #4 (errmask) | X'4B' | C 8 | '00000000' |
| CONF5 | Configuration register #5 (erraddress) | X'4C' | C 8 | '00000000' |
| CONF6 | Configuration register #6 (fscr control) | X'4D' | C 8 | '00000001' |
| CONF7 | Configuration register #7 (DCC control) | X'4E' | C 4 | '0000' |
| CONF8 | Configuration register #8 (ThrLoW) | X'4F' | C 6 | '000011' |
| CONF9 | Configuration register #9 (ThrNoW) | X'50' | C 6 | '010001' |
| CONF10 | Configuration register #10 (ThrHiW) | X'51' | C 6 | '100000' |
| SOH-A11 | First A1 | X'100' | 8 | |
| SOH-A12 | Second1 A1 | X'101' | 8 | |
| SOH-A13 | Third A1 | X'102' | 8 | |
| SOH-A21 | First A2 | X'103' | 8 | |
| SOH-A22 | Second A2 | X'104' | 8 | |
| SOH-A23 | Third A2 | X'105' | 8 | |
| SOH-J0 | J0 | X'106' | 8 | |
| | Reserved for national use and not included in frame scrambling | X'107-8' | 8 | |
| SOH-B1 | B1 | X'109' | 8 | |

## OFP_Tx GPP Handler Address Mapping  Base Address = x'400 (Page 2 of 3)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
|  | Media dependant bytes | X'10A-0B' | 8 |  |
| SOH-E1 | E1 | X'10C' | 8 |  |
|  | Media dependant byte | X'10D' | 8 |  |
|  | Reserved for future standardization | X'10E' | 8 |  |
| SOH-F1 | F1 | X'10F' | 8 |  |
|  | Reserved for national use | X'110-11' | 8 |  |
| SOH-D1 | D1 | X'112' | 8 |  |
|  | Media dependant bytes | X'113-14' | 8 |  |
| SOH-D2 | D2 | X'115' | 8 |  |
|  | Media dependant byte | X'116' | 8 |  |
|  | Reserved for future standardization | X'117' | 8 |  |
| SOH-D3 | D3 | X'118' | 8 |  |
|  | Reserved for future standardization | X'119-1A' | 8 |  |
| SOH-H1 | H1 | X'11B' | 8 |  |
| SOH-J0 | b'1001SS11' with S unspecified | X'11C-1D' | 8 |  |
| SOH-H2 | H2 | X'11E' | 8 |  |
| SOH-1s | x'FF' | X'11F-20' | 8 |  |
| SOH-H31 | First H3 | X'121' | 8 |  |
| SOH-H32 | Second H3 | X'122' | 8 |  |
| SOH-H23 | Third H3 | X'123' | 8 |  |
| SOH-B21 | First B2 | X'124' | 8 |  |
| SOH-B22 | Second B2 | X'125' | 8 |  |
| SOH-B23 | Third B2 | X'126' | 8 |  |
| SOH-K1 | K1 | X'127' | 8 |  |
|  | Reserved for future standardization | X'128-29' | 8 |  |
| SOH-K2 | K2 | X'12A ' | 8 |  |
|  | Reserved for future standardization | X'12B-2C' | 8 |  |
| SOH-D4 | D4 | X'12D' | 8 |  |
|  | Reserved for future standardization | X'12E-2F' | 8 |  |
| SOH-D5 | D5 | X'130' | 8 |  |
|  | Reserved for future standardization | X'131-32' | 8 |  |
| SOH-D6 | D6 | X'133' | 8 |  |
|  | Reserved for future standardization | X'134-35' | 8 |  |
| SOH-D7 | D7 | X'136' | 8 |  |
|  | Reserved for future standardization | X'137-38' | 8 |  |
| SOH-D8 | D8 | X'139' | 8 |  |
|  | Reserved for future standardization | X'13A-3B' | 8 |  |
| SOH-D9 | D9 | X'13C' | 8 |  |
|  | Reserved for future standardization | X'13D-3E' | 8 |  |
| SOH-D10 | D10 | X'13F' | 8 |  |
|  | Reserved for future standardization | X'140-41' | 8 |  |
| SOH-D11 | D11 | X'142' | 8 |  |

## OFP_Tx GPP Handler Address Mapping   Base Address = x'400 (Page 3 of 3)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
|  | Reserved for future standardization | X'143-44' | 8 |  |
| SOH-D12 | D12 | X'145' | 8 |  |
|  | Reserved for future standardization | X'146-47' | 8 |  |
| SOH-S1 | S1 | X'148' | 8 |  |
| SOH-Z11 | Reserved for future standardization | X'149-4C' | 8 |  |
| SOH-M1 | M1 | X'14D' | 8 |  |
| SOH-M1 | E2 | X'14E' | 8 |  |
|  | Reserved for future standardization | X'14F-50' | 8 |  |
|  | Justification stuff bytes | X'151-53' | 8 |  |
| POH-J1 | J1 | X'154' | 8 |  |
| POH-B3 | B3 | X'155' | 8 |  |
| POH-C2 | C2 | X'156' | 8 |  |
| POH-G1 | G1 | X'157' | 8 |  |
| POH-F2 | F2 | X'158' | 8 |  |
| POH-H4 | H4 | X'159' | 8 |  |
| POH-F3 | F3 | X'15A' | 8 |  |
| POH-K3 | K3 | X'15B' | 8 |  |
| X'POH-N1' | N1 | 15C | 8 |  |
|  | Reserved | X'15D-5F' | 8 |  |
| POH-J0-16 | 16 byte J0 section trace | X'160-6F' | 8 |  |
|  | Reserved | X'170-7F' | 8 |  |
| POH-J1-16 | 16 byte J1 path trace (3) | X'180-8F' | 8 |  |
| POH-J1-64 | 64 byte J1 path trace (3) | X'190-BF' | 8 |  |

1. Meaning of counter address marked as (*) Independant of the counter width, given that a counter has chiplet address N as a base, reading address N or address N-1 both yield the Least Significant Byte of the counter. Reading address N has no effect on the counter but reading address N-1 resets the counter after read operation
2. Address range 100-17F located in 128x8 GRA Address range 180-1BF located in 64x8 GRA.
3. The 64 byte J1 path trace processing uses the 16 byte addresses of 16 byte J1 path trace to map a full 64 byte space

IBM

## Counter Registers

### PTRINC

Number of pointer increment events (eight-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 404/405 |
| **Power On Value** | X'00' |

PTRINC

```
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
```
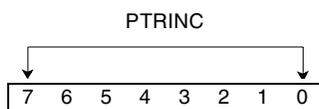
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | PTRINC(7-0) | Pointer increment counter |

### PTRDEC

Number of pointer decrement events (eight-bit counter) Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 406/407 |
| **Power On Value** | X'00' |

PTRDEC

```
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
```
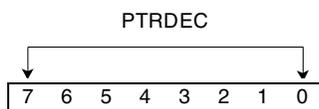
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | PTRDEC(7-0) | Pointer decrement counter |

**ND_EVCNT**

Number of new data events (eight-bit counter) Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 408/409 |
| **Power On Value** | X'00' |
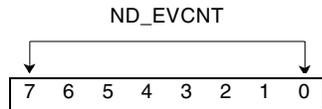
ND_EVCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ND_EVCNT(7-0) | New data event counter |

**JUSCNT**

Number of justification errors detected (eight-bit counter) Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 40A/40B |
| **Power On Value** | X'00' |
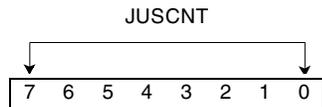
JUSCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | JUSCNT(7-0) | Justification error counter |

### JUSCNTTh11

Threshold for number of justification errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 40C |
| **Power On ValueX'80'** | |



JUSCNTTh11
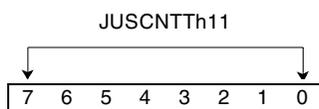
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | JUSCNTTh11(7-0) | Threshold for justificication error counter |

### CntEn1

Counter On/Off control register for OFP_Tx. For each bit position, 0 = Counter is disabled, 1 = Counter is enabled.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 402 |
| **Power On Value** | X'00' |



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | Reserved | Reserved |
| 3 | EN-JUSCNT | Justification error counter enable |
| 2 | EN-ND_EVCNT | New data event counter enable |
| 1 | EN-PTRDEC | Pointer decrement counter enable |
| 0 | EN-PTRINC | Pointer increment counter enable |
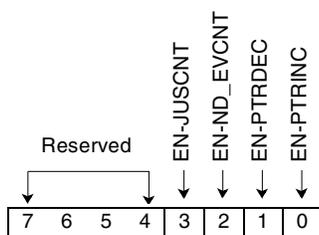
**Reset Register**

**RESET**

Reset/Halt chiplet control register. This register is automatically preset to the default value by the reset signal ResOT coming from GPPINT chiplet. For each bit position, 0 = Reset/Halt not active, 1 = Reset/Halt active.

**Length**                8 bits

**Type**                  Read/Write

**Address**               430

**Power On Value**        X'01'
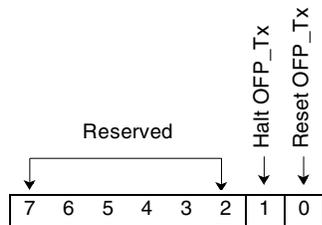


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | Reserved | Reserved |
| 1 |  | Halt (freeze) OFP_Tx chiplet |
| 0 |  | Reset (disable) OFP_Tx chiplet |

**Command Register**

**CMD1**

Command register for the chiplet. Single-cycle active if b'1' is written into bit position.

**Length**                8 bits

**Type**                  Read/Write

**Address**               431

**Power On Value**        X'00'
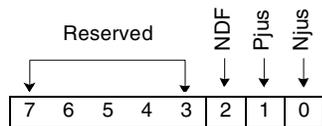


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-3 | Reserved | Reserved |
| 2 | NDF | Force a start-of-new-VC-4 event |
| 1 | Pjus | Perform a positive frequency justification |
| 0 | Njus | Perform a negative frequency justification |

IBM

**Status Registers**

### STAT1

Status register #1 of the chiplet. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 433 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-2 | Reserved | Reserved |
| 1 | HUG | 0  Higher order unequipped generator inactive<br>1  Higher order unequipped generator active |
| 0 | Init | 0  Default GRA initialization not completed<br>1  Default GRA initialization completed' |
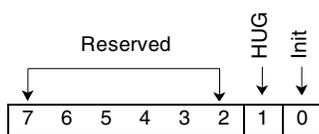
### STAT2

Status register #2 of the chiplet. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 434 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7-3 | Reserved | Reserved |
| 2 | NDF | 0    NO NDF transmitted<br>1    NDF transmitted |
| 1 | Pjus | 0    NO positive frequency justification transmitted<br>1    Positive frequency justification transmitted |
| 0 | Njus | 0    NO negative frequency justification transmitted<br>1    Negative frequency justification transmitted |
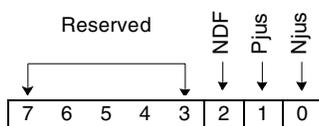
**Interrupt and Mask Registers**

**MainIRQ**

Register to indicate fatal interrupt events and to point to user IRQ registers with active requests. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 438 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|--------|----------|-------------------------------|
| 7-3 | Reserved | Reserved |
| 2 | IRQ3 | Active request in IRQ3 register |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |
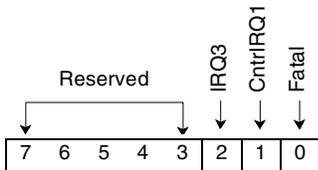
**M_MainIRQ**

Register to mask pending interrupt requests. A masked request will not generate an outgoing IRQ to the GPPINT. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates signal IRQOT to GPPINT.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 439 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-3 | Reserved | Reserved |
| 2 | IRQ3 | Active request in IRQ3 register |
| 1 | CntrlRQ1 | Active request in CntrlRQ1 register |
| 0 | Fatal | Fatal event occurred |
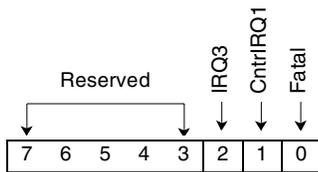
**CntrIRQ1**

Register to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

**Length**            8 bits

**Type**              Read/Write

**Address**           43A

**Power On Value**    -

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-5 | Reserved | Reserved |
| 4 | TH-JUSCNT | Threshold overstep justification error counter |
| 3 | OV-JUSCNT | Overflow justification error counter |
| 2 | OV-ND_EVCNT | Overflow new data event counter |
| 0 | OV-PTRINC | Overflow pointer increment counter |
| 1 | OV-PTRDEC | Overflow pointer decrement counter |

## M_CntrIRQ1

Register to mask pending counter interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 43B |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7-5 | Reserved | Reserved |
| 4 | TH-JUSCNT | Threshold overstep justification error counter |
| 3 | OV-JUSCNT | Overflow justification error counter |
| 2 | OV-ND_EVCNT | Overflow new data event counter |
| 0 | OV-PTRINC | Overflow pointer increment counter |
| 1 | OV-PTRDEC | Overflow pointer decrement counter |
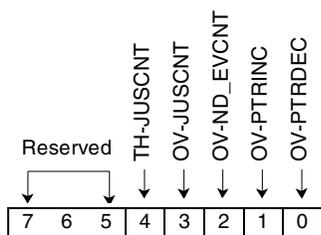
**IRQ3**

Register to indicate active user interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

**Length**              8 bits

**Type**                Read/Write

**Address**           43C

**Power On Value**    -

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Reserved | DLoss | FLow | FHigh | FrmErr | SPCIR | TxLPow | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | DLoss | Data loss = Data FIFO empty |
| 4 | FLow | FIFO low threshold overflow |
| 3 | FHigh | FIFO high threshold overflow |
| 2: | FrmErr | Framing error detected |
| 1 | SPCIR | SPC FSM interrupt request |
| 0 | TxLPow | Low Power indication from Optical/Electrical module |

**M_IRQ3**

Register to mask pending user interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 43D |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | DLoss | Data loss = Data FIFO empty |
| 4 | FLow | FIFO low threshold overflow |
| 3 | FHigh | FIFO high threshold overflow |
| 2 | FrmErr | Framing error detected |
| 1 | SPCIR | SPC FSM interrupt request |
| 0 | TxLPow | Low Power indication from Optical/Electrical module |
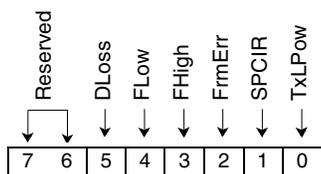
**Configuration Registers**

**CONF1**

Configuration register #1. General OFP_Tx configuration signals A.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 448 |
| **Power On Value** | X'03' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | SPCI(7-4) | Specifies STM-N row number in which an interrupt request will be issued |
| 3 | J1Mode | 0   Transmit 16 byte J1 path trace<br>1   Transmit 64 byte J1 path trace |
| 2 | JusFrm | 0   Allow pointer modification to be performed on frame to frame basis<br>1   Enforces three frames being interleaved between two pointer modification operations |
| 1 | AutRst_Sta | 0   No action on read access<br>1   Auto-reset status register upon read access |
| 0 | AutRst_Int | 0   No action on read access<br>1   Auto-reset interrupt request registers upon read access |
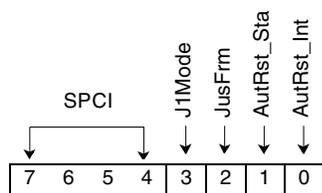
**IBM**

## CONF2

Configuration register #2. General OFP_Tx configuration signals B.

**Length**            8 bits

**Type**              Read/Write

**Address**           449

**Power On Value**    X'00'
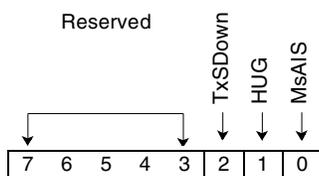
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-3 | | Reserved |
| 2 | TxSDown | Directly connected to output pin :<br>0   Optical/Electrical normal operation<br>1   transmit shutdown for Optical/Electrical module |
| 1 | HUG | 0   NO unequipped STM-N signal<br>1   Enforce unequipped STM-N signal |
| 0 | MsAIS | 0   NO multiplex section AIS<br>1   Enforce multiplex section AIS |

## CONF3

Configuration register #3

**Length**            8 bits

**Type**              Read/Write

**Address**           44A

**Power On Value**    X'FE'
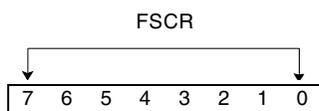
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | FSCR(7-0) | Reload pattern for frame scrambler |

## CONF4

Configuration register #4.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 44B |
| **Power On Value** | X'00' |
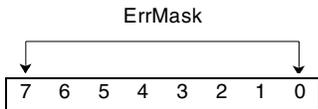
ErrMask

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ErrMask(7-0) | Mask register forcing bit error insertion. XORed with retrieved SOH/POH |

## CONF5

Configuration register #5.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 44C |
| **Power On Value** | X'00' |

ErrAddr

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ErrAddr(7-0) | Error mask address register. Indicates address of SOH/POH byte to be corrupted |

**CONF6**

Configuration register #6 Frame scrambling control register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 44D |
| **Power On Value** | X'01' |



| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | CIDnum(7-2) | Number of all - '1'/'0' bytes |
| 1 | CIDEn | CID Insertion Enable<br>0   NO CID insertion<br>1   perform CID insertion |
| 0 | ScrEn | Scramble Enable<br>0   NO scrambling<br>1   perform scrambling |
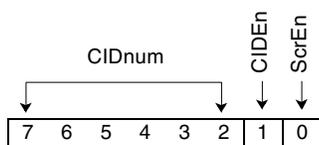
## CONF7

Configuration register #7 DCC control register.

| **Length** | 8 bits |
|---|---|
| **Type** | Read/Write |
| **Address** | 44E |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | Reserved | Reserved |
| 3 | EdgeMode | 0    active falling edge<br>1    active rising edge |
| 2 | ClkMode | 0    continuous clock<br>1    strobed clock |
| 1 | OpMode | 0    DCC1 channel (D1 - D3)<br>1    CC2 channel (D4 - D12) |
| 0 | Enable | 0    disable DCC1 processing<br>1    enable DCC1 processing |
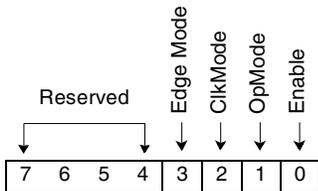
## CONF8

Configuration registers #8 Low water FIFO threshold register.

| **Length** | 8 bits |
|---|---|
| **Type** | Read/Write |
| **Address** | 44F |
| **Power On Value** | X'03' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5-0 | ThrLoW(7-2) | Low Water FIFO threshold, Default value is 3 |
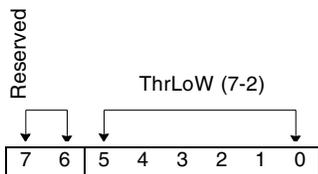
## CONF9

Configuration registers #9 Normal water FIFO threshold register.

**Length**          8 bits

**Type**            Read/Write

**Address**         450

**Power On Value**  X'11'
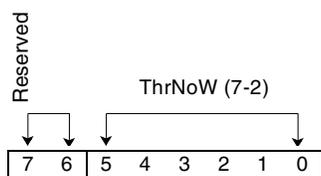


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-6 | Reserved | Reserved |
| 5-0 | ThrNoW(7-2) | Normal Water FIFO threshold, Default value is 17 |

## CONF10

Configuration registers #10 High water FIFO threshold register.

**Length**          8 bits

**Type**            Read/Write

**Address**         451

**Power On Value**  X'20'
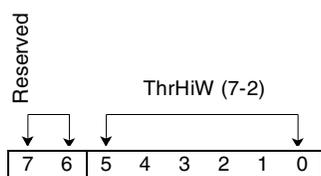


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-6 | Reserved | Reserved |
| 5-0 | ThrHiW(7-2) | High Water FIFO threshold, Default value is 32 |

## Overhead Frame Processor Architecture: Receive Direction

## OFP_Rx GPP Handler Address Mapping  Base Address = x'800c (Page 1 of 4)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| ROFmid | Read-on-the-fly register | X'0' | F 8 | '00000000' |
| CntEn1 | COUNT ENABLE register #1 | X'2' | X 8 | '00000000' |
| CntEn2 | COUNT ENABLE register #2 | X'3' | X 3 | '000' |
| B1BITCNT | BIP-8 B1 bit error counter | X'4/5' * | N 16 | 'x'0000'' |
| B1BITCNTTh12 | Threshold register Byte2 (Least significant Byte) for B1BITCNT | X'6' | X 8 | '00000000' |
| B1BITCNTTh11 | Threshold register Byte1 for counter B1BITCNT | X'7' | X 8 | '01111101' |
| B1BLKCNT | BIP-8 B1 block error counter (1) | X'8/9' * | N 16 | 'x'0000'' |
| B1BLKCNTTh12 | Threshold register Byte2 (Least significant Byte) for B1BLKCNT | X'A' | X 8 | '00000000' |
| B1BLKCNTTh11 | Threshold register Byte1 for counter B1BLKCNT | X'B' | X 8 | '01111101' |
| B2BITCNT | BIP-24 B2 bit error counter, 2 thresholds (1) | X'C/D' * | N 16 | 'x'0000'' |
| B2BITCNTTh12 | Degradation threshold Byte2 (Least significant Byte) for B2BITCNT | X'E' | X 8 | '00100000' |
| B2BITCNTTh11 | Degradation threshold Byte1 for B2BITCNT | X'F' | X 8 | '01001110' |
| B2BITCNTTh22 | Failure threshold Byte2 (Least significant Byte) for B2BITCNT | X'10' | X 8 | '00000000' |
| B2BITCNTTh21 | Failure threshold Byte1 for B2BITCNT | X'11' | X 8 | '01111101' |
| B2BLKCNT | BIP-24 B2 block error counter, 2 thresholds (1) | X'12/13' * | N 16 | 'x'0000'' |
| B2BLKCNTTh12 | Degration threshold Byte2 (Least significant Byte) for B2BLKCNT | X'14' | X 8 | '00100000' |
| B2BLKCNTTh11 | Degradation threshold Byte1 forB2BLKCNT | X'15' | X 8 | '01001110' |
| B2BLKCNTTh22 | Failure threshold Byte2 (Least significant Byte) for B2BLKCNT | X'16' | X 8 | '00000000' |
| B2BLKCNTTh21 | Failure threshold Byte1 for B2BLKCNT | X'17' | X 8 | '01111101' |
| B3BITCNT | BIP-8 B3 bit error counter (1) | X'18/19' * | N 16 | 'x'0000'' |
| B3BITCNTTh12 | Threshold register Byte2 (Least significant Byte) for B3BITCNT | X'1A' | X 8 | '00000000' |
| B3BITCNTTh11 | Threshold register Byte1 for counter B3BITCNT | X'1B' | X 8 | '01111101' |
| B3BLKCNT | BIP-8 B3 block error counter (1) | X'1C/1D' * | N 16 | 'x'0000'' |
| B3BLKCNTTh12 | Threshold register Byte2 (Least significant Byte) for B3BLKCNT | X'1E' | X 8 | '00000000' |
| B3BLKCNTTh11 | Threshold register Byte1 for counter B3BLKCNT | X'1F' | X 8 | '01111101' |
| MSREICNT | Multiplex section remote error indication counter (1) | X'20/21' * | N 16 | 'x'0000'' |
| MSREICNTTh12 | Threshold register Byte2 (Least significant Byte) for MSREICNT | X'22' | X 8 | '00000000' |
| MSREICNTTh11 | Threshold register Byte1 for counter MSREICNT | X'23' | X 8 | '01111101' |
| HPREICNT | Higher-order path remote error indication counter (1) | X'24/25' * | N 16 | 'x'0000'' |
| HPREICNTTh12 | Threshold register Byte2 (Least significant Byte) for HPREICNT | X'26' | X 8 | '00000000' |
| HPREICNTTh11 | Threshold register Byte1 for counter HPREICNT | X'27' | X 8 | '01111101' |
| PJ_EVCNT | Positive justification counter, no threshold (1) | X'28/29' * | N 8 | '00000000' |
| NJ_EVCNT | Negative justification counter, no threshold (1) | X'2A/2B' * | N 8 | '00000000' |
| ND_EVCNT | New data event counter, no threshold (1) | X'2C/2D' * | N 8 | '00000000' |
| RESET | Default RESET register | X'30' | R 2 | '01' |
| STAT1 | Status register #1 (Mode) | X'33' | S 3 | |
| STAT2 | Status register #2 (AU pointer) | X'34' | S 6 | |
| STAT3 | Status register #3 (SOH) | X'35' | S 6 | |
| STAT4 | Status register #4 (POH) | X'36' | S 4 | |

## OFP_Rx GPP Handler Address Mapping  Base Address = x'800c (Page 2 of 4)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| MainIRQ | MAIN INTerrupt register | X'38' | I 7 | |
| M_MainIRQ | INTerrupt MASK register for MainIRQ | X'39' | X 7 | '0000000' |
| CntrIRQ1 | COUNTER INTerrupt register | X'3A' | I 8 | |
| M_CntrIRQ1 | INTerrupt MASK register for CntrIRQ1 | X'3B' | X 8 | '00000000' |
| CntrIRQ2 | COUNTER INTerrupt register | X'3C' | I 8 | |
| M_CntrIRQ2 | INTerrupt MASK register for CntrIRQ2 | X'3D' | X 8 | '00000000' |
| CntrIRQ3 | COUNTER INTerrupt register | X'3E' | I 5 | |
| M_CntrIRQ3 | INTerrupt MASK register for CntrIRQ3 | X'3F' | X 5 | '00000' |
| IRQ6 | USER INTerrupt register | X'40' | I 4 | |
| M_IRQ6 | INTerrupt MASK register for IRQ6 | X'41' | X 4 | '0000' |
| IRQ7 | USER INTerrupt register | X'42' | I 8 | |
| M_IRQ7 | INTerrupt MASK register for IRQ7 | X'43' | X 8 | '00000000' |
| IRQ8 | USER INTerrupt register | X'44' | I 8 | |
| M_IRQ8 | INTerrupt MASK register for IRQ8 | X'45' | X 8 | '00000000' |
| CONF1 | Configuration register #1 (general) | X'48' | C 8 | '00111111' |
| CONF2 | Configuration register #2 (SOH processing) | X'49' | C 6 | '0000' |
| CONF3 | Configuration register #3 (POH processing) | X'4A' | C 4 | '0000' |
| CONF4 | Configuration register #4 (APS processing) | X'4B' | C 8 | '00000000' |
| CONF7 | Configuration register #7 (miscellaneous) | X'4E' | C 8 | '00100000' |
| CONF8 | Configuration register #8 (FSCR) | X'4F' | C 8 | '11111110' |
| CONF9 | Configuration register #9 (SL) | X'50' | C 8 | '00010011' |
| SOH-A11 | First A1 | X'100' | 8 | |
| SOH-A12 | Second1 A1 | X'101' | 8 | |
| SOH-A13 | Third A1 | X'102' | 8 | |
| SOH-A21 | First A2 | X'103' | 8 | |
| SOH-A22 | Second A2 | X'104' | 8 | |
| SOH-A23 | Third A2 | X'105' | 8 | |
| SOH-J0 | J0 | X'106' | 8 | |
| | Reserved for national use and not included in frame scrambling (C1) | X'107-8' | 8 | |
| SOH-B1 | B1 | X'109' | 8 | |
| | Media dependant bytes | X'10A-0B' | 8 | |
| SOH-E1 | E1 | X'10C' | 8 | |
| | Media dependant byte | X'10D' | 8 | |
| | Reserved for future standardization | X'10E' | 8 | |
| SOH-F1 | F1 | X'10F' | 8 | |
| | Reserved for national use | X'110-11' | 8 | |
| SOH-D1 | D1 | X'112' | 8 | |
| | Media dependant bytes | X'113-14' | 8 | |
| SOH-D2 | D2 | X'115' | 8 | |
| | Media dependant byte | X'116' | 8 | |

## OFP_Rx GPP Handler Address Mapping  Base Address = x'800c (Page 3 of 4)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| | Reserved for future standardization | X'117' | 8 | |
| SOH-D3 | D3 | X'118' | 8 | |
| | Reserved for future standardization | X'119-1A' | 8 | |
| SOH-H1 | H1 | X'11B' | 8 | |
| SOH-J0 | b'1001SS11' with S unspecified | X'11C-1D' | 8 | |
| SOH-H2 | H2 | X'11E' | 8 | |
| SOH-1s | x'FF' | X'11F-20' | 8 | |
| SOH-H31 | First H3 | X'121' | 8 | |
| SOH-H32 | Second H3 | X'122' | 8 | |
| SOH-H23 | Third H3 | X'123' | 8 | |
| SOH-B21 | First B2 | X'124' | 8 | |
| SOH-B22 | Second B2 | X'125' | 8 | |
| SOH-B23 | Third B2 | X'126' | 8 | |
| SOH-K1 | K1 | X'127' | 8 | |
| | Reserved for future standardization | X'128-29' | 8 | |
| SOH-K2 | K2 | X'12A' | 8 | |
| | Reserved for future standardization | X'12B-2C' | 8 | |
| SOH-D4 | D4 | X'12D' | 8 | |
| | Reserved for future standardization | X'12E-2F' | 8 | |
| SOH-D5 | D5 | X'130' | 8 | |
| | Reserved for future standardization | X'131-32' | 8 | |
| SOH-D6 | D6 | X'133' | 8 | |
| | Reserved for future standardization | X'134-35' | 8 | |
| SOH-D7 | D7 | X'136' | 8 | |
| | Reserved for future standardization | X'137-38' | 8 | |
| SOH-D8 | D8 | X'139' | 8 | |
| | Reserved for future standardization | X'13A-3B' | 8 | |
| SOH-D9 | D9 | X'13C' | 8 | |
| | Reserved for future standardization | X'13D-3E' | 8 | |
| SOH-D10 | D10 | X'13F' | 8 | |
| | Reserved for future standardization | X'140-41' | 8 | |
| SOH-D11 | D11 | X'142' | 8 | |
| | Reserved for future standardization | X'143-44' | 8 | |
| SOH-D12 | D12 | X'145' | 8 | |
| | Reserved for future standardization | X'146-47' | 8 | |
| SOH-S1 | S1 | X'148' | 8 | |
| | Reserved for future standardization | X'149-9C' | 8 | |
| SOH-M1 | M1 | X'14D' | 8 | |
| SOH-M1 | E2 | X'14E' | 8 | |
| | Reserved for future standardization | X'14F-50' | 8 | |
| | Reserved | X'151-53' | 8 | |
| POH-J1 | J1 | X'154' | 8 | |

## OFP_Rx GPP Handler Address Mapping  Base Address = x'800c (Page 4 of 4)

| Register Name | Description | Address Offset | Type Width | Initial Value |
|---|---|---|---|---|
| POH-B3 | B3 | X'155' | 8 | |
| POH-C2 | C2 | X'156' | 8 | |
| POH-G1 | G1 | X'157' | 8 | |
| POH-F2 | F2 | X'158' | 8 | |
| POH-H4 | H4 | X'159' | 8 | |
| POH-F3 | F3 | X'15A' | 8 | |
| POH-K3 | K3 | X'15B' | 8 | |
| POH-N1 | N1 | X'15C' | 8 | |
| | Reserved | X'15D-5F' | 8 | |
| POH-J0-16-E | Expected 16 byte J0 section trace | X'160-6F' | 8 | |
| POH-J0-16-R | Received 16 byte J0 section trace | X'170-7F' | 8 | |
| POH-J1-16 | Expected 16 byte J1 path trace (3) | X'180-8F' | 8 | |
| POH-J1-64 | 64 byte J1 path trace (3) | X'190-BF' | 8 | |

1. Meaning of counter address marked as (*) Independant of the counter width, given that a counter has chiplet address N as a base, reading address N or address N-1 both yield the Least Significant Byte of the counter. Reading address N has no effect on the counter but reading address N-1 resets the counter after read operation.
2. Address range 100-17F located in 128x8 GRA Address range 180-1BF located in 64x8 GRA.
3. The 64 byte J1 path trace processing uses the 16 byte addresses of 16 byte J1 path trace to map a full 64 byte space.

**Counter Registers**

**ROFmid**

Read-on-the-fly registers.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 800 |
| **Power On Value** | X'00' |

ROFmid

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ROFmid(7-0) | Read-on-the-fly register Most Significant Byte |

**B1BITCNT**

Number of BIP-8 B1 bit errors counted since last counter reset (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 804/805 |
| **Power On Value** | X'00' |

B1BITCNT (8:15)
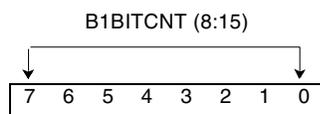
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BITCNT(8:15) | BIP-8 B1 bit error counter Least Significant Byte |

## B1BITCNTTh11

Threshold for number of BIP-8 B1 bit errors.

**Length**          8 bits

**Type**            Read/Write

**Address**         807

**Power On Value**  X'7D'

B1BITCNTTh11
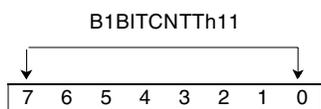
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BITCNTTh11(7-0) | threshold for BIP-8 B1 bit error counter Most Significant Byte |

## B1BITCNTTh12

Threshold for number of BIP-8 B1 bit errors. threshold overstep leads to an interrupt request.

**Length**          8 bits

**Type**            Read/Write

**Address**         806

**Power On Value**  X'00'

B1BITCNTTh12
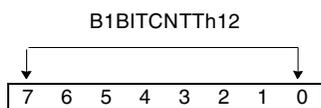
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BITCNTTh12(7-0) | threshold for BIP-8 B1 bit error error counter Least Significant Byte |

## B1BLKCNT

Number of BIP-8 B1 block errors counted since last counter reset (16-bit counter). Overflow leads to an inter-rupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 808/809 |
| **Power On Value** | X'00' |

B1BLKCNT (8:15)
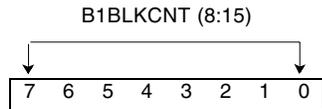
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BLKCNT(8:15) | BIP-8 B1 block error counter Least Significant Byte |

## B1BLKCNTTh11

Threshold for number of BIP-8 B1 block errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 80B |
| **Power On Value** | X'7D' |

B1BLKCNTTh11
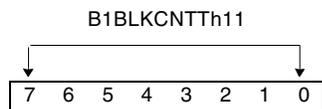
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BLKCNTTh11(7-0) | threshold for BIP-8 B1 block error counter Most Significant Byte |

## B1BLKCNTTh12

Threshold for number of BIP-8 B1 block errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 80A |
| **Power On Value** | X'00' |

B1BLKCNTTh12

```
  7  6  5  4  3  2  1  0
```
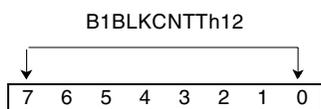
| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B1BLKCNTTh12(7-0) | threshold for BIP-8 B1 block error counter Least Significant Byte |

## B2BITCNT

Number of BIP-24 B2 bit errors counted since last counter reset (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 80C/80D |
| **Power On Value** | X'00' |

B2BITCNT (8:15)

```
  7  6  5  4  3  2  1  0
```

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BITCNT(8:15) | BIP-24 B2 bit error counter Least Significant Byte |

### B2BITCNTTh11

Degradation threshold for number of BIP-24 B2 bit errors.

**Length** 8 bits

**Type** Read/Write

**Address** 80F

**Power On Value** X'4E'

B2BITCNTTh11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | B2BITCNTTh11(7-0) | Degradation threshold for BIP-24 B2 bit error counter Most Significant Byte |

### B2BITCNTTh12

Degradation threshold for number of BIP-24 B2 bit errors. Threshold overstep leads to an interrupt request.

**Length** 8 bits

**Type** Read/Write

**Address** 80E

**Power On Value** X'20'

B2BITCNTTh12
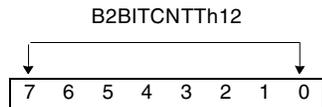
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | B2BITCNTTh12(7-0) | Degradation threshold for BIP-24 B2 bit error counter Least Significant Byte |

## B2BITCNTTh21

Failure threshold for number of BIP-24 B2 bit errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 811 |
| **Power On Value** | X'7D' |

B2BITCNTTh21

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BITCNTTh21(7-0) | Failure threshold for BIP-24 B2 bit error counter Most Significant Byte |

## B2BITCNTTh22

Failure threshold for number of BIP-24 B2 bit errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 810 |
| **Power On Value** | X'00' |

B2BITCNTTh22
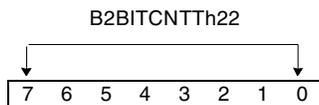
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BITCNTTh22(7-0) | Failure threshold for BIP-24 B2 bit error counter Least Significant Byte |

## B2BLKCNT

Number of BIP-24 B2 block errors counted since last counter reset (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 812/813 |
| **Power On Value** | X'00' |

B2BLKCNT

```
  7   6   5   4   3   2   1   0
```

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BLKCNT(8:15) | BIP-24 B2 block error counter Least Significant Byte |

## B2BLKCNTTh11

Degradation threshold for number of BIP-24 B2 block errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 815 |
| **Power On Value** | X'4E' |

B2BLKCNTTh11

```
  7   6   5   4   3   2   1   0
```

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BLKCNTTh11(7-0) | Degradation threshold for BIP-24 B2 block error counter Most Significant Byte |

## B2BLKCNTTh12

Degradation threshold for number of BIP-24 B2 block errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 814 |
| **Power On Value** | X'20' |

B2BLKCNTTh12

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BLKCNTTh12(7-0) | Degradation threshold for BIP-24 B2 block error counter Least Significant Byte |

## B2BLKCNTTh21

Failure threshold for number of BIP-24 B2 block errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 817 |
| **Power On Value** | X'7D' |

B2BLKCNTTh21
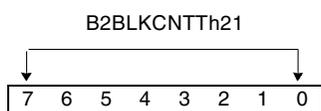
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BLKCNTTh21(7-0) | Failure threshold for BIP-24 B2 block error counter Most Significant Byte |

### B2BLKCNTTh22

Failure threshold for number of BIP-24 B2 block errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 816 |
| **Power On Value** | X'00' |

B2BLKCNTTh22

```
7  6  5  4  3  2  1  0
```

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B2BLKCNTTh22(7-0) | Failure threshold for BIP-24 B2 block error counter Least Significant Byte |

### B3BITCNT

Number of BIP-8 B3 bit errors counted since last counter reset (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 818/819 |
| **Power On Value** | X'00' |

B3BITCNT

```
7  6  5  4  3  2  1  0
```

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B3BITCNT(8:15) | BIP-8 B3 bit error counter Least Significant Byte |

## B3BITCNTTh11

Threshold for number of BIP-8 B3 bit errors.

**Length**        8 bits

**Type**        Read/Write

**Address**        81B

**Power On Value**        X'7D'

B3BITCNTTh11
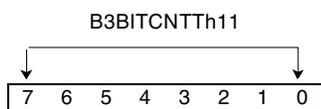
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | B3BITCNTTh11(7-0) | threshold for BIP-8 B3 bit error counter Most Significant Byte |

## B3BITCNTTh12

Threshold for number of BIP-8 B3 bit errors. Threshold overstep leads to an interrupt request.

**Length**        8 bits

**Type**        Read/Write

**Address**        81A
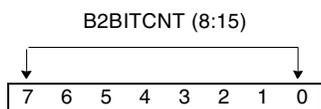
**Power On Value**        X'00'

B3BITCNTTh12
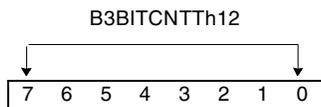
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | B3BITCNTTh12(7-0) | threshold for BIP-8 B3 bit error counter Least Significant Byte |

## B3BLKCNT

Number of BIP-8 B3 block errors counted since last counter reset (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 81C/81D |
| **Power On Value** | X'00' |

B3BLKCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B3BLKCNT(8:15) | BIP-8 B3 block error counter Least Significant Byte |

## B3BLKCNTTh11

Threshold for number of BIP-8 B3 block errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 81F |
| **Power On Value** | X'7D' |

B3BLKCNTTh11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B3BLKCNTTh11(7-0) | Threshold for BIP-8 B3 block error counter Most Significant Byte |

## B3BLKCNTTh12

Threshold for number of BIP-8 B3 block errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 81E |
| **Power On Value** | X'00' |

B3BLKCNTTh12

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | B3BLKCNTTh12(7-0) | Threshold for BIP-8 B3 block error counter Least Significant Byte |

## MSREICNT

Multiplex Section Remote Error Indication counter (16-bit Counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 820/821 |
| **Power On Value** | X'00' |

MSREICNT (8:15)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | MSREICNT(8:15) | Multiplex Section Remote Error Indication counter Least Significant Byte |

**MSREICNTTh11**

Threshold for number of Multiplex Section Remote Errors.

**Length**            8 bits

**Type**              Read/Write

**Address**           823

**Power On Value**    X'7D'

MSREICNTTh11
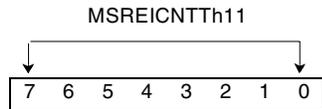
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | MSREICNTTh11(7-0) | Threshold for Multiplex Indication counter Section Remote Error Most Significant Byte |

**MSREICNTTh12**

Threshold for number of Multiplex Section Remote Errors. Threshold overstep leads to an interrupt request.

**Length**            8 bits

**Type**              Read/Write

**Address**           822

**Power On Value**    X'00'

MSREICNTTh12
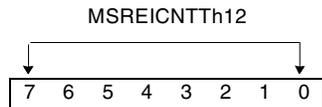
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | MSREICNTTh12(7-0) | Threshold for Multiplex Indication counter Section Remote Error Least Significant Byte |

## HPREICNT

Higher-order Path Remote Error Indication counter (16-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 824/825 |
| **Power On Value** | X'00' |

HPREICNT (8:15)
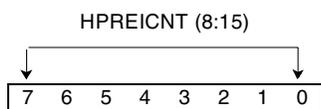
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | HPREICNT(8:15) | Higher-order Path Remote Error Indication counter Least Significant Byte |

## HPREICNTTh11

Threshold for number of Higher-order Path Remote Errors.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 827 |
| **Power On Value** | X'7D' |

HPREICNTTh11
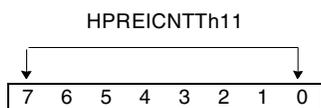
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | HPREICNTTh11(7-0) | Threshold for Higher-order Path Remote Error Indication counter Most Significant Byte |

**HPREICNTTh12**

Threshold for number of Higher-order Path Remote Errors. Threshold overstep leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 826 |
| **Power On Value** | X'00' |

HPREICNTTh12
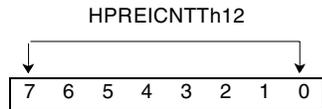
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | HPREICNTTh12(7-0) | Threshold for Higher-order Path Remote Error Indication counter Least Significant Byte |

**PJ_EVCNT**

Positive Justification Event counter (eight-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 828/829 |
| **Power On Value** | X'00' |
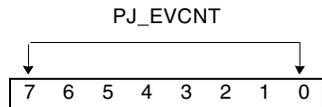
PJ_EVCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-0 | PJ_EVCNT(7-0) | Positive Justification Event counter |

## NJ_EVCNT

Negative Justification Event counter (eight-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 82A/82B |
| **Power On Value** | X'00' |

NJ_EVCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | NJ_EVCNT(7-0) | Negative Justification Event counter |

## ND_EVCNT

New Data Event counter (eight-bit counter). Overflow leads to an interrupt request.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read |
| **Address** | 82C/82D |
| **Power On Value** | X'00' |

ND_EVCNT

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | ND_EVCNT(7-0) | New Data Event counter |

**CntEn1**

Counter On/Off control register #1 for OFP_Rx. For each bit position, 0 = Counter is disabled,
1 = Counter is enabled.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 802 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|---|---|---|
| 7 | EN-HPREICNT | Higher-order Path Remote Error Indication counter enable |
| 6 | EN-MSREICNT | Multiplex Section Remote Error Indication counter enable |
| 5 | EN-B3BLKCNT | BIP-8 B3 block error counter enable |
| 4 | EN-B3BITCNT | BIP-8 B3 bit error counter enable |
| 3 | EN-B2BLKCNT | BIP-24 B2 block error counter enable |
| 2 | EN-B2BITCNT | BIP-24 B2 bit error counter enable |
| 1 | EN-B1BLKCNT | BIP-8 B1 block error counter enable |
| 0 | EN-B1BITCNT | BIP-8 B1 bit error counter enable |
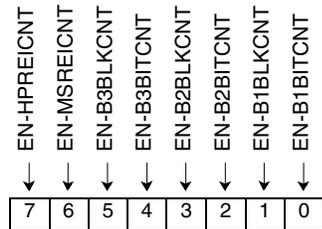
### CntEn2

Counter On/Off control register #2 for OFP_Rx. For each bit position, 0 = Counter is disabled,
1 = Counter is enabled.

**Length**          8 bits

**Type**            Read/Write

**Address**         803

**Power On Value**  X'00'
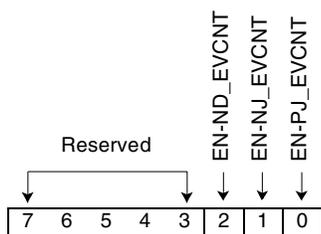
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-3 | Reserved | Reserved |
| 2 | EN-ND_EVCNT | New Data Event counter enable |
| 1 | EN-NJ_EVCNT | Negative Justification Event counter enable |
| 0 | EN-PJ_EVCNT | Positive Justification Event counter enable |

### Reset Register (RESET)

Reset/Halt chiplet control register. This register is automatically preset to the default value by the reset signal
ResOT coming from GPPINT chiplet. For each bit position, 0 = Reset/Halt not active, 1 = Reset/Halt active.

**Length**          8 bits

**Type**            Read/Write

**Address**         830

**Power On Value**  X'01'
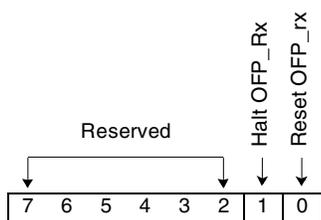
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-2 | Reserved | Reserved |
| 1 | | Halt (freeze) OFP_Rx chiplet |
| 0 | | Reset (disable) OFP_Rx chiplet |

**Status Registers**

**STAT1**

Status register #1 of the chiplet. OFP_Rx Mode status information. This is an event latch register.

| **Length** | 8 bits |
| **Type** | - |
| **Address** | 833 |
| **Power On Value** | - |
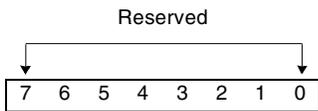
Reserved

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | Reserved | Reserved |

**STAT2**

Status register #2 of the chiplet. AU pointer status information of OFP_Rx. This is an event latch register.

| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 834 |
| **Power On Value** | - |

Reserved / CONCrx / NewPtr / InvPtr / NDF / Pjus / Njus
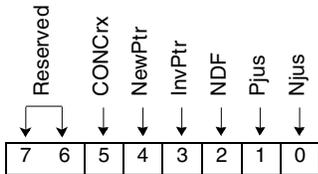
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | CONCrx | Concatenation indication received |
| 4 | NewPtr | Valid New Pointer received |
| 3 | InvPtr | Invalid Pointer received |
| 2 | NDF | NDF received |
| 1 | Pjus | Positive frequency justification received |
| 0 | Njus | Negative frequency justification received |

**STAT3**

Status register #3 of the chiplet. Section Overhead (SOH) status of OFP_Rx. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 835 |
| **Power On Value** | - |



| Bit(s) | Name | Description |
|---|---|---|
| 7-6 | Reserved | Reserved |
| 5 | E1chg | Orderwire channel E1 content changed |
| 4 | E2chg | Orderwire channel E2 content changed |
| 3 | F1chg | User communication channel F1 content changed |
| 2 | D1chg | D1-D3 communication channel content changed |
| 1 | D4chg | D4-D12 communication channel content changed |
| 0 | M1chg | Number of bit blocks in error changed |
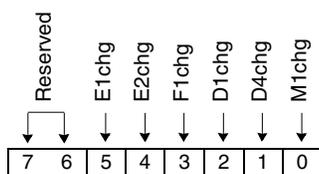
**STAT4**

Status register #4 of the chiplet. Path Overhead (POH) status of OFP_Rx. This is an event latch register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 836 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | Reserved | Reserved |
| 3 | C2chg | Payload composition indication changed |
| 2 | G1chg | Path status indication changed |
| 1 | F2chg | User communication channel F2 content changed |
| 0 | Z5 | Z5 content changed |
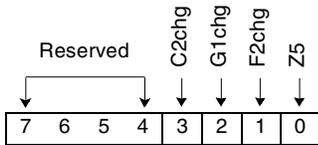
**Interrupt and Mask Registers**

### MainIRQ

Register to indicate fatal interrupt events and to point to user IRQ registers with active requests. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 838 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | Reserved | Reserved |
| 6 | IRQ8 | Active request in IRQ8 register |
| 5 | IRQ7 | Active request in IRQ7 register |
| 4 | IRQ6 | Active request in IRQ6 register |
| 3 | CntrIRQ3 | Active request in CntrIRQ3 register |
| 2 | CntrIRQ2 | Active request in CntrIRQ2 register |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occured |

**M_MainIRQ**

Register to mask pending interrupt requests. A masked request will not generate an outgoing IRQ to the GPPINT. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates signal IRQOR to GPPINT.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 839 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|:---:|:---:|---|
| 7 | Reserved | Reserved |
| 6 | IRQ8 | Active request in IRQ8 register |
| 5 | IRQ7 | Active request in IRQ7 register |
| 4 | IRQ6 | Active request in IRQ6 register |
| 3 | CntrIRQ3 | Active request in CntrIRQ3 register |
| 2 | CntrIRQ2 | Active request in CntrIRQ2 register |
| 1 | CntrIRQ1 | Active request in CntrIRQ1 register |
| 0 | Fatal | Fatal event occurred |
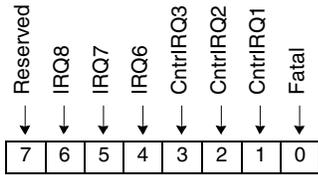
## CntrIRQ1

Register #1 to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83A |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7 | OV-B2BLKCNT | Overflow BIP-24 B2 block error counter |
| 6 | T2-B2BITCNT | Failure threshold overstep BIP-24 B2 bit error counter |
| 5 | TH-B2BITCNT | Degradation threshold overstep BIP-24 B2 bit error counter |
| 4 | OV-B2BITCNT | Overflow BIP-24 B2 bit error counter |
| 3 | TH-B1BLKCNT | Threshold overstep BIP-8 B1 block error counter |
| 2 | OV-B1BLKCNT | Overflow BIP-8 B1 block error counter |
| 1 | TH-B1BITCNT | Threshold overstep BIP-8 B1 bit error counter |
| 0 | OV-B1BITCNT | Overflow BIP-8 B1 bit error counter |

**M_CntrIRQ1**

Register to mask pending counter interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT),1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83B |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | OV-B2BLKCNT | Overflow BIP-24 B2 block error counter |
| 6 | T2-B2BITCNT | Failure threshold overstep BIP-24 B2 bit error counter |
| 5 | TH-B2BITCNT | Degradation threshold overstep BIP-24 B2 bit error counter |
| 4 | OV-B2BITCNT | Overflow BIP-24 B2 bit error counter |
| 3 | TH-B1BLKCNT | Threshold overstep BIP-8 B1 block error counter |
| | 2OV-B1BLKCNT | Overflow BIP-8 B1 block error counter |
| 1 | TH-B1BITCNT | Threshold overstep BIP-8 B1 bit error counter |
| 0 | OV-B1BITCNT | Overflow BIP-8 B1 bit error counter |

**CntrIRQ2**

Register #2 to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83C |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7 | TH-MSREICNT | Threshold overstep Multiplex Section Remote Error Indication counter |
| 6 | OV-MSREICNT | Overflow Multiplex Section Remote Error indication counter |
| 5 | TH-B3BLKCNT | Threshold overstep BIP-8 B3 block error counter |
| 4 | OV-B3BLKCNT | Overflow BIP-8 B3 block error counter |
| 3 | TH-B3BITCNT | Threshold overstep BIP-8 B3 bit error counter |
| 2 | OV-B3BITCNT | Overflow BIP-8 B3 bit error counter |
| 1 | T2-B2BLKCNT | Failure th. overstep BIP-24 B2 block error counter |
| 0 | TH-B2BLKCNT | Degrad. th. overstep BIP-24 B2 block error counter |

**M_CntrIRQ2**

Register to mask pending counter interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT),1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83D |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7 | TH-MSREICNT | Threshold overstep Multiplex Section Remote Error Indication counter |
| 6 | OV-MSREICNT | Overflow Multiplex Section Remote Error indication counter |
| 5 | TH-B3BLKCNT | Threshold overstep BIP-8 B3 block error counter |
| 4 | OV-B3BLKCNT | Overflow BIP-8 B3 block error counter |
| 3 | TH-B3BITCNT | Threshold overstep BIP-8 B3 bit error counter |
| 2 | OV-B3BITCNT | Overflow BIP-8 B3 bit error counter |
| 1 | T2-B2BLKCNT | Failure th. overstep BIP-24 B2 block error counter |
| 0 | TH-B2BLKCNT | Degrad. th. overstep BIP-24 B2 block error counter |

### CntrIRQ3

Register #3 to indicate active counter interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83E |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7-5 | Reserved | Reserved |
| 4 | OV-ND_EVCNT | Overflow New Data event counter |
| 3 | OV-NJ_EVCNT | Overflow Negative Justification event counter |
| 2 | OV-PJ_EVCNT | Overflow Positive Justification event counter |
| 1 | TH-HPREICNT | Threshold overstep Higher-order Path Remote Error indication counter |
| 0 | OV-HPREICNT | Overflow HPR error indication counter |

**M_CntrIRQ3**

Register to mask pending counter interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT),1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 83F |
| **Power On Value** | X'00' |

```
                  OV-ND_EVCNT
                    OV-NJ_EVCNT
                      OV-PJ_EVCNT
                        TH-HPREICNT
                          OV-HPREICNT
Reserved
  ┌──────┐   ↓   ↓   ↓   ↓   ↓
  ↓      ↓
┌───┬───┬───┬───┬───┬───┬───┬───┐
│ 7 │ 6 │ 5 │ 4 │ 3 │ 2 │ 1 │ 0 │
└───┴───┴───┴───┴───┴───┴───┴───┘
```

| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-5 | Reserved | Reserved |
| 4 | OV-ND_EVCNT | Overflow New Data event counter |
| 3 | OV-NJ_EVCNT | Overflow Negative Justification event counter |
| 2 | OV-PJ_EVCNT | Overflow Positive Justification event counter |
| 1 | TH-HPREICNT | Threshold overstep Higher-order Path Remote Error indication counter |
| 0 | OV-HPREICNT | Overflow HPR error indication counter |

### IRQ6

Register to indicate active user interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

**Length**          8 bits
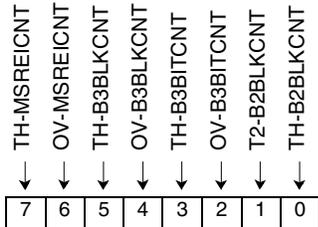
**Type**            Read/Write

**Address**         840

**Power On Value**  -
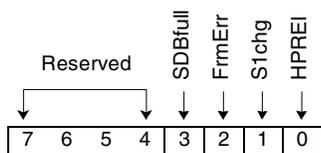
| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | Reserved | Reserved |
| 3 | SDBfull | SDB_Rx FIFO full |
| 2 | FrmErr | Interrupt from ORxAUG FSM |
| 1 | S1chg | Synchronization status changed |
| 0 | HPREI | Higher-order Path Remote Error Indication |

**M_IRQ6**

Register to mask pending user interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT),1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

**Length**          8 bits

**Type**            Read/Write
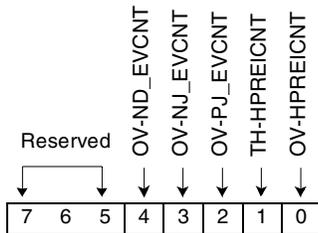
**Address**         841

**Power On Value**  X'00'
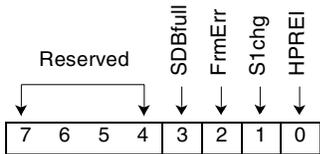


| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | Reserved | Reserved |
| 3 | SDBfull | SDB_Rx FIFO full |
| 2 | FrmErr | Interrupt from ORxAUG FSM |
| 1 | S1chg | Synchronization status changed |
| 0 | HPREI | Higher-order Path Remote Error Indication |

### IRQ7

Register to indicate active user interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 842 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|---|---|---|
| 7 | OOF | Out of frame alarm |
| 6 | LOS | Loss of signal alarm |
| 5 | LOF | Loss of frame alarm |
| 4 | LOP | Loss of pointer alarm |
| 3 | HPAIS | Higher-order path AIS |
| 2 | HPRDI | Higher-order path RDI |
| 1 | UNEQ | Unequipped signal |
| 0 | SLM | Signal label mismatch alarm |
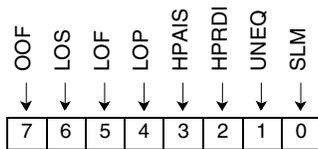
**M_IRQ7**

Register to mask pending user interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT), 1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 843 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|:---:|:---:|:---|
| 7 | OOF | Out of frame alarm |
| 6 | LOS | Loss of signal alarm |
| 5 | LOF | Loss of frame alarm |
| 4 | LOP | Loss of pointer alarm |
| 3 | HPAIS | Higher-order path AIS |
| 2 | HPRDI | Higher-order path RDI |
| 1 | UNEQ | Unequipped signal |
| 0 | SLM | Signal label mismatch alarm |
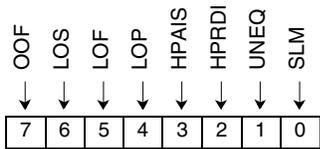
**IRQ8**

Register to indicate active user interrupt requests of this chiplet. For each bit position, 0 = No interrupt request pending, 1 = Interrupt request pending.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 844 |
| **Power On Value** | - |

| Bit(s) | Name | Description |
|:---:|:---:|---|
| 7 | MSAIS | Multiplex Section AIS |
| 6 | MSRDI | Multiplex Section RDI |
| 5-3 | Reserved | Reserved |
| 2 | MSTIM | Multiplex Section trace identifier mismatch |
| 1 | HPTIM | Higher-order path trace identifier mismatch |
| 0 | PtrErr | Pointer processing error |
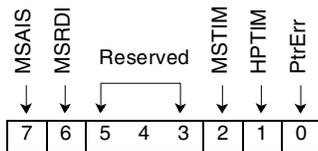
**M_IRQ8**

Register to mask pending user interrupt requests. For each bit position, 0 = The corresponding pending request bit is masked (DEFAULT),1 = The corresponding pending request bit activates the pointer bit in MainIRQ register.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 845 |
| **Power On Value** | X'00' |

| Bit(s) | Name | Description |
|--------|----------|----------------------------------------------|
| 7 | MSAIS | Multiplex Section AIS |
| 6 | MSRDI | Multiplex Section RDI |
| 5-3 | Reserved | Reserved |
| 2 | MSTIM | Multiplex Section trace identifier mismatch |
| 1 | HPTIM | Higher-order path trace identifier mismatch |
| 0 | PtrErr | Pointer processing error |
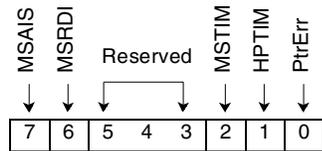
IBM

## Configuration Registers

### CONF1

Configuration register #1. General OFP_Rx configuration signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 848 |
| **Power On Value** | X'3F' |

| Bit(s) | Name | Description | |
|---|---|---|---|
| 7 | ResHunt | 0 | Hunt free running |
| | | 1 | Reset Hunt to PIM |
| 6 | Bellcore | 0 | Operate according to ITU standard |
| | | 1 | Operate according to Bellcore specification |
| 5 | J1GRA | 0 | Do not write J1 section trace to GRA |
| | | 1 | Write J1 section trace to GRA |
| 4 | J0GRA | 0 | Do not write J0 section trace to GRA |
| | | 1 | Write J0 section trace to GRA |
| 3 | FIFOen | 0 | Do not write C4 payload to FIFO |
| | | 1 | Write C4 payload to FIFO |
| 2 | GRAen | 0 | Do not write SOH/POH info to GRA |
| | | 1 | Write received SOH/POH info to GRA |
| 1 | AutRst_Sta | 0 | No action on read access |
| | | 1 | Auto-reset status register upon read access |
| 0 | AutRst_Int | 0 | No action on read access |
| | | 1 | Auto-reset interrupt request registers upon read access |
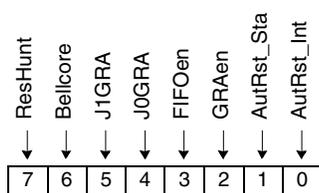
**CONF2**

Configuration register #2. SOH processing configuration signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 849 |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|---|---|---|
| 7-4 | | Reserved |
| 3 | K2en | 0    Disable K2 AIS processing<br>1    Enable K2 AIS processing |
| 2 | S1en | 0    Disable S1 synchronization status processing<br>1    Enable S1 synchronization status processing |
| 1 | M1en | 0    Disable M1 REI processing<br>1    Enable M1 REI processing |
| 0 | 0proc | 0    Disable J0 section trace processing<br>1    Enable J0 section trace processing |
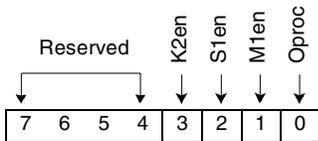
## CONF3

Configuration register #3. POH byte processing configuration signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 84A |
| **Power On Value** | X'00' |



| Bit(s) | Name | Description |
|--------|------|-------------|
| 7-4 | | Reserved |
| 3 | C2en | 0    Disable C2 signal label processing<br>1    Enable C2 signal label processing |
| 2 | G1en | 0    Disable G1 path status processing<br>1    Enable G1 path status processing |
| 1 | J1mode64 | 0    16 byte J1 trace<br>1    64 byte J1 trace |
| 0 | J1proc | 0    Disable J1 path trace processing<br>1    Enable J1 path trace processing |
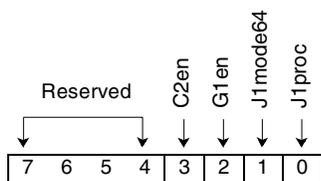
**CONF4**

Configuration register #4. APS processing configuration signals.

**Length**            8 bits

**Type**              Read/Write

**Address**           84B

**Power On Value**    X'00'
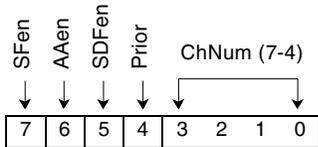
| Bit(s) | Name | Description | |
|--------|------|-------------|---|
| 7 | SFen | 0 | Disable SF K2 MS_RDI processing |
| | | 1 | Enable SF K2 MS_RDI processing |
| 6 | AAen | 0 | Disable automatic Alarm processing for K2 |
| | | 1 | Enable automatic Alarm processing for K2 |
| 5 | SDFen | 0 | Disable automatic SDF K1 processing |
| | | 1 | Enable automatic SDF K1 processing |
| 4 | Prior | Priority level | |
| 3-0 | ChNum(7-4) | Channel Number | |

IBM

### CONF7

Configuration register #7. Miscellaneous OFP_Rx configuration signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 84E |
| **Power On Value** | X'20' |

| Bit(s) | Name | Description | |
|:---:|:---:|:---|:---|
| 7 | EdgeMode | 0 | Active falling edge |
| | | 1 | Active rising edge |
| 6 | ClkMode | 0 | Continuous clock mode |
| | | 1 | Strobed clock mode |
| 5 | OpMode | 0 | DCC 1 channel selected |
| | | 1 | DCC 2 channel selected |
| 4 | DCCen | 0 | Disable DCC processing |
| | | 1 | Enable DCC processing |
| 3-2 | OUTnum(7-6) | Number of µs for in-frame to out-of-frame transition | |
| 1-0 | INnum(7-6) | Number of µs for out-of-frame to in-frame transition | |
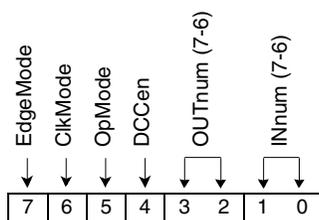
**CONF8**

Configuration register #8. Pattern register signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 84F |
| **Power On Value** | X'FE' |

FSCRrx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | FSCRrx(7-0) | Frame descrambling reload pattern |

**CONF9**

Configuration register #9. Pattern register signals.

| | |
|---|---|
| **Length** | 8 bits |
| **Type** | Read/Write |
| **Address** | 850 |
| **Power On Value** | X'13' |
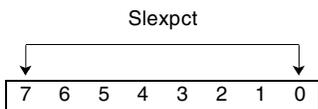
Slexpct

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| Bit(s) | Name | Description |
|---|---|---|
| 7-0 | Slexpct(7-0) | Expected signal label |

## Memory Map for Registers and Arrays

| Address | Entity | Elements Accessed |
|---|---|---|
| XXXX 0000 - FF | PCINT | Registers |
| XXXX 0400 - FF | INTST | Registers |
| XXXX 0500 - FF | CRSET | Registers |
| XXXX 0600 - 7FF | DMAQS | Registers & Array |
| XXXX 0100 - FF | GPDMA | Registers |
| XXXX 0900 - FF | COMET/PAKIT | Registers |
| XXXX 0A00 - FF | CHKSM | Registers |
| XXXX 0B00 - FF | LINKC | Registers |
| XXXX 0C00 - FF | RAALL | Registers |
| XXXX 0D00 - FF | VIMEM | Registers |
| XXXX 0E00 - FF | ARBIT | Registers |
| XXXX 1000 - 1FF | BCACH | Registers & Array |
| XXXX 1200 - 3FF | CSKED | Registers & Array |
| XXXX 1400 - 5FF | SEGBF | Registers & Array |
| XXXX 1600 - 7FF | REASM | Registers & Array |
| XXXX 1800 - FFF | RXQUE | Registers |
| XXXX 2000 - FFF | NPBUS/FRAMR | Registers & External EEPROM |
| XXXX 3000 - BFF | POOLS | Registers & Arrays |
| XXXX 4C00 - FFF | PCORE | Registers & Arrays |

## Printed Circuit Board Considerations

- Connect the BIST0DI1 output to the IBDINH1 input so that the device drivers are driven to high impedance during a device reset or while the IBM2520L8767 BIST is running. Add a pullup resistor to this net.
- A filter circuit should be added to the VDDA input for the PLL.
- Connect MFORCEBP to MBYPASS, with a 1-K pullup on this net.
- Pullups are required on the IBDRINH signal.
- Pulldowns (or grounds) are required on the PPLLTI signal.

IBM

## Pin Assignments and DC Characteristics

This table lists the complete pinout of the IBM2520L8767 by pin name and gives the book name for the electrical driver code associated with the pin. Please refer to *Pinout Viewed from Above* on page 6 for an illustration of the grid positions. Refer to *Book Definitions* on page 530 to learn the DC characteristics for the books.

### Signal Pin Listing by Family  (Page 1 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| MBYPASS | | 0N15 | PLL | IN |
| PVDDA | | 0M17 | PLL | IN |
| PPLLTI | | 0M19 | PLL | IN |
| BIST0DI1 | | 0P03 | F | INOUT |
| CMADDR | CMADDR<17> | 0G01 | C | INOUT |
| | CMADDR<16> | 0K08 | | |
| | CMADDR<15> | 0F02 | | |
| | CMADDR<14> | 0H04 | | |
| | CMADDR<13> | 0G02 | | |
| | CMADDR<12> | 0G03 | | |
| | CMADDR<11> | 0K07 | | |
| | CMADDR<10> | 0J05 | | |
| | CMADDR<9> | 0K06 | | |
| | CMADDR<8> | 0D02 | | |
| | CMADDR<7> | 0H05 | | |
| | CMADDR<6> | 0L05 | | |
| | CMADDR<5> | 0F04 | | |
| | CMADDR<4> | 0D01 | | |
| | CMADDR<3> | 0D03 | | |
| | CMADDR<2> | 0C01 | | |
| | CMADDR<1> | 0B01 | | |
| | CMADDR<0> | 0C02 | | |

## Signal Pin Listing by Family  (Page 2 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| CMCAS | CMCAS<1> | 0C07 | B | INOUT |
| | CMCAS<0> | 0E07 | | |
| CMCLK | | 0D04 | Z | INOUT |
| CMCLKRP | | 0B02 | Z | INOUT |
| CMCS | CMCS<6> | 0F08 | C | INOUT |
| | CMCS<5> | 0G07 | | |
| | CMCS<4> | 0F06 | | |
| CMCS7 | | 0A07 | B | INOUT |

IBM

## Signal Pin Listing by Family (Page 3 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| | CMDATA<35> | 0Y02 | | |
| | CMDATA<34> | 0R09 | | |
| | CMDATA<33> | 0W01 | | |
| | CMDATA<32> | 0U03 | | |
| | CMDATA<31> | 0R07 | | |
| | CMDATA<30> | 0V02 | | |
| | CMDATA<29> | 0T04 | | |
| | CMDATA<28> | 0T02 | | |
| | CMDATA<27> | 0R03 | | |
| | CMDATA<26> | 0P10 | | |
| | CMDATA<25> | 0U01 | | |
| | CMDATA<24> | 0R02 | | |
| | CMDATA<23> | 0R01 | | |
| | CMDATA<22> | 0P08 | | |
| | CMDATA<21> | 0P04 | | |
| | CMDATA<20> | 0N03 | | |
| | CMDATA<19> | 0P07 | | |
| CMDATA | CMDATA<18> | 0N09 | H | INOUT |
| | CMDATA<17> | 0P06 | | |
| | CMDATA<16> | 0P02 | | |
| | CMDATA<15> | 0R05 | | |
| | CMDATA<14> | 0N01 | | |
| | CMDATA<13> | 0N05 | | |
| | CMDATA<12> | 0M02 | | |
| | CMDATA<11> | 0M04 | | |
| | CMDATA<10> | 0M07 | | |
| | CMDATA<9> | 0L01 | | |
| | CMDATA<8> | 0L02 | | |
| | CMDATA<7> | 0J01 | | |
| | CMDATA<6> | 0M08 | | |
| | CMDATA<5> | 0L03 | | |
| | CMDATA<4> | 0K02 | | |
| | CMDATA<2> | 0K04 | | |
| | CMDATA<2> | 0H02 | | |
| | CMDATA<1> | 0L07 | | |

## Signal Pin Listing by Family  (Page 4 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| CMDATA2 | CMDATA2<38> | 0W03 | H | INOUT |
|  | CMDATA2<37> | 0W02 |  |  |
|  | CMDATA2<36> | 0V04 |  |  |
| CMRAS | CMRAS<1> | 0A05 | B | INOUT |
|  | CMRAS<2> | 0C05 |  |  |
| CMSYNCAS |  | 0M06 | C | INOUT |
| CMSYNRAS |  | 0N07 | C | INOUT |
| CMWE | CMWE<1> | 0C09 | B | INOUT |
|  | CMWE<0> | 0E09 |  |  |
| CTS |  | 0K05 | K | INOUT |
| DSR |  | 0P01 | K | INOUT |
| DTR |  | 0K03 | D | INOUT |

## Signal Pin Listing by Family (Page 5 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| | ENSTATE<63> | 0H10 | | |
| | ENSTATE<62> | 0K13 | | |
| | ENSTATE<61> | 0H16 | | |
| | ENSTATE<60> | 0K18 | | |
| | ENSTATE<59> | 0H18 | | |
| | ENSTATE<58> | 0L17 | | |
| | ENSTATE<57> | 0R18 | | |
| | ENSTATE<56> | 0V18 | | |
| | ENSTATE<55> | 0T18 | | |
| | ENSTATE<54> | 0W18 | | |
| | ENSTATE<53> | 0P12 | | |
| | ENSTATE<52> | 0U17 | | |
| | ENSTATE<51> | 0W17 | | |
| | ENSTATE<50> | 0R13 | | |
| | ENSTATE<49> | 0V16 | | |
| | ENSTATE<48> | 0T13 | | |
| ENSTATE | ENSTATE<47> | 0Y17 | G | INOUT |
| | ENSTATE<46> | 0Y19 | | |
| | ENSTATE<45> | 0T14 | | |
| | ENSTATE<44> | 0V15 | | |
| | ENSTATE<43> | 0V13 | | |
| | ENSTATE<42> | 0V12 | | |
| | ENSTATE<41> | AE17 | | |
| | ENSTATE<40> | AD16 | | |
| | ENSTATE<39> | 0V14 | | |
| | ENSTATE<38> | 0U13 | | |
| | ENSTATE<37> | AB15 | | |
| | ENSTATE<36> | AB16 | | |
| | ENSTATE<35> | 0W11 | | |
| | ENSTATE<34> | AE16 | | |
| | ENSTATE<33> | AD15 | | |
| | ENSTATE<32> | 0Y14 | | |

## Signal Pin Listing by Family  (Page 6 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| FY0EMP | | 0C10 | L | INOUT |
| FY0FUL | | 0L09 | L | INOUT |
| FY0RENB | | 0M18 | E | INOUT |
| FY0TENB | | 0N19 | E | INOUT |
| FYDISCRD | | 0K12 | J | INOUT |
| FYDTCT | | 0G17 | L | INOUT |
| FYRCA | | 0K16 | L | INOUT |
| FYRDAT | FYRDAT<11> | 0P15 | W | IN |
| | FYRDAT<10> | 0T15 | | |
| | FYRDAT<9> | 0T17 | | |
| | FYRDAT<8> | 0T19 | | |
| | FYRDAT<7> | 0V17 | | |
| | FYRDAT<6> | 0V19 | | |
| | FYRDAT<5> | AC15 | | |
| | FYRDAT<4> | AE07 | | |
| | FYRDAT<3> | AA13 | | |
| | FYRDAT<2> | AC13 | | |
| | FYRDAT<1> | AE13 | | |
| | FYRDAT<0> | AA11 | | |
| FYRDATB | FYRDATB<15> | 0M12 | L | INOUT |
| | FYRDATB<14> | 0M14 | | |
| | FYRDATB<13> | 0L19 | | |
| | FYRDATB<12> | 0M16 | | |
| FYRPAR | FYRPAR<1> | 0M13 | L | INOUT |
| | FYRPAR<0> | 0L18 | | |
| FYRRDB | | 0N17 | E | INOUT |
| FYRSCLKN | | 0E19 | N | INOUT |
| FYRSCLKP | | 0C19 | N | INOUT |
| FYRSDATP | | 0J19 | N | INOUT |
| FYRSDATN | | 0G19 | N | INOUT |
| FYRSOC | | 0L15 | J | INOUT |
| FYSETCLP | | 0K10 | J | INOUT |
| FYTCA | | 0F18 | L | INOUT |

## Signal Pin Listing by Family (Page 7 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| FYTDAT | FYTDAT<15> | 0N13 | G | INOUT |
| | FYTDAT<14> | 0P14 | | |
| | FYTDAT<13> | 0T16 | | |
| | FYTDAT<12> | 0P16 | | |
| | FYTDAT<11> | 0N11 | | |
| | FYTDAT<10> | 0R19 | | |
| | FYTDAT<9> | 0P13 | | |
| | FYTDAT<8> | 0R15 | | |
| | FYTDAT<7> | 0R17 | | |
| | FYTDAT<6> | 0Y18 | | |
| | FYTDAT<5> | 0U15 | | |
| | FYTDAT<4> | AB18 | | |
| | FYTDAT<3> | 0Y16 | | |
| | FYTDAT<2> | AB19 | | |
| | FYTDAT<1> | AB17 | | |
| | FYTDAT<0> | AA17 | | |
| FYTPAR | FYTPAR<1> | AD19 | G | INOUT |
| | FYTPAR<0> | AC18 | | |
| FYTSCLKN | | AC19 | N | INOUT |
| FYTSCLKP | | AA19 | N | INOUT |
| FYTSDATP | | 0W19 | P | INOUT |
| FYTSDATN | | 0U19 | P | INOUT |
| FYTSOC | | 0Y15 | G | INOUT |
| FYTWRB | | 0P18 | E | INOUT |
| IBDINH1 | | 0V01 | Q | IN |
| IBDINH2 | | 0V03 | R | IN |
| IBDRINH | | 0K01 | T | IN |
| JTAG0RST | | 0K17 | W | IN |
| JTAG0RSTC | | 0K15 | K | IN |
| JTAGTCK | | 0H17 | K | IN |
| JTAGTCKC | | 0C15 | K | IN |
| JTAGTDI | | 0C11 | W | IN |
| JTAGTDIC | | 0E11 | W | IN |
| JTAGTDO | | 0A13 | F | INOUT |

## Signal Pin Listing by Family (Page 8 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| JTAGTDOC | | 0C13 | F | INOUT |
| JTAGTMS | | 0A15 | K | IN |
| JTAGTMSC | | 0E13 | K | IN |
| LEAKTST | | 0K19 | V | IN |
| MACK64 | | 0D17 | A | INOUT |
| MDEVSEL | | 0D10 | A | INOUT |
| MFORCEBP | | 0H19 | F | INOUT |
| MFRAME | | 0L11 | A | INOUT |
| MGNT | | 0D14 | M | INOUT |
| MHALT401 | | 0T01 | K | IN |
| MINT2 | | 0F15 | A | INOUT |
| MINTA | | 0E15 | A | INOUT |
| MIRDY | | 0D11 | A | INOUT |
| MLOCK | | 0C17 | M | INOUT |
| MPCIRST | | 0A11 | X | INOUT |
| MPEGCLK | | AE09 | W | IN |
| MPERR | | 0D09 | A | INOUT |
| MREQ | | 0D13 | A | INOUT |
| MREQ64 | | 0B11 | A | INOUT |
| MSERR | | 0A08 | A | INOUT |
| MSTOP | | 0F10 | A | INOUT |
| MTRDY | | 0F11 | A | INOUT |
| NSELFT | | 0P05 | S | IN |
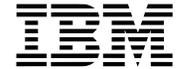
## Signal Pin Listing by Family (Page 9 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| PAD | PAD<31> | 0F17 | A | INOUT |
| | PAD<30> | 0F19 | | |
| | PAD<29> | 0K14 | | |
| | PAD<28> | 0H15 | | |
| | PAD<27> | 0H13 | | |
| | PAD<26> | 0A18 | | |
| | PAD<25> | 0G15 | | |
| | PAD<24> | 0B17 | | |
| | PAD<23> | 0E16 | | |
| | PAD<22> | 0J13 | | |
| | PAD<21> | 0A17 | | |
| | PAD<20> | 0B16 | | |
| | PAD<19> | 0H14 | | |
| | PAD<18> | 0H12 | | |
| | PAD<17> | 0D15 | | |
| | PAD<16> | 0D16 | | |
| | PAD<15> | 0G11 | | |
| | PAD<14> | 0A16 | | |
| | PAD<13> | 0B15 | | |
| | PAD<12> | 0F14 | | |
| | PAD<11> | 0H11 | | |
| | PAD<10> | 0C14 | | |
| | PAD<9> | 0G13 | | |
| | PAD<8> | 0A14 | | |
| | PAD<7> | 0F13 | | |
| | PAD<6> | 0F12 | | |
| | PAD<5> | 0K11 | | |
| | PAD<4> | 0B13 | | |
| | PAD<3> | 0D12 | | |
| | PAD<2> | 0J11 | | |
| | PAD<1> | 0A12 | | |
| | PAD<0> | 0M10 | | |

## Signal Pin Listing by Family

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| PAD64 | PAD64<63> | 0F09 | A | INOUT |
| | PAD64<62> | 0K09 | | |
| | PAD64<61> | 0B08 | | |
| | PAD64<60> | 0B07 | | |
| | PAD64<59> | 0H09 | | |
| | PAD64<58> | 0D08 | | |
| | PAD64<57> | 0D07 | | |
| | PAD64<56> | 0A06 | | |
| | PAD64<55> | 0G09 | | |
| | PAD64<54> | 0F07 | | |
| | PAD64<53> | 0C06 | | |
| | PAD64<52> | 0H08 | | |
| | PAD64<51> | 0D06 | | |
| | PAD64<50> | 0B05 | | |
| | PAD64<49> | 0A04 | | |
| | PAD64<48> | 0H07 | | |
| | PAD64<47> | 0E05 | | |
| | PAD64<46> | 0D05 | | |
| | PAD64<45> | 0J07 | | |
| | PAD64<44> | 0F05 | | |
| | PAD64<43> | 0B04 | | |
| | PAD64<42> | 0A03 | | |
| | PAD64<41> | 0H06 | | |
| | PAD64<40> | 0E04 | | |
| | PAD64<39> | 0B03 | | |
| | PAD64<38> | 0G05 | | |
| | PAD64<37> | 0C03 | | |
| | PAD64<36> | 0A02 | | |
| | PAD64<35> | 0F03 | | |
| | PAD64<34> | 0F01 | | |
| | PAD64<33> | 0E03 | | |
| | PAD64<32> | 0E01 | | |
| PB0EPRM | | AB12 | E | INOUT |
| PB0PHY1 | | AD13 | E | INOUT |

## Signal Pin Listing by Family (Page 11 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| PB0PHY2 | | AD18 | E | INOUT |
| PBADDR16 | | AD09 | E | INOUT |
| PBADDR17 | | 0U11 | E | INOUT |
| PBALE1 | | 0T11 | E | INOUT |
| PBALE2 | | AE12 | E | INOUT |
| PBDATA | PBDATA<7> | 0V11 | M | INOUT |
| | PBDATA<6> | AC14 | | |
| | PBDATA<5> | 0W13 | | |
| | PBDATA<4> | 0T12 | | |
| | PBDATA<3> | AE14 | | |
| | PBDATA<2> | 0Y13 | | |
| | PBDATA<1> | 0Y12 | | |
| | PBDATA<0> | 0V10 | | |
| PBINTRA | | AD17 | L | INOUT |
| PBPHYRST | | AC10 | E | INOUT |
| PBRDRDY | | 0W15 | L | INOUT |
| PBRNWRT | | AD11 | E | INOUT |
| PBSCLK | | 0R11 | E | INOUT |
| PBSDATA | | AE18 | M | INOUT |
| PCBE | PCBE<3> | 0D19 | A | INOUT |
| | PCBE<2> | 0F16 | | |
| | PCBE<1> | 0E17 | | |
| | PCBE<0> | 0B18 | | |
| PCBE64 | PCBE64<7> | 0D18 | A | INOUT |
| | PCBE64<6> | 0A10 | | |
| | PCBE64<5> | 0B09 | | |
| | PCBE64<4> | 0J15 | | |
| PCICLK | | 0A09 | X | IN |
| PDBLCLK | | 0L13 | E | INOUT |
| PFFCFG | PFFCFG<2> | 0M15 | W | IN |
| | PFFCFG<1> | 0P17 | | |
| | PFFCFG<0> | 0P19 | | |
| PFFOSC | | 0T05 | W | IN |
| PIDSEL | | 0B12 | M | INOUT |

## Signal Pin Listing by Family (Page 12 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| PINTCLK | | 0J17 | E | INOUT |
| PMADDR | PMADDR<17> | AC03 | C | INOUT |
| | PMADDR<16> | AD02 | | |
| | PMADDR<16> | AC02 | | |
| | PMADDR<16> | AD01 | | |
| | PMADDR<16> | AC01 | | |
| | PMADDR<16> | AB03 | | |
| | PMADDR<16> | AB01 | | |
| | PMADDR<16> | 0Y04 | | |
| | PMADDR<16> | 0T08 | | |
| | PMADDR<16> | AA03 | | |
| | PMADDR<16> | AA01 | | |
| | PMADDR<16> | 0V05 | | |
| | PMADDR<16> | AB02 | | |
| | PMADDR<16> | 0T06 | | |
| | PMADDR<16> | 0Y01 | | |
| | PMADDR<16> | 0Y03 | | |
| | PMADDR<16> | 0U05 | | |
| | PMADDR<16> | 0T07 | | |
| PMCAS | PMCAS<1> | AC07 | B | INOUT |
| | PMCAS<0> | AA07 | | |
| PMCLK | | AA15 | Z | INOUT |
| PMCLKRP | | AA16 | Z | INOUT |
| PMCS | PMCS<6> | AB13 | C | INOUT |
| | PMCS<5> | AD12 | | |
| | PMCS<4> | AB14 | | |
| PMCS7 | | AE15 | B | INOUT |

## Signal Pin Listing by Family (Page 13 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| | PMDATA<35> | AC17 | | |
| | PMDATA<34> | 0T10 | | |
| | PMDATA<33> | 0U09 | | |
| | PMDATA<32> | AB09 | | |
| | PMDATA<31> | AD08 | | |
| | PMDATA<30> | 0T09 | | |
| | PMDATA<29> | AE08 | | |
| | PMDATA<28> | AB08 | | |
| | PMDATA<27> | AD07 | | |
| | PMDATA<26> | 0V09 | | |
| | PMDATA<25> | 0Y08 | | |
| | PMDATA<24> | AB07 | | |
| | PMDATA<23> | 0Y07 | | |
| | PMDATA<22> | AE06 | | |
| | PMDATA<21> | 0W09 | | |
| | PMDATA<20> | 0W07 | | |
| | PMDATA<19> | AC06 | | |
| | PMDATA<18> | 0Y09 | H | INOUT |
| PMDATA | PMDATA<17> | AB06 | | |
| | PMDATA<16> | 0Y06 | | |
| | PMDATA<15> | AD05 | | |
| | PMDATA<14> | AE04 | | |
| | PMDATA<13> | 0V08 | | |
| | PMDATA<12> | AA05 | | |
| | PMDATA<11> | AB04 | | |
| | PMDATA<10> | AB05 | | |
| | PMDATA<9> | 0U07 | | |
| | PMDATA<8> | 0Y05 | | |
| | PMDATA<7> | 0V06 | | |
| | PMDATA<6> | AD04 | | |
| | PMDATA<5> | AE03 | | |
| | PMDATA<4> | 0V07 | | |
| | PMDATA<3> | AA04 | | |
| | PMDATA<2> | AD03 | | |
| | PMDATA<1> | 0W05 | | |

## Signal Pin Listing by Family (Page 14 of 14)

| Pin Family Name | Pin Name | Grid Position | Book Name | Input/Output |
|---|---|---|---|---|
| PMDATA2 | PMDATA2<38> | 0Y10 | H | INOUT |
| | PMDATA2<37> | AE10 | | |
| | PMDATA2<36> | AB10 | | |
| PMRAS | PMRAS<1> | AE05 | B | INOUT |
| | PMRAS<0> | AC05 | | |
| PMSYNCAS | | AB11 | C | INOUT |
| PMSYNRAS | | 0Y11 | C | INOUT |
| PMWE | PMWE<1> | AC09 | B | INOUT |
| | PMWE<0> | AA09 | B | INOUT |
| PPAR | | 0G18 | A | INOUT |
| PPAR64 | | 0J09 | A | INOUT |
| PPLLOUT | | 0C18 | G | INOUT |
| RTS | | 0M05 | D | INOUT |
| RXCLK | | AE11 | W | IN |
| RXD | | 0M03 | W | IN |
| TESTCT | | 0H03 | S | IN |
| TESTM | | 0H01 | S | IN |
| TMMCORE | | 0T03 | S | IN |
| TXCLK | | AC11 | W | IN |
| TXD | | 0M01 | D | INOUT |

IBM

## Book Definitions (Page 1 of 2)

| Book Name | Instances | Description | $V_{IL}$ (V) Min | $V_{IL}$ (V) Max | $V_{IH}$ (V) Min | $V_{IH}$ (V) Max | $I_{IL}$ (@0V) Max | $I_{IH}$ (@$V_{DD}$) Max | $V_{OH}$ (V) Min | $V_{OL}$ (V) Min | $I_{OH}$ (mA) Min | $I_{OL}$ (mA) Min | Note |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 86 | -5.0V-Tolerant PCI Non-Test Three-state CIO | 0.0 | 0.8 | 2.0 | 5.5 | >0 | <0 | 2.4 | 0.5 | - | - | |
| B | 14 | -3.3V LVTTL Test 35 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | 2.4 | 0.4 | 13.0/12.0 | 9.0 | |
| C | 51 | -3.3V LVTTL Non-Test 35 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | 2.4 | 0.4 | 13.0/12.0 | 9.0 | |
| D | 3 | -3.3V LVTTL Test 50 Ohm Three-State CIO w/Pull-up | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | -250µA | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| E | 16 | -3.3V LVTTL Non-Test 50 Ohm Three-State CIO w/Pull-up | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | -250µA | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| F | 4 | -3.3V LVTTL Test 50 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| G | 52 | -3.3V LVTTL Non-Test 50 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| H | 77 | -5.0V-Tolerant LVTTL Non-Test 50 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 0 | 2.4 | 0.4 | 13.0/12.0 | 9.0 | |
| J | 3 | -5.0V-Tolerant LVTTL Non-Test 50 Ohm Three-State CIO w/Pull-down | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 400µA | 2.4 | 0.4 | 10.0 | 7.0 | |
| K | 8 | -5.0V-Tolerant LVTTL Test 50 Ohm Three-State CIO w/Pull-up | 0.0 | 0.8 | 2.0 | 5.5 | -250µA | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| L | 13 | -5.0V-Tolerant LVTTL Non-Test 50 Ohm Three-State CIO w/Pull-up | 0.0 | 0.8 | 2.0 | 5.5 | -250µA | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| M | 12 | -5.0V-Tolerant LVTTL Non-Test 50 Ohm Three-State CIO | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 0 | 2.4 | 0.4 | 10.0 | 7.0 | |
| N | 3 | -3.3V STI Non-Test Differential Receiver | -0.5 | $V_{REF}$+0.5 | $V_{REF}$+0.5 | $V_{DDQ}$+0.3 | 0 | 0 | - | - | - | - | 1 |
| P | 1 | -3.3V STI Non-Test Differential Driver | $V_{OH}$ minimum voltage: $V_{DDQ}$ = -0.4V  $V_{OL}$ maximum voltage = 0.4V | | | | | | - | - | 10.0 | 7.0 | 1 |
| Q | 1 | -3.3V LVTTL Test DI1 Receiver w/Pull-up | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 400µA | - | - | - | - | |
| R | 1 | -3.3V LVTTL Test DI2 Receiver w/Pull-up | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 400µA | - | - | - | - | |
| S | 4 | -3.3V LVTTL Test Receiver w/Pull-up | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 400µA | - | - | - | - | |

1. $V_{REF}$ = Differential Input Voltage. $V_{DDQ}$ = Output Supply Voltage.

## Book Definitions  (Page 2 of 2)

| Book Name | Instances | Description | $V_{IL}$ (V) Min | $V_{IL}$ (V) Max | $V_{IH}$ (V) Min | $V_{IH}$ (V) Max | $I_{IL}$ (@0V) Max | $I_{IH}$ (@$V_{DD}$) Max | $V_{OH}$ (V) Min | $V_{OL}$ (V) Min | $I_{OH}$ (mA) Min | $I_{OL}$ (mA) Min | Note |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | 1 | -3.3V LVTTL Test RI Receiver | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 400µA | - | - | - | - | |
| V | 1 | -5.0V-Tolerant LVTTL Test LT w/Pull-up | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 400µA | - | - | - | - | |
| W | 23 | -5.0V-Tolerant LVTTL Test Receiver w/Pull-up | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 400µA | - | - | - | - | |
| X | 2 | -5.0V-Tolerant LVTTL Test Receiver | 0.0 | 0.8 | 2.0 | 5.5 | 0 | 0 | - | - | - | - | |
| Z | 4 | -3.3V Non-Test 20 Ohm | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | 2.4 | 0.4 | 23.0/12.0 | 16.0 | |
| PLL | 3 | PLL (Phase Locked Loop that Connects at I/O Pads) | 0.0 | 0.8 | 2.0 | $V_{DD}$+0.6 | 0 | 0 | - | - | - | - | |

1. $V_{REF}$ = Differential Input Voltage. $V_{DDQ}$ = Output Supply Voltage.

# AC Timing Characteristics

## PHY Timing

| Description | Min | Max | Units |
|---|---|---|---|
| FYTWRB High to FYTDAT[15:0] | | 20 | ns |
| FYTWRB High to FYTPAR[1:0] | | 20 | ns |
| FYTWRB High to FYTSOC | | 20 | ns |
| FYTWRB High to FY0TENB | | 20 | ns |
| FYTCA to FYTWRB Setup | 9 | -- | ns |
| FYTCA to FYTWRB Hold | 0 | -- | ns |
| FY0FUL to FYTWRB Setup | 9 | -- | ns |
| FY0FUL to FYTWRB Hold | 0 | -- | ns |
| FYRRDB High to FYRENB | | 20 | ns |
| FYRDAT[15:0] to FYRRDB Setup | 9 | -- | ns |
| FYRDAT[15:0] to FYRRDB Hold | 0 | -- | ns |
| FYRPAR[1:0] to FYRRDB Setup | 9 | -- | ns |
| FYRPAR[1:0] to FYRRDB Hold | 0 | -- | ns |
| FYRSOC to FYRRDB Setup | 9 | -- | ns |
| FYRSOC to FYRRDB Hold | 0 | -- | ns |
| FY0FUL to FYRRDB Setup | 9 | -- | ns |
| FY0FUL to FYRRDB Hold | 0 | -- | ns |
| FY0EMP to FYRRDB Setup | 9 | -- | ns |
| FY0EMP to FYRRDB Hold | 0 | -- | ns |
| FY0DISCRD to FYRRDB Setup | 9 | -- | ns |
| FY0DISCRD to FYRRDB Hold | 0 | -- | ns |
| FY0SETCLP to FYRRDB Setup | 9 | -- | ns |
| FY0SETCLP to FYRRDB Hold | 0 | -- | ns |
| FY0DTCT to FYRRDB Setup | 9 | -- | ns |
| FY0DTCT to FYRRDB Hold | 0 | -- | ns |

## NPBUS Sideband Interface Timing

### NPBUS Timing

| Description | Min | Max | Units |
|---|---|---|---|
| REFCLK High to PBDATA[7:0] | | 35 | ns |
| REFCLK High to PBDATAP | | 35 | ns |
| REFCLK High to PBRNWRT | | 15 | ns |
| REFCLK High to PBRDRDY | | 25 | ns |
| REFCLK High to PBADDR[15:0] | | 30 | ns |
| PBDATA[7:0] to REFCLK Setup | 30 | -- | ns |
| PBDATA[7:0] to REFCLK Hold | 0 | -- | ns |
| PBDATAP to REFCLK Setup | 30 | -- | ns |
| PBDATAP to REFCLK Hold | 0 | -- | ns |
| PBRDRDY to REFCLK Setup | 15 | -- | ns |
| PBRDRDY to REFCLK Hold | 0 | -- | ns |
| PBRNWRT to REFCLK Setup | 15 | -- | ns |
| PBRNWRT to REFCLK Hold | 0 | -- | ns |

## I/O PCI Bus Timing

### PCI Bus Timing  (Page 1 of 2)

| Description | Min | Max | Units |
|---|---|---|---|
| REFCLK High to PAD[31:0] | 2 | 11 | ns |
| REFCLK High to PPAR | 2 | 11 | ns |
| REFCLK High to PCBE[3:0] | 2 | 11 | ns |
| REFCLK High to MFRAME | 2 | 11 | ns |
| REFCLK High to MTRDY | 2 | 11 | ns |
| REFCLK High to MIRDY | 2 | 11 | ns |
| REFCLK High to MSTOP | 2 | 11 | ns |
| REFCLK High to MDEVSEL | 2 | 11 | ns |
| REFCLK High to U0REQ | 2 | 11 | ns |
| REFCLK High to MPERR | 2 | 11 | ns |
| REFCLK High to MSERR | 2 | 11 | ns |
| REFCLK High to MINITA | 2 | 11 | ns |
| REFCLK High to U0INT2 | 2 | 12 | ns |
| REFCLK High to ULE0BE | 2 | 11 | ns |

## PCI Bus Timing  (Page 2 of 2)

| Description | Min | Max | Units |
|---|---|---|---|
| REFCLK High to U0DIR | 2 | 11 | ns |
| MFRAME to REFCLK Setup | 7 | -- | ns |
| MFRAME to REFCLK Hold | 0 | -- | ns |
| PCBE[3:0] to REFCLK Setup | 7 | -- | ns |
| PCBE[3:0] to REFCLK Hold | 0 | -- | ns |
| PAD[31:0] to REFCLK Setup | 7 | -- | ns |
| PAD[31:0] to REFCLK Hold | 0 | -- | ns |
| PPAR to REFCLK Setup | 7 | -- | ns |
| PPAR to REFCLK Hold | 0 | -- | ns |
| MPERR to REFCLK Setup | 7 | -- | ns |
| MPERR to REFCLK Hold | 0 | -- | ns |
| PIDSEL to REFCLK Setup | 7 | -- | ns |
| PIDSEL to REFCLK Hold | 0 | -- | ns |
| MDEVSEL to REFCLK Setup | 7 | -- | ns |
| MDEVSEL to REFCLK Hold | 0 | -- | ns |
| MTRDY to REFCLK Setup | 7 | -- | ns |
| MTRDY to REFCLK Hold | 0 | -- | ns |
| MIRDY to REFCLK Setup | 7 | -- | ns |
| MIRDY to REFCLK Hold | 0 | -- | ns |
| MLOCK to REFCLK Setup | 7 | -- | ns |
| MLOCK to REFCLK Hold | 0 | -- | ns |
| MSTOP to REFCLK Setup | 7 | -- | ns |
| MSTOP to REFCLK Hold | 0 | -- | ns |
| U0EXTIO to REFCLK Setup | 27 | -- | ns |
| U0EXTIO to REFCLK Hold | 0 | -- | ns |
| U0EXTPM to REFCLK Setup | 27 | -- | ns |
| U0EXTPM to REFCLK Hold | 0 | -- | ns |
| U0GNT to REFCLK Setup | 10 | -- | ns |
| U0GNT to REFCLK Hold | 0 | -- | ns |

## Synchronous DRAM Timing

### SDRAM Read Cycle (1 of 4)

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | |
|---|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | | |
| CMSRAS | | | | | | | | |
| CAMRAS0 | | | | | | | | |
| CMSCAS | | | | | | | | |
| CMWE0 | | | | | | | | |
| CMCAS0 | | | | | | | | |
| CMADDR | | Row | | Column | | | | |
| CMDATA | | | | | | | Data Out | |
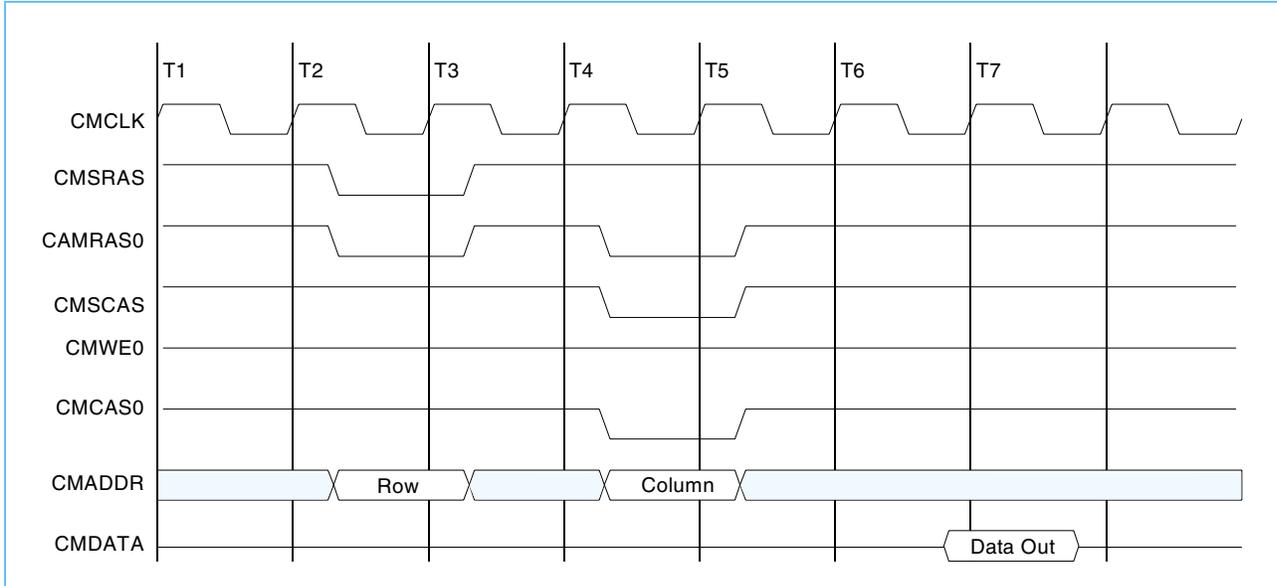
### SDRAM Read Cycle (2 of 4)

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
|---|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | | |
| CMSRAS | | | | | | | | |
| CAMRAS0 | | | | | | | | |
| CMSCAS | | | | | | | | |
| CMWE0 | | | | | | | | |
| CMCAS0 | | | | | | | | |
| CMADDR | | Row | | Column | | | | |
| CMDATA | | | | | | Data Out | | |
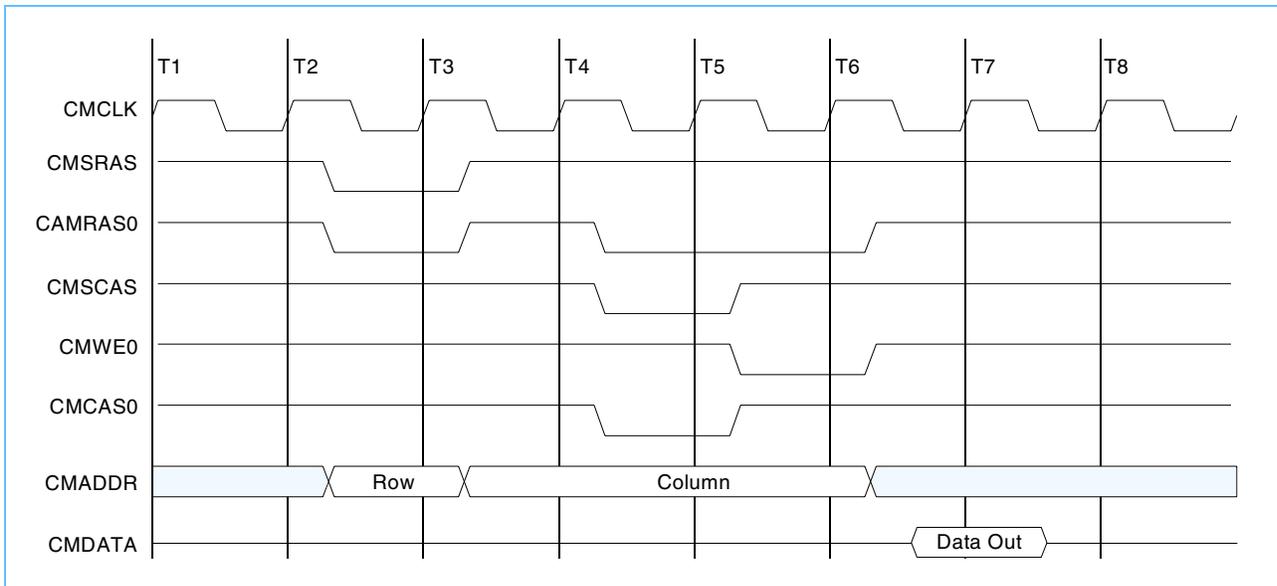
## SDRAM Read Cycle  (3 of 4)
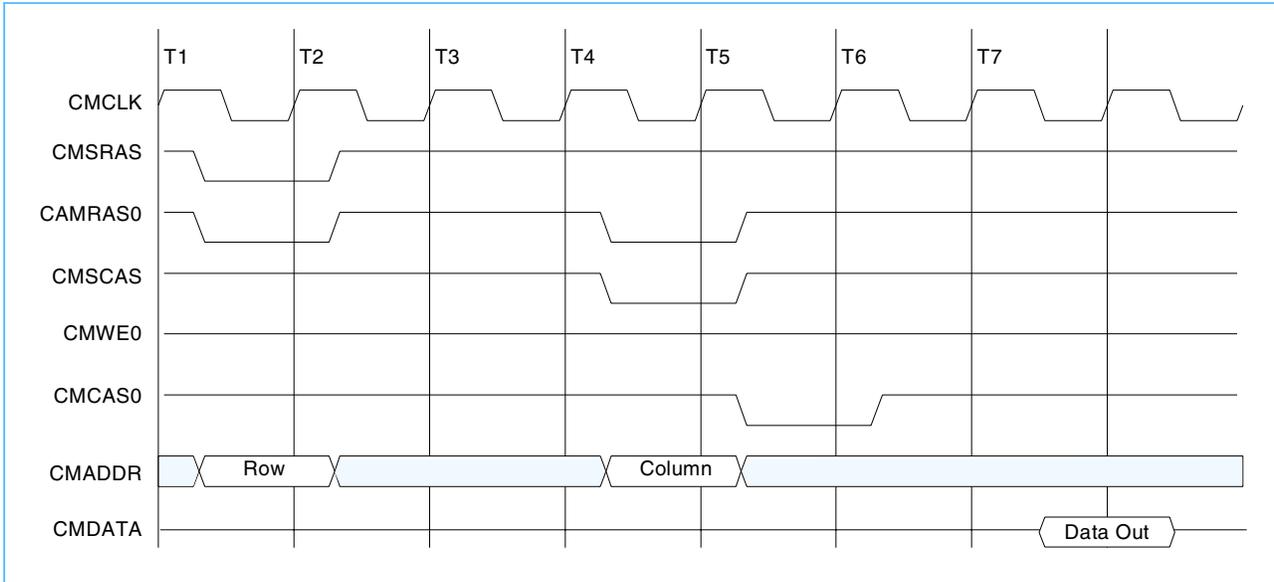


## SDRAM Read Cycle (4 of 4)
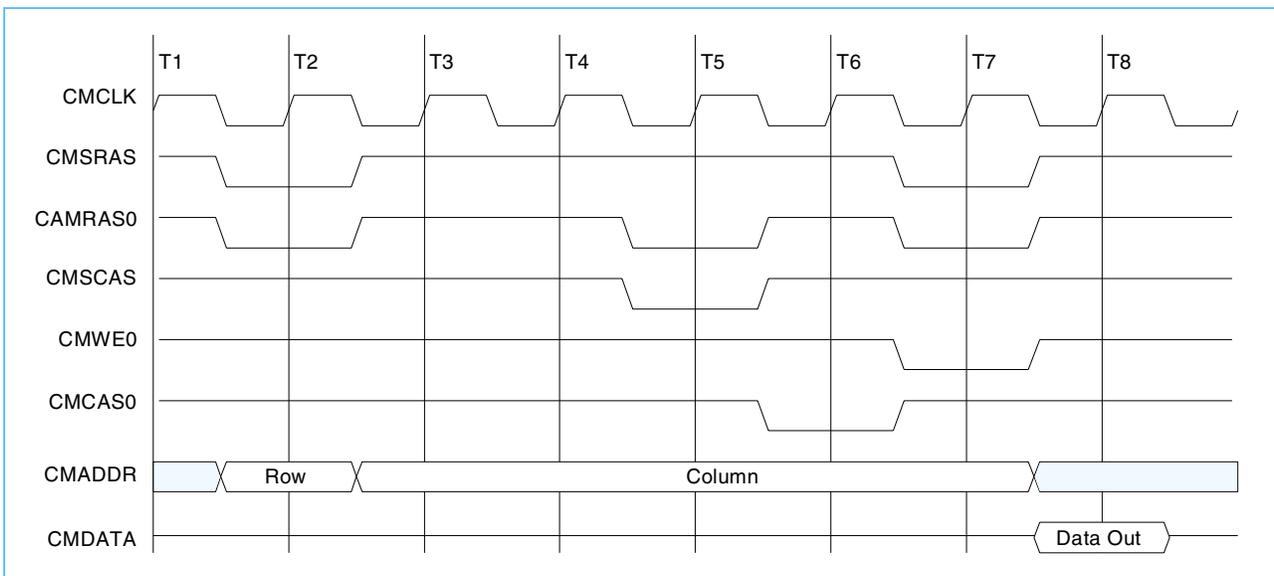
## SDRAM Write Cycle (1 of 4)



CAS Latency=2, Burst Length=1

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | |
| CMSRAS | | | | | | | |
| CMRAS0 | | | | | | | |
| CMSCAS | | | | | | | |
| CMWE0 | | | | | | | |
| CMCAS0 | | | | | | | |
| CMADDR | XXXXXXXX | Row | XXXXXX | Col | XXXXXXXXXX | | |
| CMDATA | ZZZZZZZ | | XXXXXX | Data | ZZZZZZZZZZ | | |

## SDRAM Write Cycle (2 of 4)

**CAS Latency=3, Burst Length=1**

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | |
| CMSRAS | | | | | | | |
| CMRAS0 | | | | | | | |
| CMSCAS | | | | | | | |
| CMWE0 | | | | | | | |
| CMCAS0 | | | | | | | |
| CMADDR | XXXXXXXX | Row | XXXXXX | | Col | XXXXXXXXXX | |
| CMDATA | ZZZZZZZZ | | XXXXXX | | Data | ZZZZZZZZZZ | |

## SDRAM Write Cycle  (3 of 4)

**CAS Latency=2, Burst Length=2**

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | |
| CMSRAS | | | | | | | |
| CMRAS0 | | | | | | | |
| CMSCAS | | | | | | | |
| CMWE0 | | | | | | | |
| CMCAS0 | | | | | | | |
| CMADDR | XXXXXXXXX | Row | 00000000 | Col | | XXXXXXXXX | |
| CMDATA | ZZZZZZZZZZ | | | Data | | ZZZZZZZZZZ | |

## SDRAM Write Cycle  (4 of 4)



**CAS Latency=3, Burst Length=2**

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 |
|---|---|---|---|---|---|---|---|---|---|
| CMCLK | | | | | | | | | |
| CMSRAS | | | | | | | | | |
| CMRAS0 | | | | | | | | | |
| CMSCAS | | | | | | | | | |
| CMWE0 | | | | | | | | | |
| CMCAS0 | | | | | | | | | |
| CMADDR | XXXXXXXX | Row | | Col | | | | XXXXXXX | |
| CMDATA | ZZZZZZZ | | | | 00000000 | Data | ZZZZZZZZ | | |

## SDRAM Write of 64-byte Burst with CAS latency=2

**CAS Latency=2, Burst Length=2**

| | T1 | T2 | T3 | T4 | T5 | T6 | T17 | T18 | T19 | T20 | T21 | T22 | T23 | T24 | T25 |
|---|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

CMCLK

CMSRAS

CMRAS0

CMSCAS

CMWE0

CMCAS0

CMADDR: XXX Row Col0 Col1 Col2 Col 14 Col 15 XXXXXXXX

CMDATA: ZZZZZZZ Data 0 Data 1 Data 2 Data 14 Data 15 ZZZZZZZ

## SDRAM Write of 64-byte Burst with CAS latency=3

## SRAM Timing

## SRAM Read Cycle
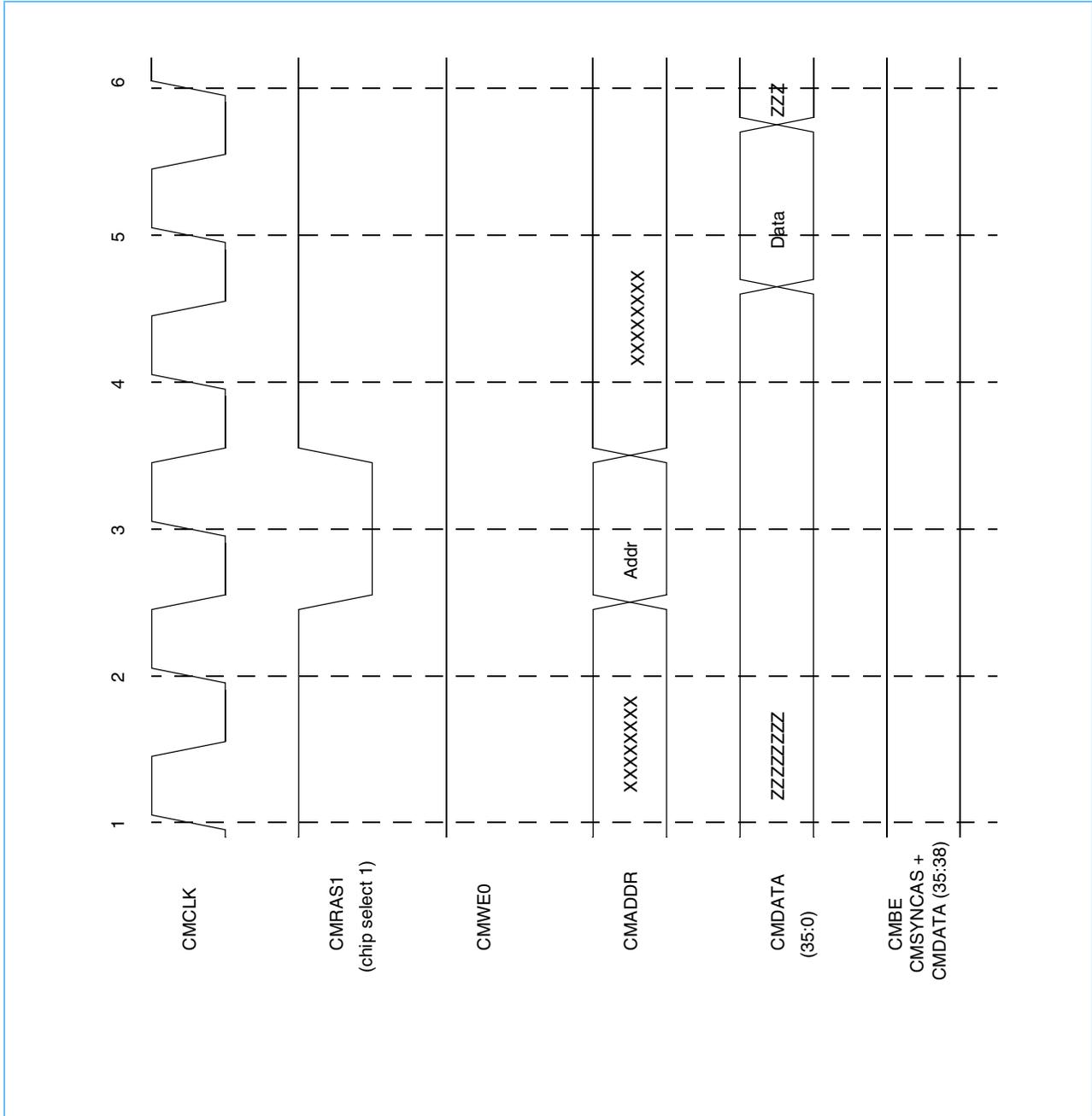
## SRAM Write Cycle

## SRAM Read Cycle with Byte Enables



SRAM Read Cycle with Byte Enables timing diagram showing signals CMCLK, CMRAS1 (Chip Select 1), CMWE0, CMADDR, CMDATA (35:0), and CMBE CMSYNCAS + CMDATA (38:35) across clock cycles 1 through 6.
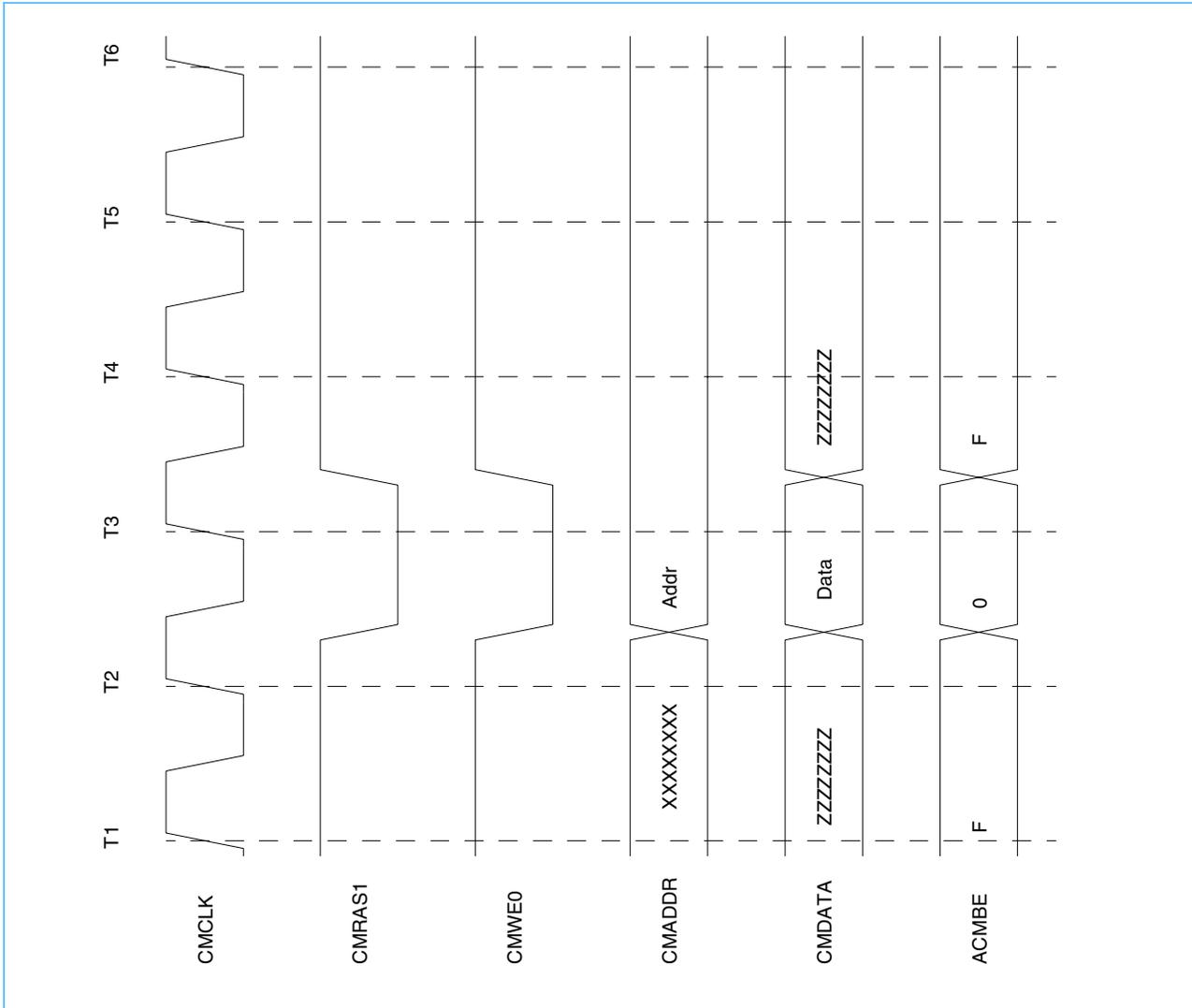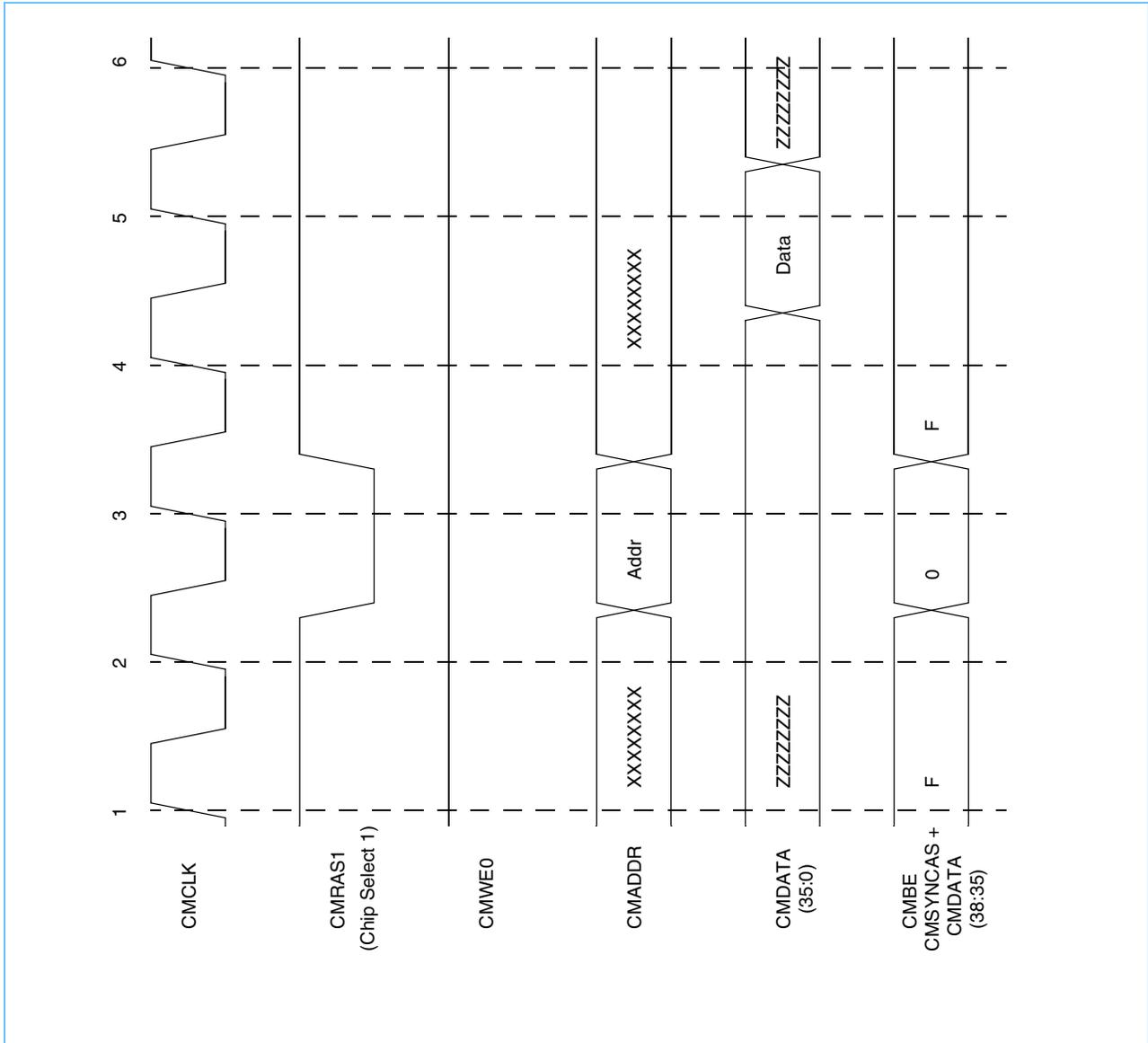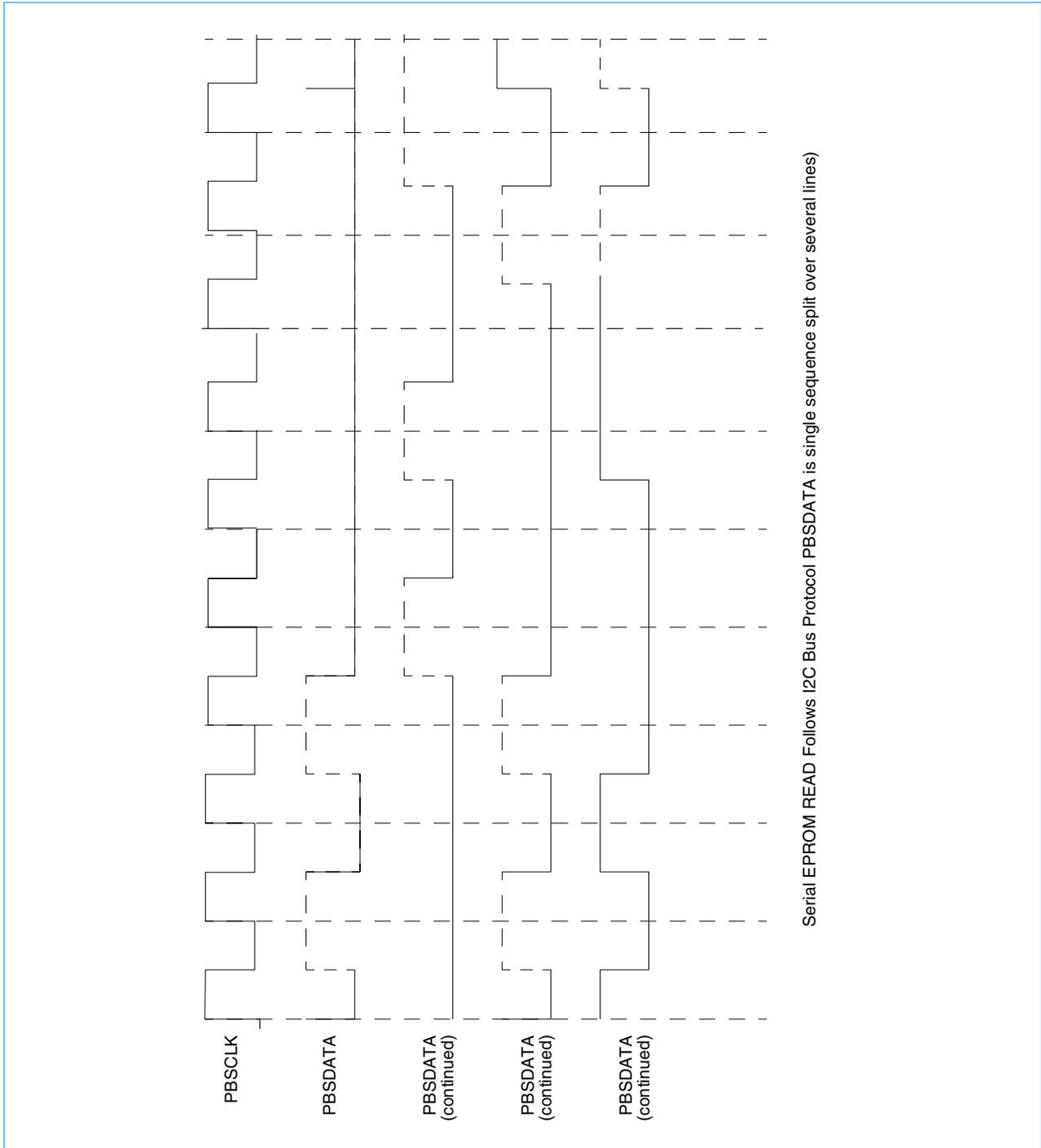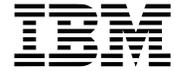
## SRAM Write Cycle with Byte Enables
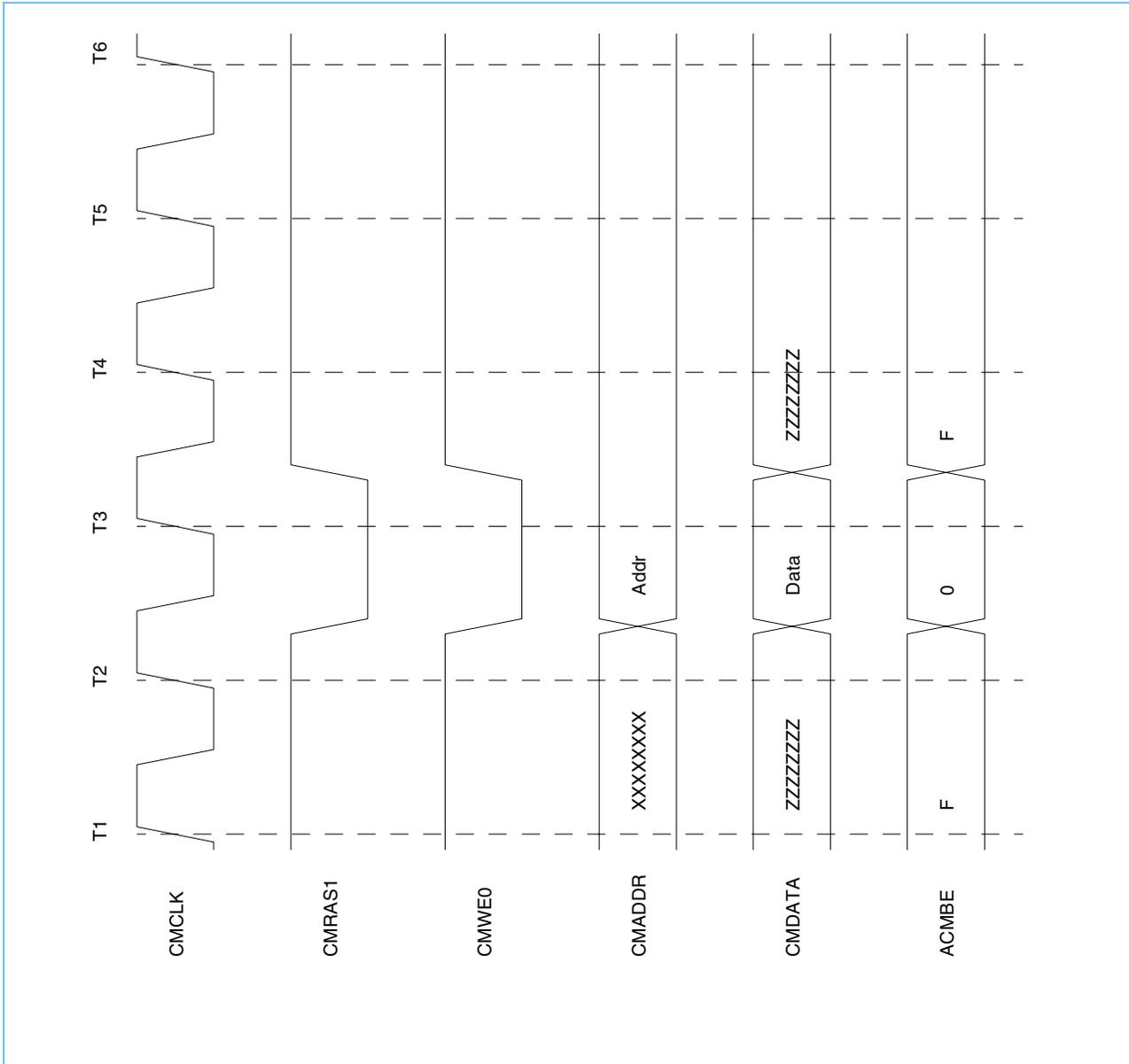
## EPROM Timing

## Parallel EPROM Read
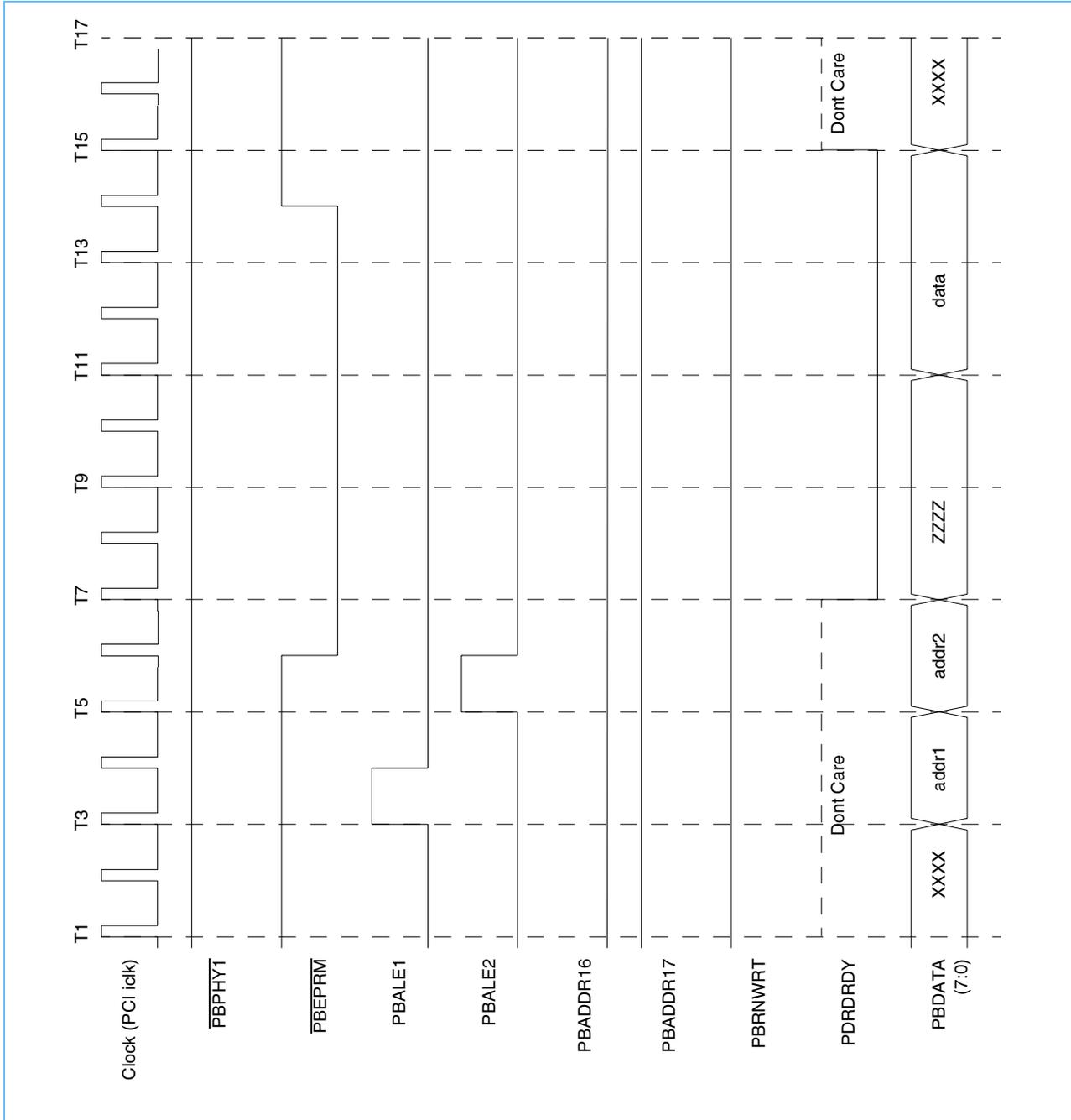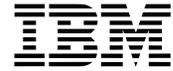
## Parallel EPROM Write

## Serial EPROM Read



PBSCLK

PBSDATA

PBSDATA
(continued)

PBSDATA
(continued)

PBSDATA
(continued)

Serial EPROM READ Follows I2C Bus Protocol PBSDATA is single sequence split over several lines)

## Serial EPROM Write



Serial EPROM WRITE Follows I2C Bus Protocol PBSDATA is single sequence split over several lines

PBSCLK

PBSDATA

PBSDATA
(continued)

PBSDATA
(continued)
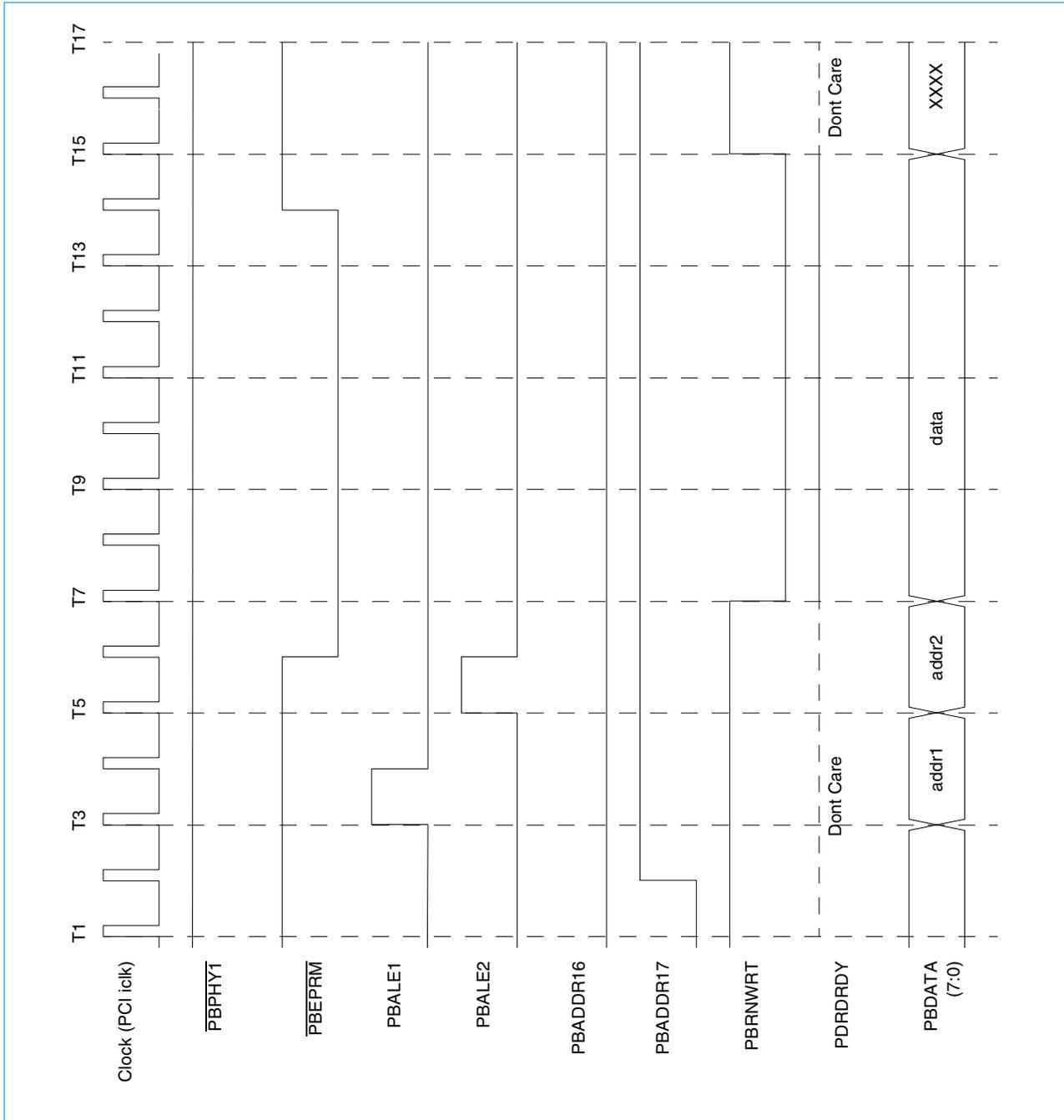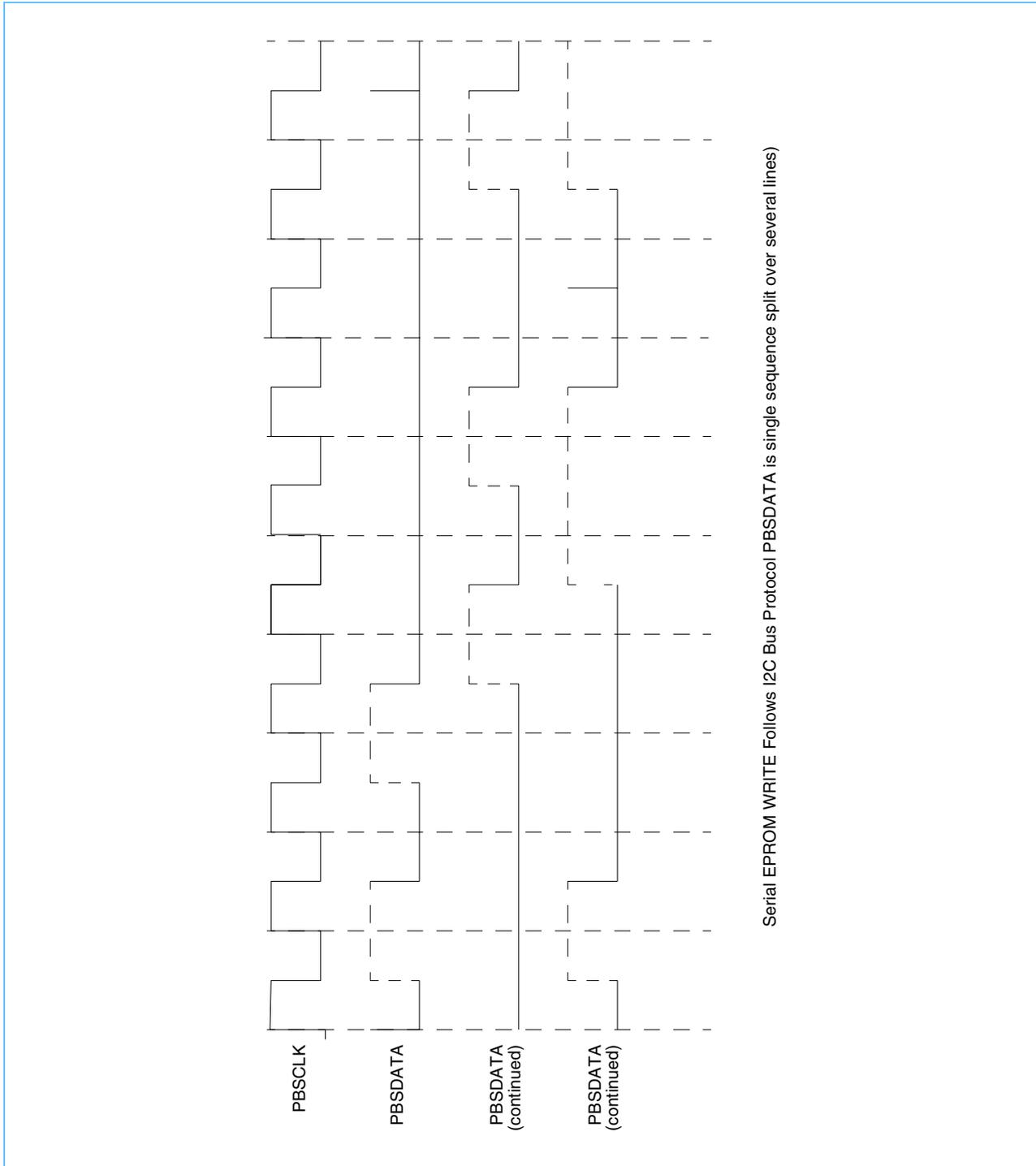
## PHY Timing

## PHY Read
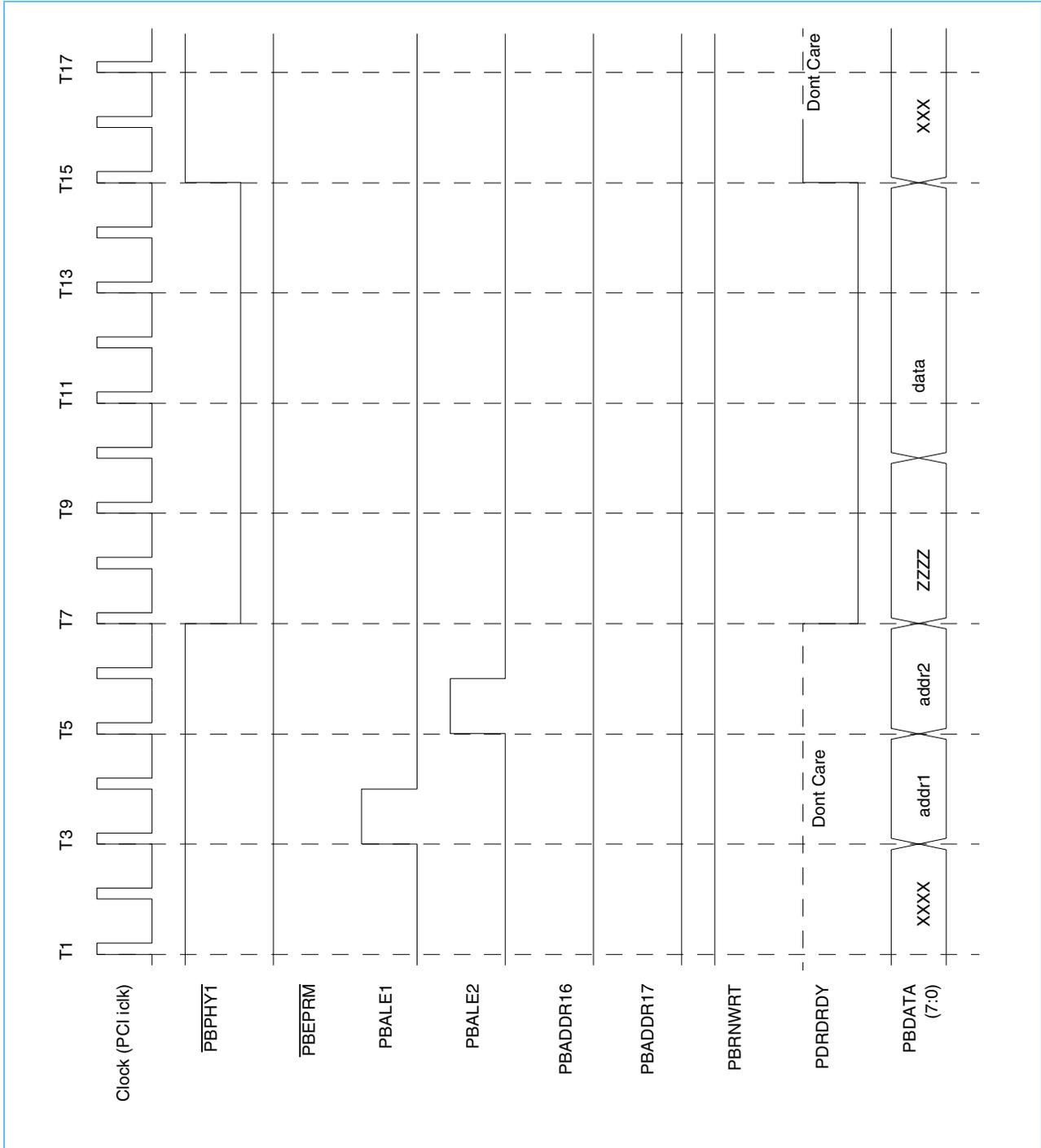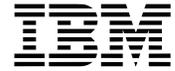
## PHY Write
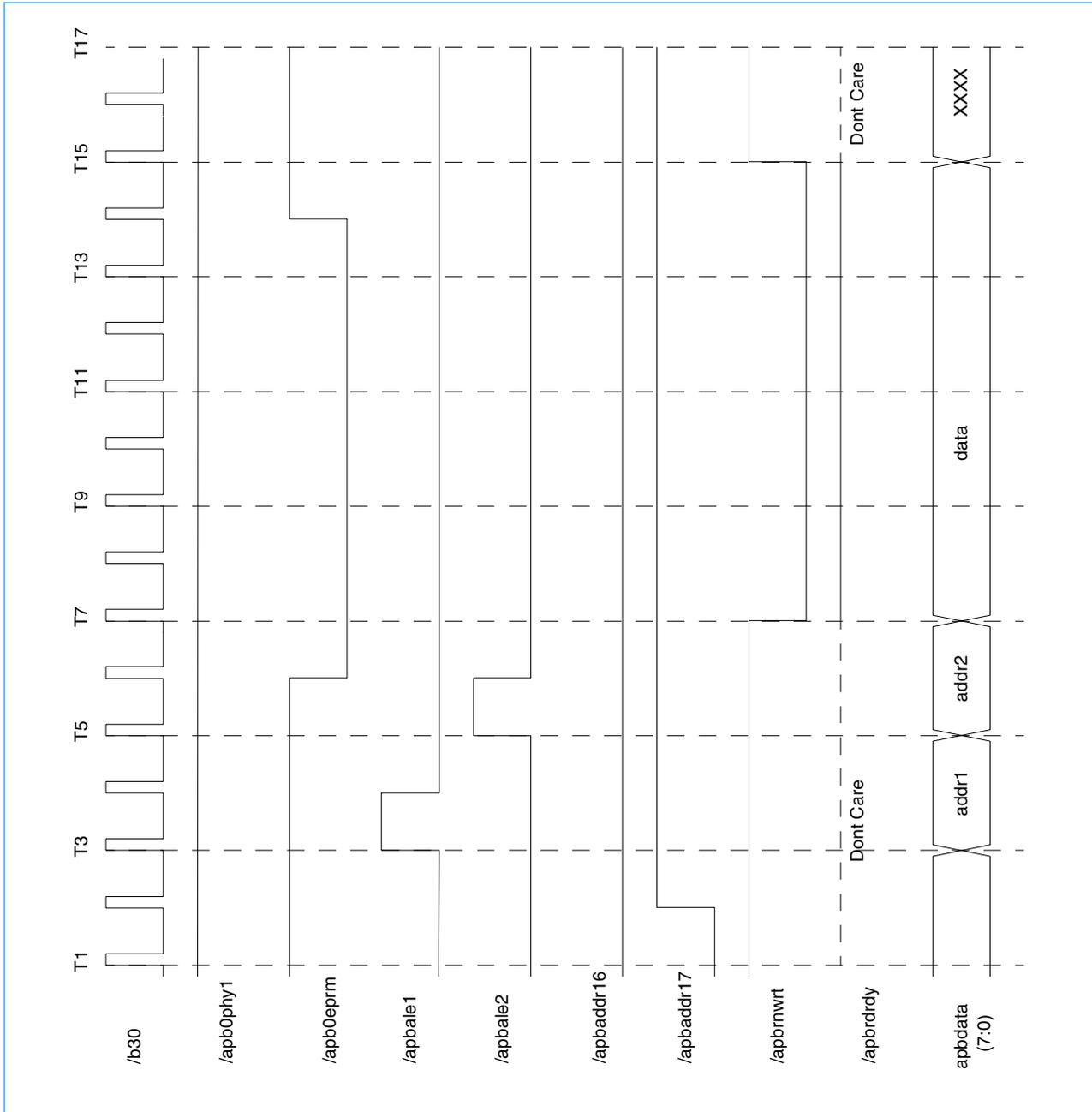
## Revision Log

| Rev. | Description |
|------|-------------|
| 12/15/98 | Initial release (00). |
| 08/27/99 | First revision (01). No content changes. Refined format and corrected typographical errors. |