1023 Channel Voice Over IP Processor

Data Sheet

DS5828 Issue 2 December 2002

Features

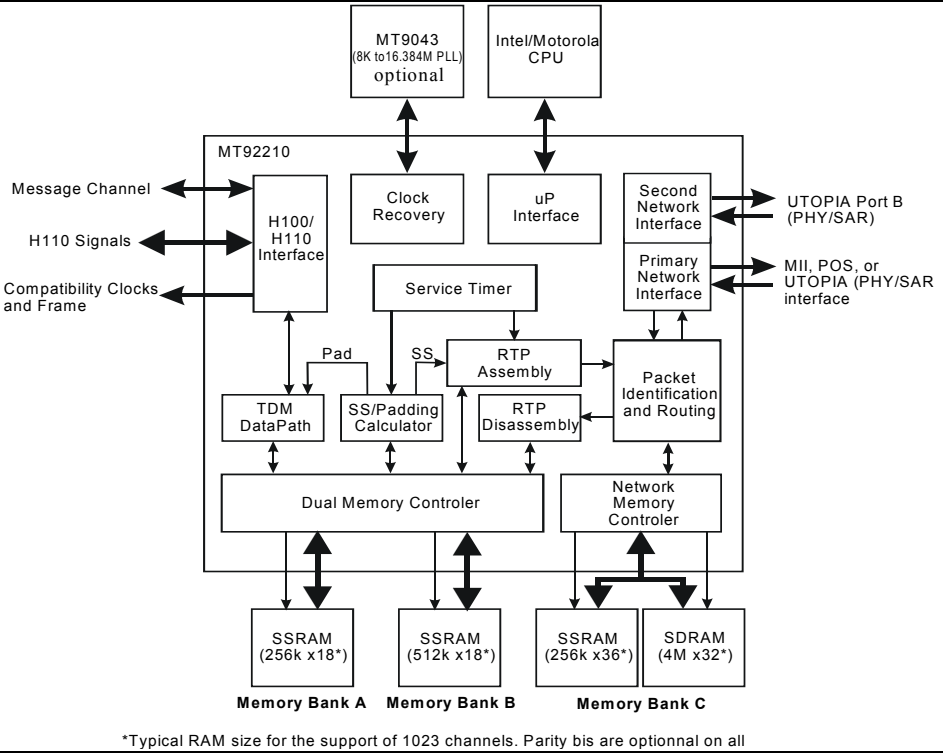
- 1023 full-duplex PCM or ADPCM voice channels over IP/UDP/RTP connections
- RTP packaging optional in IP/UDP connection
- Supports IP version 4 and version 6
- Supports IP over Ethernet, ATM (AAL5) or POS
- Support Ethernet II, IEEE 802.3, LLC/SNAP and PPP frames
- Supports Classical IP over ATM and LAN Emulation (LANE) v1/v2
- Supports MPLS, MPOA and IEEE 802.1p/Q ELAN-ID
- H.110 compliant TDM bus carrying PCM, ADPCM or HDLC channels
- HDLC channels can be used to carry UDP payload generated by external agent
- Support trunking in RTP; up to 255 PCM/ADPCM channels per RTP connection
- Support maximum 1500 bytes packet size
- Up to 4096 bytes of jitter buffer, absorbing +/- 256 ms of PDV
- Less than 250 usec of latency

Ordering Information

MT92210 608 Pin EPBGA

-40°C to +85°C

- Injection of CPU-generated RTP packets or AAL0 cells
- Reception of CPU-destinated RTP packets or AAL0 cells
- Primary and secondary network interfaces
- Primary network interface supports 10/100 MII, POS-PHY or Utopia level 1/2
- Secondary network interface supports Utopia level 1
- Proprietary Adaptive Silence Suppression
- Less than 2.5 watts of power
- 608 pin PBGA package



*Typical RAM size for the support of 1023 channels. Parity bis are optionnal on all

Figure 1 - MT92210 Block Diagram

Applications

- High density voice gateway
- Voice over IP
- 3G and UMTS
- Network processor
- IP switching
- Voice over DSL/cable

Description

The MT92210 device is a voice over IP/RTP assembly and disassembly engine that can convert up to 1023 full-duplex PCM voice channels or 4096 HDLC channels to IP packets and back, conforming to IETF RFC791 (IPv4), RFC2460 (IPv6), RFC768 (UDP) and RFC1889 (RTP). On one side, the device communicates with an H.110 TDM bus carrying voice in either PCM format, ADPCM or HDLC-encapsulated mini-packets; on the other side, it carries its packet data over Ethernet, ATM (using AAL5 cells) or Packet over SONET. A 16-bit Intel/Motorola CPU interface is used to access and configure the device. Finally, three external SSRAM banks and one external SDRAM bank are used for configuration and storage space.

Conventions

In this document, the following conventions are used:

- The transmission direction and the abbreviation TX always refer to the direction in which voice is converted into IP packets.
- The reception direction and the abbreviation RX refer to the direction in which packets or cells are converted into PCM bytes or HDLC packets.
- All numbers in this document are decimal unless otherwise specified.
- Hexadecimal number can be identified by the ‘h’ suffix (ex: A5h).
- Binary numbers are either in double quotes for multiple bits or in single quotes for individual bits (ex: “1001”, ‘0’).
- The term “byte” means 8 bits.
- The term “word” means 16 bits.
- The term “dword” means 32 bits.
- The word “high” means a binary value of ‘1’.
- The word “low” means a binary value of ‘0’.
- The verb “to clear” means to reset one or multiple bits to ‘0’
- The verb “to set” means to put one or multiple bits to ‘1’.
- All addresses are specified in hexadecimal and point to bytes. Addresses are converted from bytes to words to double words using the little endian format, unless otherwise specified.

Colour Code

In this document, the following color code is used:

- Fields in red are initialized by software when the structure is created, and are written back by the hardware.
- Fields in black are initialized by software when the structure is created, and are never written back by the hardware.
- Fields in dark yellow are initialized by software when the structure is created and are written back at the same value by the hardware.

This shade denotes a Reserved Field.

This shade denotes an Unimplemented Field.

The field outlined in red is only written back by the chip when one of the bits, contained within the field and in red, was set and will then be cleared by the chip when it is done acting upon the set request bit.

Document Organization

This data sheet is divided into the following sections:

- **CPU Interface** (Chapter 2.0) describes the main external interface of the MT92210 chip.
- **Network Interface** (Chapter 3.0) describes the interface to the 3 different types of link interfaces, Ethernet, UTOPIA, and Packet over SONET, that are supported.
- **Link Layers** (Chapter 4.0) describes the 3 different types of link layers, Ethernet, ATM AAL5, and Packet over SONET, that are supported.
- **RX/TX Data Flows** (Chapter 5.0) describes the data flows for all packets received and transmitted.
- **Packet Identification** (Chapter 6.0) describes the process by which packets are identified.
- **Packet Assembly** (Chapter 7.0) describes the collected bytes written in the circular buffers by the TX TDM, and how they are assembled into RTP packets.
- **Packet Disassembly** (Chapter 8.0) describes how RTP packets are transformed into PCM bytes, ADPCM samples, or HDLC/CPU-destined mini-packets.
- **TX/RX TDM Data Paths** (Chapter 9.0) describes the data paths for all bytes transmitted and received with the H.110 interface.
- **H.110 Interface** (Chapter 10.0) describes the compatibility of the TDM interface with the H.110 bus.
- **Clocking** (Chapter 11.0) describes the clocks used for the Network Interface and the SAR portion of the device.
- **Pin-out** is in Chapter 12.0.
- **Electrical Characteristics** (Chapter 13.0) describes the electrical characteristics of all the interfaces.
- **Register List** and **Memory Map** are contained in the MT92210 Design Manual.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

1.0 Features

The MT92210 device supports the following features:

1.1 General Features

- Up to 1023 full-duplex PCM or ADPCM voice channels over IP/UDP/RTP connections
- Up to 4096 HDLC channels carrying application data (UDP payload) converted to IP/UDP connections
- Simultaneous support of PCM, ADPCM and HDLC connections
- 16-bit Intel/Motorola CPU Interface
- Fully H.110 compliant TDM interface
- Network Interface A: UTOPIA Level 1/2, POS-PHY Level 2 or MII
- Network Interface B: UTOPIA Level 1
- Full chip capacity can be achieved with two 18 bit data bus ZBT SSRAM, each ranging in size from 128K to 1M bytes; one 36 bit data bus ZBT SSRAM, ranging in size from 128K to 1M bytes; and two 16 bit data bus SDRAM, each ranging in size from 8M to 16M bytes

1.2 Data Formats

- Simultaneous support of IP version 4 and 6
- Chip packages voice in RTP, UDP, and IP to support RFC791, RFC2460, RFC768, and RFC1889
- IP packets can be sent over 3 different types of physical links (Ethernet, ATM, Packet over SONET)
- RTP packaging is optional per connection
- HDLC mini-packets are encapsulated in IP and UDP (RTP packaging performed by external agent)
- IP/UDP layers are optional and can be eliminated to reduce overhead (i.e., null encapsulation) in either ATM AAL5 or Ethernet (using custom EtherType)
- IP over AAL5 can be performed using Classical IP over ATM with SNAP/LLC headers or Ethernet LAN Emulation (LANE) v1 or v2
- Support of MPLS. On reception, MPLS label can be used to establish data format following it, as well as logical subnet number and quality of service. MPLS multicast can be treated as unicast or routed to software
- Support of MPOA. On reception, MPOA tag can be used to establish data format following it, as well as logical subnet number and quality of service
- Logical subnet number can be established using ATM header, MPLS flow header, MPOA tag or ELAN-ID in LANE v2 header
- Quality of service can be determined using ATM header, Ethernet user priority or IP Type Of Service (TOS)

1.3 Voice Treatment Functions

- Up to 1023 PCM/ADPCM channels
- Support for up to 4096 HDLC channels distributed over 512 streams and 2046 time slots
- Up to 255 PCM/ADPCM channels per connection
- HDLC packets contain application data (UDP payload) converted to IP/UDP datagram
- Packets sizes up to 1500 bytes for IP/UDP
- Support of up to 1500 TDM samples of data per packet
- Jitter Absorption Buffer size up to 4096 bytes allowing absorption of up to ±256 ms of PDV
- Injection of CPU-generated RTP packets
- Reception of CPU-destined RTP packets in buffers of up to 64K bytes
- Packet Delay Variation monitoring to diagnose and reduce delay
- Packet loss & misinsertion compensation for PCM and ADPCM packets
- Network jitter monitoring allows support of RTCP for PCM, ADPCM, HDLC and CPU connections
- Policing on HDLC channels and CPU channels protects against misbehaving connections
- PCM, ADPCM, HDLC and CPU mini-packets can all be transported on the same connection with chip's RTP engine to guarantee consistency among the packets
- In the disassembly module, synchronization deltas allow multiple independent connections to be

synchronized end-to-end, allowing, for example, transparent transport of a DS3 over many IP connections

1.4 Network Functions

- IP packet identification can be performed using any combination of IP source address, IP destination address, UDP source port, UDP destination port and RTP Synchronization Source Identifier and are programmable on a per-connection basis
- Non-voice packets can be injected and received via the CPU Interface
- Non-voice packets can be injected and received via the secondary UTOPIA port
- The MT92210 can be daisy-chained to other UTOPIA devices to increase capacity
- Off-the-shelf AAL5 SAR can be used to terminate data connections on a PCI bus
- Support of 16 different look-up profiles, each one of which can use different fields from the packet headers
- Look-up can be performed on a priority basis: for example, a packet can be looked-up using IP, UDP and RTP headers, then the look-up result can request a second lookup using only IP and UDP headers
- Binary tree of up to 128K nodes is used to route packets using packet identification key
- Dynamically balanced tree system ensures optimal performance
- IP, UDP and RTP header verification is performed
- Multihoming is supported with any number of local IP addresses
- Payload Type & Marker bit routing allows different compression formats as well as signaling packets to be transported on the same connection
- MPLS labels, MPOA tags and ELAN ID can be looked-up in binary tree to establish data format that will follow them, logical subnet number and quality of service

1.5 Silence Suppression and Padding

- Proprietary Adaptive Silence Suppression
- Supported in both PCM and ADPCM formats
- Built-in detection of energy level
- Padding with matched-energy comfort noise
- 64 tone buffers used to generate tones (1 byte to 64Kb each)
- 32 large comfort noise buffers (16Kb to 64Kb)
- Suppression indication can be generated by chip or fed externally to synchronize with off-chip compression CODEC

1.6 H.110 Interface

- Fully H.110 compatible
- H.110 Master and Slave capability
- Support of message channel
- Low Latency Loop-back (H.110 to H.110) of 128 channels (delay <= 375 us)
- Redundant Adaptive Clock Recovery Circuit
- Support of 2/4/8 MHz bus speed in groups of 4 streams (8 separate groups)
- Generation of H.110 compatibility signals
- Dual **ct_netref** signals
- Programmable fsync and TDM clocks for compatibility with other TDM buses

1.7 TDM data formats

- Support of plain PCM in u-law and A-law
- Translation between u-law and A-law on a per connection basis
- Support of ADPCM at 40, 32, 24 or 16 kbps
- Dual time-slot mode allows dynamic, error-free switching between PCM and ADPCM formats with silence suppression
- Support of HDLC encapsulated mini-packets with asynchronous timing

- Support of HDLC streams ranging from 1 to 2046 time slots
- Support of HDLC packets with optional Address byte or word, optional Control byte and optional 16-bit CCITT-CRC
- Routing of HDLC streams according to HDLC address byte, with up to 512 channels per stream
- Support of HDLC packets up to 1500 bytes in length

1.8 Link Interface

- Ethernet support for MII interface
- Support for Ethernet MIB
- ATM using twin UTOPIA interfaces allow secondary data SAR to be daisy-chained with device for data connections
- Packet over SONET support for 16-bit POS-PHY bus allowing interoperation with PHY at speeds up to 155 Mbps
- Support for packets of up to 1500 bytes (plus MAC header) in Ethernet and up to 65535 bytes in ATM and Packet Over SONET
- Secondary UTOPIA port can be used in all modes allowing the same data support architecture to be used independently of the link layer with minimal changes
- Transmission of voice to secondary port allows H.110/PCM bridging when coupled with AAL5 SAR
- Pin-out allows designs that support Ethernet, ATM and Packet over SONET with only software configuration deciding on the link layer used

2.0 CPU interface

The CPU module serves as the main external interface of the MT92210 device. Through the CPU interface, external agents can program the MT92210 registers, and read or write to the internal or external memories. The interface is programmable to allow interaction with various types of external agents.

2.1 CPU Interface Description

The CPU interface is comprised of:

- Direct Access Select (DAS) as the MSB bit concatenated with a 15-bit address bus
- 16-bit data bus
- 2 interrupt signals
- associated control signals.

The CPU interface can be configured to operate in either Intel or Motorola mode, The MT92210 supports both 8-bit or 16-bit data bus and multiplexed or non-multiplexed address/data pins.

Internally, a subset of registers -- CPU Interface Registers (000h to 00Ah), can be accessed with very low latency. These registers contain address indirection and data indirection bits. The controlling CPU can choose to launch an indirect access through these registers. Indirect reads will complete in due time when the data is available, while indirect writes are performed almost instantaneously.

Direct accesses to the device can also be made. In these cases, accesses may take longer to complete. Any time a direct access is done, the CPU interface will delay the access using the **cpu_rdy_ndtack** pin until the access has completed internally. Note that direct writes are likely to complete very quickly as long as the write cache is not full.

2.2 CPU Interrupts

The CPU interface provides a programmable global interrupt capability. The interrupt signal are ‘interrupt1’and ‘interrupt2’, pins Y5 and W4 respectively. Both interrupts have programmability to select their active polarity (open-collector drive) via registers ‘interrupt1_conf’and ‘interrupt2_conf’, addresses 214h and 216h respectively. Interrupt1 provides the capability to program a minimum acceptable period between interrupts. The period is programmed in us units via the ‘interrupt1_conf’ register. This provides a ‘frequency interrupt controller’ facility and mask the assertion of further interrupts until the specified period of time has elapsed. The mask period will start when the interrupt1_treated[15] bit in the register ‘interrupt_treated’ (address 212h) is set. When Interrupt2 is enabled it is always activated when an interrupt condition occurs.

The operation of the CPU interrupt network is common for all modules. When an interrupt is asserted, an interrupt flag is set to identify the module where the interrupt was generated. On completion of the ISR the interrupt must be cleared as the interrupt will remain asserted until it is de-asserted by the user. All Interrupt Enable Registers have a mirror Status Register. Hence, the bit positioning of the interrupt enables and the corresponding status bits are identical.

Interrupt pins are always tri-stated when inactive.

2.2.1 Example Interrupt Flow

Upon the initialization of the Globe Interrupt pins the following methodology is adopted to identify the source of the interrupt. For this example Interrupt2 is employed and the CPU module will be the source of the interruption.

2.2.1.1 Interrupt Initialization

Set interrupt polarity, register interrupt2_conf[15:14], address 216h.

Enable Inetrrupt2 for the CPU module by setting bit 0 in inetrrupt2_enble register (21Ch). The MT92210 will generate an interrupt on interrupt2 pin according to the modules enabled in inetrrupt2_enable.

2.2.1.2 Interrupt Servicing

When interrupt2 is asserted (‘inetrrupt2’ pin):

1. Read the interrupt flags to ascertain the module raising the interrupt. The CPU module interrupt flag is located in register inetrrupt_flags(210h), this bit is named cpureg_interrupt_active.
2. If the cpureg_interrupt_active bit is set, check the source of the CPU interrupt by reading the ‘status0’ register at 102h, either internal_read_timeout_sar, and/or inmo_read_done, and/or interrnal_read_timeout_net
3. To de-assert the interrupt the user must write 1 to corresponding bit in register 102h, ether internal_read_timeout_sar, and/or inmo_read_done, and/or internal_read_timeout_net. Only then will the interrupt be de-asserted.

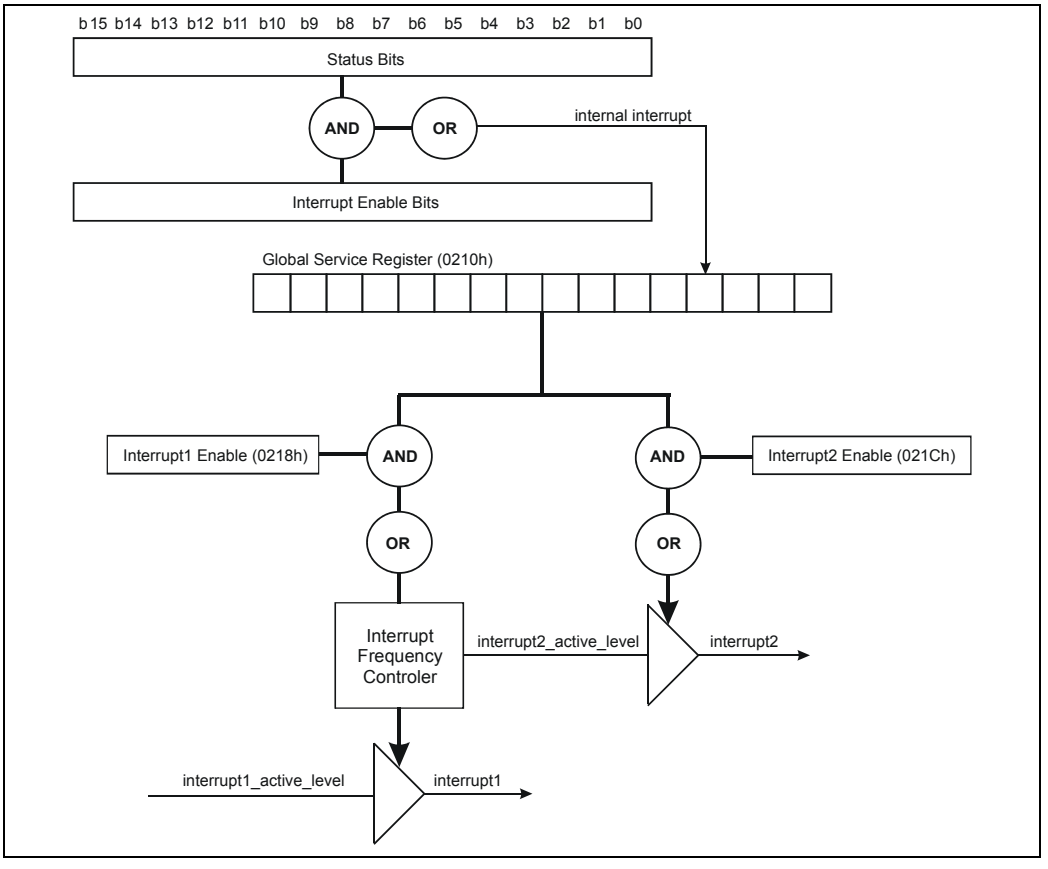


Figure 2 - Internal Interrupt Network

2.3 Intel/Motorola Interface

The MT92210 CPU interface supports both Intel and Motorola modes, in both 8-bit or 16-bit data bus and multiplexed or non-multiplexed address/data pins. The MT92210 supports 128 Megabytes of addressable space, therefore extended addressing is necessary. The CPU interface directly addresses four control words, delegated for indirection accessing. All of the register spaces in this section provide extremely fast CPU access times.

000h	Control Register
004h	Read/Write Data Register
008h	Address High Register
00Ah	Address Low Register

Table 1 - Indirect Access Register

CPU_MODE [3:0]	INTERFACE TYPE	ALE[a]	ADDRESS PIN[b]	DATA PIN	DIRECT_ ACCESS
0000	Intel, 16 bit data bus, non-multiplexed	cpu_ale	cpu_a[14:0] (word address)	cpu_d[15:0]	cpu_a_das
0001	Intel, 16 bit data bus, multiplexed.	cpu_ale	cpu_d[15:1] (word address)	cpu_d[15:0]	cpu_a_das
0010	Intel, 8 data bus, non-multiplexed	cpu_ale	cpu_a[14:0] (byte address)	cpu_d[7:0]	cpu_a_das
0011	Intel, 8 data bus, multiplexed	cpu_ale	cpu_a[14:8] & cpu_d[7:0] (byte address)	cpu_d[7:0]	cpu_a_das
0100	Motorola, 16 bit data bus, non-multiplexed	cpu_ale	cpu_d[14:0] (word address)	cpu_d[15:0]	cpu_a_das
0101	Motorola, 16 bit data bus, multiplexed	cpu_ale	cpu_a[15:1] (word address)	cpu_d[15:0]	cpu_a_das
0110	Motorola, 8 bit data bus, non-multiplexed	cpu_ale	cpu_a[14:0] (byte address)	cpu_d[7:0]	cpu_a_das
0111	Motorola, 8 bit data bus, multiplexed	cpu_ale	cpu_a[14:8] & cpu_d[7:0] (byte address)	cpu_d[7:0]	cpu_a_das
1xxx	Reserved				

Table 2 - CPU Interface Mode Selection

a. The cpu_ale pin is interpreted in all modes. However, it is not necessary in the non-multiplexed modes and can be tied to VCC.
b. The address placed on the cpu_a[14:0] pin is a word address in 16-bit mode and a byte address in 8-bit mode. The address, when placed on the cpu_d pins, is always a byte address.

Field	Bit	Type	Reset	Description
read_burst_length	6:0	RW	01h	Number of words to prefetch when any read is executed. 00h = 128; 01h= 1; 02h = 2, etc. The higher this number, the longer the first read in a sequential series of reads will take; all successive accesses will be very quick and the overall read performance will be much better. This field should be set to 1 for individual (non-sequential) reads. Any read burst that crosses a 256 byte boundary will be broken up into two bursts.
Reserved	7	RO	0	
access_req	8	PC	0	Set by software when an extended access to the device must be started. Cleared by hardware when the access is completed. Used for extended indirect accesses only.
extended_a[3:1]	11:9	RW	000	Extended address bits 3:1. **extended_a**[32:0] points to bytes. Used for extended indirect accesses only.
write_enable	13:12	RW	00	Active high write enables. “00” = read access (indirect only); “01” = write to lower byte; “10”=write to upper byte; “11”=write to entire word. Used both in extended indirect and extended direct write accesses. For all extended direct read accesses, this field has no effect. For all byte wide extended direct accesses, this field has no effect.
extended_parity	15:14	RW	00	Parity bits used for both reads and writes. Used both in extended indirect and extended direct accesses.

Table 3 - Control Register (000h)

Field	Bit	Type	Reset	Description
extended_data	15:0	RW	0000h	The 16 bits of data used in an extended indirect access to the chip. For all extended direct accesses, this field is not used. During write accesses, the write data is written here by the external CPU. During read accesses, read data returns here to be read.

Table 4 - Read/Write Data Register (004h)

Field	Bit	Type	Reset	Description
extended_a[32:20]	12:0	RW	0000h	Extended address bits 32:20. **extended_a**[32:0] points to bytes. Used both for extended indirect and extended direct accesses.
Reserved	15:13	RO	000	

Table 5 - Address High Register (008h)

Field	Bit	Type	Reset	Description
extended_a[19:4]	15:0	RW	0000h	Extended address bits 19:4. **extended_a**[32:0] points to bytes. Some bits in this register are not used in direct accesses. When operating the CPU interface with a 16 bit data bus, only bits 19:16 are used. When operating the CPU interface with an 8-bit data bus, only bits 19:15 are used.

Table 6 - Address Low Register (00Ah)

2.3.1 Extended Indirect Access Procedures

Extended Indirect Accessing solely employees the registers 000h to 00Ah to access the 128 Megabytes of addressable memory space. The access address is written to registers 000h, 008h, and 00Ah. The MT92210 will read/write to that address and fetch /place the data value from/to register 004h. For all extended indirect accesses the cpu_a_das pin will be held low.

2.3.1.1 Extended Indirect Writes

1. Write the upper address, **extended_a**[32:20], to register 008h. This may not be required if previous value holds true.
2. Write the lower address, **extended_a** [19:4], to register 00Ah. This may not be required if previous value holds true.
3. Write the write data, **extended_data**[15,0], to register 004h.This may not be required if previous value holds true.
4. Write **write_enable**, **extended_parity**, **access_req=**'1' and **extended_a** [3:1] in a single access to register 000h.
5. Read the **access_req** bit located in the Control Register[8] to determine when the write cycle has completed.

The software will set access_req[8] in register 000h. The hardware will reset the bit when the data write cycle has completed. Therefore, this bit can be polled to determine when the data write cycle has completed.

2.3.1.2 Extended Indirect Reads

1. Write the upper address, **extended_a**[32:20], to register 008h.This may not be required if previous value holds true.
2. Write the lower address, **extended_a** [19:4], to register 00Ah. This may not be required if previous value holds true.
3. Write **write_enable**="00", **access_req=**'1' and **extended_a** [3:1] in a single access to register 000h.
4. Wait until access_req is cleared, then read data from the data field **extended_data[15,0],** register 004h.

Optional parity check may be ascertained by performing a read on the extended_parity[15,14], register 000h. The software will set access_req[8] in register 000h and then the hardware will reset it when the data is ready to be read from register 004h.

2.3.2 Extended Direct Access Procedures

Extended Direct Accessing employs the high and low address registers to perform page addressing. The address within the page is provided directly by the CPU address bus. Similarly, the data is fetched/placed directly on the CPU data bus.

The access address is written to registers 008h and 00Ah. This will perform only the page addressing. Upon assertion of the address within the page, the MT92210 will read/write the data with respect to that address. The

cpu_a_das pin is set when the data read/write occurs. When operating the CPU interface in direct mode with a 16-bit data bus, **extended_a[19:16]**, are employed for the lower address word register 00Ah. However, when operating the CPU interface in direct mode with an 8-bit data bus, bits **[19:15]** are used for the lower address word.

2.3.2.1 Extended Direct Writes

1. Write the upper address, **extended_a**[32:20], to register 008h. This may not be required if previous value holds true.
2. Write the lower address, **extended_a** [19:16] or [19:15] to register 00Ah. The remaining bits [15:4] or [14:4] are ignored. This may not be required if previous value holds true.
3. Write **write_enable**[13:12] (This may not be required if previous value holds true) and **extended_parity**[15:14]. The extended parity write is optional.
4. Write data value to the address within the corresponding memory page with the cpu_a_das pin set.

2.3.2.2 Extended direct reads

1. Write the upper address, **extended_a**[32:20], to register 008h. This may not be required if previous value holds true.
2. Write the lower address, **extended_a** [19:16] or [19:15] to register 00Ah. The remaining bits [15:4] or [14:4] are ignored. This may not be required if previous value holds true.
3. Assert the lower address within the memory page and fetch the read data with cpu_a_das set.
4. Read the **extended_parity** field (optional), extended_parity[15:14], register 000h.

2.4 MT92210 Reset Procedure

The reset procedure for the MT92210 requires several steps, mostly due to the fact that there are several levels of hardware and software resets in the chip. All register accesses in the reset procedure maybe performed in either Direct or Indirect mode. The procedure to configure the chip is as follows:

1. Assert the nreset pin for at least one 1 ms.
2. De-assert the nreset pin.
3. Clear nreset bit in Register 100h, set Bit 9 (mem_oe), Bit 10 (ethernet_enable, if necessary), Bit 13 (low_latency_cpu_accesses) in Register 100h.
4. Configure upclk frequency in Register 10Ah.
5. Configure the fast_clock PLLs in Register 110h, 170h, 172h.
6. Configure H.110 PLL in Register 174h.
7. Set proper divisors in Register 164h, 166h.
8. Reset Bit 9 (mem_oe) in Register 100h.
9. Set Bit 0 (nreset_registers) in CPU Register 100h.
10. Set active levels for interrupt pins in the Main Registers (214h, 216h).
11. Configure external memories in the Main Registers (230h, 232h, 234h, 236h, 240h).
12. Set Bit 1 (nreset_chip) in CPU Register 100h.
13. Configure all the other registers.
14. Set Bit 2 (nreset_network) in CPU Register 100h.

3.0 Network Interface

The objective of the MT92210 device is to transport voice information encapsulated in IP packets over network connections. Therefore, to allow maximum flexibility, it can support 3 different types of link interfaces: Ethernet, UTOPIA and Packet over SONET.

The network module of the chip is responsible for the identification and routing of packets, deciding which packets should be kept and treated as voice, which should be routed to the data packet buffer and which should be discarded.

The network module accepts packets that are generated by the Packet Assembly module, as well as packets received from either of the two RX link ports. In the TX direction, it can send packets to the Packet Disassembly module, as well as to any of 4 TX link buffers: TX link A High-Priority, TX link A Low-Priority, TX link B High-Priority and TX link B Low-Priority.

It can also receive cells from its twin UTOPIA ports or from the TX CPU cell queue and can route them to the RX CPU cell buffer, or one of 4 TX link A cell queues (in priority) or one of 2 cell queues going to TX link B.

The following figure gives an overview of the data path in the network module, including all the queues that are used to buffer the data along the way:

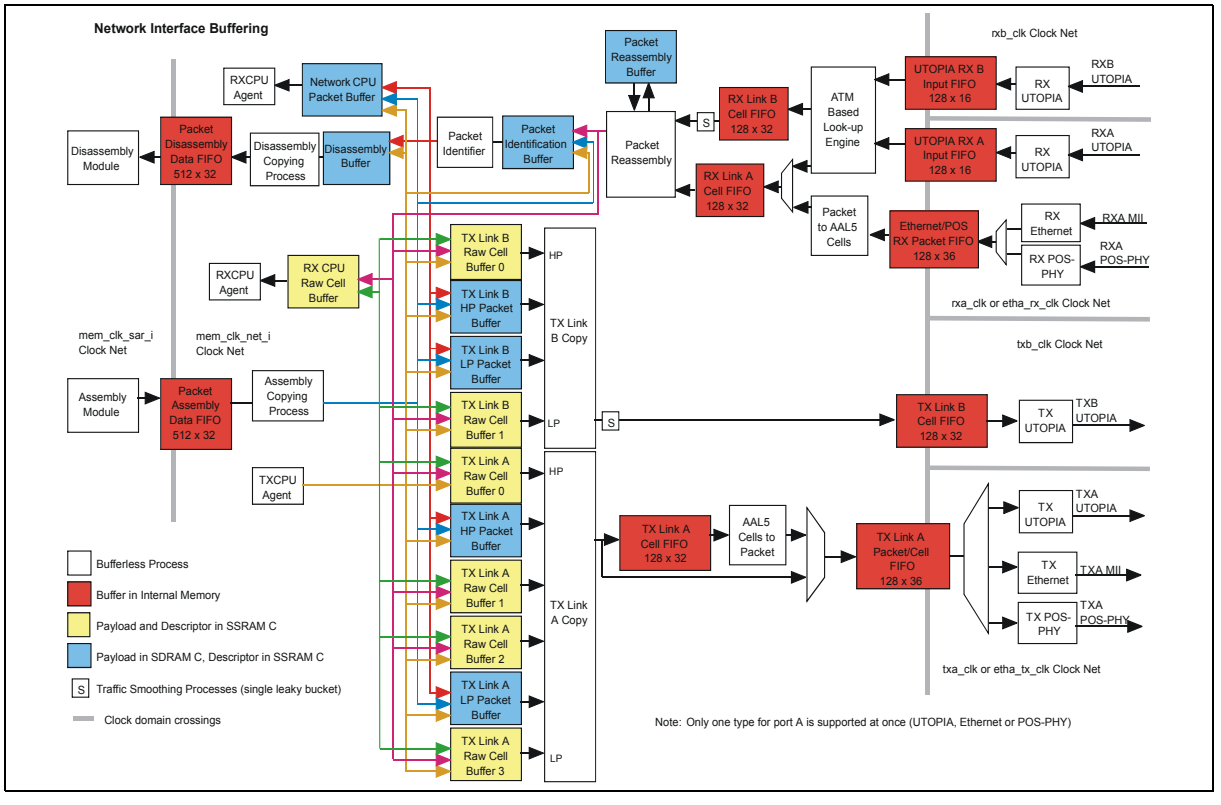


Figure 3 - Network Interface Buffering

The module uses an external 32-bit SDRAM to buffer all the packets in transit and applies a linked list technique to allocate the blocks of memory in the SDRAM. Each packet, as it enters the module, is broken down into 48-byte payload blocks. These blocks are stored one at a time in SDRAM. A 19-bit link pointer links together the blocks that make up the packet. A link value of 00000h indicates that this block is the last one in the packet. Each block

occupies 64 bytes of SDRAM. The following figure 4 illustrates the format of blocks as they are kept in external SDRAM.

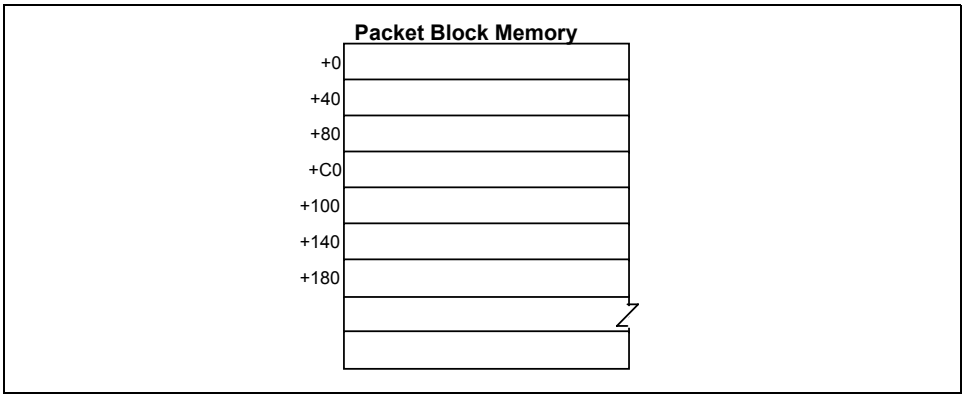


Figure 4 - Packet Block Memory and Format

	b31 b30 b29 b28 b27 b26 b25 b24	b23 b22 b21 b20 b19 b18 b17 b16	b15 b14 b13 b12 b11 b10 b9 b8	b7 b6 b5 b4 b3 b2 b1 b0
+0	Payload	Payload	Payload	Payload
+4	Payload	Payload	Payload	Payload
+8	Payload	Payload	Payload	Payload
+C	Payload	Payload	Payload	Payload
+10	Payload	Payload	Payload	Payload
+14	Payload	Payload	Payload	Payload
+18	Payload	Payload	Payload	Payload
+1C	Payload	Payload	Payload	Payload
+20	Payload	Payload	Payload	Payload
+24	Payload	Payload	Payload	Payload
+28	Payload	Payload	Payload	Payload
+2C	Payload	Payload	Payload	Payload
+30		Multicast Sum	Forward Link Handle [24:6]	
+34				
+38				
+3C				

Figure 5 - Packet Block Format

Field	Description
Payload	Packet payload. 48 bytes per block.
Forward Link Handle	Link to next block of packet. All zeros is invalid link (indicating end of packet).
Multicast Sum	Used in TX. When packet is queued in multiple TX queues this is decremented each time the packet is sent and the TX queue that decrements it to 0 frees the cell memory.

Table 7 - Packet Block Format Table

Packets are managed in 7 queues. Each queue corresponds to a possible destination of a packet within the module. In addition to the 4 queues containing packets for the 2 TX link outputs, there are also queues for packets going to the Disassembly module, the RX CPU FIFO in external memory and the Packet Identifier module. Each of these queues can be configured independently as to a maximum number of blocks it may contain This prevents a single overflowing port to grab the entire SDRAM and rob properly functioning ports of the memory they need to operate. The CPU agent can also seize blocks from the linked list to write its packets destined to the CPU. The CPU must free those blocks once it has read them.

In addition to the block queues, there are also 7 handle queues, each handle queue being associated to the block queues. The handle queues contain the handles identifying the packets contained within the block queue. These handles detail all the characteristics of the packets and are passed between agents until they reach their final destination. Each handle queue can be programmed to a 2^n size. Note that while handles may be copied to a new handle queues (especially after passing through the packet identification section), the blocks that contain the packet itself are never moved.

Each time a packet handle is added to a handle queue, the number of blocks it occupies is added to the block fill of the queue. Should the packet cause either the block fill or the handle fill to exceed their maximum values, the packet will be discarded and a per-queue status bit will be set in registers, indicating the error that has occurred. Note that the chip performs its own garbage collection, so no blocks in the linked list are ever left floating because the packet to which they were associated was lost.

The network interface also supports multicast functionality: a single packet can be sent to multiple destinations simultaneously. A notable exception is that if a packet is being sent to the packet identifier, it cannot be sent anywhere else.

A module called the packet handler manages all of these queues using a small internal memory shown in the next figure.

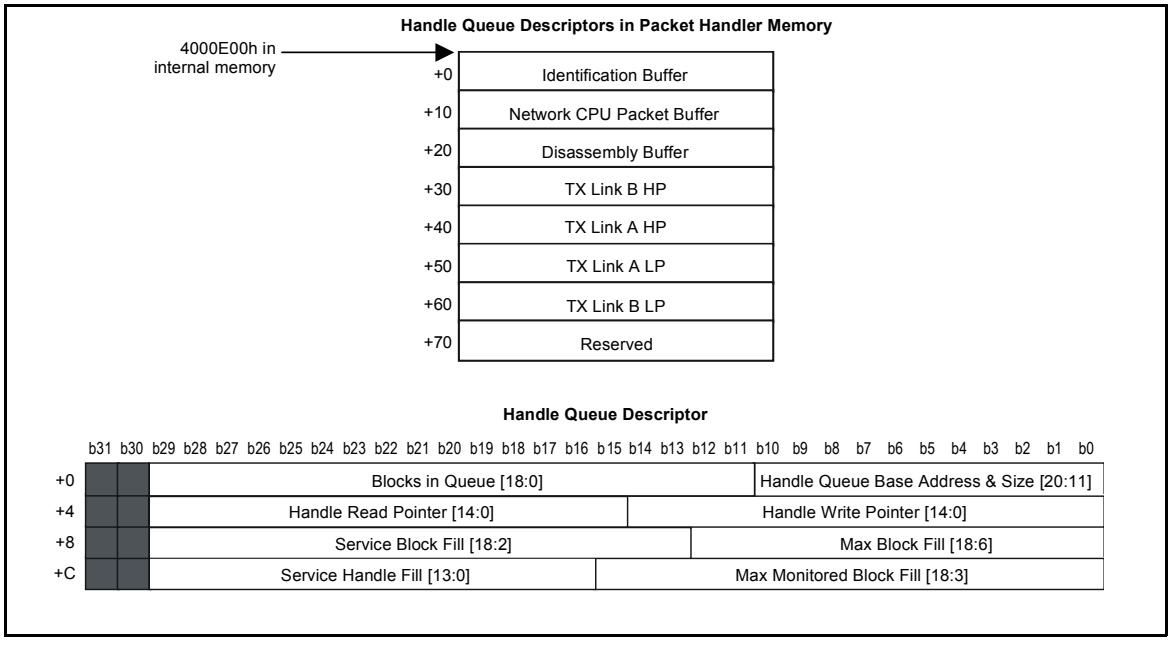


Figure 6 - Packet Handler Memory

Field	Description
Blocks in Queue	Total number of blocks currently contained in queue.
Handle Queue Base Address & Size	Indicates the base address of the handle queue as well as its size. The minimum queue size is 2K bytes, which is 128 handles. The maximum size is 256K bytes, which is 16K handles. The size is indicated by the lowest '1' in the field: for instance, if the field were xxxxxxxxxx1, this would represent a 2K byte buffer with a base address whose high bits are xxxxxxxxxx. If the field were xxxxxx10000, this would represent a 32K byte buffer with a base address whose high bits are xxxxxx.
Handle Read Pointer	Pointer within the handle queue indicating the most recently read handle.
Handle Write Pointer	Pointer within the handle queue indicating the most recently written handle.
Service Block Fill	If the Blocks in Queue is less than or equal to this value, a service bit in registers can be set, indicating that the queue has gone below a certain fill. Both this and the handle fill must be valid for the service bit to be set.
Service Handle Fill	If the difference between Handle Write Pointer and Handle Read Pointer is less than or equal to this value, a service bit in registers can be set. Both this and the block fill must be valid for the service bit to be set.
Max Block Fill	Indicates how many blocks, at most, the cache may contain before it overflows. This excludes the packet at the head of the queue, because it has been cached and will be treated shortly.

Table 8 - Handle Queue Descriptor

Handle queues are stored in SSRAM C. Each handle occupies a block of 16 bytes. Handle format is shown in Figure 7.

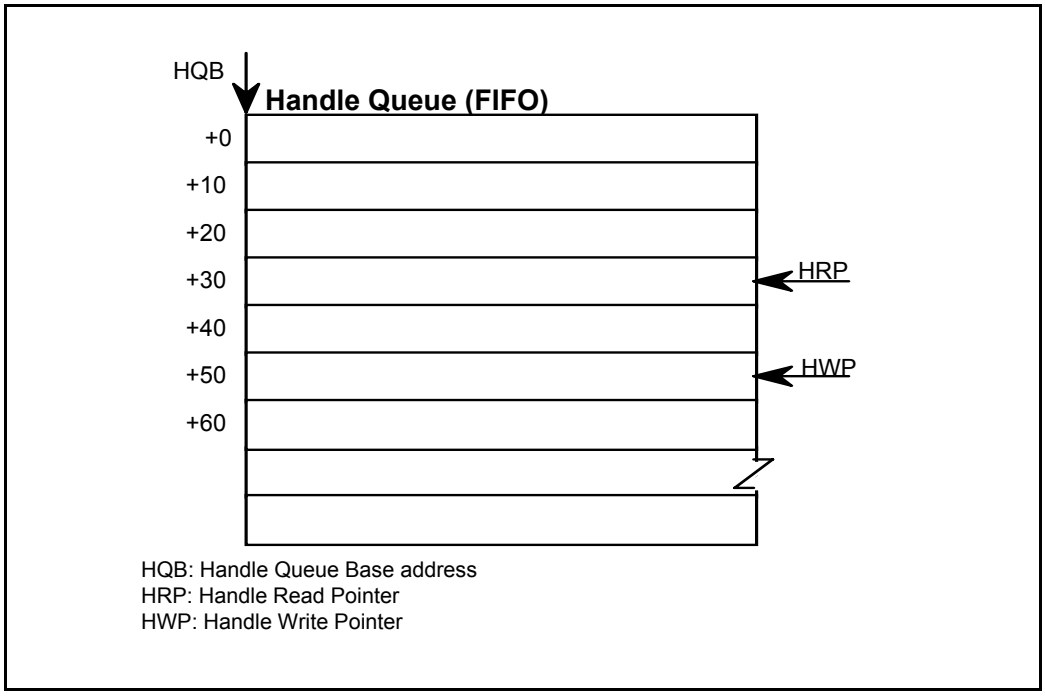


Figure 7 - Handle Queue and Handle Format

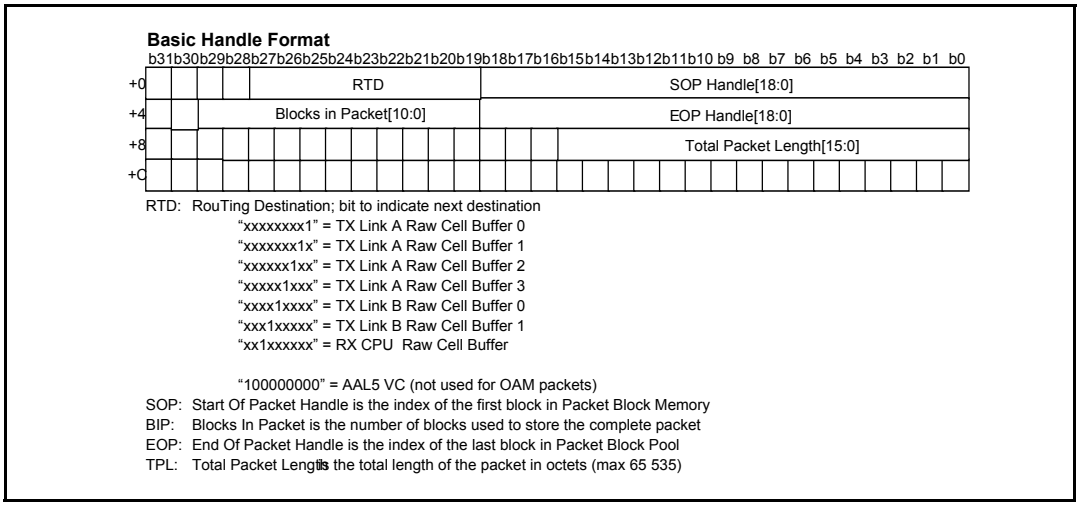


Figure 8 - Basic Handle Format

RX packets received from the packet assembly module are contained in a 2K byte internal memory. Once the full packet has been received, it is transferred to the SDRAM in block form. This internal buffer is large enough to contain an entire packet of up to 1500 bytes in length (+ link overhead). In addition to the packet memory, a small (128 bytes) handle memory contains the handles to these packets, indicating their length, base address and

destination. Packets from the packet assembly module can be routed to any one of the TX link buffers or to the packet identifier for internal loopback functionality.

Packets going to the disassembly module use a similar scheme: a 2K-byte memory is used for the packets and a 128-byte memory for the handles.

Near the TX link layer interface, the network module uses 512-byte buffers to transfer packets between the SDRAM and the link layer interfaces. There are 4 buffers used for this purpose: 1 destined to TX link A, 1 to TX link B, 1 from RX link A and 1 from RX link B. These buffers can be smaller because the system operates flawlessly even if the entire packet is not contained in the buffer at any one time.

Cells contained in external memory are stored in the SSRAM and free cells are allocated as the various modules within the chip request them. The cells are also managed in a linked list, in the same way as the packet are. However, since each cell is individual (i.e. not part of a packet comprised of many blocks) they are only linked when they are free: when a cell is allocated and contains valid data, its link is not used. Whenever cells are allocated, the pointer to the cell is added to a cell queue, which contains all cells going to a given destination. The following figures indicate the format of raw cells in queues, depending on whether the cell is allocated or free:

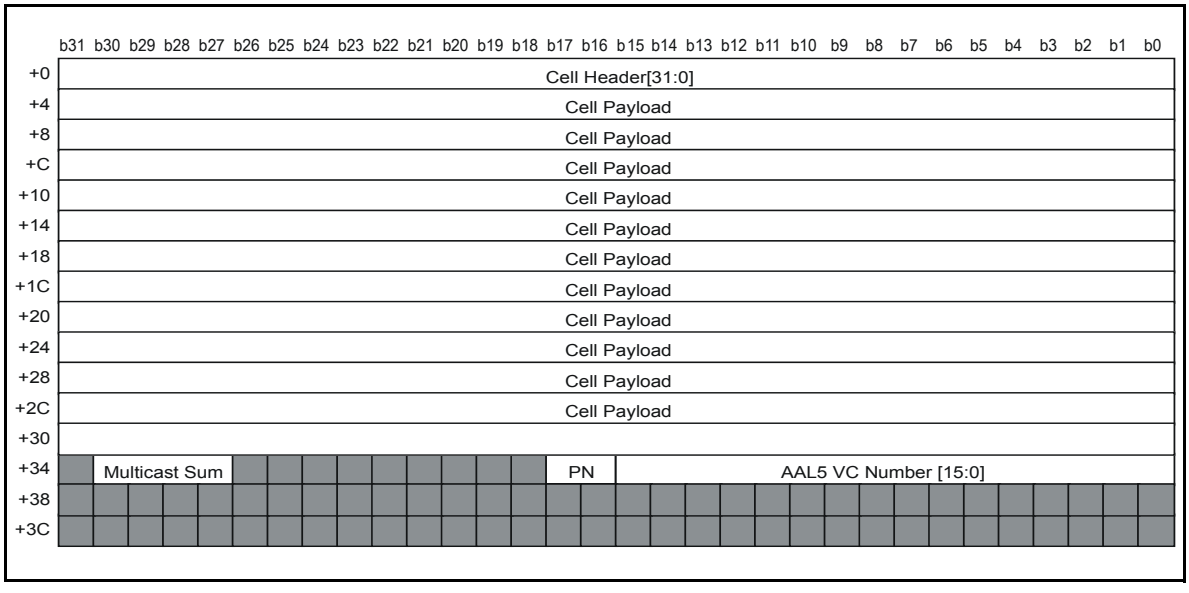


Figure 9 - Raw Cell Format (used cell)

Field	Description
Multicast Sum	Used in TX. When Cell is queued in multiple TX queues this is decremented each time the cell is sent and the TX queue that decrements it to 0 frees the cell memory.
AAL5 VC Number	This points to a Packet Reassembly structure.
PN	Port Number. Used in RX. Source port of this cell. Used to indicate, in the RX AAL0 FIFO, where cells originated. “00” = RX port A, “01” = RX port B; “11” = TX CPU.

Table 9 - Fields and Description

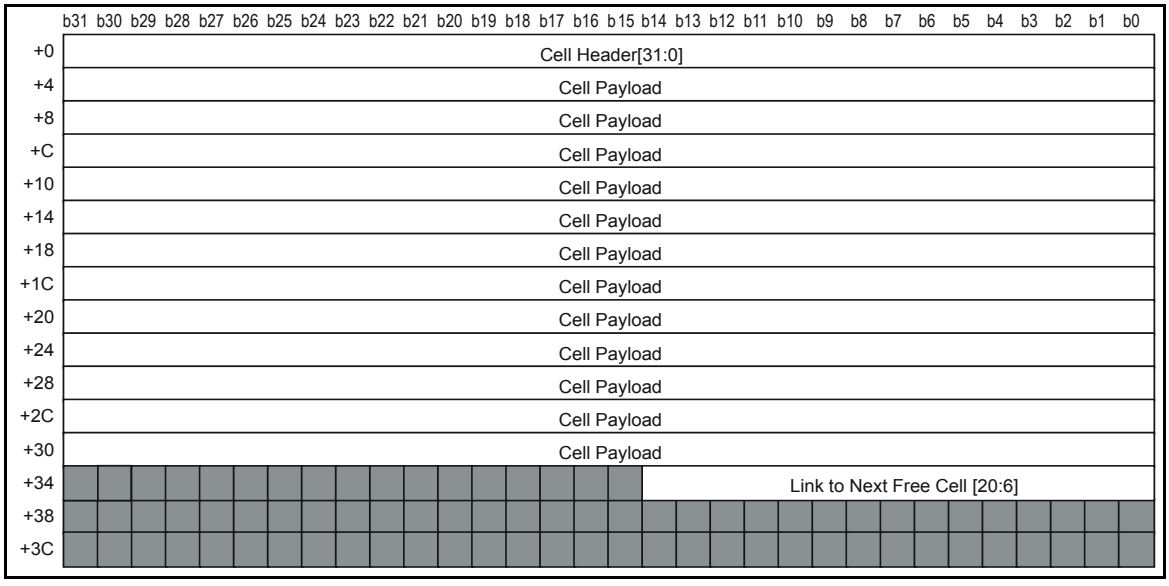


Figure 10 - Raw Cell Format (free cell)

Field	Description
Link to Next Free Cell	Points to the next free cell in SSRAM.

Table 10 - Fields and Description

There are multiple cell queues in the chip: 4 going to TX A, 2 going to TX B, and 1 to the RX CPU. Each one of them keeps a list of cells pointed to in external memory. Each cell queue is of a programmable 2^n size between 64 cells (4K bytes) and 8K cells (512K bytes). If a cell queue overflows, a status bit will be flagged in registers.

As is the case with packets, a cell handler module manages the read and write pointers to the queues. However, instead of being contained in a small internal memory, all pointers and configuration of each cell queue is contained in registers. However, a small internal memory is used to prefetch pointer to cells destined to each queue to prevent starvation. The format of this memory is described in the following figure.

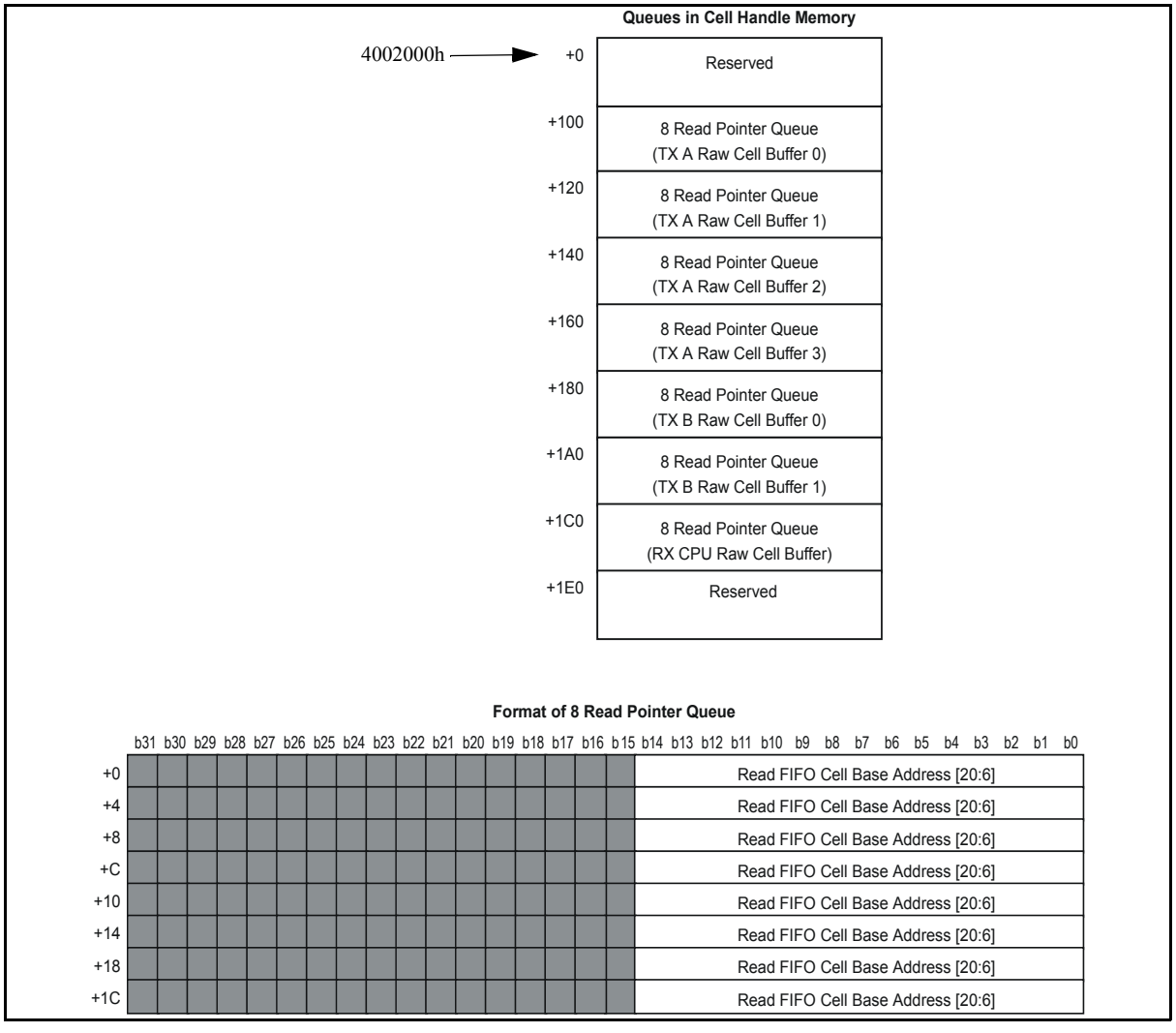


Figure 11 - Cell Handler Memory

4.0 Link Layers

This chapter describes the link-layer interfaces, as well as the packet reassembly structure used to transport their data.

4.1 Interfaces

The MT92210 device can support 3 different types of primary link layers: Ethernet, which uses the MII bus to communicate with an Ethernet PHY, ATM AAL5, which uses the UTOPIA bus, and Packet over SONET, which uses a 16-bit POS-PHY bus to interface with a SONET PHY.

Each of these modes has its own headers to identify IP packets over the link layer and the MT92210 is directly compatible to Physical Layer chips for each of the three interfaces.

In addition to its primary network interface, the MT92210 supports a secondary UTOPIA port that can be used to connect a data SAR to the chip, or to daisy chain several MT92210 chips together. This secondary port is always a UTOPIA port, independently of the configuration of the primary port. The chip is capable of treating IP packets from both ports and can convert from one link layer to the other if need be.

4.2 Ethernet Interface

The MT92210, when configured to operate in Ethernet mode, can operate using 10 or 100 Mbps Ethernet in either full- or half-duplex mode. When transmitting Ethernet packets, the chip will transmit the preamble, the packet itself and the CRC-32 that it has calculated over the entire packet. The chip will abort its packet transfer in case of collisions and it will back-off for a random period of time as specified by the Ethernet standard. If a packet retransmission is attempted 16 consecutive times, the packet will be discarded and an error will be flagged to registers.

When receiving Ethernet packets, the chip will verify the packet for the correct destination MAC address, the correct payload size, the correct CRC-32. When parsing the packet, the chip will accept destination MAC addresses that correspond to its own, as well as broadcast addresses; it can also be configured to accept all MAC addresses, as well as selecting canonical or non-canonical formats of MAC addresses. The length/type is checked and the packet is identified as IP/non-IP. The chip can also accept Ethernet headers that conform to the 802.1-p/Q specification. Packets received on the Ethernet interface can range in size between 64 bytes and 1500 bytes plus headers, in keeping with the Ethernet specification. Any larger or smaller packets will be discarded and an error bit will be flagged in registers. The reporting of these errors is necessary for the support of Ethernet MIB.

All transmission and reception of data over Ethernet is done through the MII interface that communicates with an Ethernet PHY.

4.3 Packet over SONET Interface

When using Packet over SONET, the MT92210 interfaces with a SONET PHY that allows packets to be transferred over SONET using PPP. When using this mode, transmitted packets are given a 1 or 2 byte PPP header before being sent onto the link: this header indicates that the packet is IP. When receiving packets, some filtering is done: because PPP is, by definition, point-to-point, all packets received are indeed destined to the chip, but padding packets must be deleted and non-IP packets must be flagged as such before being sent to the usual look-up flow.

The Packet over SONET interface uses a 16-bit POS-PHY bus used to communicate with SONET PHY. MT92210 doesn’t implement address bus therefore multi-PHY configuration is not supported.

The Packet over SONET interface can support packets ranging in size anywhere from 1 to 65535 bytes in length: any packet longer than 64K bytes will be discarded and an error will be flagged in registers. The chip will also discard any packets that are aborted by the PHY during transfer, as indicated by the **rxa_err** pin.

4.4 UTOPIA Interface

The MT92210 has a primary link layer port (Port A) that can be configured as Ethernet, ATM or Packet over SONET. In addition, it has a secondary port (Port B) that is always configured to operate as an ATM port. Port B can be used to interoperate with a secondary data SAR, or to daisy chain several MT92210 devices together onto a single network connection. By keeping the same secondary port configuration independently of the mode in which

the primary port operates, the MT92210 can interface with the same external SAR in all modes with minimal changes.

Cells received on the secondary UTOPIA port, as well as any cells received on the primary port when it is configured to operate in ATM mode, navigate through the UTOPIA module to reach their final destination. To multiplex and de-multiplex traffic, the UTOPIA module uses input and output FIFO to contain cells. All cells sent to the module are written into 4-cell input FIFO, which not only buffers a small amount of data but also serves the synchronization needs of the various network clocks. These cells are then passed by a look-up engine whose job is determining the VC number, as well as whether they should be kept or discarded. Cells arriving from either of the UTOPIA ports need to be looked-up. To do so, 2 look-up tables are mapped in the external SSRAM. To point to an entry in the SSRAM, a combination of bits from the VPI and VCI is used. Each look-up table can contain up to 2^16 entries = 64K entries, so up to 16 bits of the header can be used for the address.

Once the look-up engine has determined the VC number to which the cell is destined, it writes it to the appropriate output FIFO. The output FIFO are larger, since they are necessary to buffer the peaks that occur in traffic. The output FIFO are 8 cells deep. From these output FIFO, AAL5 cells are sent to the RX link agent for reassembly.

On the input side (UTOPIA reception), the UTOPIA ports can be configured through registers to behave as PHY or ATM layers on the UTOPIA bus, depending on which chip is being interfaced with. In addition, each port has an individual enable bit which prevents all traffic from being received on that port. Each port detects the parity on the UTOPIA bus and any parity error will be reported to registers. Finally, null cells can also be enabled or disabled on a per-port basis and any cells containing null headers will be deleted if this feature is enabled. Null headers can be identified under either UNI or NNI conditions.

On the output side (UTOPIA transmission), the ports can also be configured as PHY or ATM layers. The TX interfaces can also be configured to operate in multi-PHY mode, meaning that they will only drive the port's **tx_d**, **tx_prty** and **tx_soc** pins if this port has been selected to drive data onto the UTOPIA bus. Note that multi-PHY mode only applies when the port is configured to operate as a PHY layer.

The Look-Up Table (LUT) of the UTOPIA is the central element to the module's behavior. It analyzes every cell received from one of the UTOPIA ports and determines where the cell goes. A cell coming from port A or B will first have its header compared with match and mask fields which indicate what are the acceptable header values coming from this port. The way the match and mask operate is that the mask indicates which bits of the header must be fixed and which may take on any value. For all bits whose mask value is '1' (fixed), the match indicates what this value must be. This method is applied bit-wise to the VPI and VCI portions of the header. Any cell not meeting these criteria will be routed according to the **rx_normal_vc_dest** or **rx_oam_vc_dest** bits contained in registers. These fields indicate, for cells whose header does not match, where they should be routed: once again, these fields support unicast, multicast or broadcast routing of cells. Depending on the OAM bit of the header, either **rx_normal_vc_dest** or **rx_oam_vc_dest** will be used, to allow management cells to be routed differently from data cells. The routing fields are different for each input port.

However, if this validation procedure is passed, the look-up engine uses a certain number of bits from the VPI and VCI of the cell header to determine the look-up entry corresponding to the cell. The number of bits of VCI used is determined by the **vci_n** field, which is unique per port and can be configured from 0 to the full 16 bits of header: note that the bits used are always the lower bits of the VCI. The rest of the bits are taken from the low bits of the VPI, up to the full size of the look-up table (anywhere from 2^10 entries to 2^16 entries). These bits are then concatenated to form a look-up address offset, which is added to the look-up table's base address to form the absolute address of the look-up entry. The information at this address then reveals whether the cell will be kept and to which cell queue it belongs, as well as a VC number if it is AAL5. The routing can be performed differently for normal and OAM cells: in a typical application, OAM cells will be routed to one of the raw cell queues, while user cells will be sent to an AAL5 VC.

The format of the look-up table entries is described below:

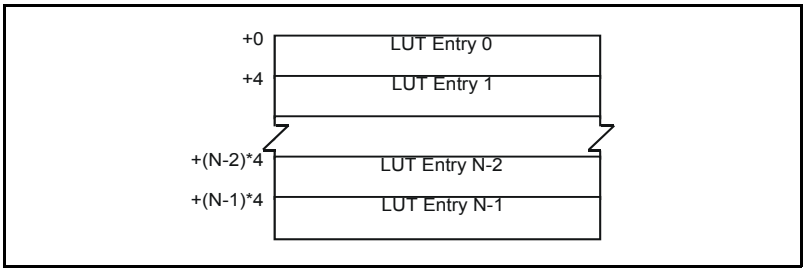


Figure 12 - UTOPIA Look Up Table

Indexing within the UTOPIA LUT is done using programmable VPI/VCI bits. See registers 420h to 42Ah and 440h to 44Ah.
N is a 2^n number between 1024 and 65536.
The base address starts on a boundary equal to the size of the LUT.
The LUT is located in SSRAM C.

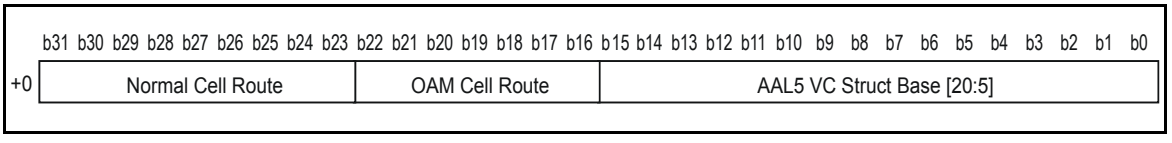


Figure 13 - UTOPIA LUT Entry Format

Field	Description
Normal Cell Route	“xxxxxxxx1” = TX Link A Raw Cell Buffer 0; “xxxxxxx1x” = TX Link A Raw Cell Buffer 1; “xxxxxx1xx” = TX Link A Raw Cell Buffer 2; “xxxxx1xxx” = TX Link A Raw Cell Buffer 3; “xxxx1xxxx” = TX Link B Raw Cell Buffer 0; “xxx1xxxxx” = TX Link B Raw Cell Buffer 1; “xx1xxxxxx” = RX CPU Raw Cell Buffer; “x1xxxxxxx” = reserved; “100000000” = AAL5 VC;
OAM Cell Route	“xxxxxx1” = TX Link A Raw Cell Buffer 0; “xxxxx1x” = TX Link A Raw Cell Buffer 1; “xxxx1xx” = TX Link A Raw Cell Buffer 2; “xxx1xxx” = TX Link A Raw Cell Buffer 3; “xx1xxxx” = TX Link B Raw Cell Buffer 0; “x1xxxxx” = TX Link B Raw Cell Buffer 1; “1xxxxxx” = RX CPU Raw Cell Buffer;
AAL5 VC Struct Base	When a VC’s Normal Cell Route indicates that its cell should be sent to the Packet Reassembly module (for AAL5 reassembly), this points to a Packet Reassembly structure.

Table 11 - Fields and Description

4.5 Packet Reassembly

When communicating only IP packets, the MT92210 can use Ethernet, Packet over SONET or ATM as its primary network interface. The primary interface must be a Utopia bus to carry AAL0 cells, other ATM cells that can carry signaling, and packets broken down and carried over AAL5. Any packets transmitted or received over AAL5 on the link will be done so one cell at a time, contrarily to Ethernet and Packet over SONET, which both transfer and receive entire IP packets. On the transmission side, this is not a problem: each packet is broken down into as many AAL5 cells as necessary and the cells are then sent one at a time over UTOPIA.

The reception side is a little trickier: as ATM cells are received, they must be targeted to a connection in order to determine if they are AAL5 carrying IP or another protocol, or "raw" cells which will be routed to another port or to software. To do so, the chip consults an ATM header look-up table. The look-up table is contained in external SSRAM and can use up to 16 header bits to determine the destination of the cell. The result of this look-up points to a destination field for non-OAM cells as well as one for OAM cells (note that OAM cells cannot be AAL5). The cells are then sent to their appropriate destinations. If the cells are AAL5, an AAL5 VC structure base number is also listed.

The AAL5 reassembly structure contains all information about the current packet and the connection to which it belongs, such as the number of cells contained in the packet so far, and diagnostic information indicating how many bytes, cells and packets have been received on this connection since start-up. It also contains a Flow Table Pointer: this pointer uniquely identifies a Flow, which corresponds to a logical subnet number. The Flow Table Pointer can be modified later in the packet's life according to any MPOA tags, MPLS labels or ELAN-IDs it may contain. If none of these modifications have taken place, then the Flow Table Pointer corresponds directly to the VC number.

When the last cell is received and the packet is reassembled successfully, the complete packet is checked for correct length and correct CRC: an error in either of these will cause it to be discarded. If the packet passes these tests, it will be sent to the packet identification queue, where it will be routed to its final destination.

When the packet is ready to be treated, the look-up table will indicate whether it is a data packet, or it needs to be sent through the regular IP packet flow. It is to be noted that, to save overhead, some implementations eliminate IP/UDP headers to save bandwidth. Like all other network interfaces, AAL5 can also support voice packets both with and without RTP headers. If the IP/UDP packet routing is chosen and the IP header is present, the look-up proceeds using the IP and UDP headers (and possibly the RTP synchronization source) to identify the packet. However, if the IP and UDP headers are absent, the 24-bit Flow Table Pointer is added to the binary tree look-up key. Only the RTP SSRC can be used along the Flow Table Pointer in the search, as long as RTP is present in the packet. Otherwise, the packet will be looked-up using only the Flow Table Pointer.

IP packets carried over AAL5 can be encapsulated using Classical IP over ATM or LANE version 1 or 2. Classical IP over ATM uses an 8-byte SNAP/LLC header at the beginning of the packet to identify the type of the packet (e.g. IP). Packets using LANE have an Ethernet-compatible header before the IP header, containing the Destination and Source MAC addresses corresponding to the packet, as well as an Ethernet Length/Type field that, in the same way as the SNAP/LLC type, identifies the protocol above it (IP or other). LANE v2 also uses LLC encapsulation and contains an ELAN-ID that may be used to resolve a logical subnet number. LANE headers may be compatible to Ethernet p/Q. The Destination MAC address in the LANE header will be checked in the same way as it would be in an Ethernet packet: the chip will accept MAC addresses corresponding to its own, as well as broadcast MAC addresses. MAC address checking can also be disabled and all packets will be accepted.

When the primary network interface is configured as Ethernet or Packet over SONET, a single Reassembly structure is used and all packets are routed to this structure. Since Ethernet and Packet over SONET do not break down packets, interleave them, or carry several packets over different VC, the packets always arrive contiguously, thus not requiring more than 1 structure.

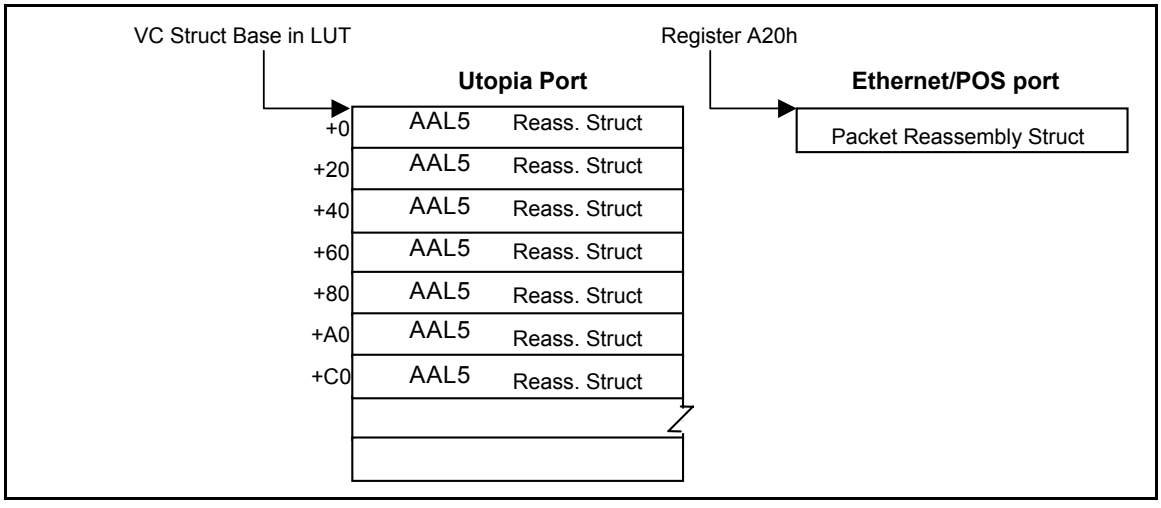


Figure 14 - Location of Reassembly Structures

This is the format of the AAL5 Packet Reassembly Structure (also used for Ethernet and POS).

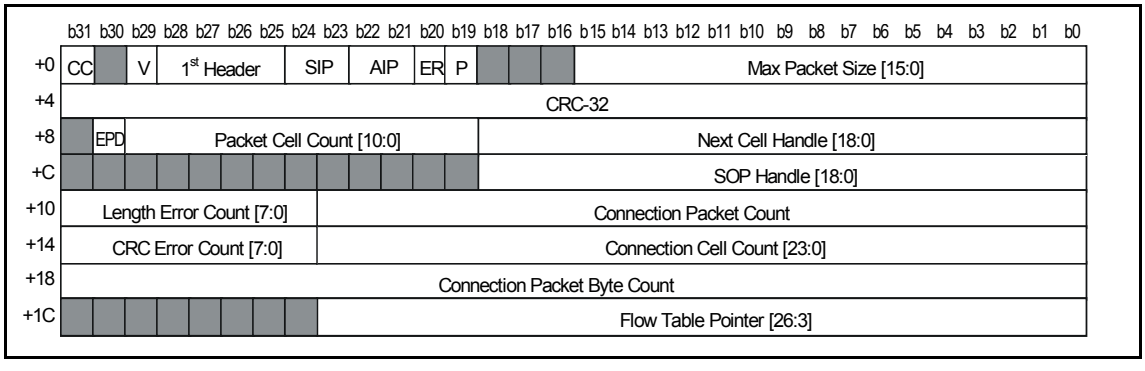


Figure 15 - Packet Reassembly Structure

Field	Description
CC	CRC Check Enable. This should always be set to ‘1’ if communicating with an AAL5 agent, and set to ‘0’ if this structure is receiving from an Ethernet or POS agent.
V	Valid bit. When ‘0’, any cell arriving on this VC number will be discarded as if it never arrived. Caution: this bit can be cached by hardware, so when it is cleared, the rxaal5_empty_cache bit in registers must be set (it will be cleared by hardware when the cache is empty).
1st Header	“0000” = VC starting with the LLC Header; “0001” = VC starting with a PPP Header; “0010” = VC starting with an IP Header; “0011” = VC starting with an MPLS Unicast Header; “0100” = VC starting with an MPLS Multicast Header; “0101” = VC starting with an MPOA Tag; “0110” = VC starting with a LANE-Ethernet/802.3 Header; “0111” = VC starting with a UDP payload (with or without RTP); others = reserved.
SIP	MPLS IP indicator. “11” = reserved; “10” = RTP/UDP payload; “01” = IP, “00” = must be looked-up. This value can be overridden later in the packet parsing process by a binary tree look-up using the MPLS label. Usually set to “00”.
AIP	MPOA IP indicator. “11” = RTP/UDP payload, “10” = IP, “01” = MPLS unicast, “00” = must be looked-up. This value can be overridden later in the packet parsing process by a binary tree look-up using the MPOA tag. Usually set to “00”.
ER	ELAN Resolved. When ‘1’, any ELAN-ID present is considered resolved, and will not have to be looked-up to generate a Flow Table Bit Pointer. Usually set to ‘0’.
P	Priority of packet. ‘0’ = LP; ‘1’ = HP.
Max Packet Size	Maximum packet size in bytes (inclusive), including link layer protocol headers and up. Minimal value 128, maximum value 65535. This number never includes the ATM header, but includes the PPP header in Packet over SONET, and includes the Ethernet header and a 2-byte LANE header in Ethernet. This number never includes the frame CRC.
CRC-32	Calculated by hardware during packet reception. Used to compare with CRC inside the received packet.
EPD	‘0’ = all is well; ‘1’ = Early Packet Discard. In this case, the packet will be thrown out when the last cell is received. This can be set because the packet length exceeded the Max Packet Size. This bit is only used by hardware; set to ‘0’ initially by software.
Packet Cell Count	Counter of the number of cells contained in the packet so far. Should be reset to 000h. This counter excludes the Next Cell Handle, since this handle has been seized for a cell that has not yet arrived.
Next Cell Handle	Pointer to the block that has been pre-allocated for storing the next cell payload when it is received.

Table 12 - Fields and Description

Field	Description
SOP Handle	Pointer to the first block of the chain that contains the packet payload.
Length Error Count	Indicates the number of length errors that have been detected on this connection. Length errors are reported when the received AAL5 length is 0, when the AAL5 length is too large to fit in the number of cells received in this packet, when the packet length (either the explicit AAL5 length, or the implicit length in Ethernet or POS) is greater than Max Packet Size, and when the number of cells received in this packet is too large for the Max Packet Size (i.e. the total number of cells received contains 0x38 or more payload bytes than the Max Packet Size).
Connection Packet Count	Counts the total number of packets received on this VC to date.
CRC Error Count	Counts the total number of CRC errors detected on this VC since initialization
Connection Cell Count	Counts the total number of cells received on this VC to date.
Connection Packet Byte Count	Counts the total number of packets received on this VC since initialization
Flow Table Pointer[26:3]	This field represents the logical subnet number on which the packet is received. The logical subnet number is initially bound either to the link (in the case of Ethernet or POS) or to the VC on which the packet is carried (in the case of ATM). This field can be overridden later on by protocols that allow VCs to carry packets on multiple subnets, such as LANEv2 or MPOA. The Flow Table Pointer points to a single bit in SSRAM C.

Table 12 - Fields and Description

5.0 RX/TX Data Flows

This chapter describes the data flows for all packets received and transmitted.

5.1 RX Data Flow

This section resumes the typical propagation of voice data throughout the MT92210 in the receive direction, from the link to the H.110 bus.

Packets arriving off the link are written into memories, either a cell input FIFO if the link is configured as a UTOPIA interface, or a packet input FIFO if the link is configured as an Ethernet or Packet over SONET interface. For the ports that are configured as UTOPIA, the cells then go through the UTOPIA look-up process, which indicates what their nature is, either raw cells going to a cell destination, or AAL5 cells to be reassembled into a packet. After they are looked-up, they are written into the RX link A or RX link B cell FIFO.

If port A is configured as Ethernet or POS, its packets go through a packet to block conversion process, where they are broken down into 48-byte blocks. In addition, Ethernet headers are converted to LANEv1 headers, and POS is mapped as PPP over AAL5. The packets are then written into the RX link A cell FIFO. As of this point, all packets look identical within the chip, independently of the link interface.

The cells are then read by the packet reassembly process and written into external SDRAM until the packet to which they belong is complete. Cells that are tagged as non-AAL5 pass by the packet reassembly process but are routed immediately to their own destination, which is already known because it was indicated in the UTOPIA look-up table. These cells may be sent back onto one of the TX links, or they may be routed directly to the CPU.

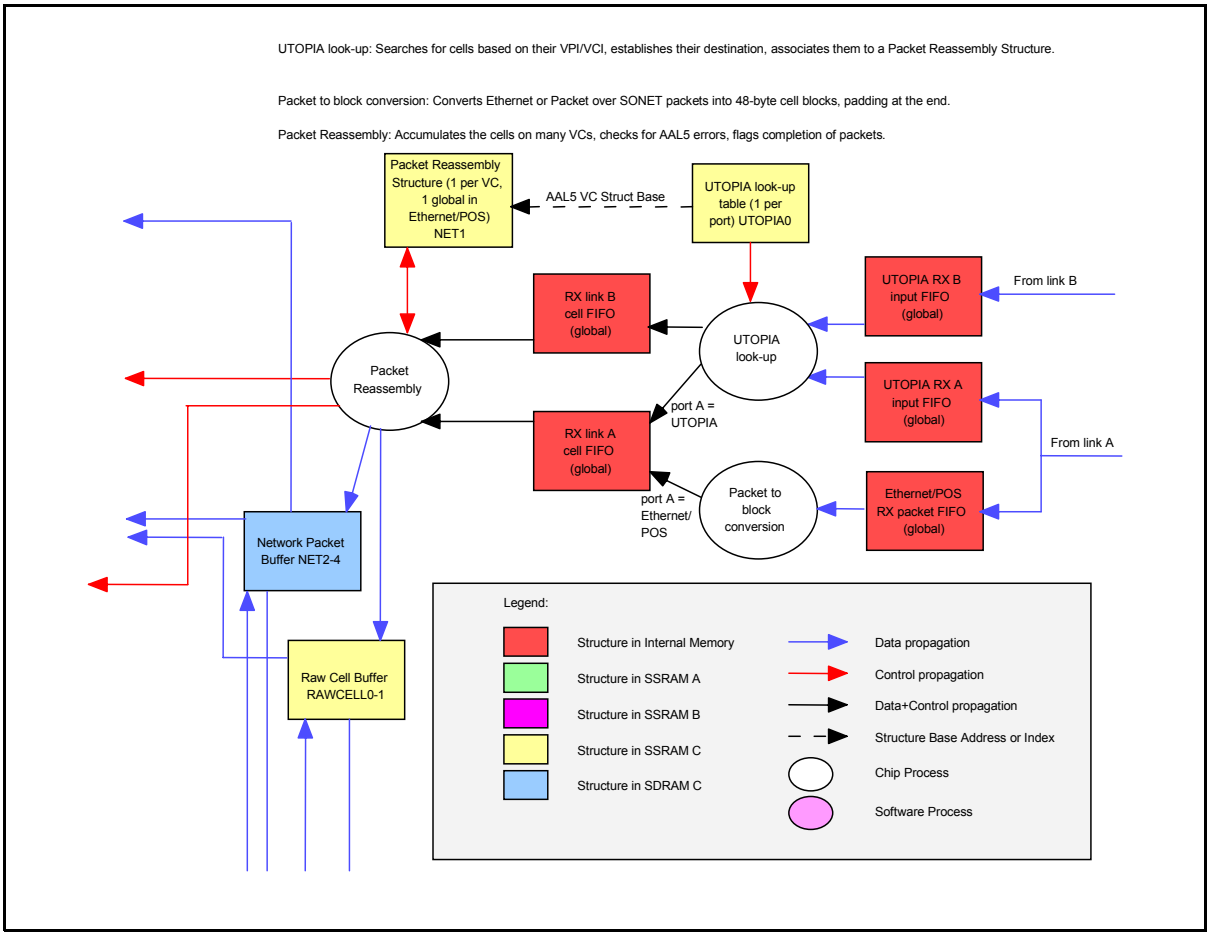


Figure 16 - Rx Flow 1

Cells of AAL5 VC will go to the packet reassembly process; when a packet completes in the packet reassembly process, its handle is sent to the packet identification buffer. The packet identifier parses through the packet, identifying various headers and extracting the identification key with which the packet will be searched for. The packet identifier also consults the Next Header memory and the Profile Memory to decide what to do if it encounters certain next header or option values.

Once the packet parser has assembled all of the necessary headers, it consults the initial search structure to decide what to do with the packet. In the case of a voice packet, the structure should tell it to search in the binary tree using a specific profile. It then passes the packet to the identification key hashing process. This process performs CRC on the identification key and annexes the profile number to it to obtain a full 60-bit key, which is then used to search for the packet in the binary tree.

When the correct node is found in the binary tree, that node points to a post-search confirmation structure. The headers contained in this structure are then compared to the headers used initially to form the packet's identification key. In addition, the flow table is searched to verify whether the destination IP address of the packet can be received on the current logical subnet. If both tests are passed, then the packet matches the post-search confirmation structure, which then indicates what its destination should be. If the packet does not match, then it keeps on being searched until it hits a post-search confirmation structure it matches with, or it hits the default structure for its profile that gives it a default destination.

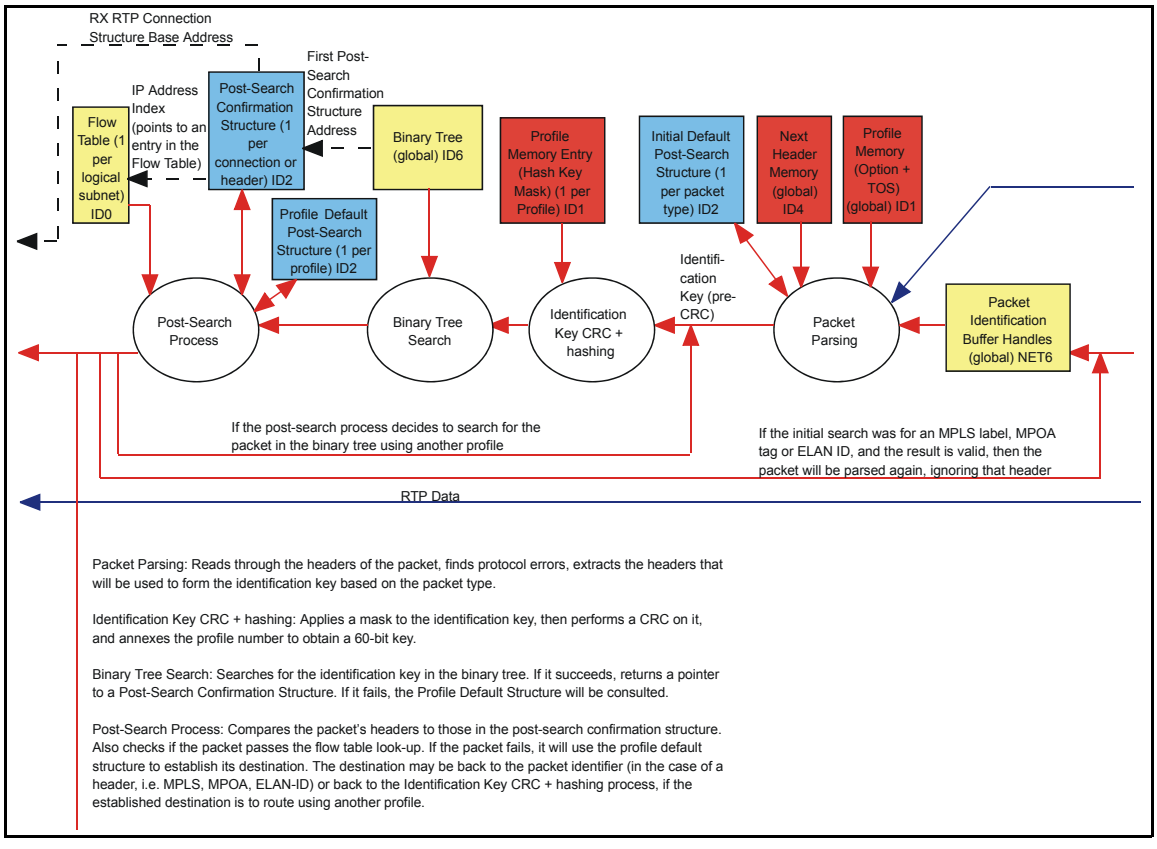


Figure 17 - Rx Flow 2

The packet's destination may be one of the TX link packet buffers, the Network CPU packet buffer, or the packet disassembly module. If the packet's destination is the packet disassembly module, then the post-search structure will contain a pointer to a valid RX RTP Connection Structure. The packet is then written into the Packet Disassembly Memory and read on the other side by the packet disassembly module.

The packet disassembly module reads the RX RTP Connection Structure, and based on the Marker bit and Payload type of the packet (or its UDP length, if it carries no RTP), it will search in the Payload Byte/Marker Bit table for the entry number that corresponds to the current packet. The entry number will point to 1 of 16 RX RTP channel structures, which may be xxPCM Channel Structures, HDLC Channel Structures, CPU Channel Structures or may indicate to delete the packet.

An RX RTP xxPCM Channel Structure will contain pointers to one or many circular buffers in which its PCM data will be written, depending on how many bearers it carries. It will also contain a pointer to an RTP Common PDV Absorption Structure, which will be used to do de-jittering on the bearer's payload. Many xxPCM Channels can share the same Common PDV Absorption Structure, which allows them to be de-jittered in common and ensures that a slip on one will force a slip on all.

If the selected channel structure is an RX HDLC Channel Structure, then it will contain a pointer to an RX HDLC Stream/Buffer Control Structure, which will itself point to a circular buffer. In HDLC, channels are policed independently, but are then merged into a common circular buffer for the entire stream. The RX HDLC Stream/Buffer Control Structure will contain the base address as well as the read and write pointers to this circular buffer.

If the selected channel structure is an RX CPU Channel Structure, it will contain a pointer to an RX CPU Buffer Control Structure. Except for the fact that the CPU Channel Structure will not request HDLC framing to be

performed on the packet and that no HDLC address or control byte insertion will be done, it is otherwise identical to the RX HDLC Channel Structure.

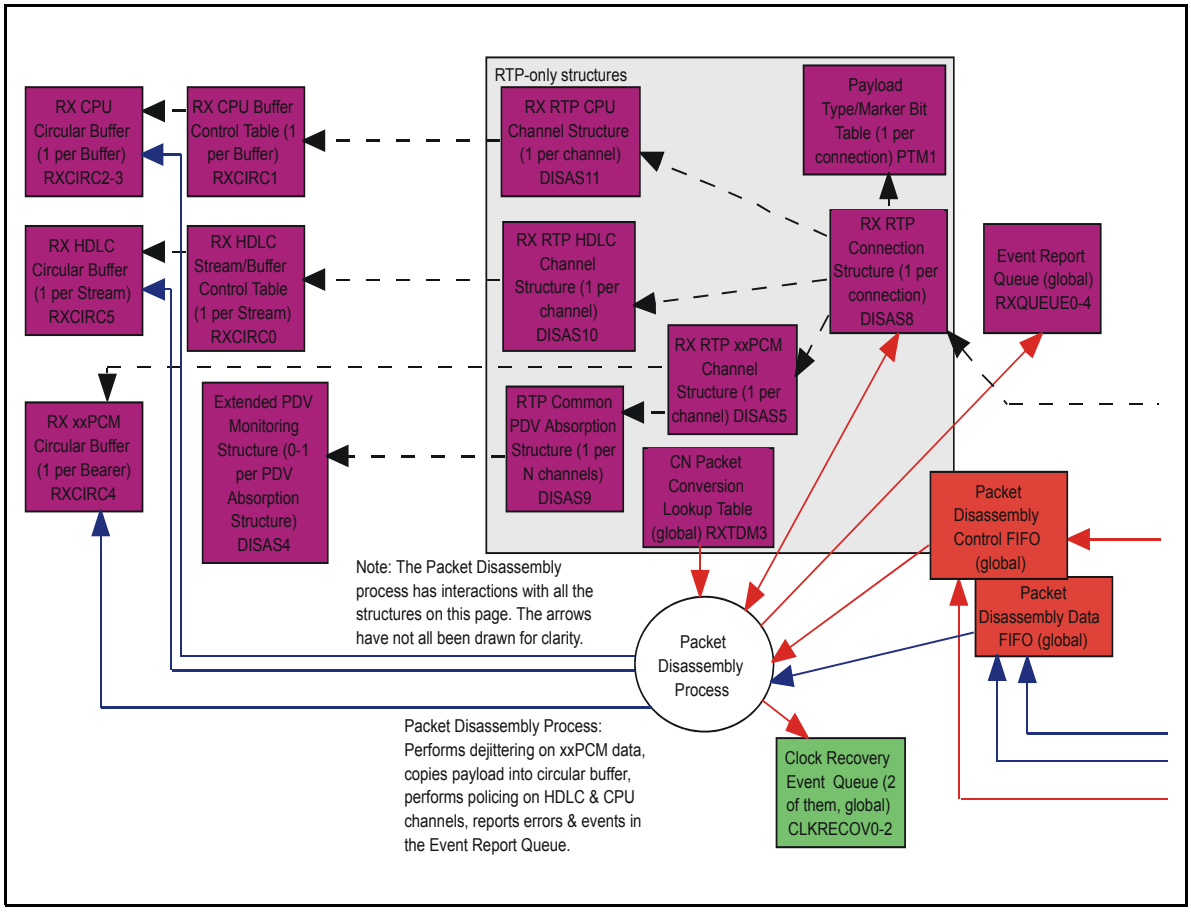


Figure 18 - Rx Flow 3

Once the bytes have been copied into the correct circular buffers, only the RX TDM process remains. The RX TDM process is not event-driven with regards to the other processes: by following the Data Flow, we see that, from the Packet Reassembly Process, each process has been triggered by the previous one completing and handing it over the packet. The RX TDM only communicates with the Packet Disassembly process through the TDM pointer.

The RX TDM process reads bytes out of the PCM and HDLC circular buffers and places them onto the H.110 bus. Each frame it reads one byte out of each xxPCM circular buffer and places it on the appropriate TSST; it uses a channel association memory to make this connection. It also compensates for underruns and packet losses by inserting SSRAM Tone Buffers, SDRAM Silence Buffers or Padding Octets. In HDLC, when there is no data to be sent on a stream, it sends the idle code.

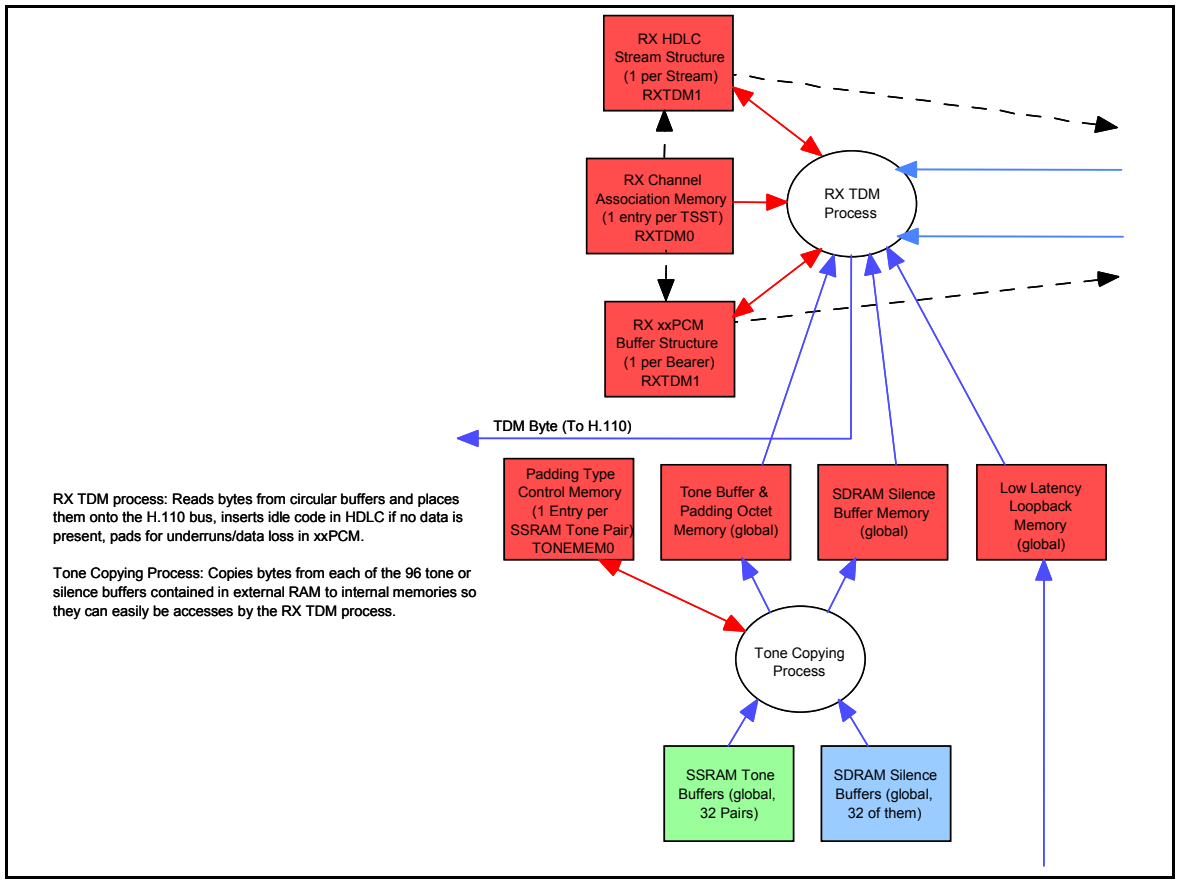


Figure 19 - Rx Flow 4

5.2 TX Data Flow

The following section summarizes the typical propagation of voice data throughout the MT92210 in the transmit direction, from the H.110 bus to the packet link.

Data from a TSST is read off the H.110 bus and is associated with either an xxPCM buffer or an HDLC stream by the TX channel association memory. This memory can associate up to 1023 TSSTs with up to 1023 xxPCM buffers or 512 HDLC streams. The byte is then treated by the corresponding entry in the TX control memory. In xxPCM, this structure contains the base address and size of the circular buffer to which the TSST is associated. In HDLC, this structure also contains the base address and size of the circular buffer, as well as all the internal context used to perform HDLC de-framing. The byte is then written into its circular buffer.

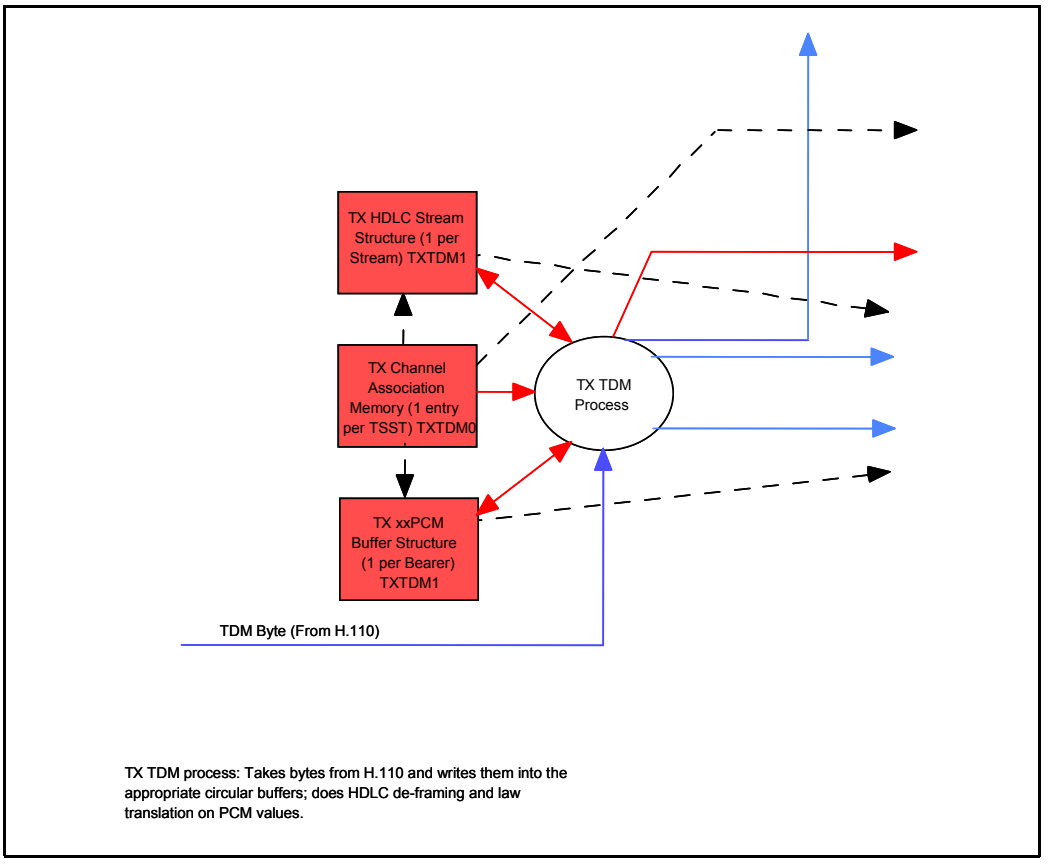


Figure 20 - Tx Flow 1

In HDLC, when a packet completes, its HDLC stream number is used to index in the HDLC Stream to HDLC Address LUT structure, and from there its HDLC address is used to index into the HDLC Address LUT. This structure, in turn, will point to the TX Connection structure that will be used by the packet assembly block to create the packet. An event is generated directly into the Assembly Event Queue; from there, the packet assembly module reads the event (and with it the address of the TX Connection Structure), then reads the packet payload itself from the HDLC circular buffer, creates the complete packet and writes its data into the Packet Assembly Data FIFO; it also writes the control associated to it in the Packet Assembly Control FIFO.

For xxPCM channels, the originator of the Assembly Events is the Service Timer. The Service Timer process scans all the Service Indicator tables and when it hits a valid indicator, writes an assembly event in the assembly event queue. From here it goes to the packet assembly process where a TX Connection structure is used to assemble the packet. The TX Connection structure may point to a TX Silence Suppression structure that would be used to perform silence suppression on the xxPCM samples.

In both HDLC and xxPCM, if the packet is RTP, a TX RTP Header Structure will be used to assemble the correct headers at the beginning of the packet. The TX RTP header structure contains the data values of all the headers, all the information needed to assemble the variable headers of the packet, as well as the packet's destination.

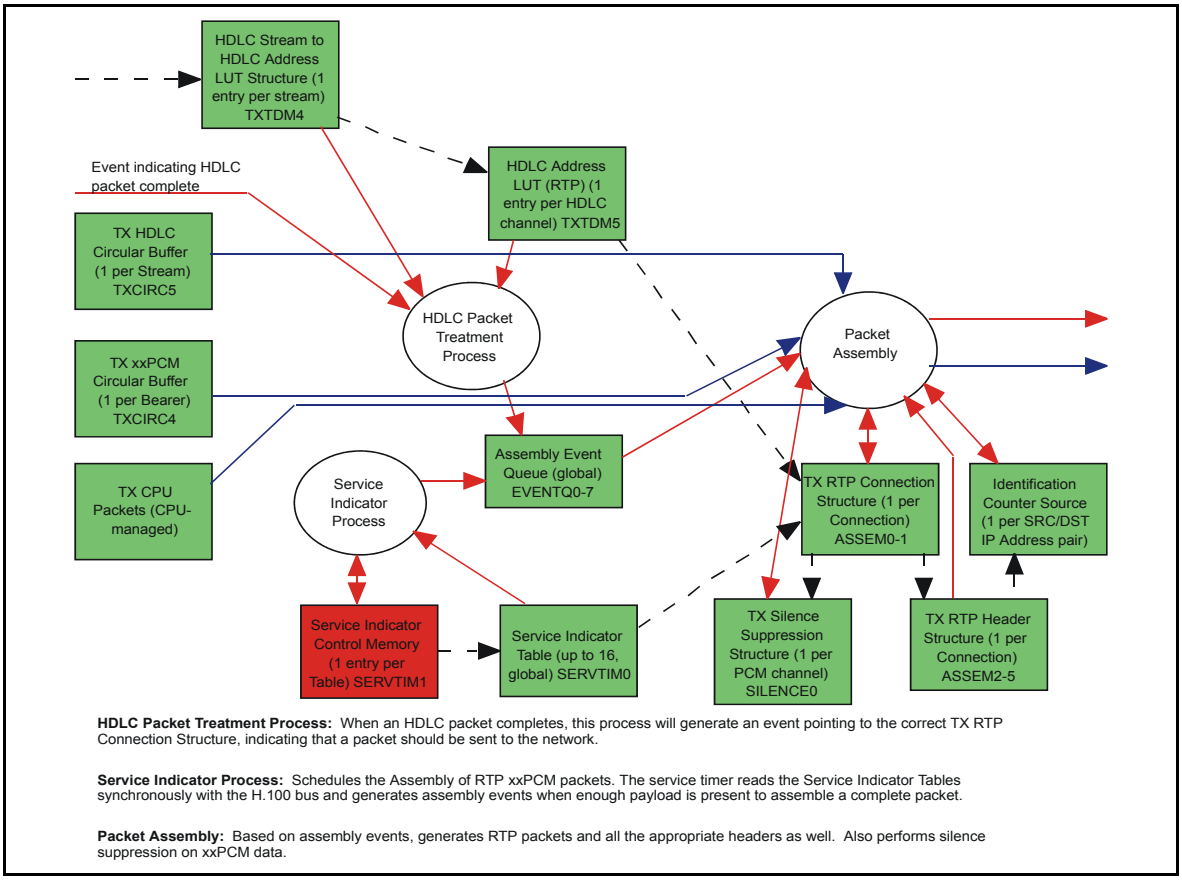


Figure 21 - Tx Flow 2

Once written in the Packet Assembly Control & Data FIFOs, RTP packets will be read by the Assembly Copying Process that then writes the packets into their destination buffer. The payload of RTP packets is written into the Network Packet Buffer, while the handle to the packet is written into one of 6 destinations: one of the 4 TX Link Packet Buffer Handles, the Network CPU Packet Buffer Handles (for internal diagnostic), or the Packet Identifier Buffer Handles (for internal loopback).

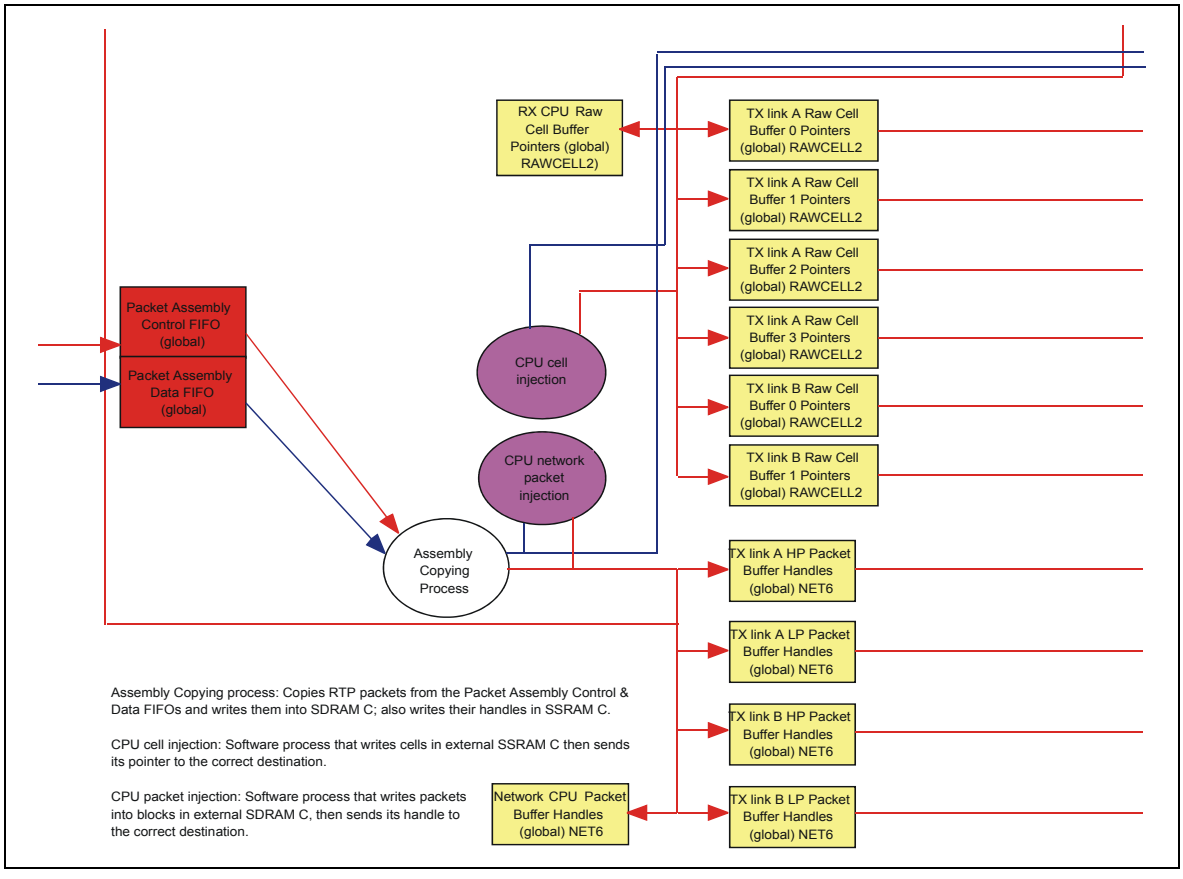


Figure 22 - Tx Flow 3

Raw cells written into one of the TX Link Raw Cell Buffers will be transmitted onto the appropriate TX link port. Raw cells can be transmitted onto any port configured as ATM UTOPIA. Either the TX Link A or TX Link B Copy Process will read the cell from the Raw Cell Buffer and copy it into either the TX Link A Packet/Cell FIFO (for port A) or the TX Link B Cell FIFO (for port B), from where it will be transmitted onto the link.

Packets written into one of the TX Link Packet Buffers will also be read out by the corresponding TX Link Copy Process: if port A is configured as Ethernet or Packet over SONET, then the data will be written into the TX Link A Cell FIFO. From there, the Block to Packet Conversion process will write the packet as a contiguous packet into the TX Link A Cell/Packet FIFO, from where it will be transmitted onto the link. If port A is configured as ATM, however, packets will get the same treatment as raw cells: written directly into the TX Link A Packet/Cell FIFO, then to the link. Packets on port B also go through the same treatment as raw cells: they are copied into the TX Link B Cell FIFO and transmitted onto the link.

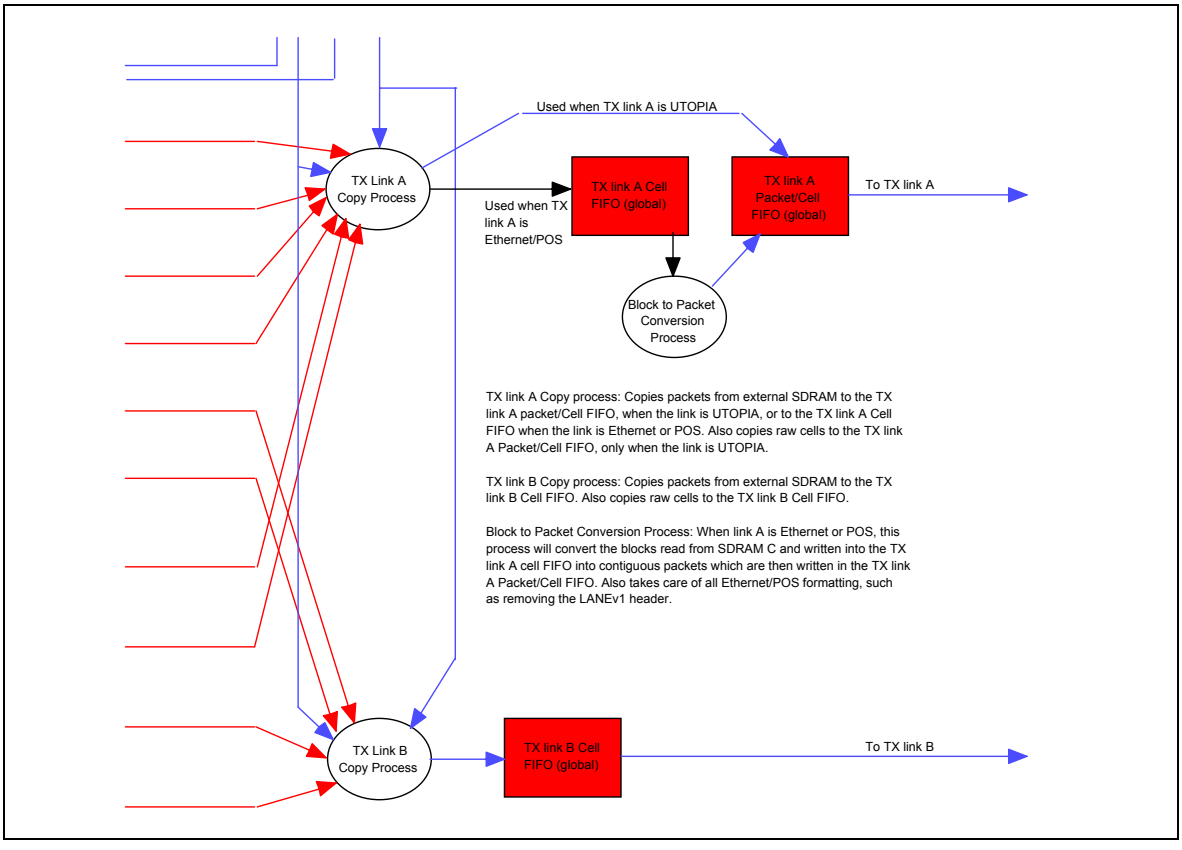


Figure 23 - Tx Flow 4

6.0 Packet Identification

Packets received from the network must go through a look-up process to determine what is their destination. The first step in choosing a final destination for a packet is determining its packet type.

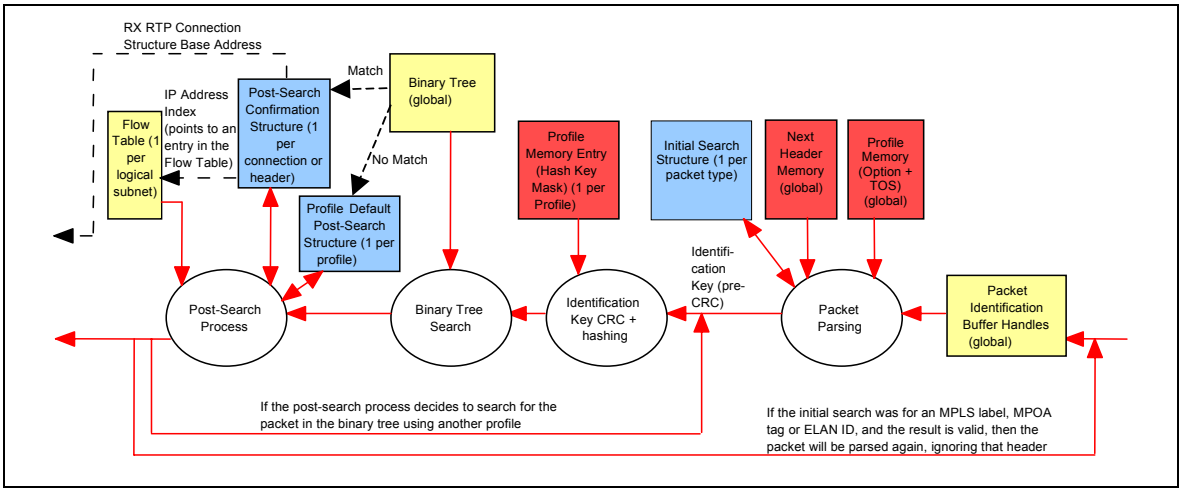


Figure 24 - Packet Identification

6.1 Packet Types

The two most important decisions are whether or not the received packet is an IP packet, and whether or not it is a null-encapsulated packet. A null-encapsulated packet is a packet that is transported over AAL5 and whose IP and UDP headers are eliminated to save bandwidth.

Regular IP packets must also be sub-classified: their IP version (v4 or v6) must be determined, and the protocol used below IP is established. Non-fragmented UDP packets are grouped together in one category, while fragmented packets or non-UDP packets are grouped separately.

IP packets with a version other than v4 or v6 are routed separately. Non-IP packets are also routed separately. And lastly, three “special” headers can be decoded and used to take decisions on packets: these are the MPLS label, the MPOA tag, and the LANEv2 ELAN-ID.

Table 13 lists all the packet types that MT92210 can identify. Each type has an associated Initial Search Structure, which has a fixed location in SDRAM C. Any received packet must fall into one of the type, and will be handled as per the instructions in Initial Search Structure. Packets that contain one or many of these headers may require multiple look-up iterations before being assigned to their final destination

No.	Packet Type	Initial Search Structure Address in SDRAM	ID Key Format
1	Null Application Data or Raw AAL5 Packets	1000h	3
2	Non-IP Packets	1080h	-
3	IPv4 with Non-Fragmented UDP Protocol	1100h	1
4	IPv4 with Fragmented UDP Protocol/non-UDP Protocol/Invalid Protocol	1180h	1
5	IPv6 with Non-Fragmented UDP Protocol	1200h	2
6	IPv6 with Fragmented UDP Protocol/non-UDP Protocol/Invalid Protocol	1280h	2
7	IPvX (Other than version 4 or 6)	1300h	-
8	ELAN-ID Lookup	1380h	3
9	MPOA Look-up	1400h	3
10	MPLS Look-up	1480h	3

Table 13 - Packet Types and Initial Search Structures

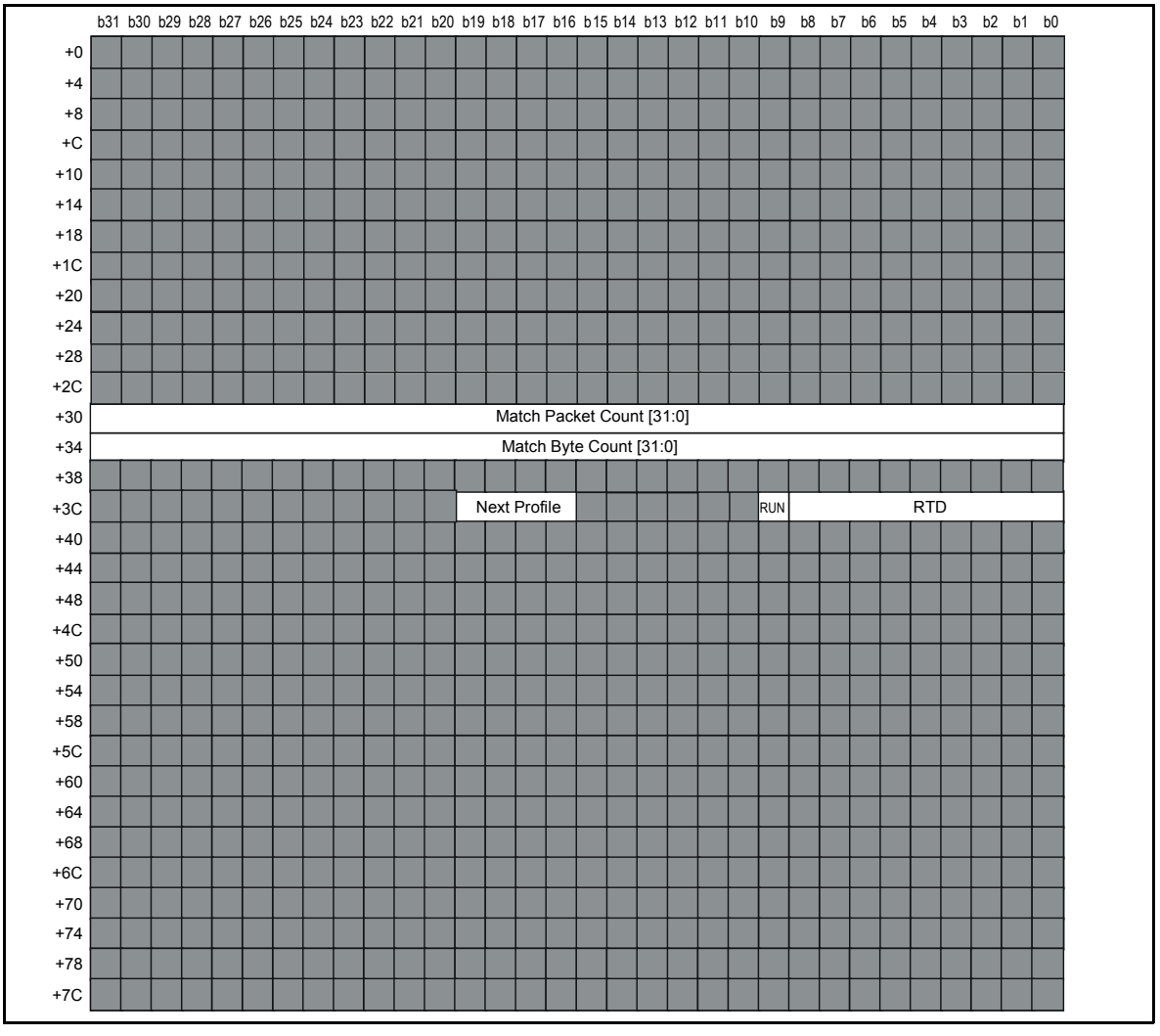


Figure 25 - Format of Initial Search Structure (Refer to Figure 19 for field descriptions)

6.2 Packet Parsing

Before getting to the IP level, the MT92210 must parse through all of the link-layer information and headers contained in the packet. The MT92210 can support packets whose link-layer information begins with an LLC header (Classical IP over ATM, LANE v2), a PPP header (Packet over SONET, PPP over ATM), an Ethernet header (Ethernet, LANE v1), an MPOA tag (MPOA over ATM), an MPLS label, an IP header or application data. It then plows through the successive layers of link layer headers and network headers to extract its identification key made up of a combination of fields such as source and destination addresses, UDP source and destination ports, RTP SSRC, MPLS,/MPOA/ELAN-ID tags as available. The parser gets the very first protocol information from “1st Header” in Packet Reassembly Structure, and is able to find out all following headers therefore determines the packet type automatically.

Some IP headers can get special treatment from the packet parser. Specifically, the “next header” field found in IPv6 (and that, in many cases, can also be used as the “protocol” field in IPv4) can force special treatment of the packet, such as deleting it or routing it as a non-UDP packet. In a similar fashion, the “options” field found at the end of the IPv4 header as well as the “options” found in the hop-by-hop options header and the destination options

header can also be used to make decisions regarding the packet. With this ability, the MT92210 can ensure that any packet containing a certain header will be deleted or always sent to the CPU buffer, for example.

The information indicating how to treat the various “next header” values is contained in the Next Header memory. The information indicating how to treat the various “options” is contained in the Profile Memory, which will be shown later in Figure 28. The format of the Next Header memory is the following:

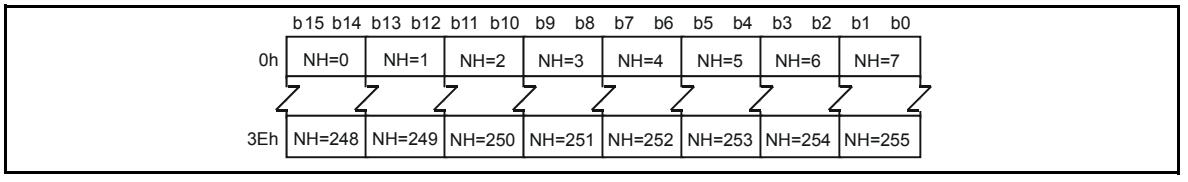


Figure 26 - Next Header Memory

Field	Description
NH	Next header. Indicates what to do when the hardware encounters this “next header” value. “00” = delete; “01” = route as non-UDP / fragmented UDP (to software); “10” = process next header in IPv6 only; “11” = process next header in IPv4 and IPv6.

Table 14 - Fields and Description

6.3 Look-up

Once the type of the packet has been found, it must be searched for in the look-up system. The packet will initially be routed to a default node (i.e. the Initial Search Structure), depending on its packet type (see Table 13 for the exhaustive list). The default node is the first step in deciding what to do with the packet: it may indicate that the packet must be discarded, routed to the CPU packet buffer or routed back onto the network on ports A or B. It may also indicate that a more detailed search must be performed on the packet: in this case, the packet will be looked-up in a binary tree. Usually, the default node will only indicate a destination for broad classes of packets that the chip cannot process itself: for example, IPvX packets, or non-IP packets. In the case of IP/UDP packets, the binary tree search should always be performed. If a tree search is to be performed, the default node will indicate which profile to use.

The MT92210 uses a binary tree approach to uniquely identify packets: each packet is given a 60-bit identification key generated by performing a 56-bit CRC on the various protocol headers that it contains and completed by adding the 4-bit profile number to the key. The headers used to generate the source key depend on the nature of the packet: while non-IP packets cannot even be passed through the binary tree, for lack of recognizable headers, null-encapsulated packets can only use the Flow Table Pointer associated to the packet and the RTP synchronization source to form the identification key (if RTP is not present, the Flow Table Pointer alone will be used). Non-UDP and fragmented packets will have a identification key based only on their source and destination IP addresses. Finally, the identification key for IP/UDP packets will include the source and destination IP addresses, source and destination UDP ports, and potentially the RTP synchronization source. The packet type determines which of the three identification key formats be selected.

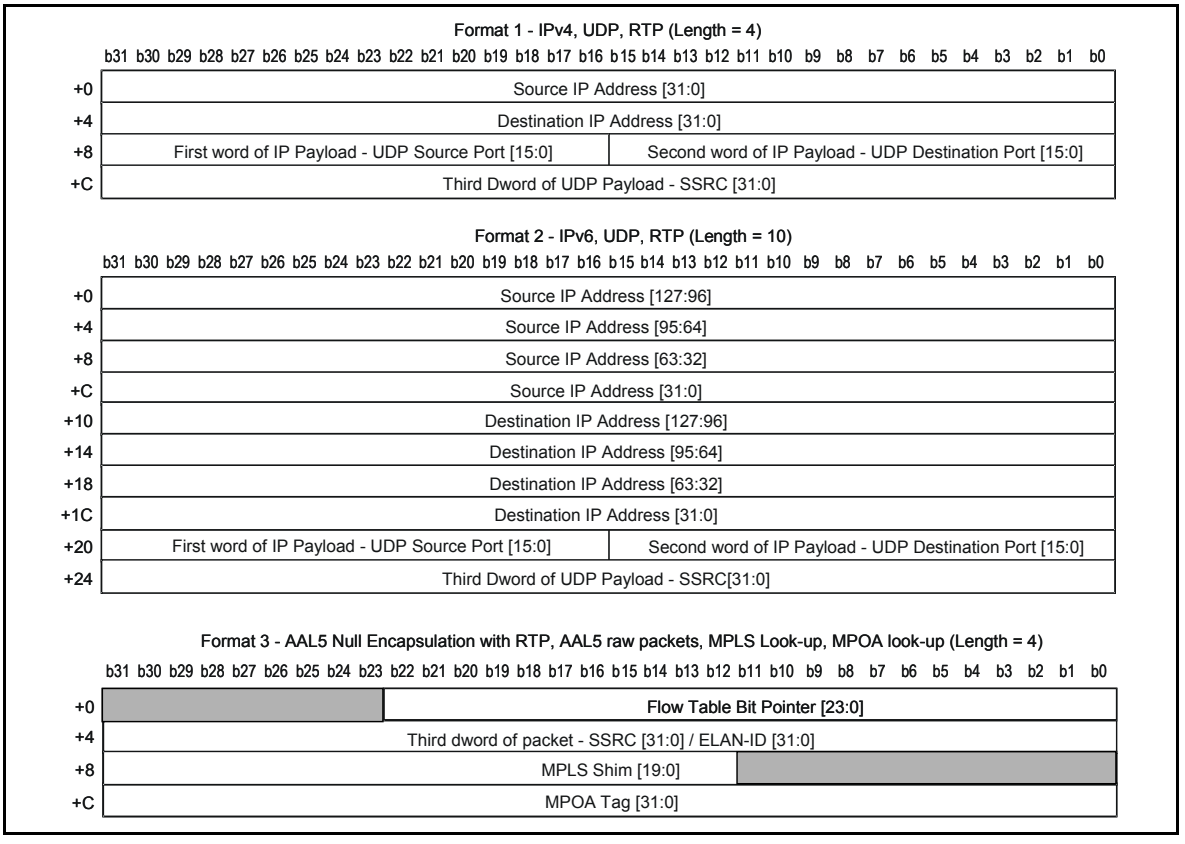


Figure 27 - Identification Key Formats (before CRC)

6.4 Masking

Before the CRC is performed on the identification key, a mask will be applied to it, depending on the profile used to search for the packet. This mask prevents certain values from being included in the final identification key. For example, if a profile wants to search for packets using only the IP source and destination addresses and the UDP source and destination ports, but not the SSRC, then dword 3 of the identification key will be masked out and the others will be allowed to pass. The mask used with each profile is contained in the Profile Memory, whose format is the following:

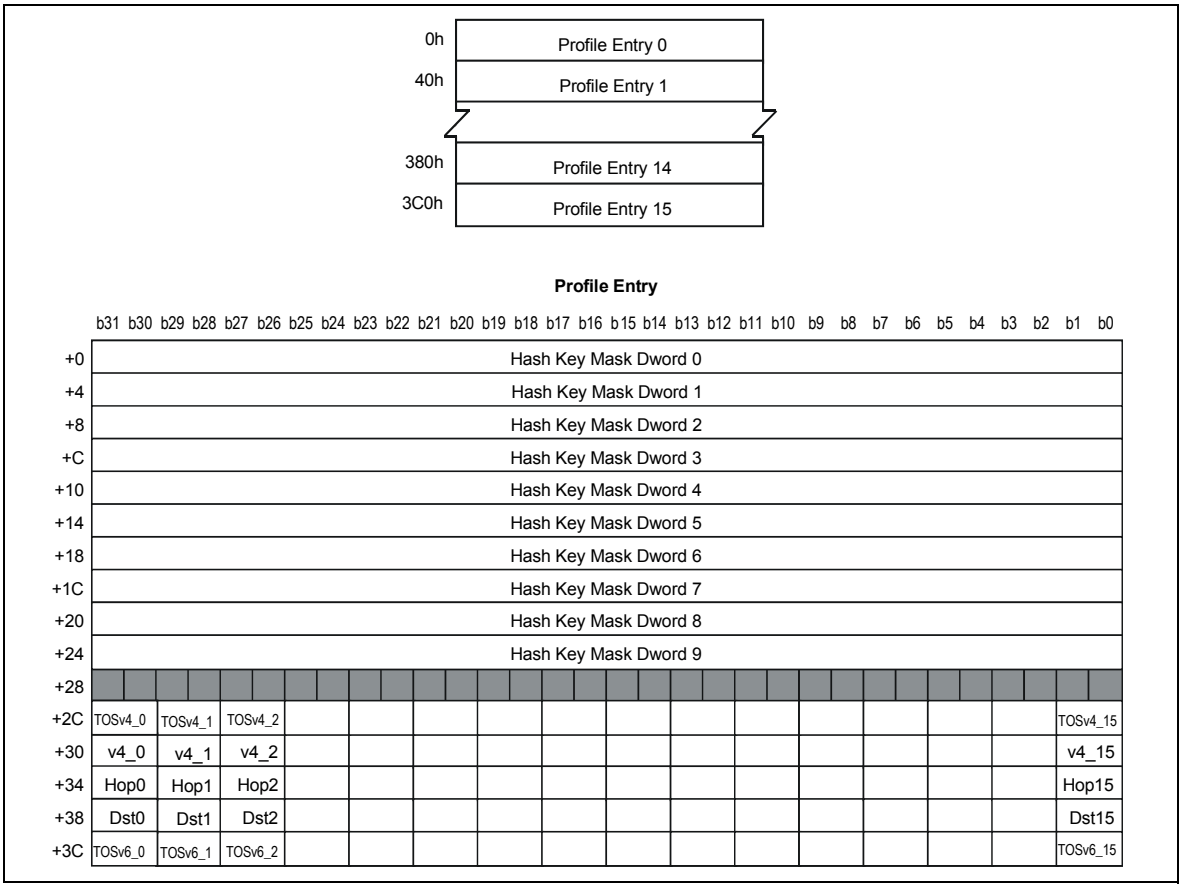


Figure 28 - Profile Memory

Field	Description
Hash Key Mask Dword	This mask will be applied to the identification key before a CRC is applied to it. When ‘0’, the corresponding bit in the identification key will be masked to ‘0’. When the identification key is only 4 dwords long, dwords 4 - 9 in this mask are ignored.
Option Header Treatment v4_X - IPv4 option field HopX - IPv6 hop-by-hop header option DstX - IPv6 destination header option	Indicates what to do when the hardware encounters these option values. “00” = unknown/no decision; “01” = skip to end; “10” = CPU through fragmented IP; “11” = delete. Each profile entry contains the treatment values for Options N*16 to N*16+15, where N is the profile entry number. In this way, all 256 values for the options are treated. These values are not in any way linked to the profile; they just reside in the same memory.

Table 15 - Fields and Description

“Type of Service” Treatment TOSv4_X - IPv4 TOS field TOSv6_X - IPv6 TOS field	Indicates what to do when the hardware encounters these TOS values. “00” = don’t change priority; “01” = reserved; “10” = change packet priority to low priority; “11” = change packet priority to high priority. Each profile entry contains the treatment values for TOS N*16 to N*16+15, where N is the profile entry number. In this way, all 256 values for the TOS are treated. These values are not in any way linked to the profile; they just reside in the same memory.

Table 15 - Fields and Description

6.5 Post-search Confirmation

Once the mask has been applied, the CRC calculated, and the profile number added, the resulting 60-bit identification key is then used to navigate through the binary tree until the node containing that key is found. When this node is found, it is associated to a post-search confirmation structure that tells the look-up system if the node actually corresponds to the headers of the given packet (because a CRC is performed, collisions between various CRC can occur).

The post-search confirmation structure contains a copy of the headers used to form the identification key. When a packet hits a post-search confirmation structure, its headers are compared to the headers contained in the structure: if they match, the packet belongs to that structure. One last verification is done before the packet is completely validated: its IP address index is looked up in the Flow Table pointed to by the packet’s Flow Table Pointer. The Flow Table is used to verify if the destination IP address of the packet is a legal address to be received on this logical subnet. If the match is ‘1’, then the packet is valid and is sent to the Routing Destination indicated by the post-search confirmation structure. If the match is ‘0’, then the packet has failed the post-search verification and continues to be searched. The Flow Table look-up is optional, and should not be performed when trying to match structures that don’t have a destination IP address, for instance if an MPLS label is being searched, or a packet containing RTP over AAL5.

The format of the Flow Table is as follows:

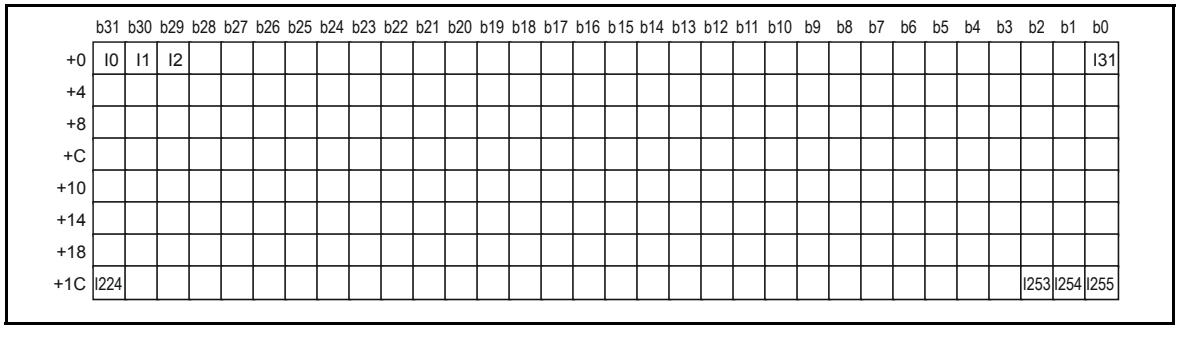


Figure 29 - Flow Table

Field	Description
I0 - I255	Index 0 to 255 in the flow table. The base address of the Flow Table is pointed to by the Flow Table Pointer. The Index is the IP Address Index from the post-search confirmation structure. When ‘1’, the destination IP address of the packet is valid to receive on this logical subnet. When ‘0’, it is invalid.

Table 16 - Fields and Description

If the confirmation structure found does not match the desired headers or the packet fails the Flow Table look-up, the structure may contain a pointer to the Next Post-Search Confirmation Structure and the matching process begins again. If no match is found and the pointer to the next structure is null, then the packet is routed according to the Default Profile Post-Search Structure routing for that profile. Because conflicts are resolved by the post-search confirmation structures, no conflicts can occur in the binary tree, so every node value can only be present once in the tree.

Each profile has a Default Post-Search Structure locating at a fixed address in SDRAM C. Any packet that didn’t find a match in binary tree, or failed the verification in Post-Search Confirmation Structures, will find its way to Default Post-Search Structure with its profile number. From there, the packet can be deleted, sent to CPU, or re-entering the binary tree by using a new profile number (i.e. a new identification key).

The following table and figure show the location and format of Profile Default Post-Search Structures.

Profile number	Profile Default Post-Search Structure Address in SDRAM
0	1800h
1	1880h
2	1900h
3	1980h
4	1A00h
5	1A80h
6	1B00h
7	1B80h
8	1C00h
9	1C80h
10	1D00h
11	1D80h
12	1E00
13	1E80
14	1F00
15	1F80

Table 17 - Profile Default Post-Search Structure

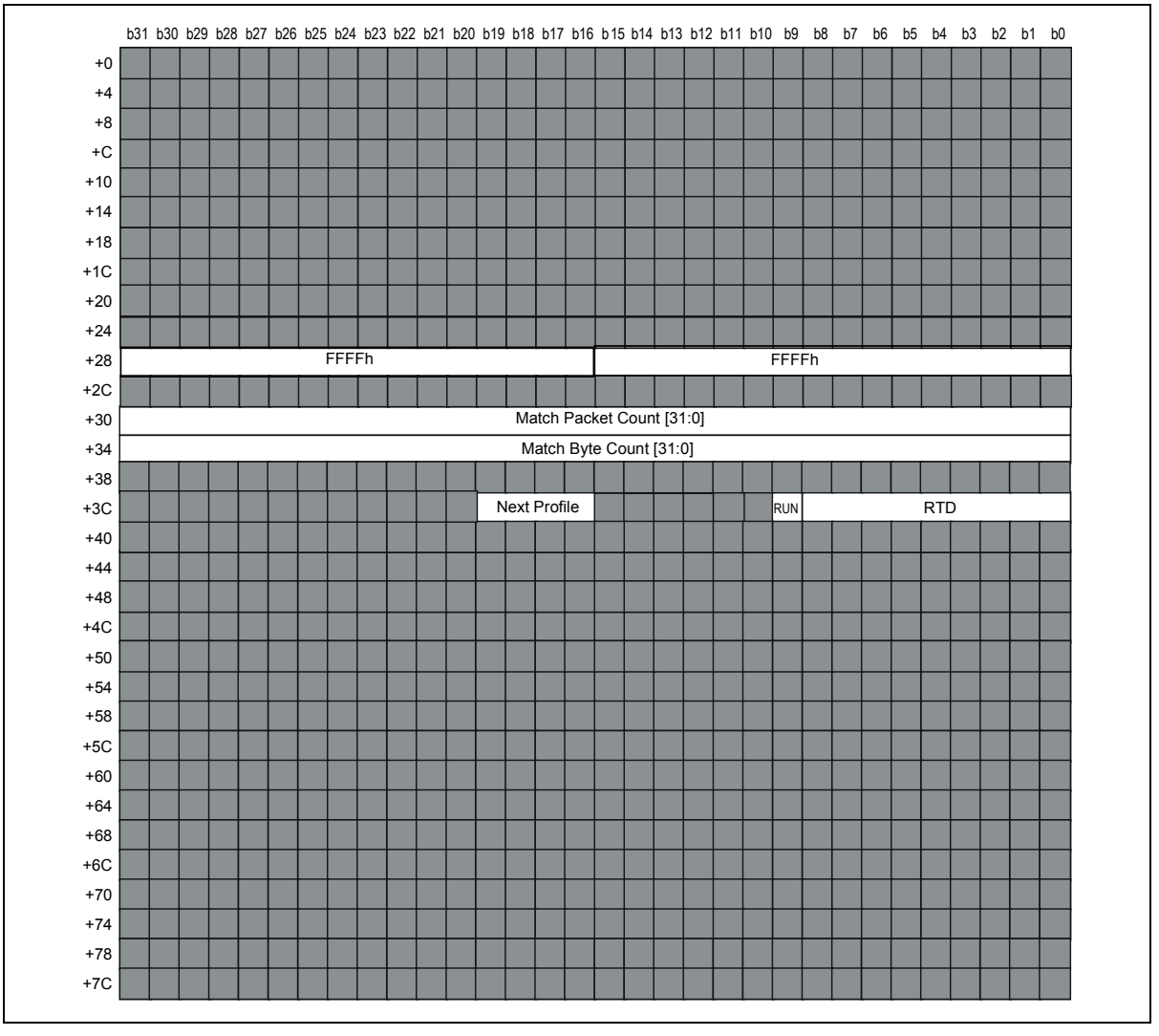


Figure 30 - Format of Profile Default Post-Search Structure (Refer to Table 19 for field descriptions)

The binary tree used is a dynamically re-balanceable tree, which means that the tree's nodes are always in the optimal distribution, minimizing search times. In addition, to minimize the impact of conflicts, two binary trees exist simultaneously in the system: one is active and used by the chip, while the other one is inactive. When the CPU decides that the current binary tree contains too many nodes with conflicts in them, it can re-create a binary tree by adding a random 32-bit number to each header dword used in the calculation of the key. This method provides a whole new distribution that is statistically very unlikely to have the same conflict problems as the original tree. The CPU can then switch the trees. The inactive one becomes active and vice-versa, and the chip beings using the new tree.

Register F18h points to the root node of binary tree. This is the format of the binary tree nodes:

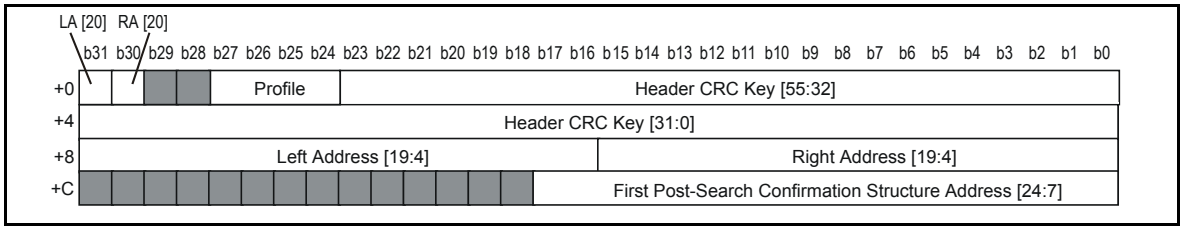


Figure 31 - Binary Tree Node

Field	Description
Header CRC key	Bits 55:48 represent a CRC-8 of the initial identification key. Bits 47:32 represent a CRC-16 of that same key, and bits 31:0 represent a CRC-32 of that same key.
Profile	Profile number being used to perform packet search. The profile number and header CRC key, between them, represent the 60-bit identification key used to search for the packet.
LA	Address of left node below the current node. The left node will contain an identification key that is smaller than the current one.
RA	Address of right node below the current node. The right node will contain an identification key that is larger than the current one.
First Post-Search Confirmation Structure Address	Indicates the address of the post-search confirmation structure that will be used to route this packet if it matches the identification key.

Table 18 - Fields and Description

When the correct node associated with the packet is found, the post-search confirmation structure will indicate what course of action to take concerning the packet. It may indicate to route the packet to the Packet Disassembly, to the RX CPU buffer or onto network port A or B. It may also indicate that another look-up needs to be performed, this time using a different look-up profile. The process then begins anew using the new profile

Other look-ups can take place in the binary tree: for example, ELAN-IDs contained in LANE v2 packets can be looked up, along with the Flow Table Pointer. A CRC will be performed on the fields, a 60-bit key will be generated, and a post-search entry will be obtained. This entry can be used to modify the Flow Table Pointer and the parsing of the packet will then continue with the new pointer.

MPOA tags and MPLS labels can also be searched for in the binary tree and post-search structures: apart from modifying the Flow Table Pointer, these searches are also used to identify the nature of the protocol above MPOA or MPLS. When packets are first assembled in the AAL5 reassembly structure, the nature of the protocols above these two are tagged in the fields MPLS_IP and MPOA_IP. When an MPOA or MPLS header is found in the packet and the nature of the following protocol has not been established, this header is looked-up along with the Flow Table Pointer. The resulting structure will identify the following protocol and may also change the Flow Table Pointer. The packet analysis then strips the MPOA or MPLS header off and continues the binary tree search with the following protocol.

To demonstrate the operation of this system, here is a simulation of the arrival of a packet in the packet identifier module. The packet arrives from RX link A and is an IPv6 voice packet carrying UDP and RTP. When parsed by the look-up module, its version number is established (6) and its IP header is searched until the protocol being carried (UDP) is found. The packet is then classified as an IP/UDP packet and routed using the default note which points to the Initial Search Structure for that type of packet, in this occurrence the structure at address 1200h.

The Initial Search Structure indicates that a binary tree search should be performed using a default profile #5. The profile entry is consulted and it indicates which fields should be used in the binary tree search through a series of mask fields. The mask fields depend on the default configuration of the packet type: for an IPv6 packet with UDP, the default headers are the Source and Destination IP address, Source and Destination UDP port and RTP

synchronization source. In this example, the look-up will be performed using only the Destination IP address and Destination UDP port; which associates with profile #11, the other fields will be masked out.

The entire 320-bit masked identification key is then passed through a series of CRC calculations that produce a 56-bit result. These CRC calculations consist of a CRC-32, a CRC-16 and a CRC-8. In addition, before passing the identification key through the CRC calculations, a 32-bit value can be added to each dword. This will allow conflict resolution if the binary tree starts getting clogged with too many conflicting nodes: adding a 32-bit value scrambles all the CRC to new random values and old conflicts will likely not reappear. Finally, the 4-bit profile value is annexed to the CRC to produce a 60-bit identification key.

This 60-bit result is then used as the new identification key for the binary tree procedure. The binary tree can have up to 128K nodes, distributed in any way. The system finds the node in the binary tree whose key is the same as the one being searched for: it may also find no matching node. Once the binary tree search is complete, the matching node's post-search confirmation structure is looked-up: it is checked to see if the headers of the looked-up packet match those in the confirmation structure, validating the node as being the correct search node. In this case, it can be assumed that there was a node in the binary tree that matched the packet, it matched in the corresponding post-search structure (there cannot be more than one match per packet in the post-search structures: this would be a fatal configuration error and would be flagged to a status bit in registers).

The format of the post-search confirmation structure is indicated in the next figure.

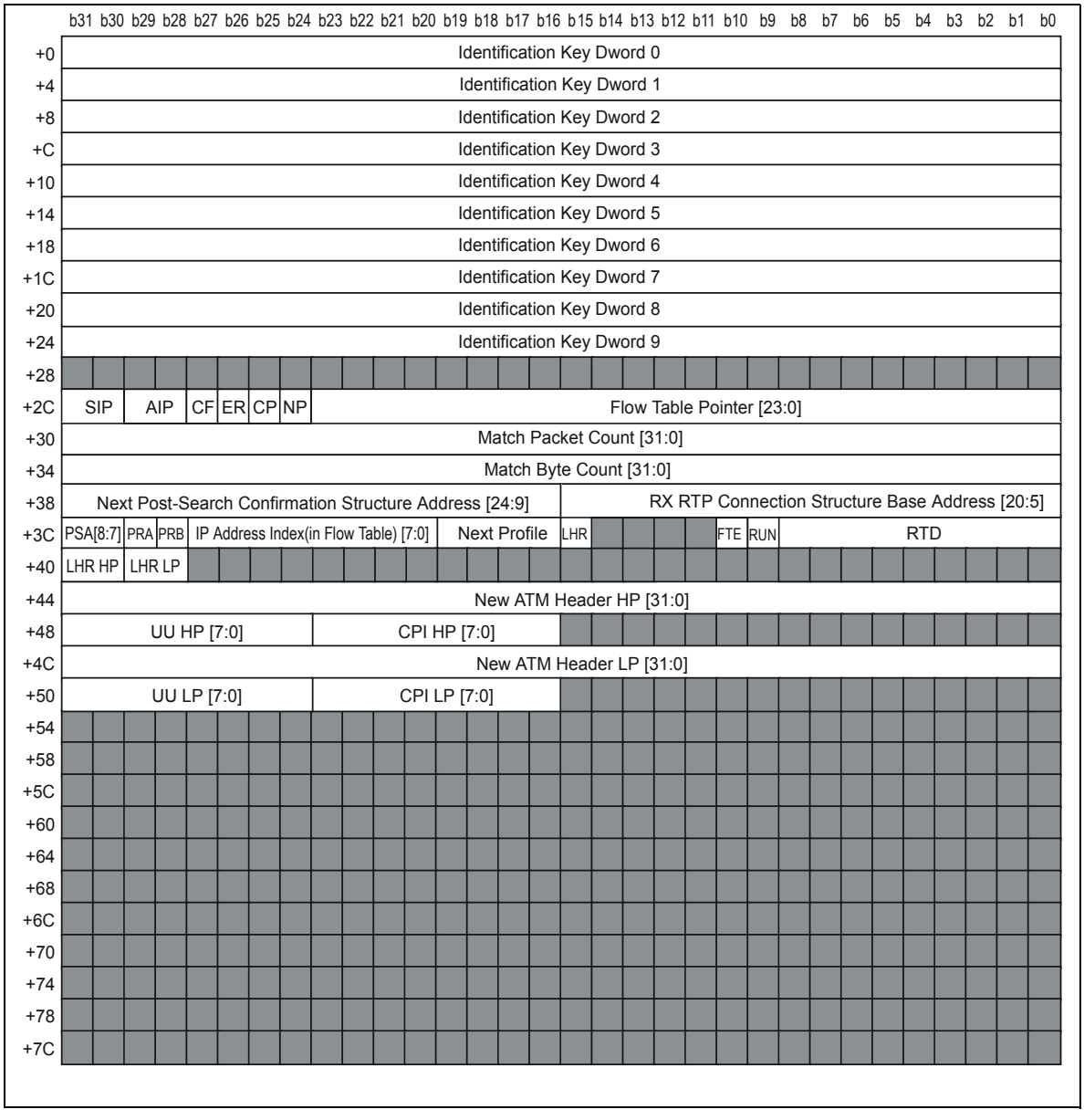


Figure 32 - Post-Search Conformation Structure

Field	Description
Identification key dword 0-9	The initial (pre-CRC) identification key of the packet should match these fields for the packet to be declared valid with this post-search structure. Note that fields which are masked out by the Profile Identification Mask should be set to 0s in this structure. The format of the identification keys is indicated in Figure 27.
SIP	MPLS IP indicator. “11” = reserved; “10” = RTP/UDP payload; “01” = IP, “00” = no change.
AIP	MPOA IP indicator. “11” = RTP/UDP payload, “10” = IP, “01” = MPLS unicast, “00” = no change.
CF	Change Flow. When ‘1’, the Flow Table Pointer in this structure will be used to replace the one to which the packet was associated.
ER	ELAN-ID Resolved Indication. When ‘0’, the ELAN-ID Resolved Indicator is left unchanged. When ‘1’, the ELAN-ID Resolved Indicator is positively identified as resolved.
CP	Change Priority of packet. When ‘1’, the priority of the packet will be changed.
NP	new priority of packet (if it was changed). ‘0’ = LP; ‘1’ = HP.
Flow Table Pointer	This field represents the logical subnet number on which the packet is received. This value will replace the current Flow Table Pointer of the packet if the CF (Change Flow) bit = ‘1’.
Match Packet Count	Number of packets that matched this post-search confirmation structure.
Match Byte Count	Number of bytes contained in all packets that matched this post-search confirmation structure (based on Total Packet Length in packet handle).
Next Post-Search Confirmation Structure Address	If a packet hits this post-search structure and does not match, this pointer points to the next structure that it can try to match with. This field is only used when more than 1 post-search confirmation structure is associated to the same binary tree node, meaning that different headers produced the same CRC. 0000h means Null link.
RX RTP Connection Structure Base Address	If the packet matches this post-search confirmation structure and its Routing Destination field includes the Disassembly module as a destination, this is the RTP Connection Structure that will be used.
PRA	Priority Route A. If this bit is high, the priority of the packet determines if it will be sent to TXA HP or TXA LP packet queue. When this bit is high, the RTD bits for TXA HP and TXA LP are ignored.
PRB	Priority Route B. If this bit is high, the priority of the packet determines if it will be sent to TXB HP or TXB LP packet queue. When this bit is high, the RTD bits for TXB HP and TXB LP are ignored.
IP Address Index (in Flow Table)	This index points to an entry within the Flow table that corresponds to the Destination IP address of this packet. Note that the old Flow Table Pointer will be used to look-up the IP Address Index, even if the Flow Table Pointer is changed by this structure.

Table 19 - Fields and Description

Field	Description
Next Profile	When this structure matches and the RUN (Route Using Next profile) bit is ‘1’, this is the profile number that will be used to search for the packet in the binary tree.
LHR	Link Header Replace. When ‘1’, the ATM header and UU/CPI may be changed before the packet is routed to its destination(s).
FTE	When ‘1’, the flow table will be consulted to know if the hit is valid.
RUN	Route Using Next profile. When ‘1’ and the packet matches this post-search confirmation structure, it will be searched for in the binary tree using Next Profile as a profile number.
RTD	Routing Destination. “000000000” = Delete; “1xxxxxxxx” = reserved; “x1xxxxxxx” = TX Link A HP; “xx1xxxxxx” = TX Link A LP; “xxx1xxxxx” = TX Link B LP; “xxxx1xxxx” = TX Link B HP; “xxxxx1xxx” = reserved; “xxxxxx1xx” = Network CPU Packet Buffer; “xxxxxxx1x” = Disassembly; “xxxxxxxx1” = Packet Identifier; others = reserved.
LHR HP	Indicates whether this packet’s ATM header and UU/CPI should be changed when the packet is routed in high priority. “1x” = change ATM header, “x1” = change UU/CPI. When one of these bits is ‘1’, LHR must also be ‘1’.
LHR LP	Indicates whether this packet’s ATM header and UU/CPI should be changed when the packet is routed in low priority. “1x” = change ATM header, “x1” = change UU/CPI. When one of these bits is ‘1’, LHR must also be ‘1’.

Table 19 - Fields and Description

The matching confirmation structure is then consulted to determine the fate of the packet: in this case, the structure indicates that a new search should be performed, using another profile!

The new profile entry is consulted and it requires a search using the Destination IP address, the Source IP address, the Destination UDP port, and the source UDP port. The new key is then generated using the CRC and profile number and searched for in the binary tree. The hit is then compared to the matching post-search confirmation structure which, in this case, indicates that a new search should be performed again with another profile number. Then the chip will repeat the above search procedure with using the Destination IP address and the Destination UDP port, and come back to the matching post-search confirmation structure. This time, the required destination is the packet disassembly module. Thus the packet is sent to the disassembly module and the next packet can be treated.

7.0 Packet Assembly

The MT92210 packet assembly module is responsible for collecting the bytes written in the circular buffers by the TX TDM and assembling them into RTP packets. Bytes will usually be encapsulated into RTP, UDP and IP, and then be shipped off on Ethernet, ATM or Packet over SONET; the packet assembly can also support a multitude of other header combinations. In addition to generating voice packets, the packet assembly must also perform silence suppression calculations on the incoming bytes to determine if the RTP packets it assembles should be transmitted or not. All of this must be scheduled correctly so that packets are generated at correct times and silence suppression information is available when necessary.

7.1 Service Timer

To ensure that all events in the packet assembly occur on time, the module contains a connection service timer that times both PCM packet assembly events and silence suppression treatment. The principle of the service timer is the following: a service timer is contained in external memory and has a length of N indicators. M indicators are read every frame. Therefore, the service timer is cyclic over N/M frames. Note that up to 16 service timers can be contained in external memory, each one of which can have any length N and any M.

Each time the TDM write pointer increments, it generates a pulse indicating to the service timer to read an entire frame of the service timer. Thus M event indicators are read and each of them can request the packet assembly module to try to assemble an xxPCM packet, and optionally, perform silence suppression calculations on a block of PCM data.

The service timer must schedule events in such a way as to minimize the end-to-end delay that CBR data encounters through the system. For example, if each RTP packet on a given connection contains 80 bytes of payload from a single channel, then an event signaling the assembly of one of these packets will be contained in a service timer of 80 frames in length. Thus, this event will be scheduled to occur once every 80 frames, matching the data rate of the incoming TDM traffic.

Because the number of frames in an RTP packet can vary and that finding a combination of 16 service timer lengths that could accommodate all possible frame fills between 1 and 1500 would be impossible, the events indicating packet assembly do not force the transmission of a packet. Instead, they indicate to the assembly module to attempt the assembly of a packet. The module will verify what is the current value of the TDM pointer and where the previous packet assembly ended, and based on this, will determine if enough TDM bytes are available to generate the desired packet. If a limited number of packet sizes is used, then the 16 service timers may be able to accommodate all connections in an exact way, adding no extra delay.

Because the assembly of RTP packets containing HDLC data is asynchronous, it does not need to be scheduled in the service indicator.

When the service timer requests a packet assembly, the event may be preceded by a silence suppression calculation event. For channels on which silence suppression is performed, a separate process takes care of reading the local and remote PCM bytes from their circular buffers, performing energy calculations on them and returning a silent/non-silent result so that the assembly module knows whether or not to send packets containing that channel. The silence suppression calculations are performed on blocks of the same size as the number of frames of data contained in the RTP packet in which these bytes will be transmitted. In this manner, the silence suppression event makes sure that the suppression decision (send / suppress) is up to date when the packet is transmitted or not transmitted. Silence suppression can only be used with connections carrying a single channel.

The service timers are configured is in an internal memory that contains 16 entries. The format of this internal memory is the following:

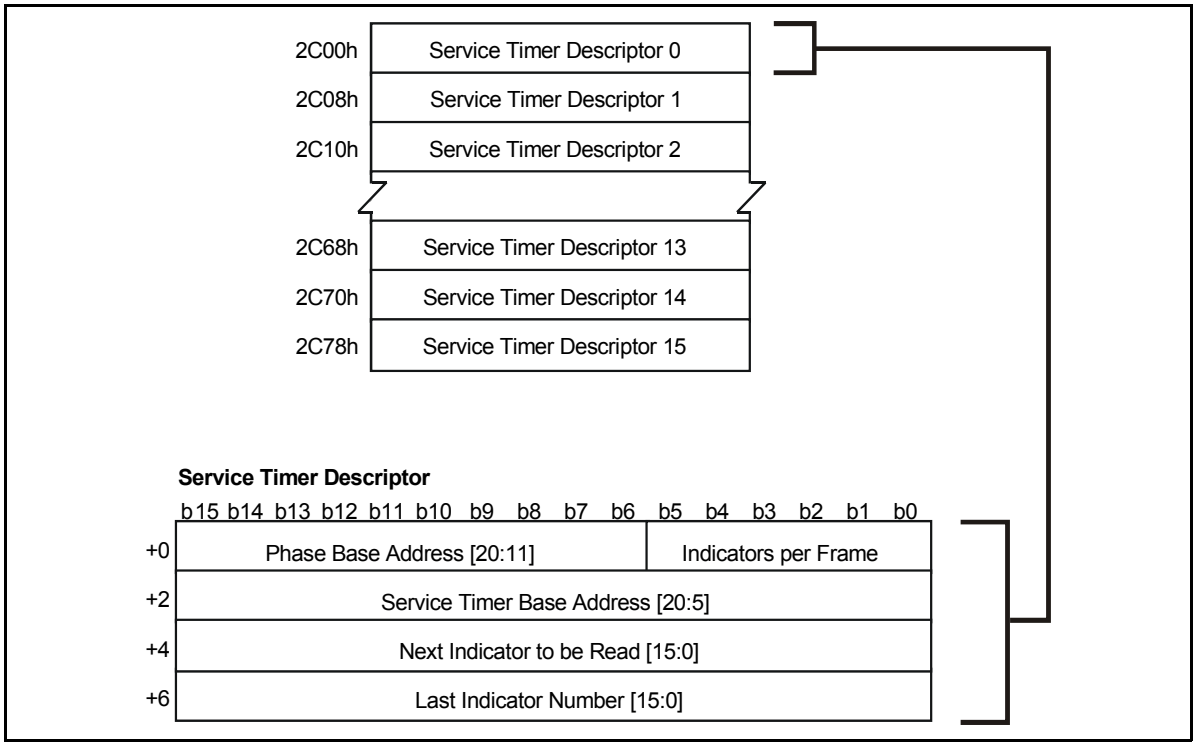


Figure 33 - Service Timer Control Memory

Phase Base Address	Points to a circular buffer that contains phase octets. An address of zero means no phasing is applied.
Indicators per Frame	This field determines how many indicators are read per TDM bus frame.
Service Timer Base Address	This field pointer to the first entry of the Service Timer in external SSRAM A. This pointers is in increments of 32 bytes.
Next Indicator to be Read	The field tells the hardware which indicator is next to be read in the Service Timer. This field should be initialized to zero at startup.
Last Indicator Number	This field defines the length of the Service Timer in increments of 1 Indicator entry. A Service Timer of 64 indicators in length would require this field to be set to 63.

Table 20 - Fields and Description

The format of the service timer in external memory is the following:

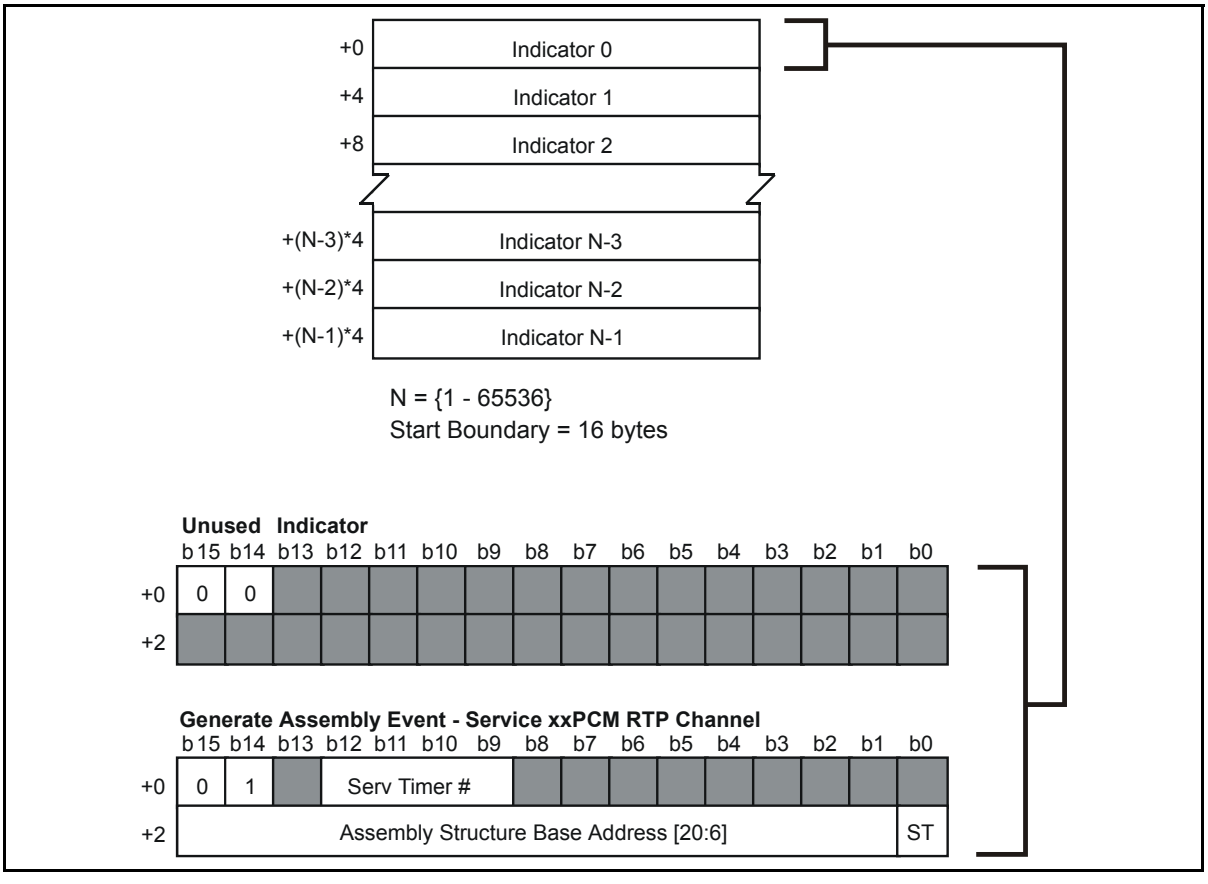


Table 21 - Service Timer

Field	Description
Serv Timer #	Service Timer number
ST:	Start Channel. A channel will only power-up on an event that has this bit set to ‘1’.
Assembly Structure Base Address [20:6]	Base address of the Tx RTP Connection Structure to which this event is being sent. The base address is pointed to in increments of 64 bytes.

7.2 Event Queue

When events indicating the assembly of RTP packets are read and treated by the service timer, they are written into a packet assembly queue to be treated in order. HDLC packets that complete also generate events in this packet assembly queue independently of the service timer.

The CPU can also generate packets to be inserted in an active connection. To do so, it writes the packets' payload in external memory (including the RTP header if any), and writes a descriptor to registers. This descriptor is then inserted into the event queue and treated by the chip. By inserting packets into an active connection instead of constructing the packets entirely itself, it allows the chip to manage the consistency of the RTP header across packets. For example, the sequence number will be incremental across all packets on an IP/RTP connection, not just on voice packets generated by the chip.

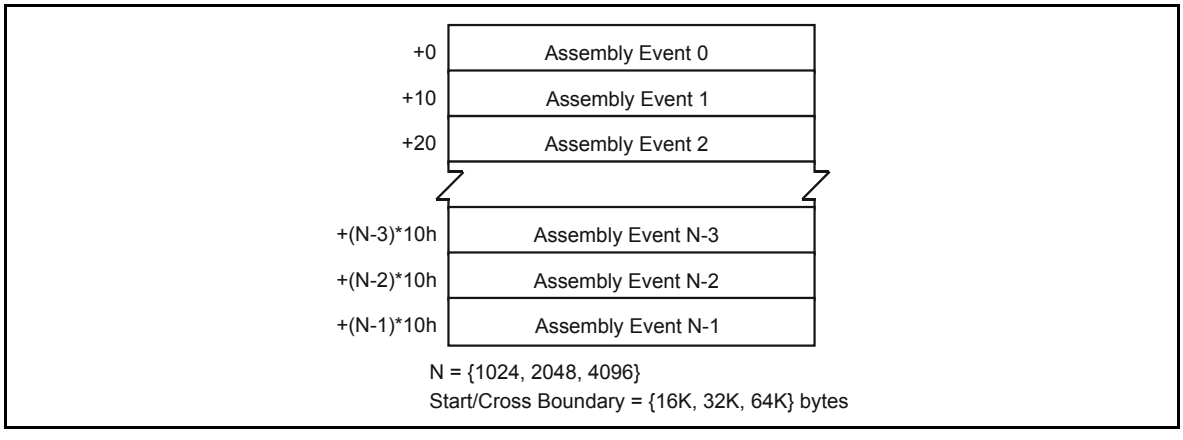


Figure 34 - Assembly Event Queue

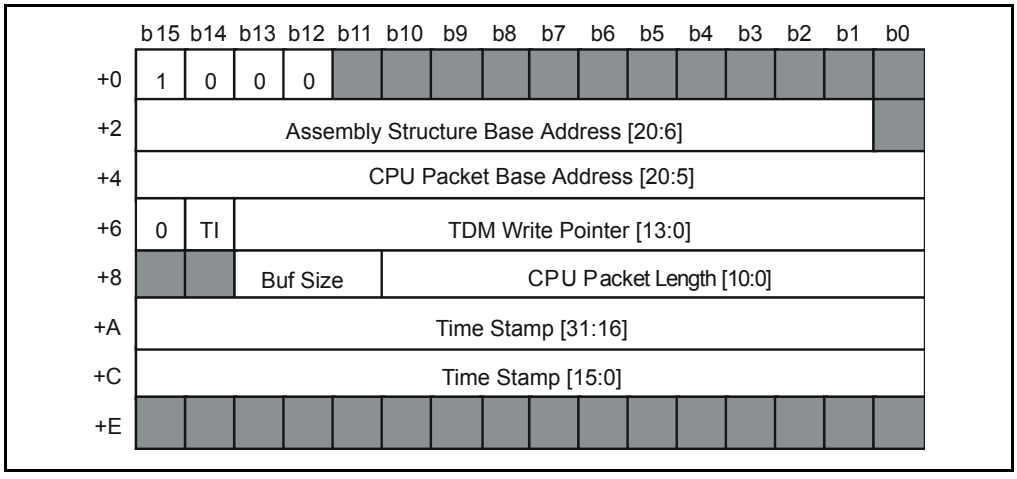


Figure 35 - Assembly Event - Send HDLC RTP Packet

Field	Description
HDLC Stream Number	Number of the HDLC Stream structure in the TX TDM Control Memory that generated this event structure.
Assembly Structure Base Address	Base address of the Tx RTP Connection Structure to which this event is being sent. The base address is pointed to in increments of 64 bytes.
HDLC Packet Base Address	Pointer to the first byte of the HDLC packet in external memory. This pointer points to 32-byte increments in SSRAM A.
TI	Time stamp Insert. When ‘0’, the time stamp will be inserted in the packet as it is received in the HDLC packet. When ‘1’, the time stamp in the HDLC packet will be treated as an offset from the current bus time-stamp.
TDM Write Pointer	TDM Write Pointer used to write xxPCM samples in the TX Circular Buffers at the time of the generation of this event.
Buf Size	Size of the HDLC circular buffer in which the HDLC packet is contained.
HDLC Packet Length	Length of the payload of the HDLC packet in bytes.
Time Stamp	Local Bus Time Stamp during which the HDLC packet completed.

Table 22 - Fields and Description

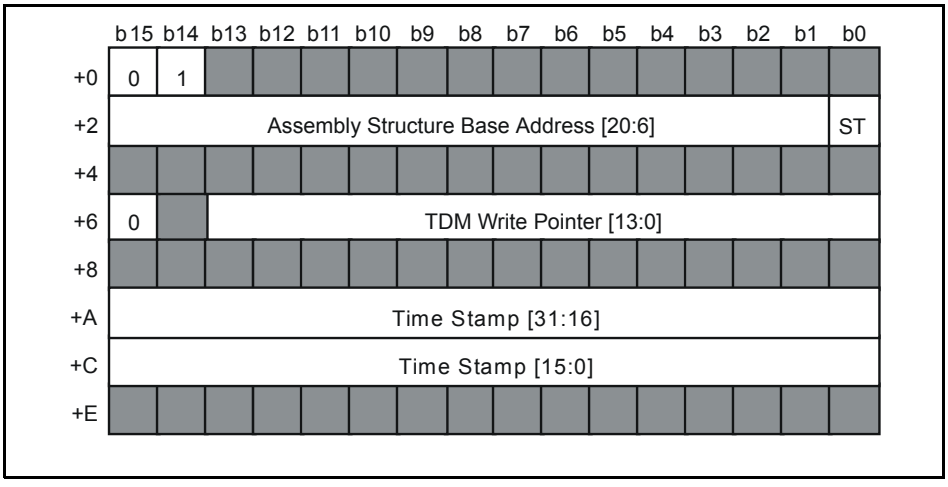


Figure 36 - Assembly Event - Service xxPCM RTP Channel

Field	Description
Assembly Structure Base Address	Base address of the Tx RTP Connection Structure to which this event is being sent. The base address is pointed to in increments of 64 bytes.
ST	Start bit. When this bit is set, the first packet of an xxPCM channel can be sent. If an event is received by an TX Connection structure with xxPCM channel waiting to startup (i.e. I bit is cleared), only events with the ST bit set will allow the first packet to be sent out (other event will be ignored).
TDM Write Pointer	TDM Write Pointer used to write xxPCM samples in the TX Circular Buffers at the time of the generation of this event.
Time Stamp	Local Bus Time Stamp during which the HDLC packet completed.

Table 23 - Fields and Description

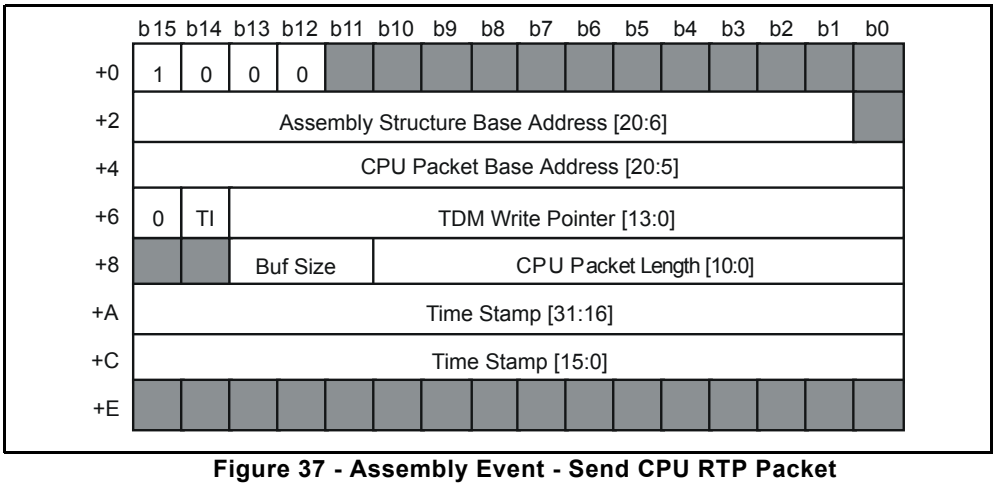


Figure 37 - Assembly Event - Send CPU RTP Packet

Field	Description
Assembly Structure Base Address	Base address of the Tx RTP Connection Structure to which this event is being sent. The base address is pointed to in increments of 64 bytes.
CPU Packet Base Address	Pointer to the first byte of the CPU packet in external memory. This pointer points to 32-byte increments in SSRAM A.
TI	Time Stamp Insert. When ‘0’, the time stamp will be inserted in the packet as it is received in the CPU packet. When ‘1’, the time stamp in the CPU packet will be treated as an offset from the current bus time-stamp.
TDM Write Pointer	TDM Write Pointer used to write xxPCM samples in the TX Circular Buffers at the time of the generation of this event.
Buf Size	Size of the CPU circular buffer in which the CPU packet is contained.
CPU Packet Length	Length of the payload of the CPU packet in bytes.
Time Stamp	Local Bus Time Stamp during which the CPU packet was queued.

Table 24 - Fields and Description

7.3 RTP Packets

PCM, HDLC and CPU-generated packets are all treated by the same structure, the TX RTP connection structure. Most of the fields of this structure manage the encapsulation protocols used (IP, UDP and RTP) as well as containing the link header. The TX RTP connection structure also contains a pointer to a secondary structure, the TX RTP header structure. This structure contains all protocol-related information, as well as anything that might need to be changed dynamically.

By having a pointer to another structure, a new TX RTP header structure can be created with new headers and new characteristics for the packet; once this is done, the pointer in the connection structure can be modified, creating a glitch-free change between the two.

7.3.1 TX RTP Header Structure

The TX RTP header structure contains a number of header words (note that headers for any layer at the network level or above are multiples of dword in length, while link headers can be byte multiples). The structure can indicate different types of encapsulation: IPv4 with UDP and with/without RTP, IPv6 with UDP and with/without RTP, or null encapsulation with/without RTP. In addition, the header words in the structure will contain a SNAP/LLC, LANE/Ethernet, LANE v2 or PPP header, or begin directly with IP, MPOA, MPLS or application data, depending on the type of the connection on which the packet will be sent.

7.3.2 Header Length

The Header Length shows how many bytes of total header will be present in the packet, including link header. This can vary from very short (a few bytes, in the case of ATM carrying null encapsulated data) to very long (when using, for example, Ethernet 802.1 p/Q with IPv6/UDP/RTP and many extension headers). The first 48 bytes of the header are reserved for ATM fields: the ATM header is contained in the first dword of link header and the first 2 bytes of the second dword contain the CPI and UU fields. These will always be ignored in Ethernet or PPP. The other 42 bytes are not used. After the ATM header, the rest of the link header follows and can be between 0 and 255 bytes in length, as indicated by Link Header Length. This length includes the 48 bytes used for the ATM header, UU and CPI, independently of whether the link is Ethernet, PPP or ATM. Note that the IP and RTP headers are always a multiple of 4 bytes in length, and the UDP header is always 8 bytes, while the link header can end on any byte boundary. In the structure, padding bytes and will be added to the link header: it must always end on a dword

boundary. The padding bytes will be ignored when the packet is transmitted on the bus. The Header Length is in dword, including the padding octets.

7.3.3 Packet Type

The Packet Type indicates with which header packets on the connection begin. The values are: “0000” = LLC, “0001” = PPP, “0010” = IP, “0011” = MPLS Unicast, “0100” = MPLS Broadcast, “0101” = MPOA, “0110” = LANEv1/Ethernet, “0111” = Application data. The most exceptional type is 7, because in this mode no IP or UDP headers are contained in the packet, meaning that several fields calculated by the chip, like IP lengths, UDP checksum, and others, do not need to be calculated.

7.3.4 Identification Counter Source Address

The Identification Counter Source Address field is used when a valid Identification value must be generated in the IP header. This is always the case in IPv4 and sometimes the case in IPv6 to allow transparent conversion between IPv6/IPv4 networks. The Identification value must be incremented for each packet that is transmitted. However, because CPU-generated packets must also use the same pool of identification values, a mechanism is put in place to allow the two to share it. Up to 2^{14} identification values can be contained in external memory, and each connection, when it transmits a packet, uses the current value of the identification indicated by Identification Counter Source Address and increments it by 1 after having transmitted the packet. The CPU may also access these values in an uninterrupted way to seize values for its own packets (see the identification_seize registers for more information on this mechanism). The IDentification Enable bit indicates if these identification values should be generated. The ID v6 Position points, in a relative way to header dword 0 within the header structure, to the dword in which the identification field is contained.

7.3.5 UDP Header Start

The UDP Header Start indicates how long the IP header is (in dword) and where the UDP checksum must be inserted in the packet. In the word that would contain the UDP checksum, a partial checksum must be written that contains the one's complement sum of the IP source and destination addresses, as well as the protocol (11h, meaning UDP). This is necessary because, in IPv6, the UDP checksum is not calculated on the IP destination address as indicated in the IP header, but on the final destination address, which may be contained in extension headers.

7.3.6 Timestamp Offset

The Timestamp Offset is a value that must be added to the generated RTP timestamp in the packet. Timestamps are generated differently in PCM, HDLC and CPU packets. This offset ensures that an outside observer cannot predict the timestamp; this offset should be set to a random value when the connection is initialized.

7.3.7 Sequence Number

The Sequence Number is incremental and is incremented and generated for all packets regardless of their source. This means that HDLC and CPU packets that generate an RTP header will have their Sequence Number thrown out and replaced by the one generated by this structure. The Sequence Number Insert bit indicates whether or not the sequence number will be inserted by the chip for HDLC and CPU-sourced packets. If this bit is '0', then the sequence number coming from the packets will be kept. Note that in this case, no PCM packets can be generated on this connection (i.e. ALL sequence numbers will be externally generated).

7.3.8 Transmitted Packet Count

The Transmitted Packet Count is a 16-bit counter indicating the number of packets that have been generated on this connection since its startup. The Transmitted Octet Count is a 32-bit counter of the number of payload bytes generated in packets on this connection. This only includes payload, but it will also include any RTP headers present in CPU or HDLC packets that are seen by the system as payload. These two fields allow good diagnostic of the connection’s bandwidth on IP.

For RTCP support and diagnostics, each time a packet is transmitted, the first 12 bytes of the packet payload (in the case of HDLC/CPU packets), or the first 12 bytes of RTP header for PCM packets, are copied into external memory. For HDLC/CPU channel carrying RTP, the first 12 bytes will represent the mandatory fields of the RTP

header. With this information, RTCP packets can be generated without requiring communication between the host on which the MT92210 is running and the source of the HDLC data stream.

7.3.9 RTD

The RTD field (RouTing Destination) indicates where this packet will be destined in the network interface. In most applications it will be set to transmit packets to port A, but the field allows flexibility by allowing the possibility of internal loopback, sending to port B, or other destinations.

The format of the basic TX RTP connection structure is the following:

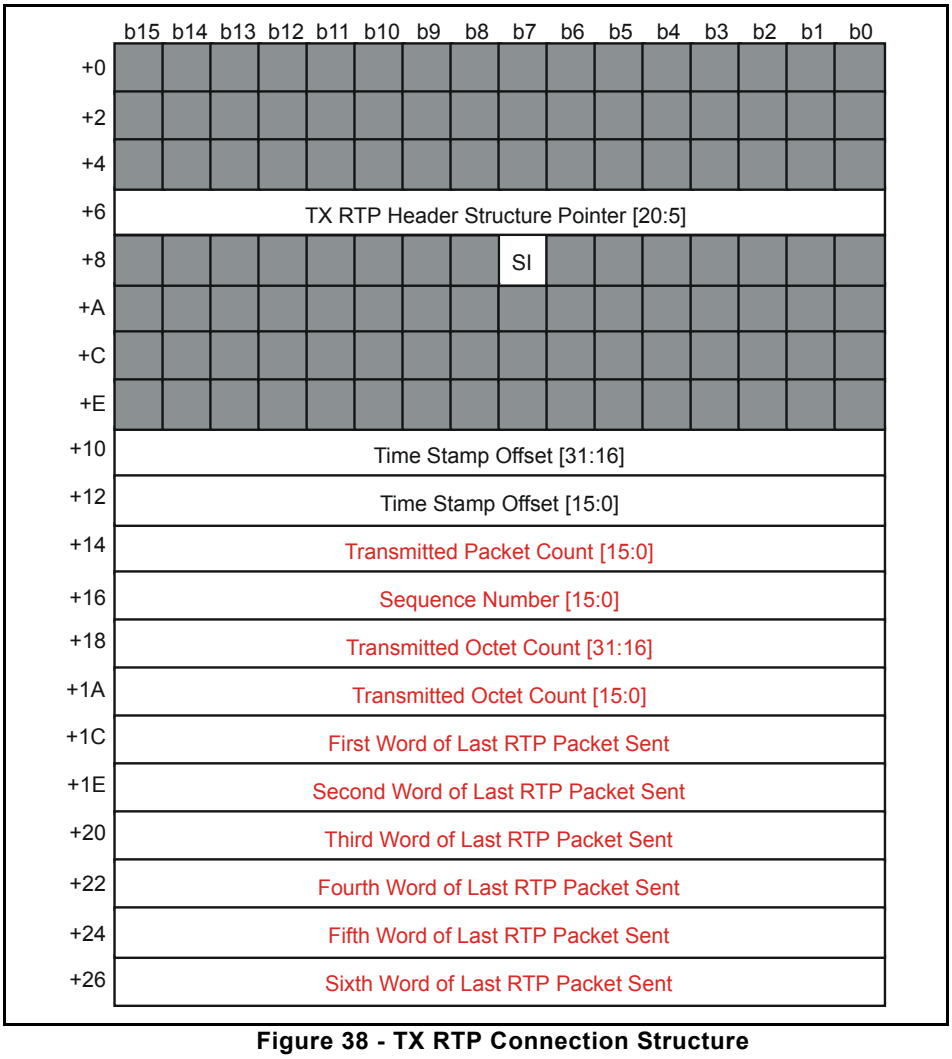


Figure 38 - TX RTP Connection Structure

Field	Description
TX RTP Header Structure Pointer	Pointer to TX Header assembly structure. This pointer points to increments of 32 bytes. All packet headers are formed based on the TX Header assembly structure. If changes in the headers must be made dynamically during a connection, this field may be rewritten atomically by software in order to point to another TX Header Structure.
SI	Sequence Number Insert. When ‘0’, the Sequence number is taken directly from the HDLC / TXCPU packet. When ‘1’, the sequence number is always inserted by this structure and the one in the original packet is ignored. This field must always be set if an xxPCM channel addition is opened over of this connection structure.
Time Stamp Offset	This number is added to the RTP time stamp of xxPCM / HDLC / TXCPU packets as they are sent. It is used to randomize the starting point of the RTP time stamp.
Transmitted Packet Count	Free-running counter of all packets transmitted on this connection.
Sequence Number	Sequence Number that will be sent in the next RTP packet if the SI bit is set. This field should be initialized to a random value by software.
Transmitted Octet Count	A Free-running counter of the total number of transmitted bytes in all packets including all header bytes.
Word of Last RTP Packet Sent	Record of first 6 words of last transmitted RTP payload. These fields can be used to monitor what is being sent on this connection.
Note	This structure must begin on a 64-byte boundary.

Table 25 - Fields and Description

If an xxPCM channel is carried by this connection, then the format of the TX RTP connection structure will be the following:

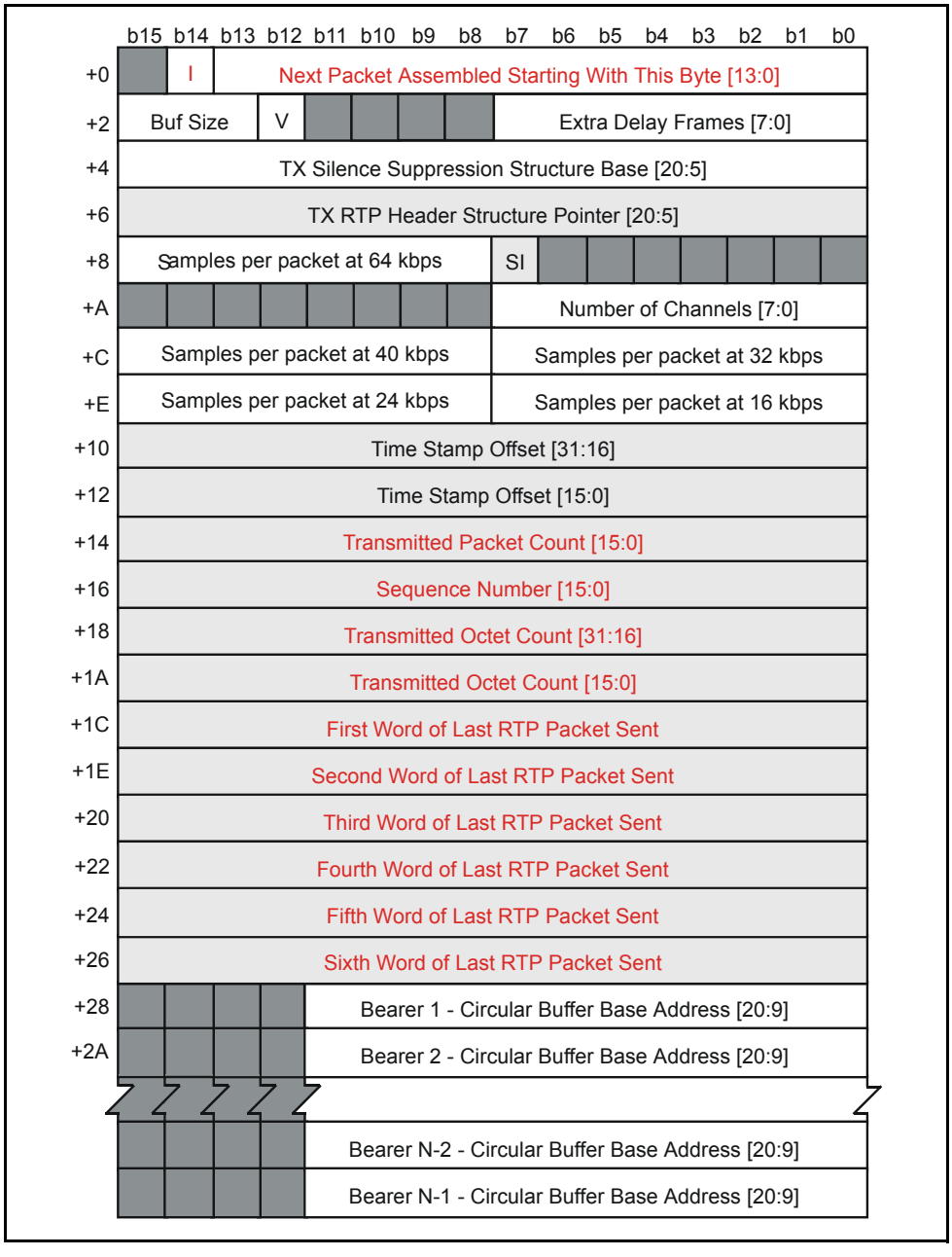


Figure 39 - xxPCM Channel Addition to TX RTP Connection Structure

Field	Definition
I	Init bit. This field is cleared by software upon initialization of an xxPCM channel addition to a TX RTP connection. It is then set by hardware when an “Assembly Event - Service xxPCM RTP Channel” is received having the ST bit set. Packets are only sent when this bit is read at ‘1’ from the xxPCM Channel addition to TX RTP Connection Structure.
Next Packet Assembled Starting With This Byte	When the I bit is cleared, this field contains the number of frames in the first virtual xxPCM that will be sent when the I bit is first set by hardware. When the first “Assembly Event - Service xxPCM RTP Channel” with ST bit set is received, no packet is sent out, but this field is initialized to its value plus the current TX TDM Circular Buffer Write pointer. In this way, the first packet to be sent out can be timed to only contain valid voice samples, since this feature enables the software to “drop” the N first packets. Setting this field to zero initially will force the hardware to start sending data one packet transmission period after setting the I bit. After the initialization of the xxPCM channel, this field is used to remember which is the oldest not yet sent byte in the TX xxPCM Circular Buffers associated with this xxPCM channel.
Buf Size	TX xxPCM Circular Buffer Size. “000” = 256 samples (512 bytes); “001” = 512 samples (1024 bytes); others = reserved.
V	Valid bit. When ‘0’, all “Assembly Event - Service xxPCM RTP Channel” will be ignored, preventing xxPCM channel packets from being sent.
Extra Delay Frames	This field is reserved for future use and should always be set zero.
TX Silence Suppression Structure Base	Pointer to a TX Silence Suppression structure. A value of 0000h indicates no silence suppression is needed on this xxPCM channel. The pointer points to 32 byte increments in SSRAM A.
Number of Bearers	Number of TDM bearers (i.e. TX xxPCM Circular Buffers) to be included in a packet. This field is usually set to 1. It specifies the number of TX xxPCM Circular Buffers that will be read for each frame that is sent, extracting a sample from each of them. This field also defined the number of “Bearer x - Circular Buffer Base Address” extension words appended to this structure.
Samples per packet at x Kbps	Number of samples to assembled into a single packet at a given CODEC rate. The total number of samples in packet is calculated as such: “Number of Bearers” * “Samples per packet”.
Bearer X - Circular Buffer Base Address	Pointer to each TX xxPCM circular buffer that must be read in order to assemble a packet. There must be one of these addresses per Bearer in the xxPCM channel. This points to circular buffers in increments of 512 byte of addressing in SSRAM A.
Bearers	For each bearer in this xxPCM channel, a single circular buffer base address must be specified in one of these extension words. All xxPCM channels will have between 1 and 255 bearers, thus between 1 and 255 extension words.

Table 26 - Fields and Description

The format of the TX RTP header structure is the following:

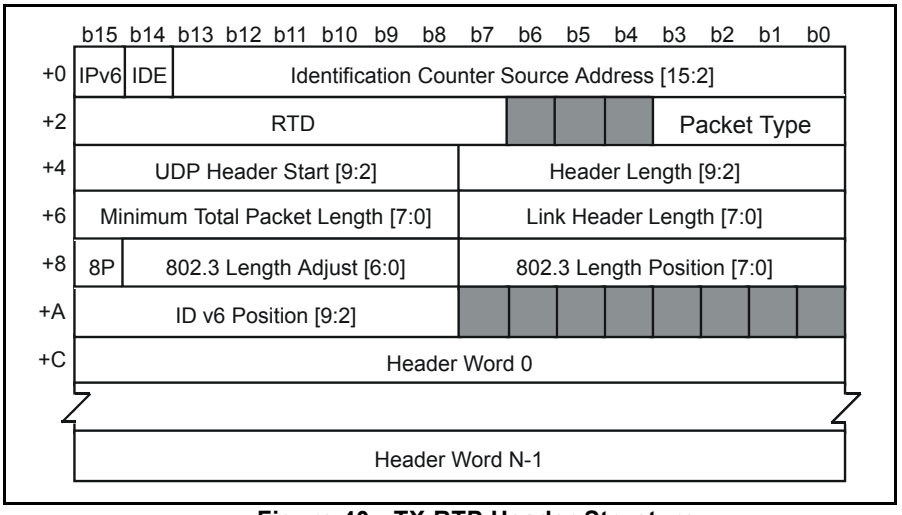


Figure 40 - TX RTP Header Structure

Field	Description
IPv6	This bit must set when the IP version for this connection is 6.
IDE	Identification Enable. When ‘1’ the identification dword will be written in the packet. When ‘0’, the identification dword will be set to an undefined value if it is present in the headers. IDE should be set when an IPv4 header is present or when an IPv6 fragmentation header is present.
Identification Counter Source Address	Points to a 4-byte field in the lower 64K of SSRAM A which will be read as the identification field that is to be inserted in the packet. When the IDE bit is set, this 32-bit counter will be incremented and written back to memory. In IPv4 only, the bottom 16 bits of this 32-bit identification counter will be written in the IPv4 header. This field is ignored when the IDE field is zero.
RTD	Routing Destination. This field indicates when packet generated with this header structure should be set. Multicast destinations are supported through an ORed combination of the following individual destinations. However, when packets are sent to the Packet Identifier, they cannot be sent anywhere else. “000000000” = Delete; “1xxxxxxxx” = reserved; “x1xxxxxxx” = TX Link A HP; “xx1xxxxxx” = TX Link A LP; “xxx1xxxxx” = TX Link B LP; “xxxx1xxxx” = TX Link B HP; “xxxxx1xxx” = reserved; “xxxxxx1xx” = Network CPU Packet Buffer; “xxxxxxx1x” = reserved; “000000001” = Packet Identifier.

Table 27 - Fields and Description

Field	Description
Packet Type	This field indicates the type of the first header present in the packet. This field is only used when the RTD is set to sent packets to the packet identifier. “0000” = LLC Header; “0001” = PPP Header; “0010” = IP Header; “0011” = MPLS Unicast Header; “0100” = MPLS Multicast Header; “0101” = MPOA Tag; “0110” = LANEv1-Ethernet/802.3 Header; “0111” = UDP payload (with or without RTP contained in payload); others = reserved.
UDP Header Start	Points to the first dword of the UDP header, relative to header word 0.
Header Length	Header length in dwords, includes all Header Words.
Minimum Total Packet Length	Minimum desired packet length including headers+48, in bytes. Hardware will pad the payload with 0’s up to this length as required, e.g. if a minimum Ethernet encapsulated packet of 46 bytes is desired, and the Link Header is 14 Bytes, this value should be set to 46+14+48=108. When no minimum packet length is desired, this field should be set to 0.
Link Header Length	This field is the byte counter of all Header words, less the initial padding bytes if there were any.
8P	802.3 Present. When ‘1’, an 802.3 header requiring a Length is present.
802.3 Length Adjust	This value is subtracted from hardware calculated packet length to obtain correct 802.3 length. The hardware calculated packet length = Payload + Header Words - Padding.
802.3 Length Position	Position in bytes of the first byte of the 802.3 Ethertype/Length with respect to Header word 0.
ID v6 Position	Points to the dword of the IPv6 header, relative to Header Word 0, in which the packet identification field will be written.

Table 27 - Fields and Description (continued)

7.4 PCM Packets

In the case of an connection carrying PCM or ADPCM, the packet assembly module uses the TX connection structure to determine how the bytes should be assembled, as well as all the different headers needed by the multiple protocols (Link, IP, UDP, RTP). The two fields that will determine the shape and size of the packet payload are the Total number of frames, that determines how many payload samples per channel will be carried by the packet, and the Number of Bearers, which tells how many xxPCM bearers will be included in the packet.

The total amount of payload samples in the packet will be Total number of frames multiplied by Number of Bearers. This is converted into bytes according to the compression rate.

7.4.1 Next TDM Write Pointer

The first word of the structure contains the Next TDM write pointer on which the next packet can be assembled. This is used when receiving a packet assembly event from the queue to determine if the packet assembly requested is valid or not. If the current TDM pointer is greater or equal to this value, then the event is valid; otherwise, it is ignored. The Initialized bit is used to detect the start-up of the connection: when a packet assembly event is read and the Initialized bit is '0', the first packet will be discarded, and the Next TDM write pointer will be initialized to the current TDM pointer + Next TDM write pointer. This means that this field should be initialized as an

offset that says: “the first packet should be assembled this many frames after the first event is read”. It allows a start-up delay on the connection.

7.4.2 Valid Bit

The Valid bit is used to ensure that no corrupt structures will be read while the scheduler is being programmed. While this bit is '0', any event that is read will simply be ignored (i.e. it will not even set the Initialized bit). The order of events should be the following: program the assembly structure (with Valid = '0'), then program the scheduler events, and finally set the Valid bit to '1'.

7.4.3 Buffer Size

The Buffer Size field indicates how large the TX Circular buffer containing the data is, from a minimum of 512 bytes up to a possible 4K bytes. The compression rate is determined implicitly from the data in the circular buffer: when it writes to the circular buffer, the TX TDM also uses auxiliary information to indicate the compression rate of the data in the buffer. The packet will then be assembled according to this compression rate. Note that a scheduling mechanism must be used to ensure that all data contained within a single packet is coded in the same compression format.

This is discussed in further detail in the TX TDM section. Note that the Total number of frames keeps its definition regardless of the compression rate, so with ADPCM 16 kbps, 4 frames of data only come out to 1 byte. The assembly process can generate one of 6 payload types for xxPCM packets: one for PCM, one for each ADPCM compression rate, and one for CN packets. In the TX RTP header structure, the payload types are written in the RTP header.

7.4.4 TX Silence Suppression Structure Base

The TX Silence Suppression Structure Base points to a structure that will be used to perform silence suppression on the xxPCM channel carried over this connection. When its value is 0000h, it means that no silence suppression will be done on this channel. Silence suppression cannot be used on channels carrying more than 1 bearer.

7.4.5 Extra Delay Frames

The Extra Delay Frames are also related to the silence suppression calculations: they indicate that the packet assembly should not use the most recent xxPCM data to assemble its packet, but instead should back up by a certain number of frames. This allows the silence suppression calculations on a block to be performed fully before the data is sent in an IP packet, ensuring, for example, that there is no start-up delay between the moment that voice begins again and the moment that packets start being sent again.

7.4.6 RTP Timestamp

The RTP timestamp sent in PCM packets is calculated in the following way: global timestamp corresponding to the first byte in the packet + Timestamp Offset written in the structure. The global timestamp is fed to the assembly module by the TX TDM and can either be a free-running counter within the chip or a value extracted from 4 time-slots on the TDM bus. See the TX TDM section for more information on sourcing the timestamp.

7.4.7 Circular Buffer Base Addresses

Finally, at the very end of the structure, are a certain number of Circular Buffer Base Addresses that indicate, for each channel in the connection, where is the circular buffer associated to it. These addresses point to 512 byte boundaries, which is the minimum size for any circular buffer, since the xxPCM data is only contained in the left byte of the circular buffer. The buffers may be larger, in which case the lower bits of the base address will not be used: the Buffer Size field indicates the size for all the circular buffers used by this connection.

7.5 HDLC Packets

HDLC and CPU-sourced packets are not payload-formatted by the TX connection structure. They are simply packaged in the link, IP and UDP (or null) headers and transmitted as such. The assembly process, in addition to encapsulating the payload in these protocols, may perform some RTP functions. The Sequence Number Insert bit indicates whether the RTP sequence number should be generated by the structure or kept as it is in the payload.

The Timestamp Insert bit, which comes from the event queue and originated in the TX TDM for HDLC packets (or was written in the descriptor by the CPU), indicates whether the timestamp from the packet should be used, or if it should be added to the global chip timestamp before being sent. To ensure that the timestamp passes through without any modifications set the Timestamp Insert bit to '0' and the Timestamp Offset to 0h as well.

By setting the Sequence Number Insert bit to '0' and making the timestamp transparent as described above, non-RTP packets can also be sent through HDLC or the CPU port.

7.6 Silence Suppression

The other operation that the packet assembly module can perform is calculating silence suppression information on data contained in circular buffers. The objective is simple: after performing calculations on a block of PCM data (using the PCM bytes), the calculator must decide if the channel currently being treated is silent or not. It then communicates this information to the packet assembler that will discard all RTP packets containing that channel until the channel re-enables.

Silence suppression calculations are performed on the block of data that is about to be sent in the xxPCM packet by the assembly process. By synchronizing the two events, the assembly process will have the most up-to-date information concerning the suppression of its packets and voice quality will be maintained to a maximum.

The silence suppression process reads its data from the same circular buffer as the packet assembly; the minimum size for any TX circular buffer is 512 bytes, because silence suppression information is only contained in the lower byte of each word in the buffer (thus the minimum amount of useful data in a circular buffer is 256 bytes).

There are 2 sequential steps to be performed in the silence suppression calculation. The first is the filtering of the input signal. In some cases, PCM signals are not centered around 0 and have a constant offset that is added to them: performing energy calculations with these offset values would lead to erroneous results and poor performance. Thus, to eliminate this error, the silence suppression module uses a first-order high-pass butterworth filter with a cutoff frequency of 10 Hz to eliminate DC and extremely low-frequency signals. The butterworth filter keeps the entire context it needs in the Local Butterworth HP 10 Hz Context field. This filter can be enabled or disabled by using the Local Butterworth enable bit.

To filter the signal, it must first be expanded into linear form. The Local Law bit indicates if the input signal is coded using u-law or A-law. Using the correct law, each PCM input sample is expanded into its linear value.

Once the filtering of the input signal has been calculated, silence suppression can be correctly performed on the data. The silence suppression process uses an adaptive algorithm to determine whether the input signal represents silence or voice. The basic approach used in the silence suppression algorithm is to smooth out the level of the signal to remove any frequency-dependent components, then to establish the minimum level at which the signal maintains itself over a reasonable period of time. Then, the current level of the signal is compared to the floor level, and if it is measured to be larger, then the signal is deemed to be voice; otherwise it is considered to be silence.

If the packet is silent, then a CN packet may potentially be generated. If the last packet was not suppressed, then a CN packet will definitely be transmitted. A CN packet will also be sent at other points in time when the padding energy at the remote end needs to be updated. The Last Suppress bit indicates the state of the last packet, which allows a CN packet transmission decision to be taken.

The CN energy is calculated by summing the energies of the most recent samples received. Depending on whether the current state is voice or silence, the energy will be summed on a different number of samples, using the First Energy Period and Subsequent Energy Period respectively. This is done because energy on background noise and silence can be calculated on a much longer period to obtain greater accuracy. The sum of samples is kept in the Energy Sum and the sample count is kept in the Energy Counter. When the Energy Counter reaches the terminal count, a linear-to-dB conversion is performed to take the Energy sum divided by the Energy Counter to obtain the average linear energy, then converted to a logarithmic scale to obtain an energy in dBov. The TX Silence Suppression Structure contains a dB Correction that allows the dB energy to be converted to any scale, like dBm0 instead of dBov, for example. There is also a Maximum dB Value and a Minimum dB Value that can be set, making sure that the value in CN packets stays within a certain range.

The silence suppression information can also be fed to the process externally. When this is the case, a special PCM code is used on the TDM bus. In this mode, 2 TSSTs must be used to feed the MT92210 with the transmit data. When the last PCM byte of the packet is read off the H.110 bus is 00h and the associated time slot upper bit also indicates 0, this means that the packet must be suppressed. Note that the TX unsigned PCM magnitude is transmitted on the lower 7 bits of the associated time slot. If the code received is normal data (upper bit 1), then the packet is valid. In this case, the external agent makes all the suppress/don't suppress decisions, but the MT92210 still calculates the CN Energy and decides when to send and when not to send CN packets.

The MT92210 also supports many modes in which it can send CN packets. In the most common configuration, it will suppress silence packets and send CN packets at the beginning of silence periods, or whenever it decides to update. It is also possible to configure it to suppress silence packets but never to send CN packets at any time. It also supports two modes in which it does not suppress packets. In the first one, it uses the marker bit in RTP to

indicate the beginning of a voice period; in the second, all silence packets are indicated with a marker bit = ‘1’, as well as the first packet at the beginning of a voice period. Finally, instead of calculating its own CN energy and generating the packet itself, the MT92210 can use a preprogrammed CN packet. This preprogrammed packet can be of variable length, allowing spectral data to be passed along as well as an energy value. Preprogrammed packets must be written into external SSRAM A by an agent through the CPU interface and may be updated periodically. Because a pointer is given to the preprogrammed packet, its payload may be changed in a glitch-free, atomic way. Spectral CN packets may be up to 64 bytes in length.

The silence suppression process also monitors and counts underruns that occur on the RX TDM side. The underrun information from the RX side is passed along and is written to the TX circular buffer. The silence suppression process can thus keep a 16-bit Underrun Count of the number of underrun events that have occurred.

The format of the TX Silence Suppression Structure varies according to whether or not the silence suppression decision is being taken internally or if it is being fed from an outside agent, and according to whether or not the generated CN packet will be a white energy value or a preprogrammed spectral value.

The possible formats for this structure follow:

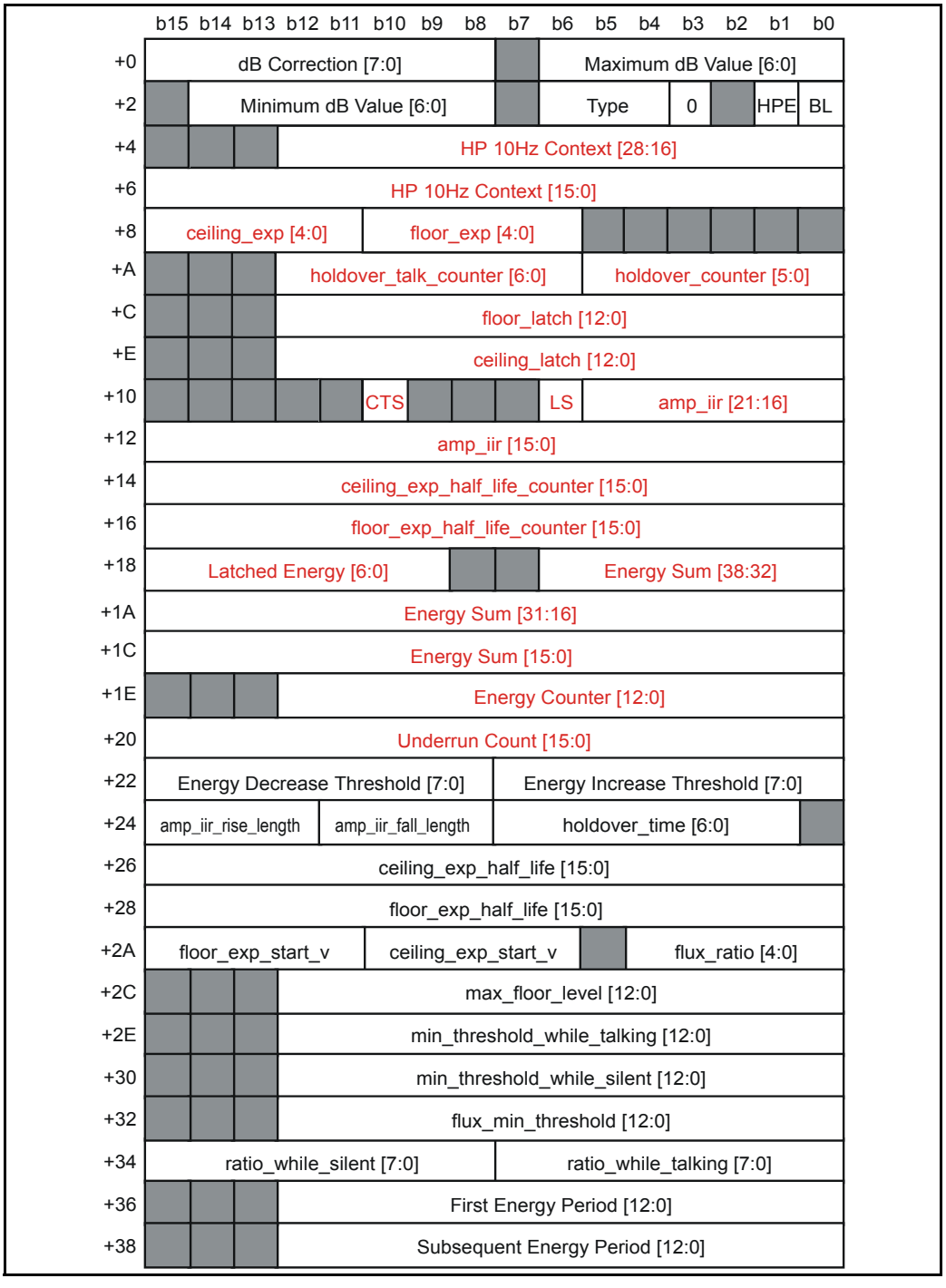


Figure 41 - TX Silence Suppression Structure: Internal VAD & White Energy Estimation

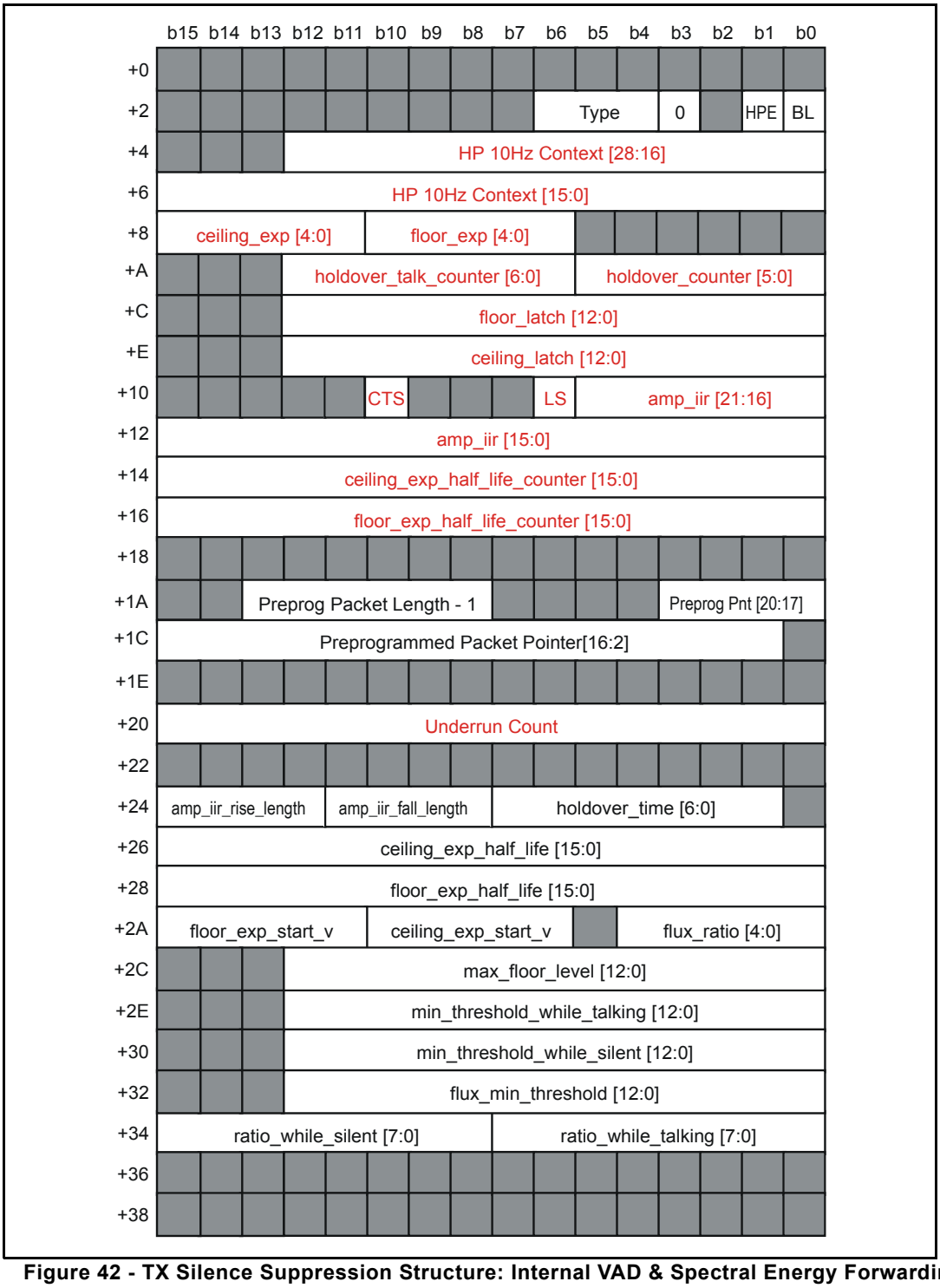


Figure 42 - TX Silence Suppression Structure: Internal VAD & Spectral Energy Forwarding

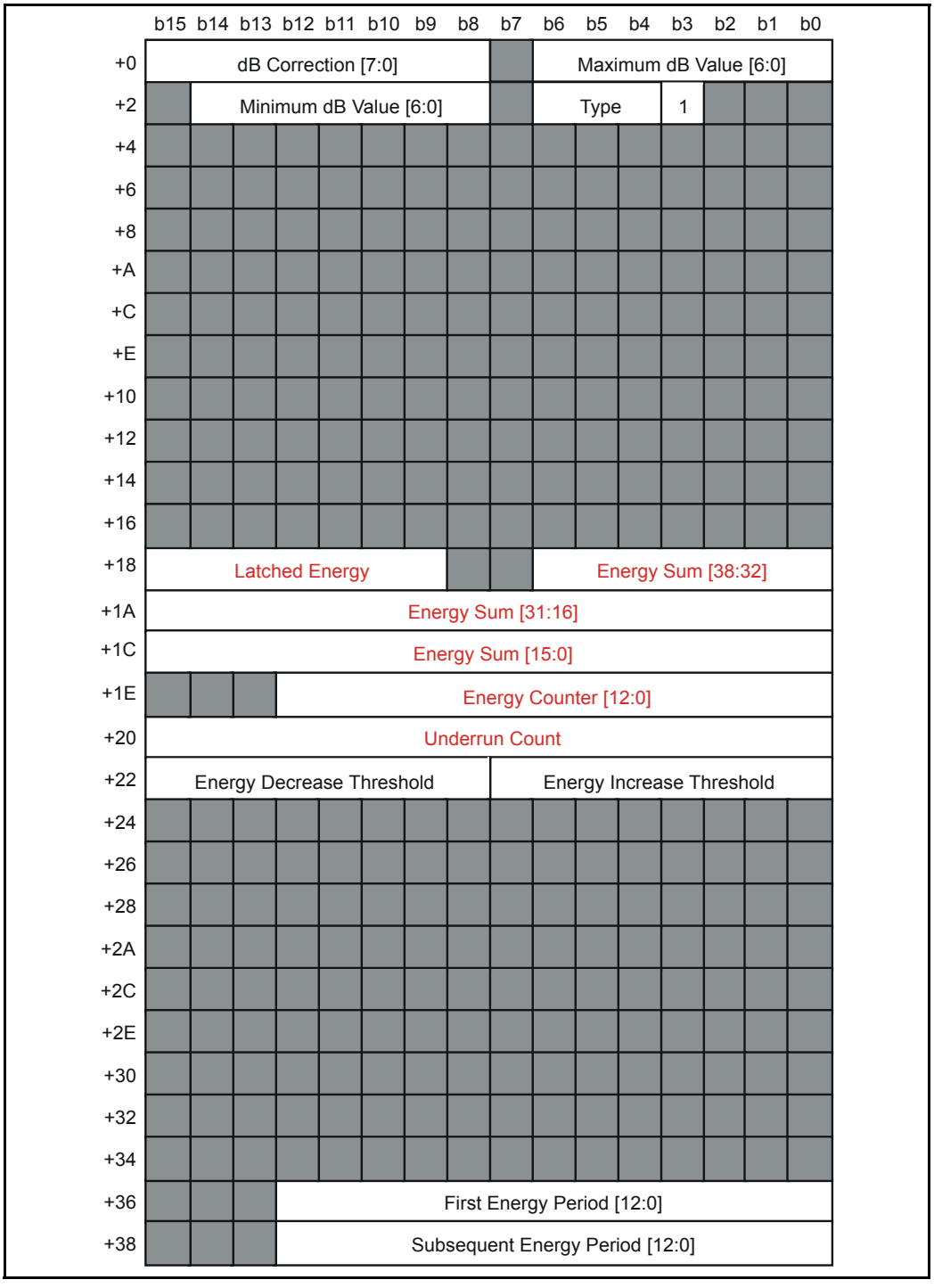


Figure 43 - TX Silence Suppression Structure: External VAD & White Energy Estimation

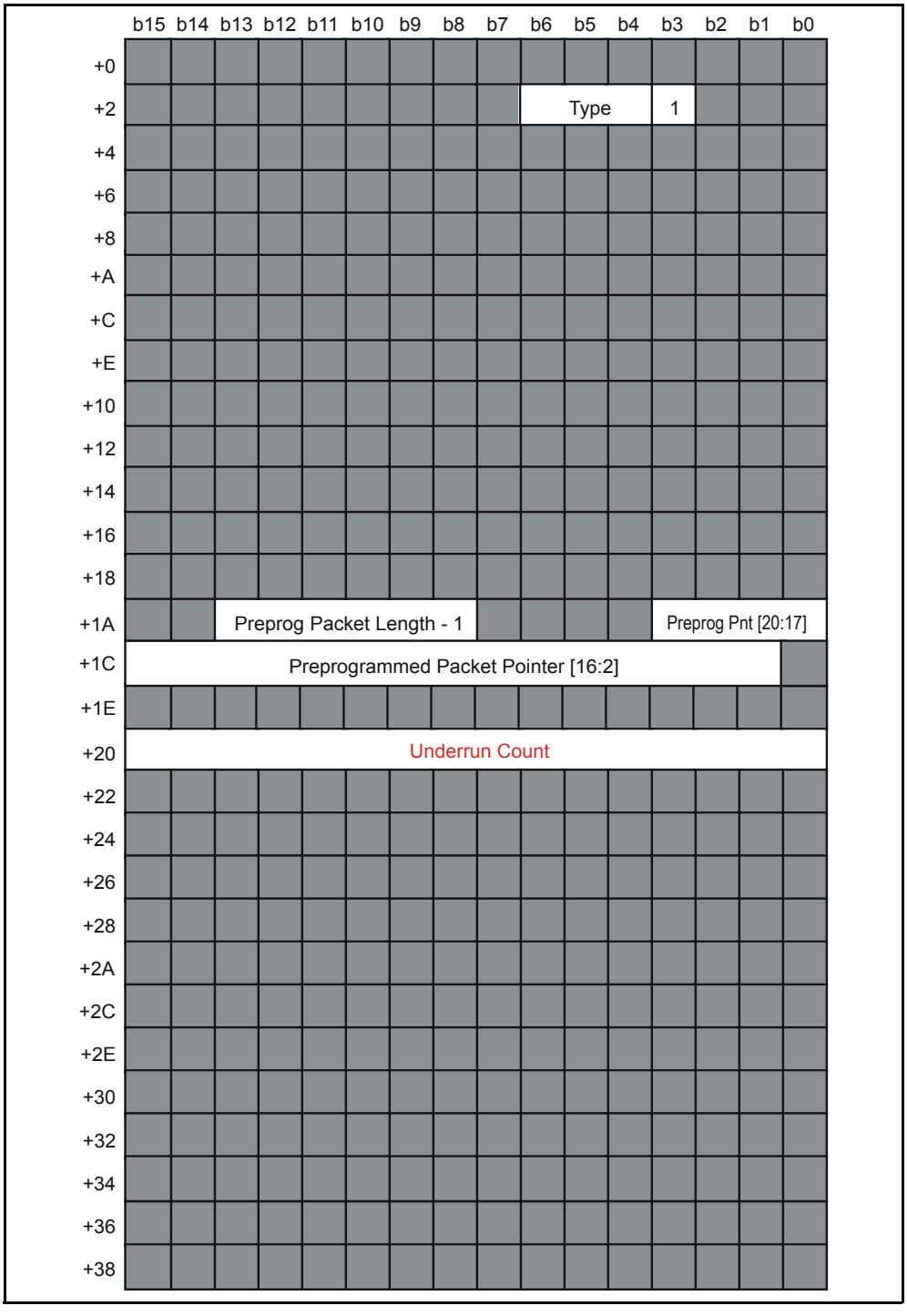


Figure 44 - TX Silence Suppression Structure: External VAD & Spectral Energy Forwarding

Field	Description
dB Correction	Two's complement signed number. Added to the dB energy before going to the Min/Max value masks. When this field is worth +1, an energy of -65 dBov will be converted to an energy of -66 dBov. Note: The dB Correction is applied first; the Maximum/Minimum dB Value range verification is applied next (and truncation may be done to force the value within the range); and the Decrease/Increase Threshold calculation is performed last.
Maximum dB Value	The maximum dB value is the highest energy dB value that can be generated. Generally set to either 0 or 30.
Minimum dB Value	The minimum dB value is the lowest energy dB value that can be generated. Generally set to either 127 or 78.
Type	How to send comfort noise packets. "000" = do not sent comfort noise packets; "001" = send internally calculated white spectrum energy comfort noise packets; "010" = send preprogrammed comfort noise packets; "011" = send no comfort noise packets (send all voice packets regardless), but set marker bit at beginning of voice period, "100" = send no comfort noise packets (send all voice packets regardless), and set marker bit in all silent packets and at beginning of voice period; others = reserved.
HPE	High Pass Filter Enable. When '1', a 10 Hz High Pass filter will be applied to the input signal before going through the VAD process and the energy estimation process. If the high pass filter is not done internally, it must have been done externally to achieve proper energy matching.
BL	Circular Buffer Law. '0' = samples in TX Circular Buffer are in u-law; '1' = samples in TX Circular Buffer are in A-law.
HP 10Hz Context	Context of the high pass filter. Must be initialized to zero.
ceiling_exp	Should be initialized to zero by software.
floor_exp	Should be initialized to zero by software.
holdover_talk_counter	Should be initialized to zero by software.
holdover_counter	Should be initialized to zero by software.
floor_latch	Should be initialized to zero by software.
ceiling_latch	Should be initialized to zero by software.
amp_iir	Should be initialized to zero by software.
LS	Last packet Suppressed. When '1', the last packet was suppressed.
CTS	Current talk status. When '0', the VAD is currently suppressing. When '1', the VAD is not suppressing.
ceiling_exp_half_life_counter	Should be initialized to zero by software.
floor_exp_half_life_counter	Should be initialized to zero by software.
Latched Energy	Last sent energy level exactly as it was sent in the comfort noise description packet.
Energy Sum	Accumulator used to estimate the idle line energy level. Must be initialized to zero by software.

Table 28 - Fields and Description

Field	Description
Energy Counter	Counter used to frame energy estimation period. This counter starts off at zero and counts up to either First Energy Period or Subsequent Energy Period. When it reaches it terminal counter, the comfort noise level is updated internally and may be resent. This field is reset to zero when the VAD identifies a packet as being non-silent. This field must be initialized to 0 by software.
Underrun Count	Free-running 16-bit underrun counter. This counter monitors underruns that occur on the TSST adjacent to the TSST that is being silence suppressed. When the AS bit is cleared, TSST0 & TSST1 are associated. When the AS bit is set, TSST0 & TSST2 are associated.
Energy Decrease/Increase Threshold	Number of dB of difference (vs the last sent comfort noise packet) required to do retransmission of a new (updated) comfort noise packet. 00h is illegal; 01h means retransmit on any difference; 02h means retransmit on a 2 dB difference; writing this value to 128 will prevent all retransmission.
amp_iir_rise_length	Should be initialized to 3 by software.
amp_iir_fall_length	Should be initialized to 7 by software.
holdover_time	Should be initialized to 16 by software.
ceiling_exp_half_life	Should be initialized to 300 by software.
floor_exp_half_life	Should be initialized to 2716 by software.
ceiling_exp_start_v	Should be initialized to 17 by software.
floor_exp_start_v	Should be initialized to 17 by software.
flux_ratio	Should be initialized to 8 by software.
max_floor_level	Should be initialized to 4000 by software.
min_threshold_while_talking	Should be initialized to 8 by software.
min_threshold_while_silent	Should be initialized to 32 by software.
flux_min_threshold	Should be initialized to 32 by software.
ratio_while_silent	Should be initialized to 57 by software.
ratio_while_talking	Should be initialized to 32 by software.
First Energy Period	Number of frames that will be averaged out to produce the estimation of the comfort noise energy when silence is first detected.
Subsequent Energy Period	Number of frames that will be averaged out to produce the estimation of the comfort noise energy for comfort noise refresh packets.
Preprog Packet Length - 1	Length of the preprogrammed comfort noise packet in bytes (minus 1). To send single byte comfort noise packets (the basic non-spectral mode), this field must be set to 00h.
Preprogrammed Packet Pointer	This field points to the location in SSRAM A at which the preprogrammed packet is located.

Table 28 - Fields and Description (continued)

8.0 Packet Disassembly

After packets have gone through the look-up process on the reception side, they can be routed to the packet disassembly module, which will transform RTP packets into PCM bytes, ADPCM samples, or HDLC/CPU-destined packets. The packet disassembly is not responsible for any header verification except for RTP headers (this is done in the network module) but it performs PDV monitoring to diagnose and reduce delay, packet loss compensation (for connections using RTP) and some RTCP functions in hardware.

The module logs all of its errors and events into the RX disassembly Event Report Queue. This queue contains structures that may be generated by RTP connections, xxPCM channels, HDLC channels and CPU channels. Each event type has its own status bits and errors that may be flagged. The Event Report queue is contained in SSRAM B and is of a variable size between 256 events (4K bytes) and 32K events (512K bytes).

The format of the RX disassembly Event Report Queue is the following:

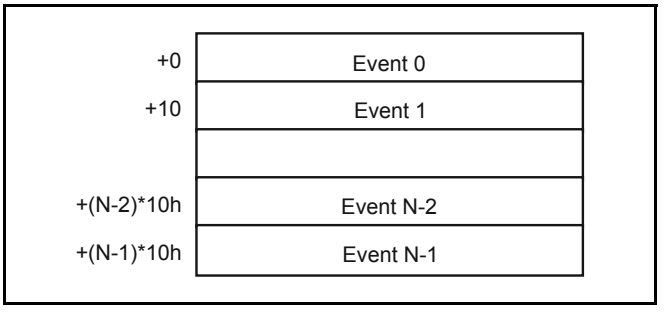


Figure 45 - RX Disassembly Event Report Queue

The RX Disassembly Event Report Queue is a circular buffer in SSRAM B used to report disassembly events to the host processor. It can be programmed to sizes of 4K bytes to 512K bytes in steps of 2^k. It must be mapped on a base address of its size. The position and size of this queue can be programmed at register address 0x720.

8.1 RTP Treatment

The disassembly module performs a 2-step process when treating received RTP packets: the first portion is performing all UDP/RTP de-encapsulation. The packet may be discarded if the UDP checksum is incorrect, depending on the polarity of the UDP Checksum Discard bit. Next, if RTP is present, the RTP version is checked and the packet is discarded if it is not equal to 2. Finally, the RTP headers are parsed if they are present and the packet is discarded if the headers extend beyond the end of the packet (i.e. the length of the packet and the info indicated in the headers cannot concord).

If a packet passes all of these tests, then the general RTP functions are performed. The packet is sent to an RX RTP Connection Structure; firstly, the packet's Payload Type and Marker bit are used to index in a Payload Type /Marker Bit Table, to determine what is the type of the received packet. Note that in RTP, payload types are often determined dynamically, and thus one table per connection might be required.

Packets received by the disassembly module may be tagged as xxPCM, HDLC or CPU packets, all depending on their payload type and marker bit. The information associated to the extension address also indicates, for xxPCM packets, what type of compression rate the packet is coded at (PCM, ADPCM 40, 32, 24, 16, or CN packet), as well as whether law translation should take place. For some combinations of payload type and marker bit, the table indicates that a report structure should be generated for this packet; this structure will report the payload type and marker bit in it. The table also indicates, for each payload type in RTP, if it should be included in the Network Jitter calculations (described in greater detail below), and which RX RTP Channel Structure Address it should use. Each disassembly structure can point to up to 16 RX RTP Channel Structure Addresses, and each of these can route packets to a different destination. Since a single connection will usually only carry a single voice channel (or aggregation of channels), this can use up to 6 extension structures (1 per voice coding rate), leaving 9 extension structures to route various associated signaling or control payload types to other extension structures (which can then send them to control agents like the CPU, or on the HDLC bus).

The table is pointed to by the Payload Type/Marker Bit Table Base field in the structure. Since the payload type and marker bit are, together, 8 bits, there are 256 entries in a table.

Conversely, in UDP-only connections, the UDP payload length is used to point in this structure. Since the pointer is only 8 bits, the UDP length will be capped, and the value 255 will be used if the payload length is greater than 255.

RTP Sequence number checking is also performed on each packet received by the structure. When a new packet arrives its sequence number is compared to the Last Received Sequence Number, and if they do not follow each other sequentially, a sequence number error will be reported and an error structure containing both sequence numbers will be reported. If the LR (Loss Report) bit is set, the module will report any packet loss by generating an error report structure that will contain both the previous and current sequence number.

Then, the Network Jitter may be calculated. The Network Jitter is a measure of how much the inter-arrival delay between 2 packets varies on average. When each packet is received, the module calculates the difference between its current time stamp and the Last Received Timestamp, as well as the difference in actual arrival time in the chip using the current local timestamp and the Last Local Timestamp.

Once it calculates this inter-arrival delay, it adds it to the previous network jitter using the function J = J*(15/16) + D*(1/16), where J is the old sum and D is the new value. When the software wishes to generate an RTCP packet, it can consult this value and insert it in the inter arrival jitter field of the RTCP packet.

Note that packets will only be included in the Network Jitter if, after looking them up in the treatment table, their Include Jitter bit is '1'. If this bit is '0', then the Network Jitter will not be updated and the Last Received Timestamp and Last Local Timestamp will not be modified. In other words, from a network jitter standpoint, it will be as if the packet never arrived.

All valid packets will also be included in the structure's 16-bit Received Packet count and 32-bit Received Octet Count. These are used to monitor the size and frequency of packets that are received. In the case of the octet count, it helps to see what is the average size of the packets being received. The RX Channel structures will also have their own counters.

When packets are looked-up in the table, the result will give an RX Channel Structure Number as well as the type of the RX Channel Structure. The RX Channel Structure Number is then used to select the correct RX Channel Structure Address from the RX connection structure and the RX Channel structure is read. This structure may be in the xxPCM, HDLC or CPU format. If the Type field in the RX Connection Structure associated to the RX Channel Structure is delete, then no RX Channel structure will be read and the packet will be deleted. Otherwise, the RX Channel structure is read and interpreted according to its type.

The format of the RX RTP Connection structure is the following:

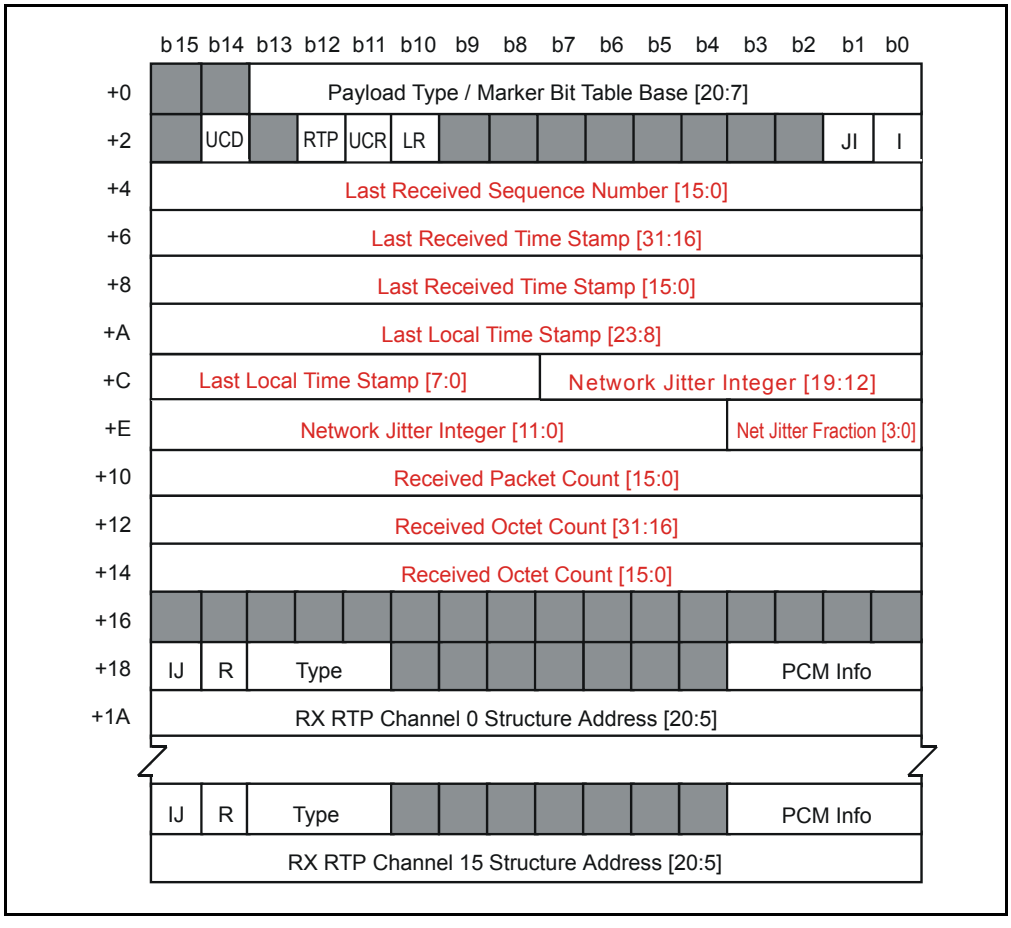


Figure 46 - RX RTP Connection Structure

Field	Description
Payload Type / Marker Bit Table Base	This field is a pointer to a Payload Type / Marker Bit Table Structure and is used to determine the index used in the 16-way split to a RX RTP Channel Structure. This pointer is defined in increments of 128 bytes.
UCD	UDP Checksum Error Discard. When this bit is ‘1’ and a packet is received with an invalid UDP checksum, the packet will be discarded prior to updating the content of this structure.
RTP	This bit must be set when packets received through this RX RTP Connection Structure are in the RTP format. This bit must be cleared when packets are non-RTP (i.e., any other protocol over UDP). The indexing to the Payload Type / Marker Bit table will be done with the Payload Type / Marker Bit for RTP, and will be done with the UDP Payload Length for all other protocols over UDP.

Table 29 - Fields and Description

Field	Description
UCR	UDP Checksum Error Report. When this bit is ‘1’, any packet received with an erroneous UDP checksum will generate an event in the RX Disassembly Event Report Queue.
LR	Loss Report. When this bit is set and two consecutive packets do not have sequential RTP sequence numbers, an event in the RX Disassembly Event Report Queue will be generated. This bit should be cleared if packets processed by this structure are not in the RTP format.
JI	Jitter init. This bit must be cleared by software when this structure is first created. It will be set by hardware when the first packet is received that has the IJ bit set in the RX RTP Channel Structure branching entry.
I	Initialized bit. This bit must be cleared by software when this structure is first created. It will be set by hardware when the first packet is received.
Last Received Sequence Number	Sequence number contained in the last received RTP packet. This field is not defined if packets processed by this structure are not in the RTP format.
Last Received Time Stamp	Time Stamp contained in the last received RTP packet. This field is not defined if packets processed by this structure are not in the RTP format.
Last Local Time Stamp	This field contains the local TDM Bus Time Stamp at the arrival time of the last packet processed by this structure. Only the bottom 24 bits of the local TDM Bus Time Stamp are kept. This field is used to calculate the RTP interpacket jitter.
Network Jitter Integer/Fraction	This field contains the interpacket jitter as monitored for all packets received via this structure, as defined by the RTP specification. The time unit of this field is frames.
Received Packet Count	This field is a counter of the total number of packets received on this connection so far.
Received Octet Count	This field is a counter of the total number of bytes received on this connection so far. For each packet received, this field will increment by UDP Length - 8.
IJ	Include Jitter. When this bit is set, the RTP Time Stamp and Sequence Number of any packet that passes through this RX RTP Channel Entry Split will be used to update the following fields: Last Received Sequence Number, Last Received Time Stamp, Network Jitter Integer and Fraction.
R	Report UUI/LI Combination to CPU through the RX Disassembly Event Report Queue (when set).
Type	Type of RX RTP Channel Structure. “000” = PCM + ADPCM; “001” = HDLC; “010” = RX CPU; “111” = delete; others = reserved.

Table 29 - Fields and Description (continued)

Field	Description
xxPCM Info	This field contains additional information about PCM and ADPCM connection packets. It is used to distinguish PCM and ADPCM encoding as well as CN packet. “0000” = PCM 64 kbps (no law change); “0001” = ADPCM 40 kbps; “0010” = ADPCM 32 kbps; “0011” = ADPCM 24 kbps; “0100” = ADPCM 16; kbps; “0101” = CN Packet (one byte, white, internal); “1000” = PCM 64 kbps (u-law -> A-law); “1001” = PCM 64 kbps (A-law -> u-law); “1010” = PCM 64 kbps (A-law -> u-law (except 00h)); “1011” = PCM 64 kbps (u-law -> u-law (except 00h)); “1100” = CN Packet (one/multi byte, white/pink, external); others = reserved.
RX RTP Channel Structure Address	This field is a pointer to an RX RTP Channel Structure that will be used to complete packet processing, sending it either to the TDM bus in PCM or HDLC format, or to an RX CPU Packet queue. This address points to increments of 32 bytes.

Table 29 - Fields and Description (continued)

The format of the Payload Type/Marker Bit Table is the following:

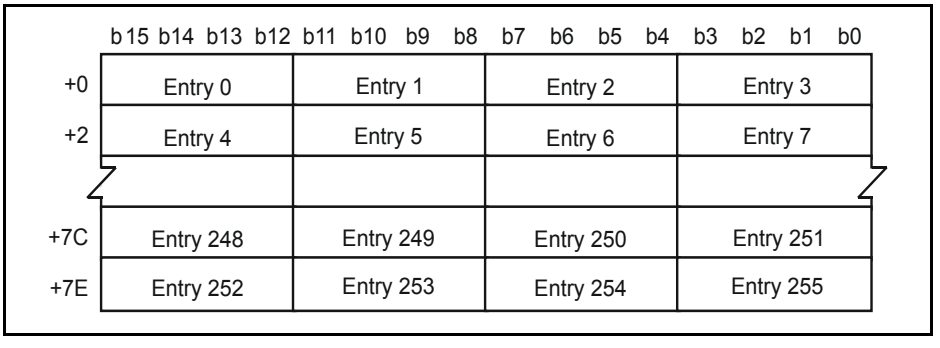


Figure 47 - Payload Type/Marker Bit Table

Field	Description
Entry 0	M = ‘0’ & PT = 0x00 (RTP), UDP Payload Len = 0 bytes (UDP)
Entry 1	M = ‘0’ & PT = 0x01 (RTP), UDP Payload Len = 1 bytes (UDP)
Entry 254	M = ‘1’ & PT = 0x7E (RTP), UDP Payload Len = 254 bytes (UDP)
Entry 255	M = ‘1’ & PT = 0x7F (RTP), UDP Payload Len = 255 or more bytes (UDP)
Note	Structure is 128 bytes long and must start on a 128-byte boundary. Each entry contains a 4-bit number used to select one of the 16 RX RTP Channel Structures to continue packet processing.

Table 30 - Fields and Description

RTP connections can generate Event Reports when special events or errors occur. These Event Reports indicate all UDP and RTP packaging errors encounters while parsing the packet. This includes fatal errors like UDP Checksum Error, RTP Version Mismatch and Bad packet Length 1 (which indicates that the RTP headers were too long for the packet length). It also reports structure Initialization, RTP packet Loss (detected through the sequence numbers) and Payload Type Report, which is set when the Report bit associated to the RX RTP Channel Structure is set. This structure reports the current packet's sequence number and the sequence number of the previous received packet (when RTP is enabled), as well as the Payload Type and Marker bit for RTP packets or the UDP Payload Length for UDP-only packets. The current Local Bus Timestamp is also written in the structure.

These errors are documented in the following figure that gives the format of these events:

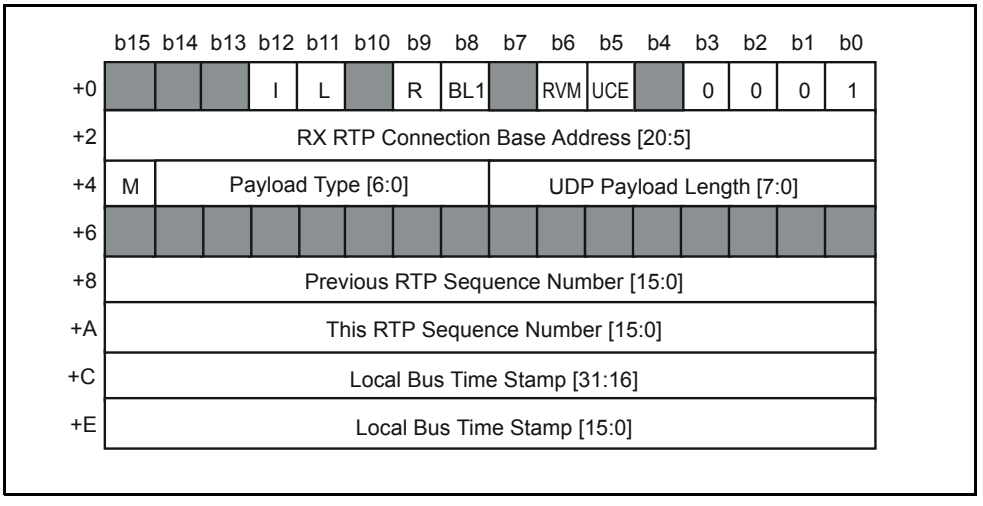


Figure 48 - RX Disassembly Event Report Queue - RTP Connection Report

Field	Description
I	Set at reception of the first packet on the RX Connection Structure.
L	Loss of one or many RTP packets detected because the sequence number fields of two consecutive packets were not incremental. The Previous and Current Sequence Numbers will indicate number of lost packets.
R	This bit is set when the R bit is set in the RX Channel Structure Split Entry (as taken for the packet).
BL1	Bad Packet Length 1. When set, a packet has been received with insufficient bytes to contain an RTP header. It will be discarded entirely.
RVM	RTP Version Mismatch. When ‘1’, this means that a packet was received with RTP version not equal to 2.
UCE	UDP Checksum Error. When set, a packet has been received with an incorrect UDP checksum.
RX RTP Connection Structure Address	Base address of the RX RTP Connection Structure that generated this report structure.
M & Payload Type	Theses two fields contain the value of the Marker bit and Payload Type of the RTP packet that caused this error structure to be generated.

Table 31 - Fields and Description

Field	Description
UDP Payload Length	This field contains the length of the UDP payload of the received packet that caused this structure to be generated. If the UDP Payload is longer than 255 bytes, this field will be saturated at value 255.
Previous RTP Sequence Number	RTP Sequence number of the second last packet that was received.
This RTP Sequence Number	RTP Sequence number of the last packet that was received.
Local Bus Time Stamp	Local Bus Time Stamp present at the time of reception of the packet that caused this report structure to be generated.

Table 31 - Fields and Description (continued)

8.2 xxPCM Treatment

In xxPCM, packet loss and adjustment can be performed by using the timestamps of the received packets. If the LE (Loss Enable) bit is set, the module will perform packet loss/misinsertion compensation by adjusting its pointer by the number of TDM frames that have elapsed. This is very useful when performing silence suppression, where the loss of packets is not only accepted but expected. In practice, Loss Enable should always be set except in UDP, where there is no timestamp to measure time with. The Last Packet Remote Timestamp, as well as the current timestamp of the packet, are used for this purpose in RTP.

The remaining fields in the xxPCM RX Channel structure are mostly used for error reporting and monitoring. AR (Always Report) and SR (Slip Report) each indicate whether or not an error of the given type will cause an error structure to be generated. Note that if an error structure is generated, all errors that occurred on that packet will be reported, regardless of their Enable bit.

The extension structure contains a 16-bit Received Packet count and a 32-bit Received Octet Count. These have the same definition as those in the disassembly structure (i.e. count UDP payload), but since all packets that are treated by this structure are valid UDP/RTP packets, they will all be included in the counts. It is important that the size of the payload in each packet be a multiple of the number of bearers present on the connection: if not, the BL2 bit (Bad Packet Length 2) in the RX error report structure will be flagged.

Finally, at the very end of the structure, are a certain number of Circular Buffer Base Addresses that indicate, for each channel in the connection, where is the circular buffer associated to it. These addresses point to 256 byte boundaries, which is the smallest allowed size for any RX circular buffer. The buffers can be between 256 bytes and 4K bytes in size, as indicated by the Buffer Size field.

xxPCM channels can receive CN packets in addition to normal packets filled with voice payload. Two types of CN packets can be received: single-byte, “white” packets that indicate only an energy level, and multi-byte, “spectral” packets containing both an energy level and spectral information.

RX xxPCM channel structures can generate clock recovery pulses to the clock recovery module to indicate the arrival of packets. The MT92210 allows 2 redundant clock recovery circuits to run simultaneously, so the extension structures contain 2 bits, clock recovery A and clock recovery B, which the structure used to generate pulses. When it generates packet arrival pulses to the clock recovery module, the disassembly process also sends it the timestamp and the sequence number of the current packet (if RTP is being used). This will more accurately allow the clock recovery process to reconstruct the relationship between mclk and the packet arrival rate.

The format of the xxPCM RX RTP Channel Structure is the following:

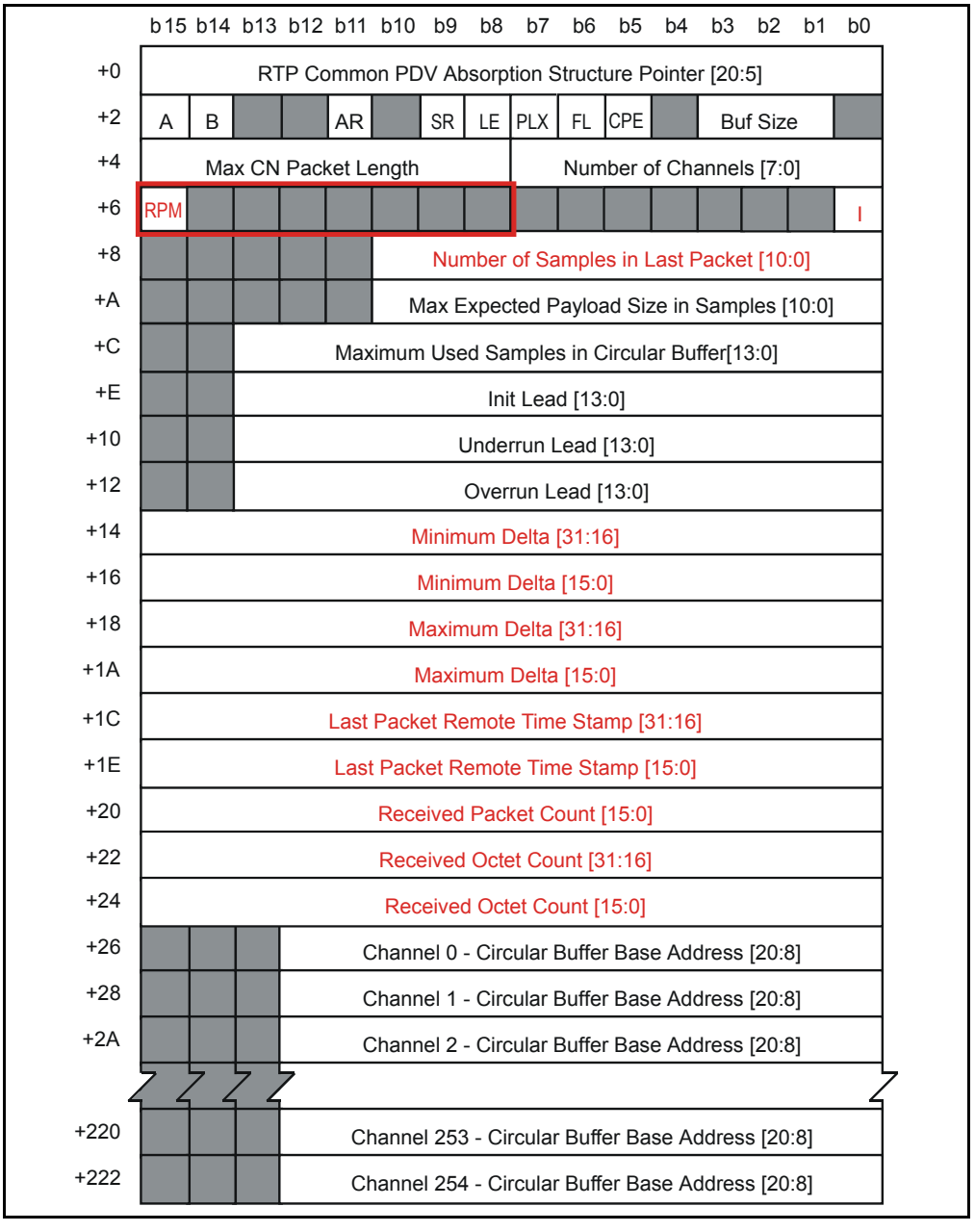


Figure 49 - RX RTP xxPCM Channel Structure

Field	Description
RTP Common PDV Absorption Structure Pointer	Pointer to PDV monitoring and absorption structure that will be used by this channel. This pointer is in units of 32 bytes.
A	Clock Recovery channel A. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue A.
B	Clock Recovery channel B. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue B.
AR	Always Report Enable. Always generate an RX Disassembly Event Report Structure when a packet is received and this field is set. Used for debugging only.
SR	Slip Report Enable. When this field is set, any underrun or overrun will be reported via an RX Disassembly Event Report Structure.
LE	Packet Loss Compensation Enable. When set, the RTP time stamp will be used to align packets in the RX circular buffer for proper dejittering. When cleared, packets will be written back-to-back in the RX Circular buffer regardless of their time stamp.
PLX	Report packets that are longer than expected. When this field is set and a packet is received containing more than Max Expected Payload Size in Frames payload samples, an RX Disassembly Event Report Structure will be generated.
FL	Report packet loss. When this field is set and Packet Loss Compensation is set, packets received that are not written immediately after the last received packet in the RX circular buffer will cause an RX Disassembly Event Report Structure to be generated (indicating a packet loss).
CPE	CN PDV Monitoring Enable. When cleared, CN packets will not affect PDV monitoring fields. When set, CN packet will be assumed to contain as many frames as the last packet that was received. This assumption may not always be correct when the number of samples in a packet changes in time.
BS	RX Circular Buffer Size. This field indicates the size of RX Circular Buffer that will be used to dejitter incoming packets. "000" = 256 bytes; "001" = 512 bytes; "010" = 1024 bytes; "011" = 2048 bytes; "100" = 4096 bytes; others = reserved.
Max CN Packet Length	CN packets longer than this value will be truncated to this length.
Number of Channels	This field contains the number of channels that are interleaved in a T1/AAL1 like manner in each packet. The field ranges between 1 and 255.
RPM	Reset PDV Monitoring. When this bit is written to '1' by the software, the next received packet will begin a new PDV monitoring period and, at the time of the reception of that packet, the RPM bit will be cleared automatically by the hardware. When PDV monitoring is reset, the Minimum Delta and Maximum Delta are set to the delta of the first packet received in the new PDV monitoring period.

Table 32 - Fields and Description

Field	Description
I	Initialized bit. This bit is written to ‘0’ by software at channel initialization. It will be written to ‘1’ by the hardware upon the reception of the first packet.
Number of Samples in Last Packet	This field contains the number of samples received in the last packet. It should be initialized to zeros upon channel initialization.
Max Expected Payload Size in Samples	This field indicates the maximum number of samples that are expected per packet received on this channel. Setting this field correctly ensures that changes in the number of samples per packet do not cause any slips. If the number of samples in a packet is unknown, this field should be set to 0.
Maximum Used Samples in Circular Buffer	This is the number of samples used for packetization delay absorption and PDV absorption. A value of 0 will accept 1 samples packets with no PDV. For the typical case of 160 byte packets with 32 ms of PDV, a value of 160 + 256 - 1 = 415 will be written here.
Init Lead	Position at which the first received sample will be written in the RX circular buffer. A value of 0 in this field means that the first received sample of the first packet will be sent on the H.110 bus immediately after reception. This field is in units of 1 frame. This field is usually set to the Expected PDV / 2.
Underrun Lead	Position at which the first received sample in an underrun packet will be written in the RX circular buffer. This field is typically 8 frames.
Overrun Lead	Position at which the first received sample in an overrun packet will be written in the RX circular buffer. This field is typically Expected PDV / 2.
Minimum Delta	PDV Monitoring Field used to indicate the minimum delta that was monitored between the local timestamp and the remote time stamp.
Maximum Delta	PDV Monitoring Field used to indicate the maximum delta that was monitored between the local time stamp and the remote time stamp.
Last Packet Remote Time Stamp	This field contains the last packet received remote time stamp.
Received Packet Count	Total number of packets received on this channel so far. Every packet that gets to this structure will be included.
Received Octet Count	Total number of bytes received on this channel so far (RTP header + payload included). Every packet that gets to this structure will be included.
Circular Buffer Base Address	Base address of the circular buffer where a given channel is written. This base address is in units of 256 bytes. When only one channel is present (i.e. Number of Channels = 1), only the first Circular Buffer Base Address needs to be programmed of the 255 possible base addresses.

Table 32 - Fields and Description (continued)

Treatment of xxPCM channels can produce Event Reports that contain xxPCM-specific errors: Underrun errors, Underrun 2 errors, Overrun errors, as well as BL2 (Bad Packet Length 2) which means that the number of payload samples was not divisible by the number of bearers, and the BL3 (Bad Packet Length 3) which means that the packet was too large to fit in the circular buffer. If any samples were lost, the SL bit will be set, and the number of lost samples will be reported in the 32-bit Samples Lost field. Initialization of the channel in the RX xxPCM Channel Structure, or the I2, Initialization in the Common PDV Absorption Structure can also be reported. In addition to reporting all these errors, the xxPCM report structure gives the base address of the RX xxPCM Channel structure (to identify the channel to which the report structure belongs), the current Local Bus Timestamp and the current Remote Timestamp.

The format of the xxPCM Event Report is the following:

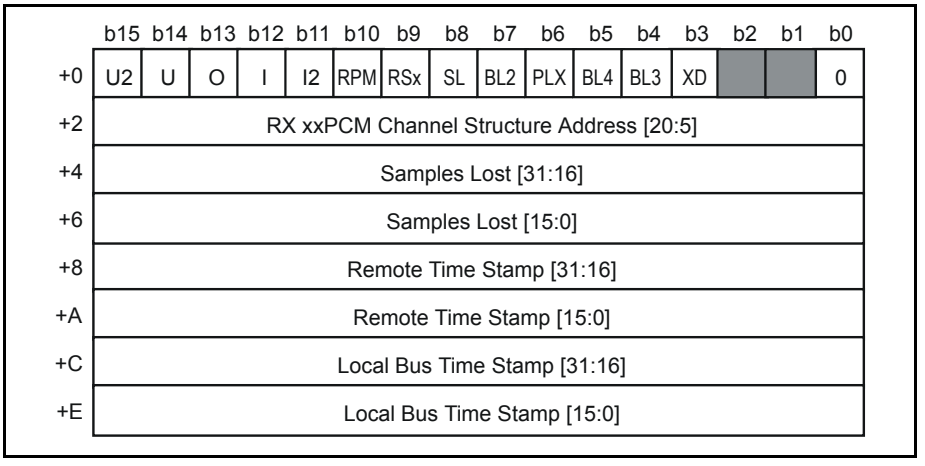


Figure 50 - RX Disassembly Event Report Queue - xxPCM Channel Report

Field	Description
U2	Underrun Detected 2. This bit is set when a packet is received with a remote time stamp that would require the first sample of the packet to be output on the TDM bus one or many frames ago. This means that the received packet was late as compared to its expected arrival time.
U	Underrun Detected. This bit is set when a packet is received with a remote time stamp that would require the first sample of the packet to be output on the TDM bus one or many frames ago after the worst expected packetization delay was added to the arrival time. For example, if a 44 sample PCM packet is processed by a structure that expects 88 samples as the maximum packet size, it may only be output on the TDM bus in 44 frames; a received time stamp that would contradict this requirement would cause an underrun slip.
O	Overrun Detected. This bit is set when a packet is received with a remote timestamp and number of samples that require the last sample of the packet to be sent out on the TDM bus with a delay greater that the Maximum Used Samples in Circular Buffer.
I	When set, a packet has been received for the first time by the RX xxPCM Channel Structure.
I2	When set, a packet has been received for the first time by the Common PDV Absorption structure.
RPM	Reset PDV Monitoring Completed. When this bit is set, the RPM bit in the RX xxPCM Channel Structure has been cleared by hardware and a new PDV monitoring period has been started.
RSx	Reset Total Slip Offset Delta Completed. When this bit is set, the Total Slip Offset Delta has been reset in the Common PDV Absorption structure.
SL	Samples lost. When set, the Samples Lost[31:0] field is non-zero, meaning that one or many samples have been lost between packets.

Table 33 - Fields and Description

Field	Description
BL2	Bad Packet Length 2. When set, the number of “payload bytes” * 8 was not divisible “Number of channels” * “Number of bits per sample”. Packets affected by this error will be discarded.
PLX	Packet Longer than Expected. The number of samples in the received packet exceeded the “Max Expected Payload Size in Samples” field of the RX xxPCM Channel Structure,
BL4	Bad Packet Length 4. The number of samples in the received packet exceeded 2047 frames.
BL3	Bad Packet Length 3. The number of samples in the received packet was 0 frames.
XD	Extreme Delay. When ‘1’, extreme delay has been detected in the Extended PDV monitoring structure.
RX xxPCM Channel Structure Address	Base address of the RX xxPCM Channel Structure that generated this report structure.
Samples Lost	Number of samples lost due to packet loss as detected by the difference between remote time stamps in two consecutive packets.
Remote Time Stamp	Remote time stamp of the packet received that cause the generation of this report structure.
Local Bus Time Stamp	Local Bus Time Stamp present at the time of reception of the packet that caused this report structure to be generated.

Table 33 - Fields and Description (continued)

8.3 Packet Delay Variation (PDV) Monitoring

The MT92210's PDV monitoring writes packets within a PDV window, minimizing delay while trying to avoid slips. To do this, it writes packets based on their remote timestamp: a packet's position in the circular buffer is decided by its RTP timestamp. This allows compensation for packet loss and misinsertion.

Because some profiles in RTP use varying-length packets (the term varying-length here refers to changing number of samples per packet, not length in bytes; compressed packets are converted to a number of samples before PDV monitoring is applied to them), changing packets lengths will show up as PDV on the network. To compensate for this, the xxPCM channel structure contains a Max Expected Payload Size in Samples field. Any variation in packet length up to this size will be compensated for. Setting this field to too large a value will insert delay; setting this field to too small a value will allow the changing length to show up as PDV. If the value is not known with certainty, a smaller value is usually better.

The default mode of the PDV monitoring system allows a PDV window of length Maximum Used Samples in Circular Buffer. (Note: the Max Expected Payload Size in Samples - 1 must be added to Maximum Used Samples in Circular Buffer to get a window of the desired size. As an example, if the Max Expected Payload Size in Samples is 160 and the desired PDV window is 10 ms (80 samples) long, then Maximum Used Samples in Circular Buffer should be set to 239 (80 + (160-1)).

An underrun occurs when not enough data is received by the disassembly module, so the desired write pointer's lead vs. the TDM read pointer would be smaller than 0. In this case, the write pointer's position is reset to Underrun Lead + current TDM read pointer.

An overrun occurs when too much data is received by the disassembly module, so the desired write pointer's lead vs. the TDM read pointer would be greater than Maximum Used Samples in Circular Buffer. In this case, the write pointer's position is reset to Overrun Lead + current TDM read pointer.

Finally, the Init Lead is used to reset the write pointer when the first packet is received on the channel.

The Underrun Lead and Overrun Lead are used to control the size of slips when they occur. Some algorithms prefer to place both the Underrun Lead and Overrun Lead at the center of the buffer, which minimizes the number of slips, but causes large slips and may add as much as (Maximum Used Samples in Circular Buffer / 2) of extra delay. Another possible algorithm is to place both the Underrun Lead and Init Lead to a small value K (e.g. 8 samples) and the Overrun Lead to (Maximum Used Samples in Circular Buffer - K). This causes slips of 8 bytes, may results in multiple slips, but ensures that no more than K samples of delay are ever inserted on the data.

The MT92210 PDV monitoring may be used to control the end-to-end delay experienced on a given channel. To do so, the software initializes the Desired Remote/Local Timestamp Delta field to 0. This gives the expected delta between the timestamp in the RTP packet and the timestamp on the H.110 bus. The chip then Overruns and Underruns according to the need and settles within its PDV window. Each slip affects the Total Slip Offset Delta, which measures by how many samples the actual delta is off from the Desired Remote/Local Timestamp Delta.

Software can then come and fix the Desired Remote/Local Timestamp Delta by adding to it the Total Slip Offset Delta and resetting the Total Slip Offset Delta. Note that a Total Slip Offset Delta of 12 and a Desired Remote/Local Timestamp Delta of 35 is the same as a Total Slip Offset Delta of 0 and a Desired Remote/Local Timestamp Delta of 47.

The Minimum Delta gives the smallest value of Remote vs. Local delta seen so far (Minimum Delta means earliest packet, so the packet most likely to cause an overrun) while the Maximum Delta gives the largest value of Remote vs. Local delta seen so far (Maximum Delta means latest packet, so the packet most likely to cause an underrun).

Thanks to this diagnostic, software can choose to change the Desired Remote/Local Timestamp Delta by another value, if, for example, the Maximum Delta indicates that a packet has been seen that would cause the current delta values to slip (this might be an indication that the PDV window is too small). If so, this will cause a slip and the software can choose when it wants the slip to occur. It can occur immediately, by setting the RSO (Reset Total Slip Offset Delta Once) bit, which will cause it to slip on the next packet. It can also wait for a packet with a marker bit = '1' by setting the RSM (Reset Total Slip Offset Delta on Marker). Finally, it can choose to reset when a significant amount of data has been lost by setting the RSL (Reset Total Slip Offset on Loss) bit. The Min Slip field defines what a “significant” amount of data is in 2^n increments, between 2 ms and 256 ms.

Some application may want the end-to-end delay to be fixed and unchanging, even in the case of slips. If that is the case, setting the RSA (Reset Total Slip Offset Delta Always) will ensure that the Total Slip Offset Delta never changes, which disables slips. This could lead to a loss of data integrity in the case of Underruns (or Overruns, but only if they manage to overflow the entire physical circular buffer, not only the section reserved by Maximum Used Samples in Circular Buffer).

This end-to-end delay control may be used in several applications:

- **End-to-end slipless framing**. This setup allows one or multiple PCM bearers to be transported end-to-end over a packet network without slips.
- **Clear-channel DS3**. Because the PDV monitoring allows multiple PCM channels to connect to a single Common PDV Absorption Structure, up to 1023 individual PCM bearers could use the same PDV information; this environment could tolerate slips where a single channel could cause a slip on all channels, maintaining the end-to-end delay across all channels.
- **Interchip synchronization**. Because the desired Remote/Local Timestamp delta is an absolute value (not relative to anything inside the chip), multiple chips can maintain the same end-to-end delay. The only requirement here is that all sources be capable of generating synchronized timestamps. MT92210 is capable of this because multiple chips can slave off the same H.110 bus timestamp.
- **Glitchless fallback**. Because multiple chips can maintain delay consistency, channels and connections can be swapped from one chip to another without a single byte loss.

The format of the RX RTP Common PDV Absorption structure is the following:

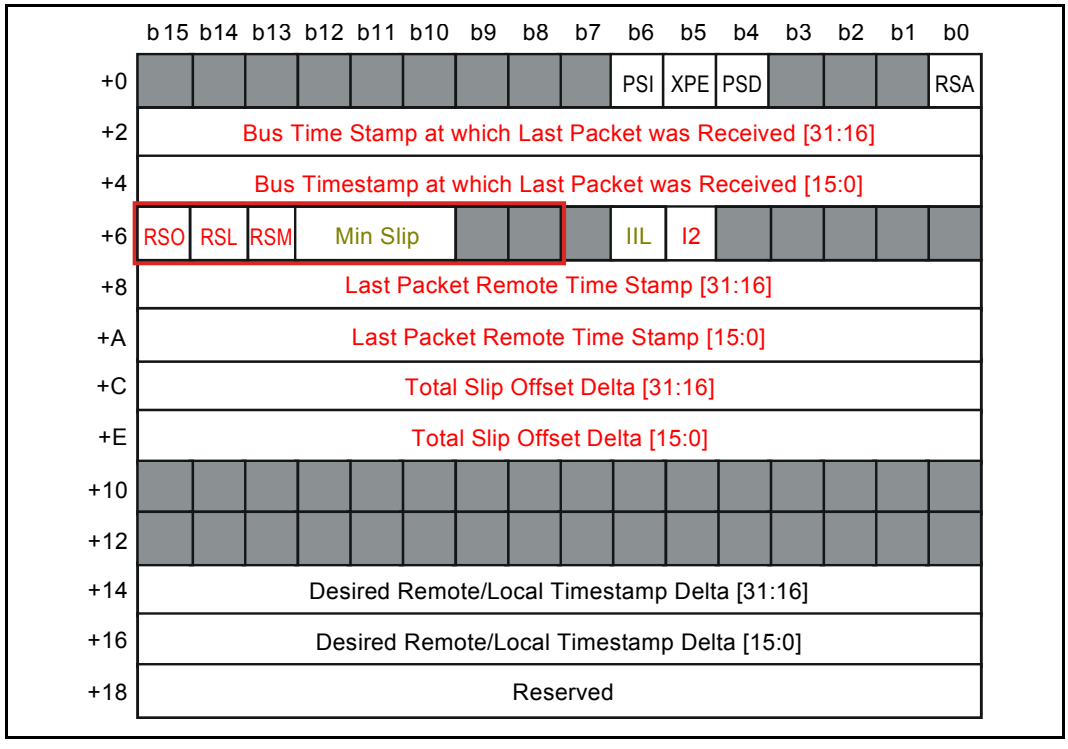


Figure 51 - RTP Common PDV Absorption Structure

Field	Description
PSI	Prevent Total Slip Offset Delta from incrementing. When ‘1’, overrun slips will not cause the Total Slip Offset Delta from being changed. Overruns in this mode can only generate an RX Disassembly Event Structure. For proper data integrity behaviour, the software should correct the Total Slip Offset Delta manually when overruns occur and this bit is set.
XPE	Write 0 for normal operation.
PSD	Prevent Total Slip Offset Delta from decrement. When ‘1’, underrun slips will not cause the Total Slip Offset Delta from being changed. In this mode, underruns can only generate an RX Disassembly Event Structure. For proper data integrity behaviour, the software should correct the Total Slip Offset Delta manually when underruns occur and this bit is set.
RSA	Reset Total Slip Offset Delta Always. When this bit is set, all packets received will be written to the TDM bus at a time stamp delta exactly equal to the Desired Remote/Local Time Stamp Delta. Setting this bit gives controlling software full control of the de-jittering functionality.
Bus Time Stamp at which last Packet was received	This field contains the local 32-bit bus time stamp that was present when the last received packet was processed by the PDV monitoring block.

Table 34 - Fields and Description

Field	Description
RSO	Reset Total Slip Offset Delta when the next packet is received. When this bit is set by software, the next packet to be processed by the PDV monitoring block will cause the Total Slip Offset Delta to be reset to zero. The hardware will clear this bit immediately after the Desired Remote/Local Time Stamp delta has been applied.
RSL	Reset Total Slip Offset Delta when the next packet is received after a significant silence period. When this bit is set by software, the next packet processed by the PDV monitoring block after a long period on inactivity (as defined by the Min Slip field) will cause the Total Slip Offset Delta to be reset to zero. The hardware will clear this bit immediately after the Desired Remote/Local Time Stamp Delta has been applied.
RSM	Reset Total Slip Offset Delta when the next packet is received with the RTP marker bit set. When this bit is set by software, the next packet to be processed by the PDV monitoring block and whose RTP marker bit is set will cause the Total Slip Offset Delta to be reset to zero. The hardware will clear this bit immediately after the Desired Remote/Local Time Stamp delta has been applied.
Min Slip	This field defines how many frames of lost data are considered enough to identify the gap as a silence period in order to cause the Total Slip Offset Delta to be reset when the RSL bit is set. “000” = 16 frames; “001” = 32 frames; “010” = 64 frames; “011” = 128 frames; “100” = 256 frames; “101” = 512 frames; “110” = 1024 frames; “111” = 2048 frames.
IIL	Init with Init Lead. When this bit is set and a packet is received with the I2 bit cleared, the Total Slip Offset Delta will be set to the shortest allowed delay between the remote and local time stamp plus the Init Lead programmed in the RX RTP Disassembly Extension Structure. When this bit is cleared, the Desired Remote/Local Time Stamp Delta will be assumed to be correct and the normal underrun/overrun slipping will take place.
I2	Init bit. This bit must be written to zero by software when the structure is initialized. When the first packet is processed by this structure, the I2 bit will be written back to ‘1’ by the hardware.
Last Packet Remote Time Stamp	32-bit time stamp at which the packet was sent by the remote end (i.e. the RTP time stamp). When the RTP protocol is not present, this field simply increments by the number of frames received in the last packet. This “emulates” an RTP time stamp when no packet loss, misinsertion or silence suppression has occurred since the last packet arrival.
Total Slip Offset Delta	This value is the current number of frames either added or dropped due to overruns and underruns. A positive number represents overruns and a negative number represents underruns. Note that underrun frames and overruns frames cancel-out in this field.

Table 34 - Fields and Description (continued)

Field	Description
Desired Remote-Local Time Stamp Delta	This field contains the desired local/remote time stamp delta that should be applied to all received packets. A time stamp delta of zero means that a packet received with time stamp of zero would see its first byte output on the bus when the local bus time stamp is zero. This field should be written to zero at initialization time.
Reserved	Write 0 for normal operation.
Note	all remote/local time stamp deltas are calculated with the following equation: delta = remote_packet_timestamp - local_bus_timestamp;

Table 34 - Fields and Description (continued)

8.4 HDLC Treatment

In the other case, the RX Channel Structure Address used points to an RX HDLC Channel structure. The RX HDLC Channel structure can do policing, report diagnostic, and indicate what type of framing to perform.

The HDLC stream number indicates to which of the 512 HDLC streams the HDLC packets obtained here belong: this will allow a packet descriptor to be written in the queue of the correct stream. The Header Type bits indicate what should be the form of the HDLC header inserted on the RTP packets: zero, one or two bytes of address, and zero or one control bytes. The CRC bit indicates if a 16-bit CRC should be appended at the end of the RTP packet. If these fields are added, their values are contained in the HDLC Address and HDLC Control Byte fields in the control structure. Finally, the 16-bit Received Packet count and a 32-bit Received Octet Count are also present, to allow solid diagnostics.

Before writing the HDLC RTP packet in its circular buffer, the disassembly performs all of the header, CRC and zero-insertion functions. The MT92210 supports 2 types of HDLC framing: bit-wise and byte-wise framing (see Annex for more details). The appropriate form of framing is applied to the entire packet (including headers and CRC) before the packet is written in its circular buffer. When the packet is ready, the disassembly consults an HDLC control structure that contains the base address of the circular buffer, as well as the current read and write pointers. If there is enough room left in the circular buffer, the disassembly will write the entire packet into the buffer, then write a packet descriptor indicating where the packet can be read and what its length is. It will also write back the new write pointer. The RX TDM can then read the packet descriptor and send the packet onto the TDM bus.

RX HDLC channel structures also implement a policing mechanism that prevents misbehaving channels from flooding an entire HDLC stream. This policing is implemented using a leaky bucket approach: a Maximum Bucket Fill indicates how full the bucket may be before packets start to be discarded, and the Discharge Rate indicates how quickly the bucket should be emptied. The bucket's fill is always positive, so in the best-case scenario the fill is 0. This ensures that, even if a connection has not received a packet for an infinite amount of time, it will not accept an infinite size packet when it does receive one! Whenever a packet is received, the bucket is first “discharged”: this means that the amount of time elapsed since the last received packet is calculated and is subtracted from the Current Bucket Fill. Then, the new packet's size is added to the bucket. If the result is larger than the Maximum Bucket Fill, the packet will be discarded, an Event Report structure will be generated containing the Policing Error bit set, and the current packet's fill will not be added to the Current Bucket Fill. A well-configured policing mechanism can ensure that the circular buffer never overflows.

The format of the RX RTP HDLC channel structure is the following:

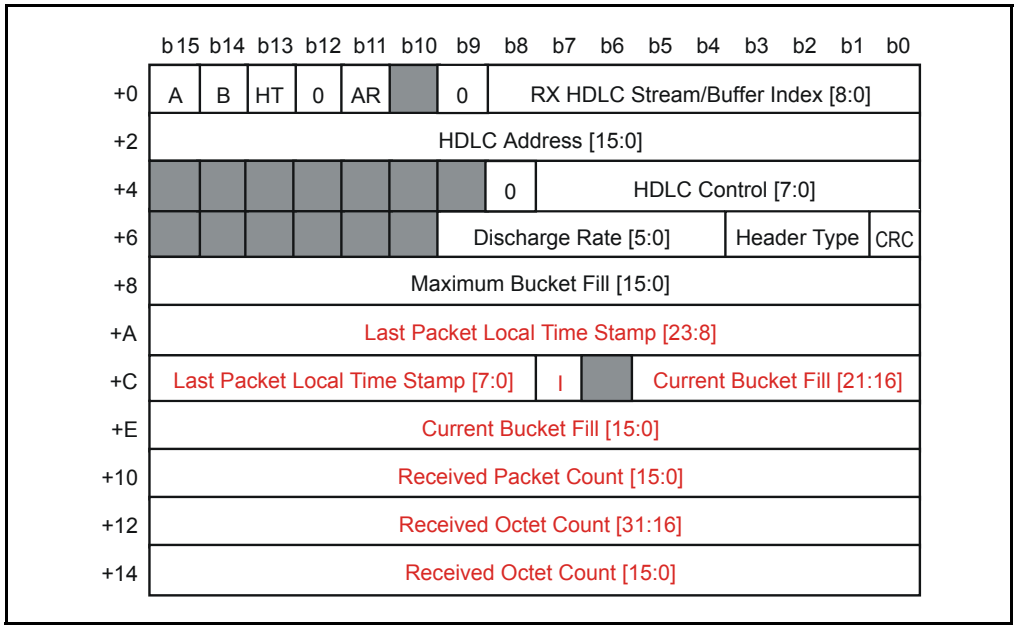


Figure 52 - RX RTP HDLC Channel Structure

Field	Description
A	Clock Recovery channel A. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue A.
B	Clock Recovery channel B. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue B.
HT	HDLC Encapsulation type. '0' = HDLC Bit-wise; '1' = HDLC Byte-wise.
AR	Always Report Enable. Always generate an RX Disassembly Event Report Structure when a packet is received and this field is set. Used for debugging only.
RX HDLC Stream/Buffer Index	Index in the RX HDLC Stream/Buffer Control Table in SSRAM B.
HDLC Address	Field that may be inserted in the HDLC packet header, depending on the Header Type.
HDLC Control	Field that may be inserted in the HDLC packet, depending on the Header Type.
Discharge Rate	Long term rate at which bytes can be received on this HDLC channel. The current bucket fill is reduced at this rate. The rate is defined as a number of bytes per frame that can be sent on the HDLC stream (including all forms of HDLC framing). This is a floating point number; it has two bits of mantissa (excluding the hidden leading '1') and 4 bits of exponent. Zero is not a legal value. The minimum rate is 4 kbps, which is defined as mantissa = "100" and exponent = "0001". So to calculate the rate, the following equation can be used: 4 kbps * (mantissa/8) * (2^exponent).
Header Type	This can be set to: "000"=Plain; "001"=One byte address; "010"=Two byte address; "011"=One byte address + Control; "100"=Two byte address + Control; others = reserved.
CRC	'0' = no CRC bytes will be appended to the HDLC packet; '1' = a 16-bit CRC-16 (otherwise known as CRC-CCITT) will be inserted at the end of the HDLC packet.
Maximum Bucket Fill	Up to this many bytes can be contained in the "policing bucket" of this HDLC channel before packet discarding takes place. Largest value is 0x0000 (which means 65536 bytes), smallest value is 0x1 (which means 1 byte).
CBF	Current Bucket Fill. Current fill of the "policing bucket" in 1/64th of a byte. This field should be initialized to all zeros by software.
Last Packet Local Time Stamp	Time stamp in frames of the last received packet used to determine the time delta between the reception of HDLC packets. Used to discharge the Current Bucket Fill.
I	Initialized bit. Written to zero by software channel initialization. Written to '1' by hardware as soon at the first packet is received on this channel.
Received Packet Count	Total number of packets received on this channel so far. Every packet that gets to this structure will be included.

Table 35 - Fields and Description

Field	Description
Received Octet Count	Total number of bytes received on this channel so far. Every packet that gets to this structure will be included. This field increments by the UDP Length - 8 each time packet is received.

Table 35 - Fields and Description (continued)

As in xxPCM, the RX HDLC channel structure can request the generation of clock recovery events. It can also generate Event Reports, the format of which is the following:

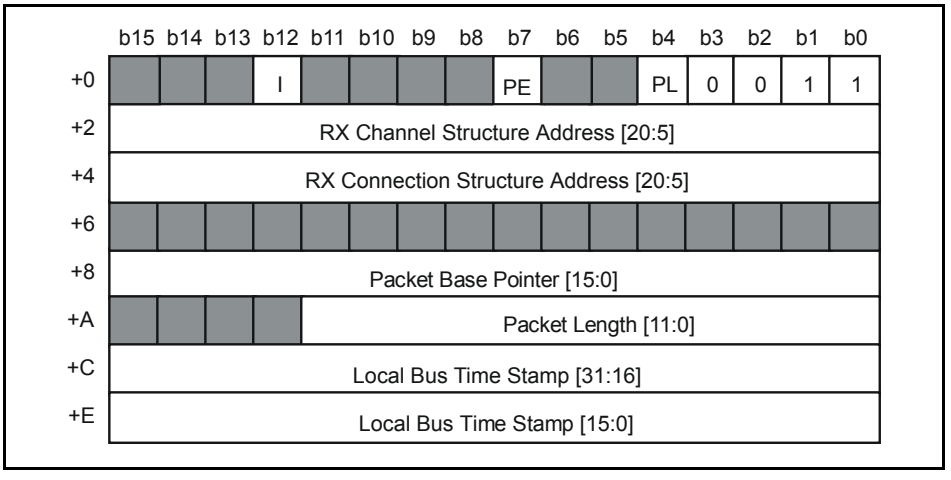


Figure 53 - RX Disassembly Event Report Queue - HDLC / CPU Channel Report

Field	Description
I	Set at the reception of the first packet on this RX HDLC/CPU Channel Structure.
PE	Policing Error. Indicates that the HDLC / CPU packet was discarded because it caused the policing bucket to overflow.
PL	Packet Lost. When set, this bit indicates that a packet was lost due to the lack of room in the Circular buffer to which this packet was to be written.
RX Channel Structure Address	Base address of the RX HDLC/CPU Channel Structure that generated this report structure.
RX Connection Structure Address	Base address of the RX Connection Structure that led to the RX Channel Structure which generated this report structure.
Packet Base Pointer	Pointer to the first voice byte of this packet in the RX Circular buffer. The pointer is relative to the base address of the RX Circular Buffer. This field is only defined for RX CPU Packets.
Packet Length	Length of the packet in bytes defined by the number of actual bytes used in the circular buffer. This field is only defined for RX CPU Packets.
Local Bus Time Stamp	Local Bus Time Stamp present at the time of reception of the packet that caused this report structure to be generated.

Table 36 - Fields and Description

8.5 CPU Treatment

The CPU manages its packets in much the same way as an HDLC stream does: it reserves a circular buffer of a certain size and even reserves RX CPU Buffer Control Tables in much the same was HDLC streams use RX HDLC Stream/Buffer Control Tables. All channels may write their packets into that circular buffer. Each CPU packet received will generate an error report structure and they can be retrieved through the pointers these structures contain. The CPU may even choose to route its packets to several different circular buffers. Finally, the CPU can also receive an interrupt informing it that its circular buffers are getting “too full”: if any of the CPU-destined circular buffers becomes more than half full, an interrupt can be generated. The CPU can then read the report structure FIFO and empty all circular buffers of their packets.

The RX CPU Channel Structure also implements the same type of policing found in HDLC, for exactly the same reasons. A misbehaving channel should not be able to flood the CPU with packets, preventing other, well-behaved channels from getting attention. The fields in the RX CPU Channel Structure are identical to those in the RX HDLC Channel Structure.

The format of the RX CPU Buffer Control Table is the following:

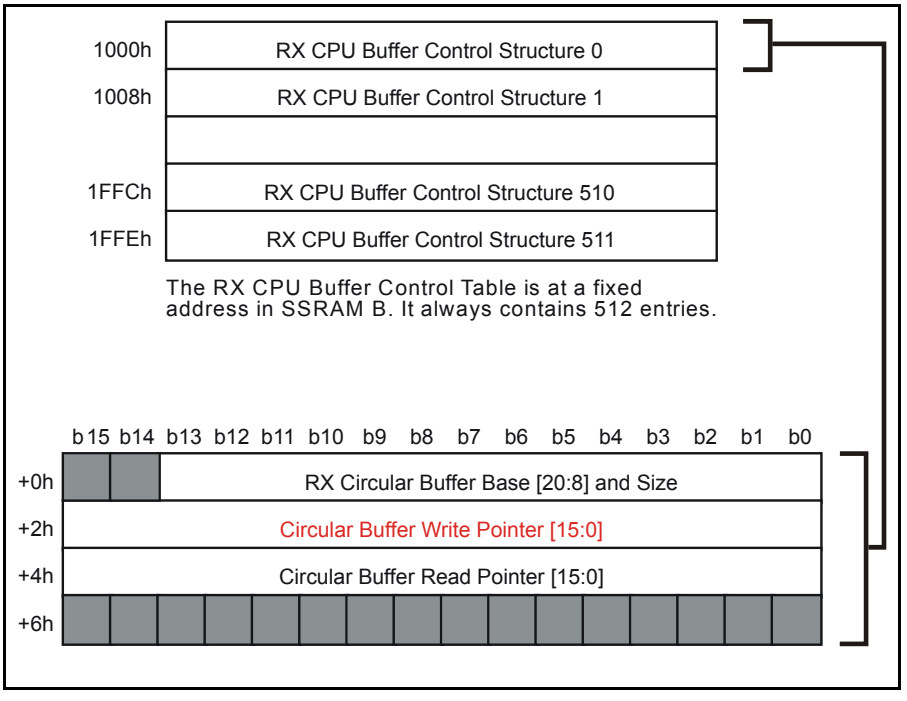


Figure 54 - Rx CPU Buffer Control Table

Add

Field	Description
RX Circular Buffer Base and Size	This field indicates the location and the size of the buffer that it used to store packets before they are read by the software. Supported size are between 256 bytes and 64K bytes in step of 2^k.
Circular Buffer Write Pointer	This pointer is used to remember the position of the next byte to be written in the circular buffer. It thus points to an invalid byte. The pointer is defined as a pointer to bytes. This field should be initialized to zero upon buffer creation.

Table 37 - Fields and Description

Field	Description
Circular Buffer Read Pointer	This pointer is used to remember the position of the next byte to be read in the circular buffer. It thus points to a valid byte. The pointer is defined as a pointer to bytes. When the read and write pointers are equal, the buffer is deemed empty. This field should be initialized to zero upon buffer creation. It is control by software and should be updated each time a packet is read from the circular buffer.

Table 37 - Fields and Description (continued)

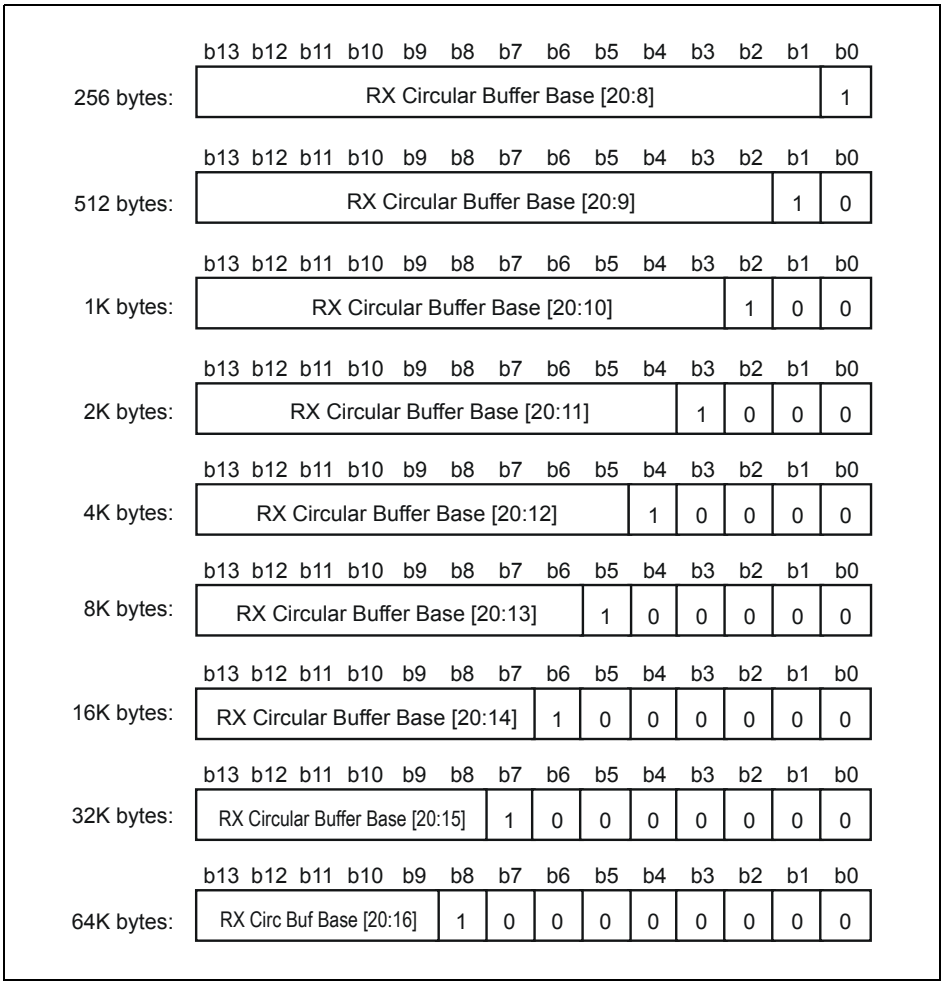


Figure 55 - RX Circular Buffer Base and Size

The format of the RX RTP CPU Channel Structure is the following:

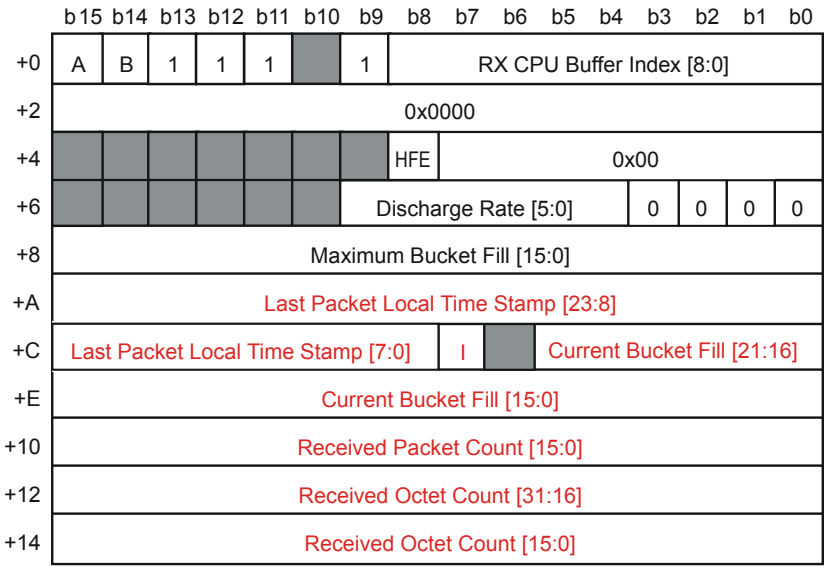


Figure 56 - RX RTP CPU Channel Structure

Field	Description
A	Clock Recovery channel A. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue A.
B	Clock Recovery channel B. When this field is set, an Adaptive Clock Recovery RTP Event Structure using this packet's sequence number and time stamp fields will be written to the Adaptive clock recovery queue B.
RX CPU Buffer Index	Index in the RX CPU Buffer Control Table in SSRAM B.
HFE	Half Full Interrupt Enable. When '1', if the RX CPU Buffer for this channel is more than half full, the interrupt_error_alarm will be forced active. This alarm should force the software to empty out packets in all RX CPU buffers.
Discharge Rate	Long term rate at which bytes can be received on this RX CPU channel. The current bucket fill is reduced at this rate. The rate is defined as a number of bytes per frame that can be sent into the RX CPU buffer. This is a floating point number; it has two bits of mantissa (excluding the hidden leading '1') and 4 bits of exponent. Zero is not a legal value. The minimum rate is 4 kbps, which is defined as mantissa = "100" and exponent = "0001". So to calculate the rate, the following equation can be used: 4 kbps * (mantissa/8) * (2^exponent).
Maximum Bucket Fill	Up to this many bytes can be contained in the "policing bucket" of this RX CPU channel before packet discarding takes place. Largest value is 0x0000 (which means 65536 bytes), smallest value is 0x1 (which means 1 byte).
CBF	Current Bucket Fill. Current fill of the "policing bucket" in 1/64th of a byte. This field should be initialized to all zeros by software.

Table 38 - Fields and Description

Field	Description
Last Packet Local Time Stamp:	Time stamp in frames of the last received packet used to determine the time delta between the reception of RX CPU packets. Used to discharge the Current Bucket Fill.
I	Initialized bit. Written to zero by software channel initialization. Written to ‘1’ by hardware as soon at the first packet is received on this channel.
Received Packet Count:	Total number of packets received on this channel so far. Every packet that gets to this structure will be included.
Received Octet Count	Total number of bytes received on this channel so far. Every packet that gets to this structure will be included. This field increments by the UDP Length - 8 each time packet is received.

Table 38 - Fields and Description (continued)

9.0 TX/RX TDM Data Paths

This chapter describes the data paths for all bytes transmitted and received with the H.110 interface.

9.1 TX TDM Data Path

The TX TDM section of the chip takes bytes from the H.110 interface and writes them into circular buffers in SSRAM A. To do so, the MT92210 uses the TX Channel Association Memory to decide which of the 1023 time slots on the H.110 bus it wants to treat. Each entry in the TX Channel Association Memory associates a time slot with either a PCM buffer or an HDLC stream; the chip can support up to 1023 PCM buffers or up to 512 HDLC streams, or any combination of the two (two PCM buffers cost the same as 1 HDLC stream). Any of the 1023 time slots supported can also be configured as Low Latency Loopback: this means that the time slot will simply be looped back onto another time slot on the H.110 bus.

This is the format of the TX Channel Association Memory:

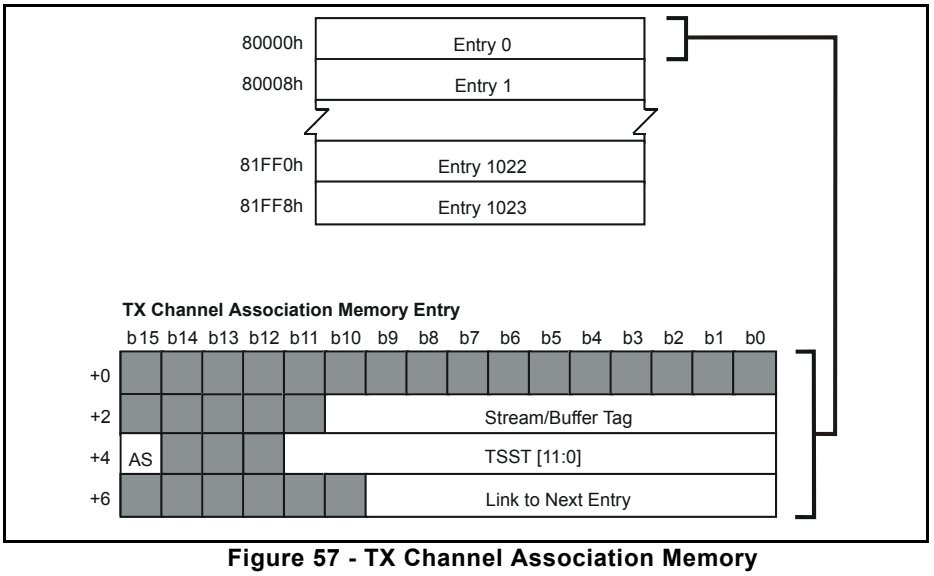


Figure 57 - TX Channel Association Memory

Field	Description
Stream/Buffer Tag	Encoded field that points either to the TX TDM Control memory (for xxPCM Buffer and HDLC stream) or to the Low-Latency Loopback Memory for Low-Latency Loopback channels.
AS	Associated Stream. When set, the least significant bit of the TSST number will be ignored and both even and odd streams pointed to by the TSST number will be read for valid data. For xxPCM circular buffers, the extra stream allows PCM / ADPCM codec changes to occur smoothly. For HDLC streams, the extra stream doubles the bandwidth that is available on the HDLC stream.
TSST	Time / Stream Number. TSST[11:5] represents the timeslot on which the data belonging to an HDLC stream or to an xxPCM buffer will be received from. TSST[4:0] represents the stream on which the data belonging to an HDLC stream or to an xxPCM buffer will be received from.
Link to next entry	Pointer to the next TX Channel Association Memory Entry. Entries must be sorted according to TSST number. Entry 0 is never considered to contain a Buffet Tag; it's TSST is hardcoded in the chip as number -1, which means that the last TX Channel Association Memory Entry must point back to Entry 0 to be ready for the next frame.

Table 39 - Fields and Description

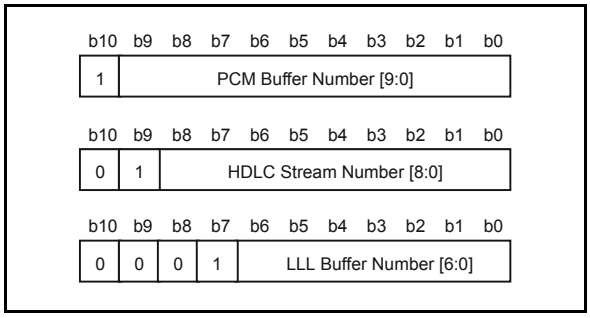


Figure 58 - Buffer Tag Format

The TX Channel Association Memory entry points to an entry in the TX TDM Control Memory that will define either a PCM buffer or an HDLC stream. If the entry defines a PCM buffer, it will define the compression type of the data to be received. The MT92210 can support TDM data in PCM A-law, PCM u-law, ADPCM-40, ADPCM-32, ADPCM-24 and ADPCM-16 formats. It can also perform auto-detection of the compression rate received using encoded values. Lastly, the TX TDM can perform law-translation on the incoming PCM values. The data can enter as either u-law or A-law and can also exit as either u-law or A-law, with any translation between the two.

If the TX TDM Control Memory entry defines an HDLC stream, then the entry will contain information used to perform HDLC de-framing. Many fields are used to keep byte-per-byte context. The entry also contains fields specifying the HDLC header format: the chip supports HDLC addresses of 0, 1 or 2 bytes, as well as 0 or 1 HDLC control bytes. Each HDLC packet may also be trailed by a 16-bit CRC, configurable per stream. The HDLC addresses are used to distinguish multiple channels on the same HDLC stream, with a maximum of 512 channels per stream (using the 9 low bits of a 2-byte address), or a maximum of 256 channels when using a single-byte address.

The TX Channel Association Memory also has an AS (Associated Stream) bit that allows greater bandwidth on HDLC streams. When this bit is '1', the TX Channel Association Memory binds 2 time slots to the corresponding TX TDM Control Memory entry instead of 1. This increases the total capacity of the TX Data Path in HDLC mode to 2046 time slots. In this mode, the 2 time slots that are bound together are 2 adjacent H.110 streams (i.e. **ct_d**[0] and **ct_d**[1], during the same time slot). The even stream contains the data that is logically first.

Each HDLC stream is associated to a circular buffer, of varying size depending on the maximum packet size expected on that stream and the bandwidth of the stream. The circular buffer sizes can vary between 512 bytes and 32K bytes, in increments of 2^n.

The format of the TX TDM Control Memory is the following:

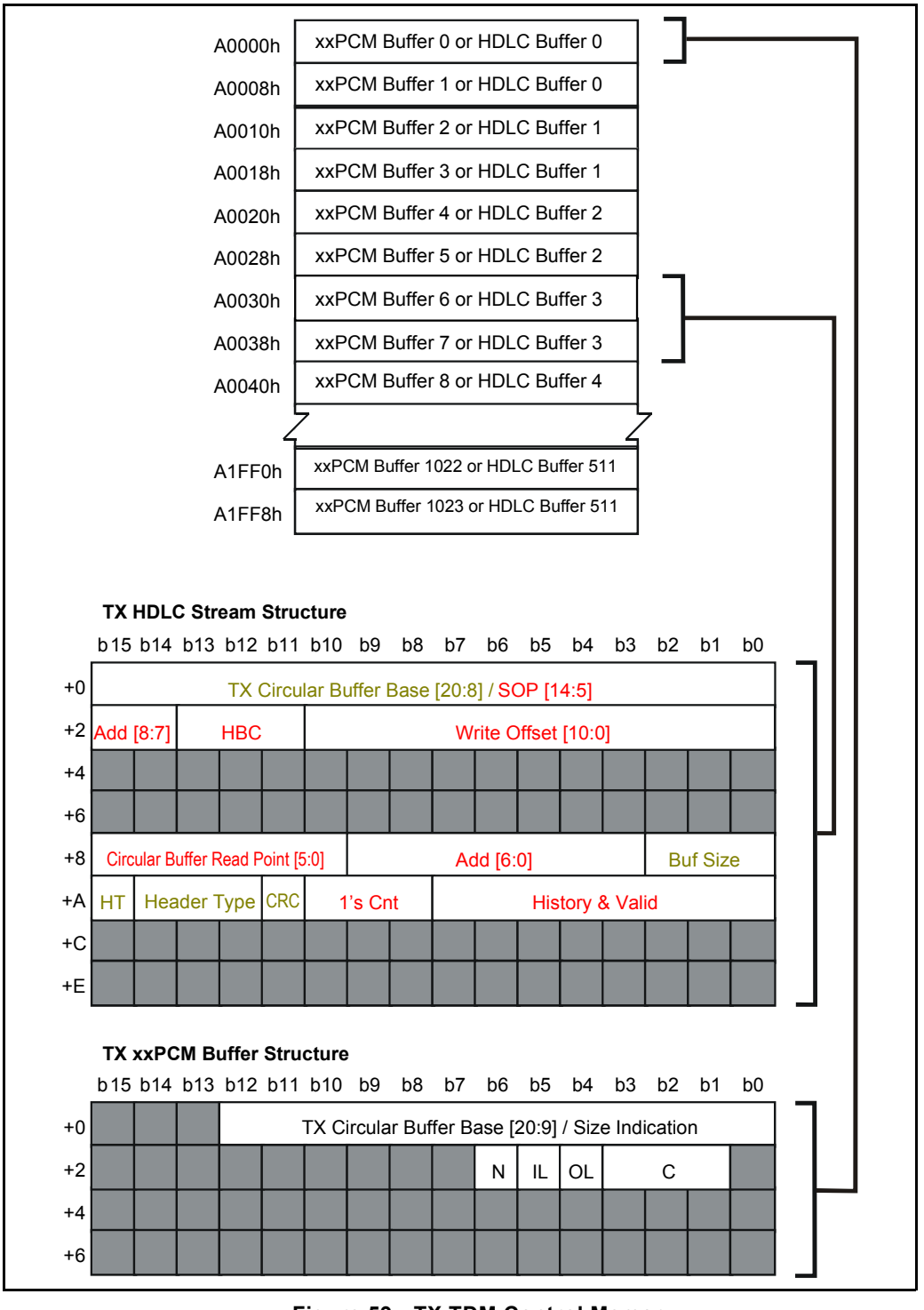


Figure 59 - TX TDM Control Memory

Field	Description
TX Circular Buffer Base / Size Indication	This field indicates the base address and size of the circular buffer in which values received from the TDM bus must be written. The size range is 256 bytes to 1K bytes.
N	Nibble mode. ‘0’ = ADPCM nibbles in high bits; ‘1’ = ADPCM nibbles in low bits.
IL	In law. Determines the PCM law of samples received from the TDM bus. ‘0’ = u-law; ‘1’ = A-law.
OL	Out law. Determines the PCM law of samples written to the TX Circular Buffer. ‘0’ = u-law; ‘1’ = A-law.
C	Compression type. “000” = PCM, “001” = ADPCM 40 kbps, “010” = ADPCM 32 kbps, “011” = ADPCM 24 kbps, “100” = ADPCM 16 kbps; “101” = ADPCM only autodetect (Associated Stream bit is cleared); “110” = ADPCM & PCM autodetect (Associated Stream bit is set); “111” = reserved.
TX Circular Buffer Base [20:8] / SOP[14:5]:	Base address of the TX Circular Buffer used for the HDLC stream. The lower order bits of this field represent the start of packet address of the packet that is currently being received on the HDLC stream. The SOP part of this field should be written to zero by software upon initialization of the HDLC stream.
Add	HDLC Address of the current packet. This field should be initialized to zero by software upon initialization of stream.
HBC	Header byte count. This field should be initialized to zero by software upon initialization of stream.
Write Offset	This field indicates where the next received byte will be written in the TX Circular Buffer. This offset is calculated using the SOP as its base. This field should be initialized to zero by software upon initialization of stream.
Circular Buffer Read Pointer	This field is updated by the TX assembly block when packets are sent out. This field should be initialized to zero by software upon initialization of stream.
Buf Size	Circular Buffer Size. “000” = 256 bytes; “001” = 512 bytes; “010” = 1K bytes; “011” = 2K bytes; “100” = 4K bytes; “101” = 8K bytes; “110” = 16K bytes; “111” = 32K bytes.
HT	HDLC Type. ‘0’ = bit wise packet framing and escape code; ‘1’ = byte wise packet framing and escape code.

Table 40 - Fields and Description

Field	Description
Header type	“000”=Plain; “001”=One byte address; “010”=Two byte address; “011”=One byte address + Control; “100”=Two byte address + Control; others = reserved.
CRC	CRC Present. When ‘1’, a CRC16 is expected at the end of each HDLC Packet that is received.
1’s Cnt	Counter of the number of consecutive ‘1’ that have been received recently on the HDLC stream. This field should be initialized to “111” by software upon initialization of stream.
History & Valid:	This field contains partially received bytes from the HDLC stream. The number of bits received but pending byte completion can be established by locating the first ‘1’ starting from the most significant bit of the field. This field should be initialized to “00000001” by software upon initialization of stream.

Table 40 - Fields and Description (continued)

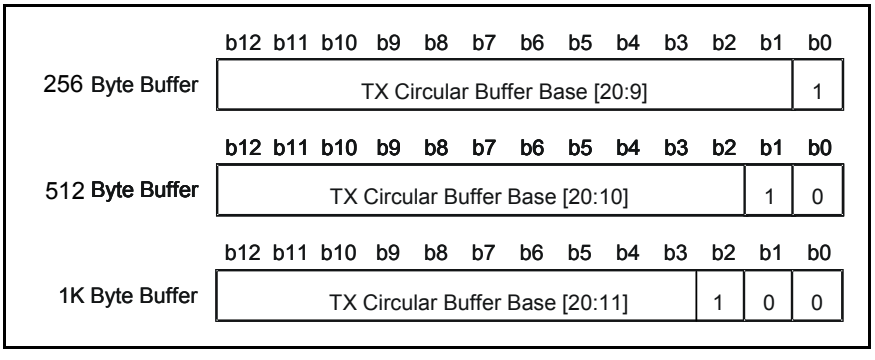


Figure 60 - TX Circular Buffer Base/Size Indication

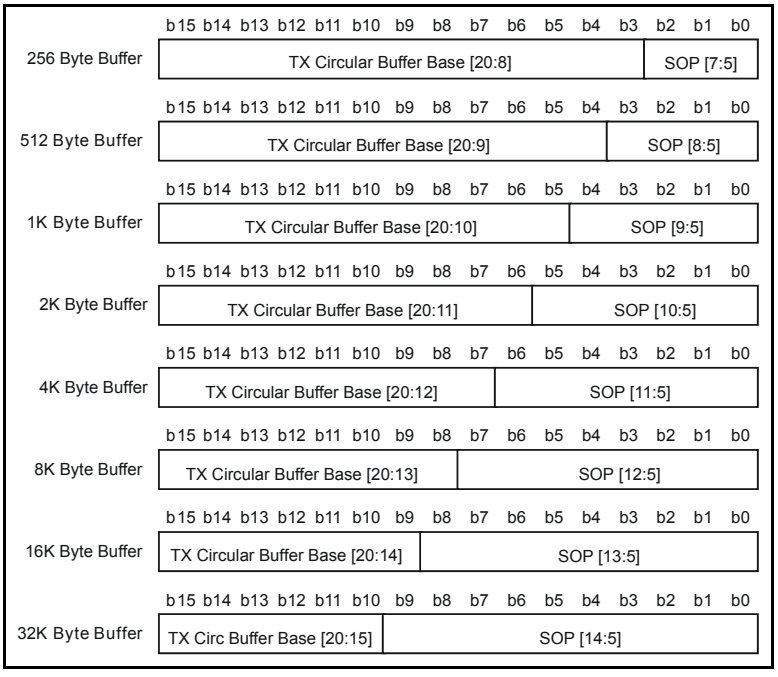


Figure 61 - TX Circular Buffer Base/SOP

When an HDLC packet completes, an event must be written to the Assembly Event Queue so the packet makes it to the network interface. Each HDLC Stream has an entry in the HDLC Stream to HDLC Address LUT Structure, which gives, for each HDLC Stream, the list of all HDLC addresses and their corresponding TX Connection Structures. For each Stream, the number of valid addresses is defined (anywhere from 16 to 512 in increments of 16) as well as a pointer to the corresponding HDLC Address LUT.

The format of the HDLC Stream to HDLC Address LUT Structure is the following:

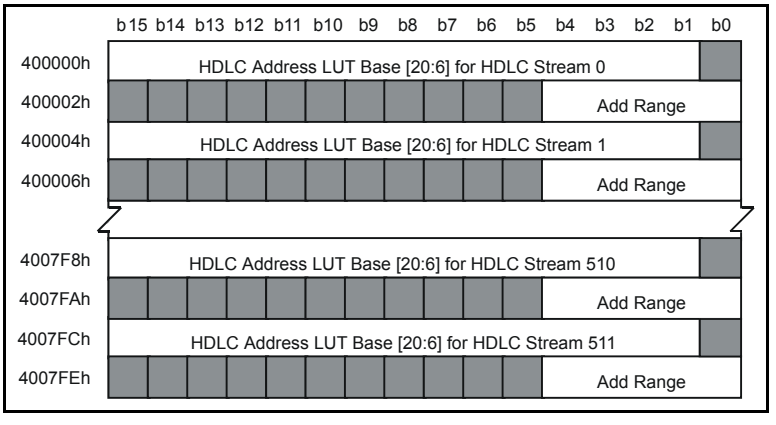


Figure 62 - HDLC Stream to HDLC Address LUT Structure

In RTP, the HDLC Address LUT Structure contains a pointer to a TX Connection Structure, as well as a TI (Timestamp Insert) bit. When this bit is ‘1’, this chip will insert its own timestamp, adding the one contained in the HDLC packet to the final timestamp. When ‘0’, the timestamp in the HDLC packet will be used as-is.

Field	Description
HDLC Address LUT Base for HDLC Stream	This field serves to locate in external SSRAM A a look-up table used to associated HDLC addresses with TX Connection Structures. It points to 64-byte increments.
Add Range	Defines the size of the HDLC Address LUT Structure. “00000” = address ranges between 0 and 511; “00001” = address ranges between 0 and 15; “00010” = address ranges between 0 and 31; “00011” = address ranges between 0 and 47;...;”11111” = address ranges between 0 and 495.
Note	Indexing in this structure is done implicitly using the HDLC stream number in the TX TDM Control Memory. This table is consulted every time an HDLC packet reception completed.

Table 41 - Fields and Description

The format of the RTP HDLC Address LUT Structure is the following:

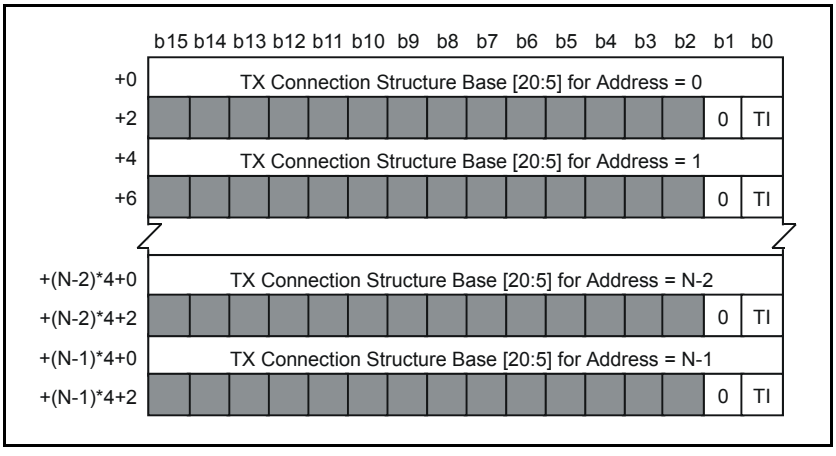


Figure 63 - HDLC Address LUT (RTP)

Field	Description
TX Connection Structure Base for Address	Base address of the TX Connection Structure that will be used to send HDLC packets with the address shown. When this field is 0000h, the address is deemed invalid and the packet is discarded.
TI	Time Stamp Insert. When ‘0’, the time stamp will be inserted in the packet as it is received in the HDLC packet. When ‘1’, the time stamp in the HDLC packet will be treated as an offset from the current bus time stamp.
N	Number of HDLC addresses supported as defined by Add Range in the HDLC Stream to HDLC Address LUT Structure.

Table 42 - Fields and Description

9.2 TX TDM Data Formats

When interfacing with external compression or silence suppression agents in xxPCM, the MT92210 uses special data formats that allow it to pass more than payload information over the H.110 bus. When receiving ADPCM data, the position of the highest ‘1’ in the data sample indicates the type of compression used: in this way, all ADPCM compression types can be encoded within a single payload byte. In addition, the 00h code is used to indicate a suppression decision. All of this information is coded within an 8-bit sample. The ADCPM sample may also be contained in the high bits of the byte: in this case, the decoding is reversed, the position of the lowest ‘1’ in the data sample indicates the compression type.

However, when the received data is ADPCM and silence suppression is being performed, the MT92210 is incapable of extracting the energy level of the silence from the ADPCM samples. Therefore, the magnitude of the original PCM value must be transmitted somehow to the MT92210. This is done on an associated stream, so a single sample of an xxPCM channel now occupies 2 H.110 TSSTs.

When the compression type can be PCM or ADPCM, then an extra indication is necessary to indicate this compression type. Because the magnitude of the original PCM value is 7 bits, there is a single free bit left on the associated stream: this bit will be used to indicate if the compression of the current data is PCM or ADPCM. The remaining encoding of the data is done in the same way.

In summary, single time-slot mode allows:

- PCM samples with internal VAD for silence suppression and energy level detection.
- ADPCM samples with compression auto-detection
- ADPCM samples with compression auto-detection and silence suppression but no energy level detection. The CN packet will have to be programmed through the CPU interface.

Dual time-slot mode allows:

- PCM and ADPCM samples with compression auto-detection
- ADPCM samples with compression auto-detection and silence suppression with energy level detection.
- PCM and ADPCM samples with compression auto-detection and silence suppression with energy level detection.

The format of the TX xxPCM TSSTs in single and dual time-slot mode is the following:

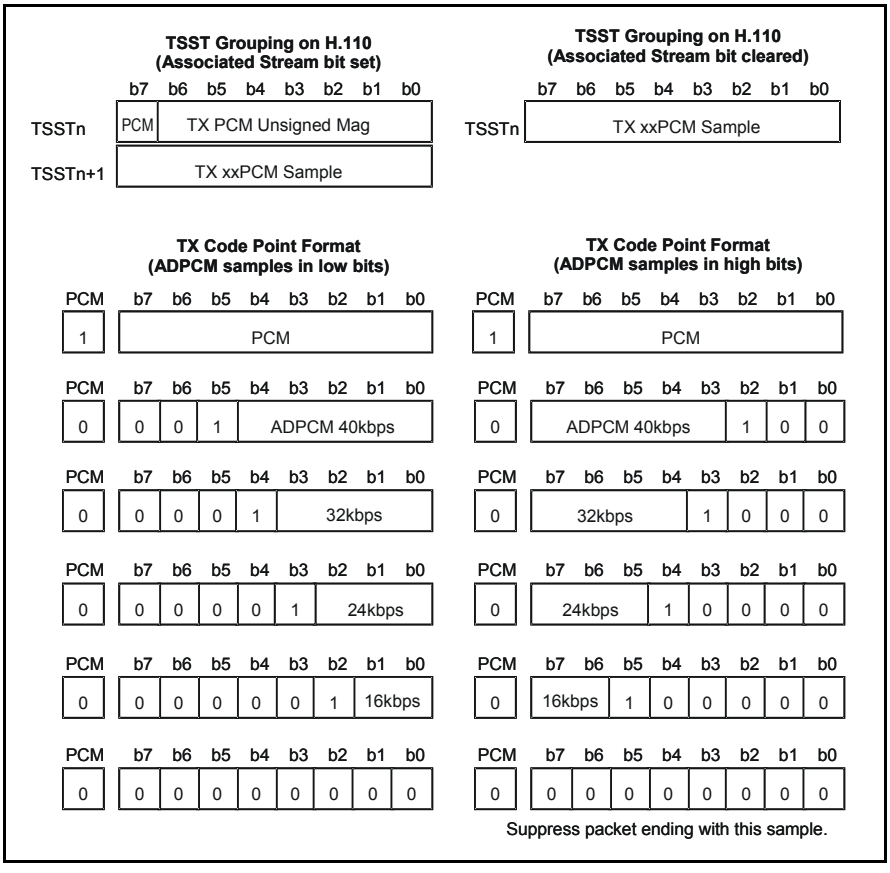


Figure 64 - Format of TX xxPCM TSSTs

9.3 RX TDM Data Path

The RX TDM module is responsible for copying data from the circular buffers contained in SSRAM B and passing them on to the H.110 interface. The RX TDM uses an RX Channel Association Memory to associate each of the 1023 time slots it supports to either a PCM buffer or an HDLC stream. The chip can support up to 1023 PCM buffers or up to 512 HDLC streams, or any combination of the two (two PCM buffers cost the same as 1 HDLC stream). Any of the 1023 time slots supported can also be configured as Low Latency Loopback: this means that the data sent onto the time slot will not come from a circular buffer but will be received from another time slot on the bus, treated by the TX TDM.

This is the format of the RX Channel Association Memory:

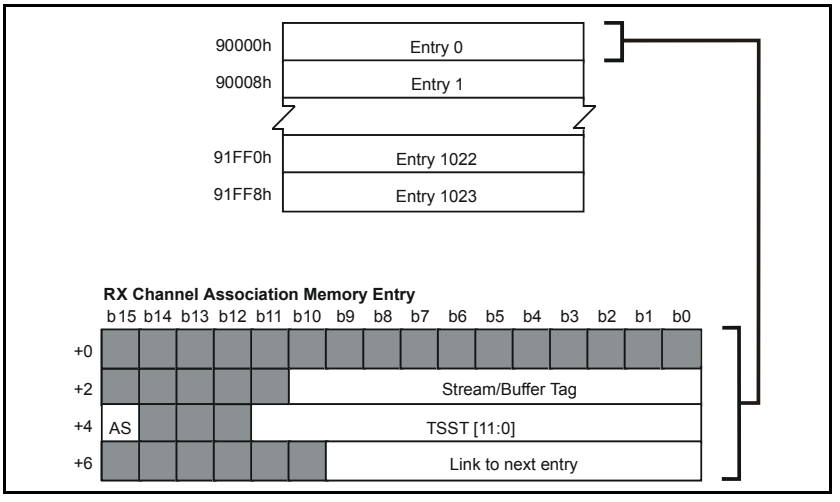


Figure 65 - RX Channel Association Memory

Field	Description
Stream/Buffer Tag	Encoded field that points either to the RX TDM Control memory (for xxPCM Buffer and HDLC stream) or to the Low-Latency Loopback Memory for Low-Latency Loopback channels.
AS	Associated Stream. When set, the least significant bit of the TSST number will be ignored and both even and odd streams pointed to by the TSST number will be driven out with valid data. For xxPCM circular buffers, the extra stream allows PCM / ADPCM codec changes to occur smoothly. For HDLC streams, the extra stream doubles the bandwidth that is available on the HDLC stream.
TSST	Time / Stream Number. TSST[11:5] represents the timeslot on which the data belonging to an HDLC stream or to an xxPCM buffer will be sent on. TSST[4:0] represents the stream on which the data belonging to an HDLC stream or to an xxPCM buffer will be sent on.
Link to next entry	Pointer to the next RX Channel Association Memory Entry. Entries must be sorted according to TSST number (incrementally). Entry 0 is never considered to contain a Buffet Tag; it's TSST is hardcoded in the chip as number -1, which means that the last RX Channel Association Memory Entry must point back to Entry 0 to be ready for the next frame.

Table 43 - Fields and Description

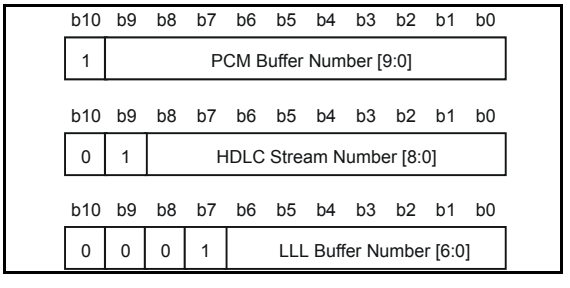


Figure 66 - Stream/Buffer Tag Format

Field	Description
PCM Buffer Number	Points to 8-byte PCM entry in RX TDM Control Structure
HDLC Stream Number	Points to 16-byte HDLC entry in RX TDM Control Structure (must be aligned on a 16-byte boundary)
LLL Buffer Number	Points to LLL 4-byte circular buffer

Table 44 - Fields and Description

When the RX Channel Association Memory points to an RX xxPCM Buffer Structure in the RX TDM Control Memory, the entry contains a pointer to the RX circular buffer used. In addition, law translation can be done in the RX direction: the data coming from the circular buffer can be u-law or A-law and it can exit on H.110 as u-law or A-law. The RX xxPCM Buffer Structure contains a Padding Type field that indicates where padding data should come from if no valid data is available in the circular buffer, in the case of underruns or packet loss. The Padding Type field can be updated by the packet disassembly module: when a CN packet is received, a new Padding Type value can be written in the circular buffer. When the RX TDM reads this value, it updates its Padding Type in the RX TDM Control Memory, then uses that value to pad from then on.

The RX xxPCM Buffer Structure also contains an Invalid Byte Counter that counts, in ms, how long it has been since a valid byte was received, with a maximum value of FFh (255 ms).

If the RX Channel Association Memory points to an HDLC Stream Buffer Structure in the RX TDM Control Memory, the information contained in that structure indicates the base address and size of the RX Circular Buffer associated to that HDLC stream, as well as the current read and write pointers to that circular buffer. If there is no data contained in the circular buffer, then an idle code will be sent onto the H.110 bus (either 7Eh in byte-framing or FFh in bit-framing). Otherwise, the next available byte in the circular buffer will be sent onto the bus. All HDLC framing has already been done by the packet disassembly module.

The RX Channel Association Memory also has an AS (Associated Stream) bit that allows greater bandwidth on HDLC streams; this bit is identical in function to the one in the TX direction. When this bit is ‘1’, the RX Channel Association Memory binds 2 time slots to the corresponding RX TDM Control Memory entry instead of 1. This increases the total capacity of the RX Data Path in HDLC mode to 2046 time slots. In this mode, the 2 time slots that are bound together are 2 adjacent H.110 streams (i.e. **ct_d**[0] and **ct_d**[1], during the same time slot). The even stream contains the data that is logically first.

The format of the RX TDM Control Memory is the following:

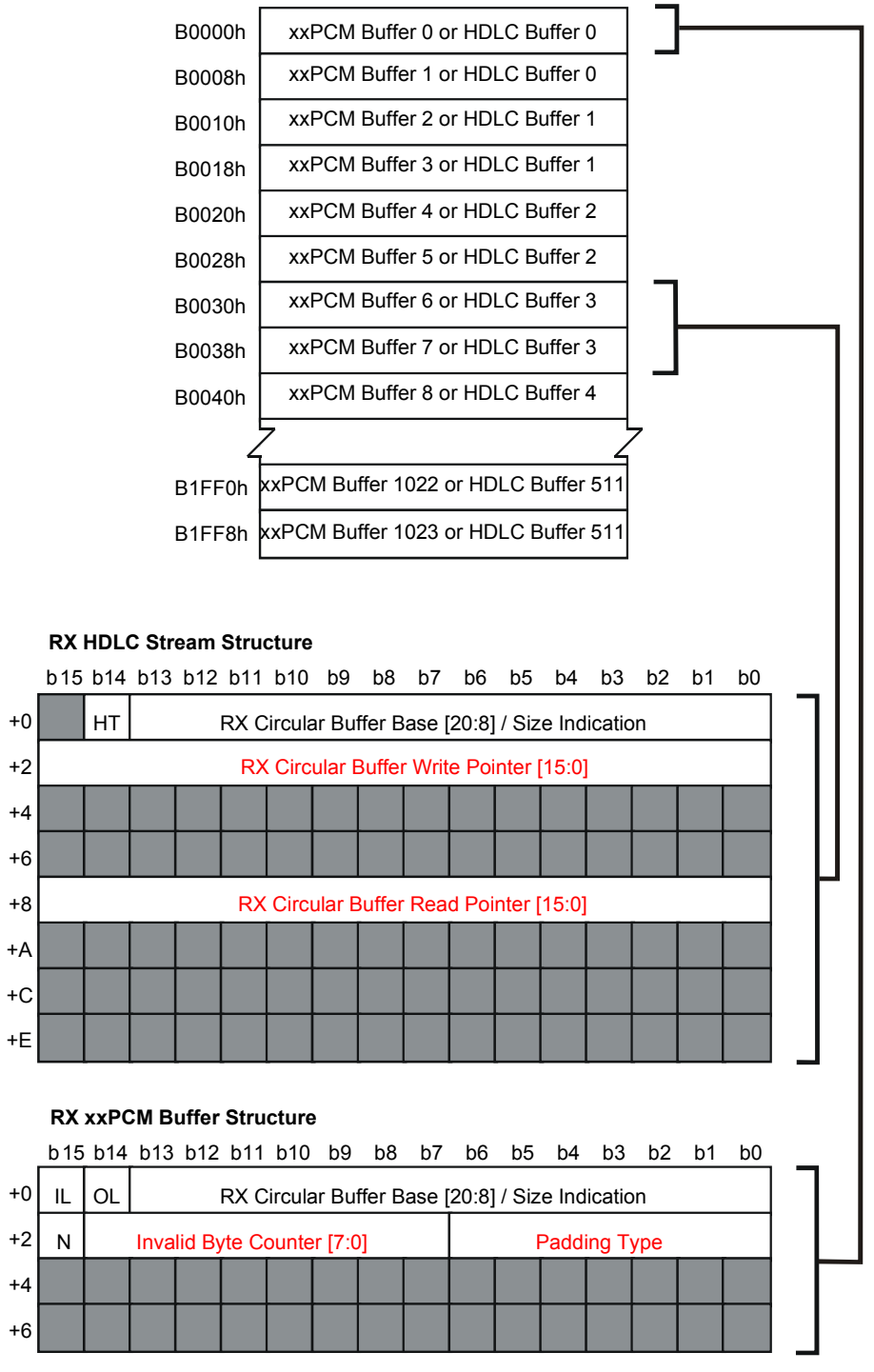


Figure 67 - RX TDM Control Memory

Field	Description
IL	In law. Determines the PCM law for samples read from the RX Circular Buffer. ‘0’ = u-law; ‘1’ = A-law.
OL	Out law. Determines the PCM law for samples written to the TDM bus. ‘0’ = u-law; ‘1’ = A-law.
RX Circular Buffer Base / Size Indication	This field indicates the size and base address of the circular buffer whose values must be output on the TDM bus. For xxPCM buffers, the size range is 256 bytes to 4K bytes. For HDLC streams, the size range is from 256 byte to 64K bytes.
N	Nibble mode. ‘0’ = ADPCM nibbles in high bits; ‘1’ = ADPCM nibbles in low bits.
Invalid Byte Counter:	Counter in ms during which no valid bytes have been received. This counter will be preset to FFh any time a CN packet is received. This counter is sticky at FFh.
Padding Type	This field indicates what to output on the bus when underruns occur. It can be updated dynamically through the reception of CN packet. 0-63 = SSRAM Tone Buffers; 64-95 = SDRAM Silence Buffers; 96-123 = single padding octets; 124-127 = free-running time stamp counter, with MSB selected with value 124. The free-running time stamp counter can be used to generate a time stamp for multiple chips on the same TDM bus to be time stamp synchronized.
HT	HDLC Type. ‘0’ = bit wise packet framing and escape code; ‘1’ = byte wise packet framing and escape code.
RX Circular Buffer Write Pointer	Copy of the write pointer contained in the RX HDLC Stream/Buffer Control Table. This pointer is used to determine how many bytes are valid in the circular buffer. RX HDLC Stream/Buffer Control Entries are associated with RX HDLC Stream Structures via direct implicit addressing (i.e. HDLC Stream # == RX HDLC Stream/Buffer Control Entry #).
RX Circular Buffer Read Pointer	Address at which the next byte to be sent out on the HDLC stream will be read from in the RX Circular Buffer. This field is written to the RX HDLC Stream/Buffer Control Entries each time it is modified.

Table 45 - Fields and Description

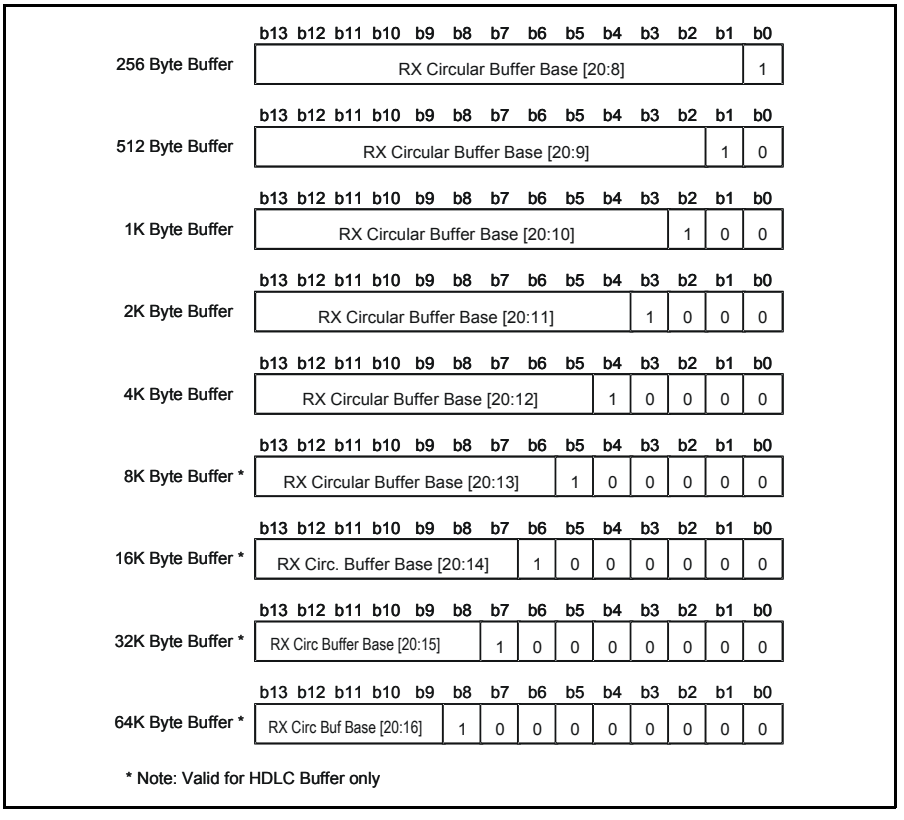


Figure 68 - RX Circular Buffer Base/Size Indication

Each RX HDLC stream structure in the RX TDM Control Memory corresponds to an entry in the RX HDLC Stream/Buffer Control Table. This entry contains almost the same information as the RX HDLC stream structure, but is updated first by the packet disassembly module. When the RX HDLC stream structure thinks its circular buffer is empty, it checks the write pointer in the RX HDLC Stream/Buffer Control entry to see if it has changed since last time. Similarly, when the RX TDM process completes a packet, it updates the value of the read pointer in the RX HDLC Stream/Buffer Control entry.

The format of the RX HDLC Stream/Buffer Control Table is the following:

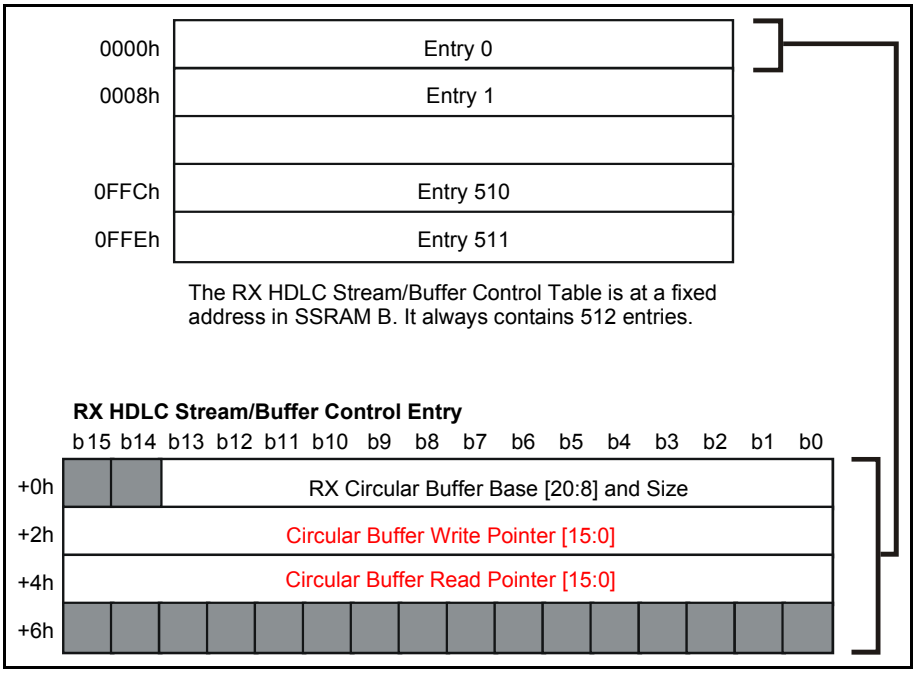


Figure 69 - RX HDLC Stream/Buffer Control Table

Field	Description
RX Circular Buffer Base and Size	This field indicates the location and the size of the buffer that it used to store packets prior to sending them on this HDLC stream. Supported size are between 256 bytes and 64K bytes in step of 2^k.
Circular Buffer Write Pointer	This pointer is used to remember the position of the next byte to be written in the circular buffer. It thus points to an invalid byte. The pointer is defined as a pointer to bytes. This field should be initialized to zero upon buffer creation.
Circular Buffer Read Pointer	This pointer is used to remember the position of the next byte to be read in the circular buffer. It thus points to a valid byte. The pointer is defined as a pointer to bytes. When the read and write pointers are equal, the buffer is deemed empty. This field should be initialized to zero upon buffer creation.
Note	Packets in the HDLC Circular Buffers have already been zero inserted/byte inserted.

Table 46 - Fields and Description

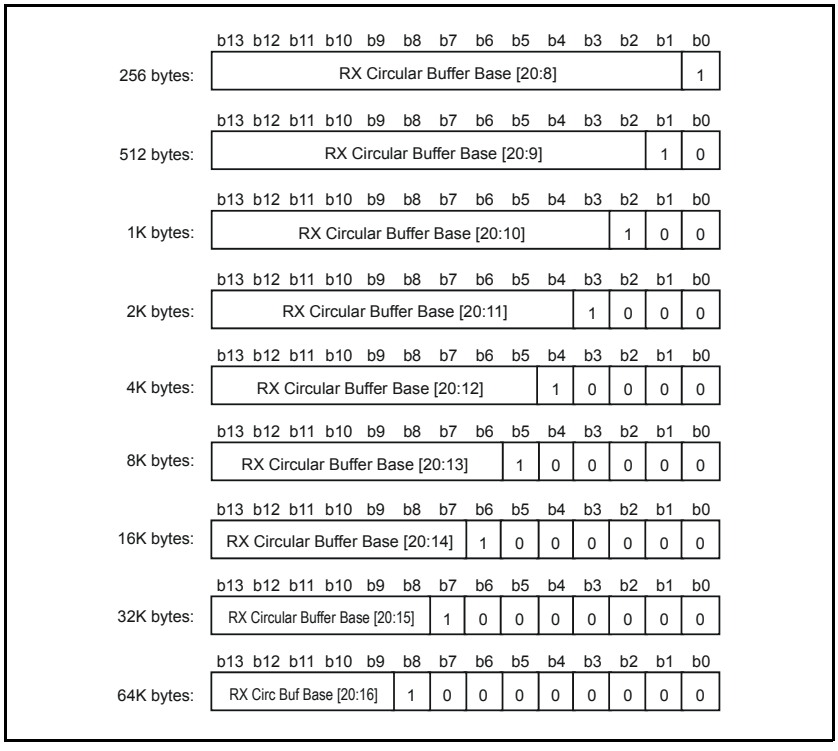


Figure 70 - RX Circular Buffer Base and Size

9.3.1 RX TDM Data Formats

When interfacing with external compression or silence suppression agents in xxPCM, the MT92210 uses special data formats that allow it to pass more than payload information over the H.110 bus. When transmitting ADPCM data, the position of the highest ‘1’ in the data sample indicates the type of compression used: in this way, all ADPCM compression types can be encoded within a single payload byte. In addition, one extra piece of information can be passed to an off-chip ADPCM decompression agent: an ADPCM reset flag. This flag is set on the first byte following a silence period and it indicates to the ADPCM decompression agent to reset the ADPCM transcoder to default values. This is a requirement after silence periods because no ADPCM data has been sent during the silence period, causing the compression and decompression agents to be out of sync. One solution to this problem is to reset both at the beginning of valid voice.

The implicit compression type as well as the ADPCM reset flag can be coded within an 8-bit sample. The ADCPM sample may also be contained in the high bits of the byte: in this case, the decoding is reversed, the position of the lowest ‘1’ in the data sample indicates the compression type.

An extra time slot is needed to encode PCM values along with the ADPCM values. If the compression rate can change to PCM as well as ADPCM, then 512 new values appear: 256 PCM values, and 256 PCM values with a reset flag. Despite the data being PCM, the ADPCM decompression agent may use it to synchronize its CODEC during the period of time where the data is PCM, so that if the decompression rate reverts to ADPCM, its context is valid. Therefore 2 extra bits on an associated stream are used.

The RX TDM is also capable of passing spectral CN packets on the H.110 bus. The encoding of these packets is done by passing 80h on the bus as a header, followed by the packet and trailed by a 16-bit CRC for data integrity purposes. When spectral CN packets are used, no padding is inserted by the chip: invalid bytes are indicated by a 00h code on the bus.

The format of the RX xxPCM TSSTs is the following:

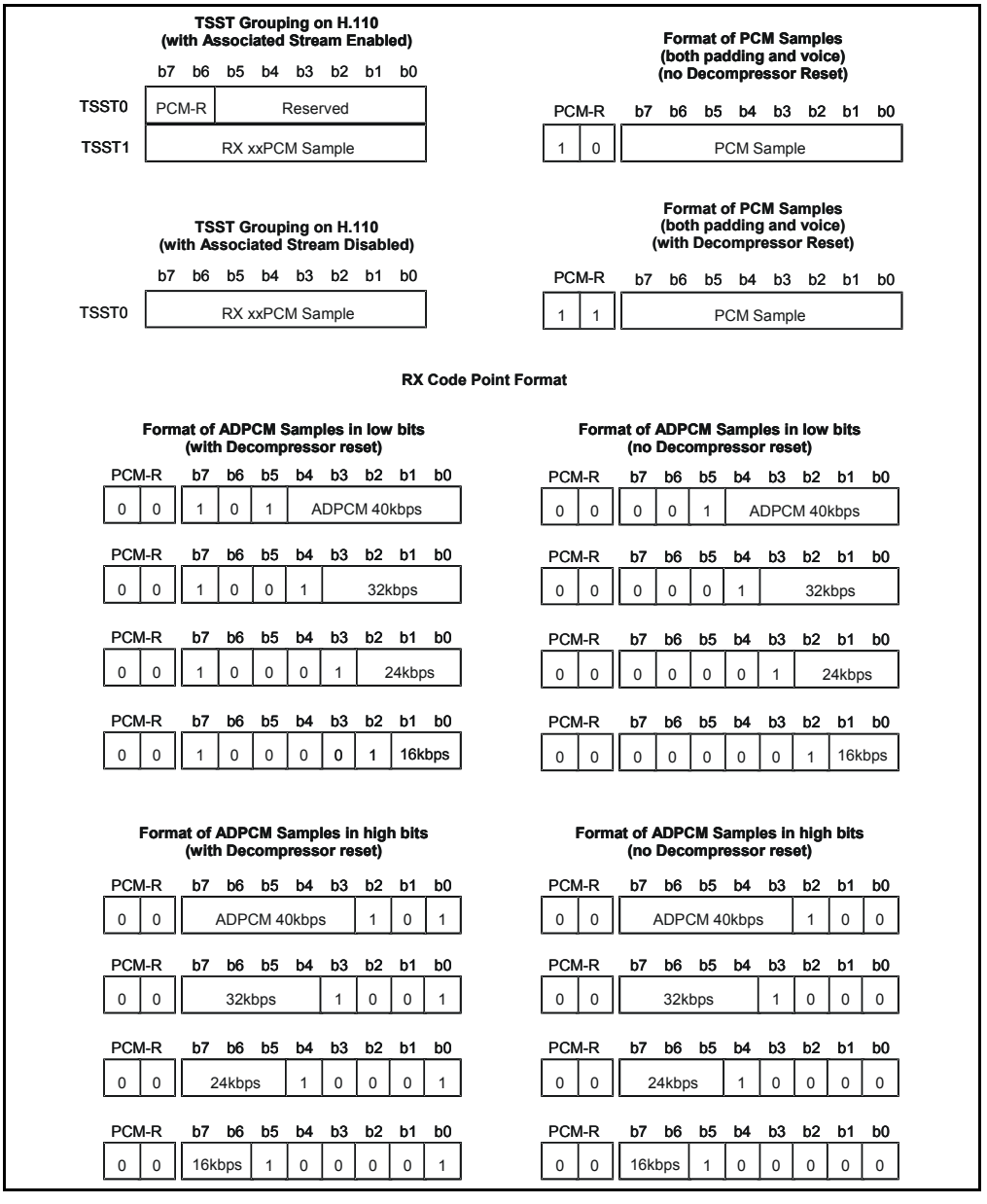


Figure 71 - Format of RX xxPCM TSSTs - 1 of 2

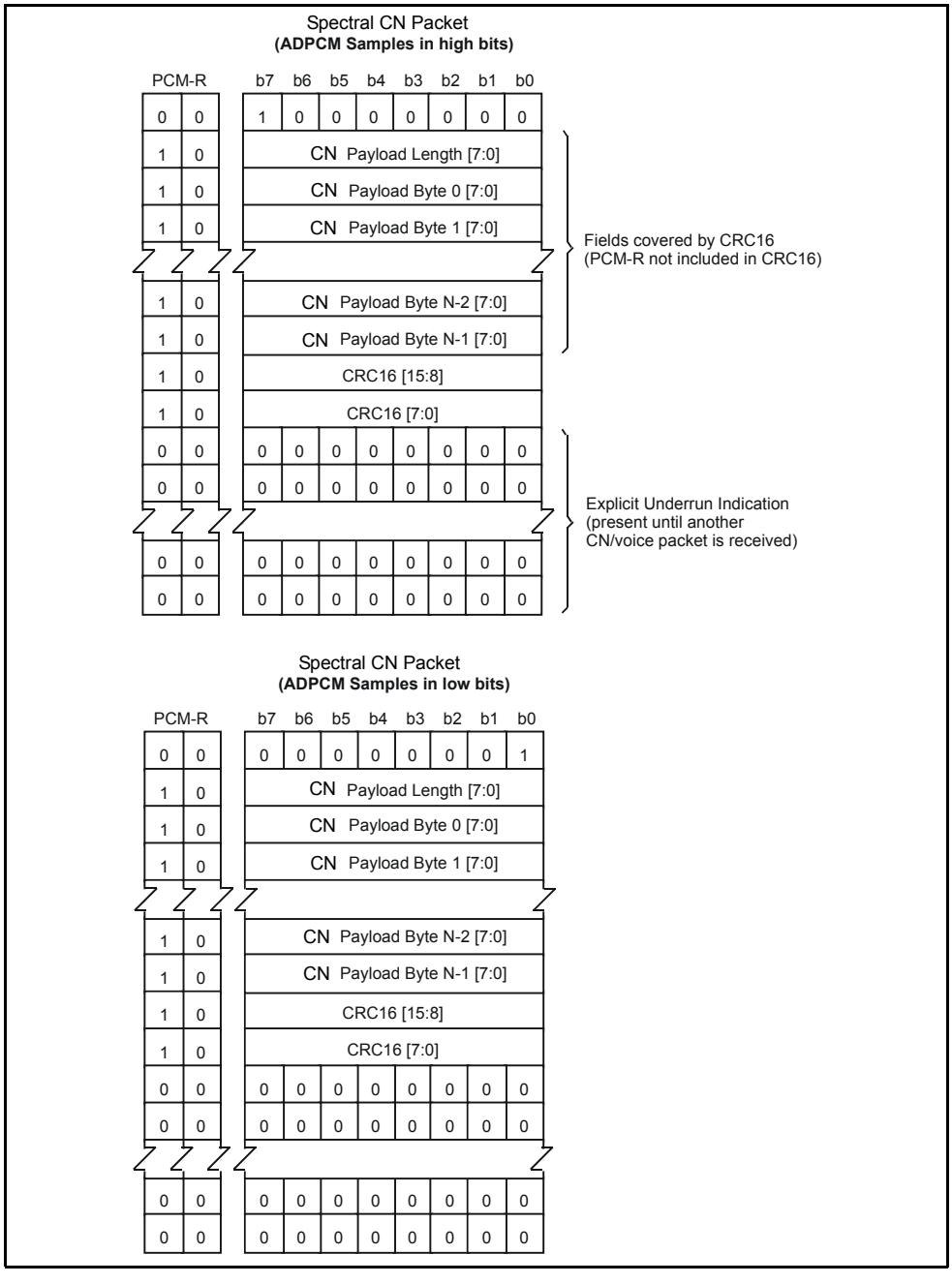


Figure 72 - Format of RX xxPCM TSSTs - 2 of 2

Two lookup tables are used to determine padding bytes during a silence period based on the received CN packet.

The following CN Lookup Table is located at fixed address in SSRAM B:

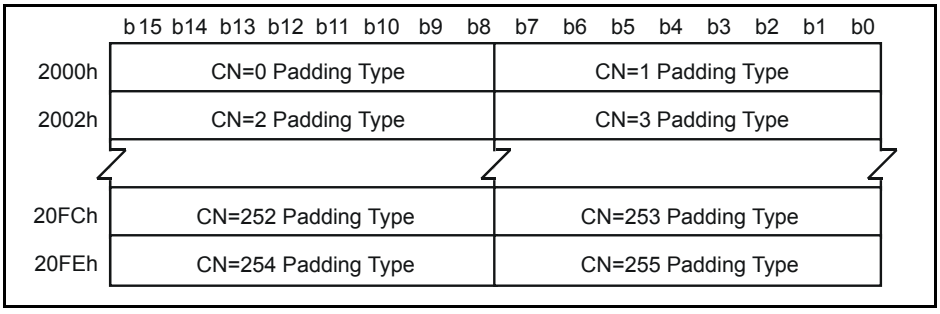


Figure 73 - CN Packet Conversion Lookup Table

Field	Description
Padding Type	This field indicates the padding type that should be placed on the TDM bus during this silence period. The first byte of the CN packet is used as an index into this table. 128-191 = SSRAM Tone Buffers; 192-223 = SDRAM Silence Buffers; 224-251 = Padding Octets; 252-254 = Reserved; 255 = Ignore CN Packet.

Table 47 - Fields and Description

10.0 H.110 Interface

The TDM interface on the MT92210 device is fully compatible with the H.110 bus and can be used to interface either as bus master or as bus slave. It respects all of the major requirements of H.110, such as supporting up to 32 TDM streams running at 8 MHz (up to 4096 time slots), the possibility of running 16 of the streams at lower frequencies (2 or 4 MHz), and the capability of passing 128 of the channels on H.110 in loopback.

The slave portion of the H.110 interface respects the timing requirements of this interface. It can sample the incoming data from the **ct_d** pins 90 ns after the rising edge of the clock as per the spec (3/4 sampling), and it can also sample on the falling edge (2/4 sampling) or rising edge of the clock (4/4 sampling). When driving its data, it can tri-state its pins early (between 20 and 0 ns before the rising edge of the clock) or it can tri-state synchronously on the rising edge of the clock. Both of these options allow flexibility in interoperation with other devices that are not fully H.110 compliant.

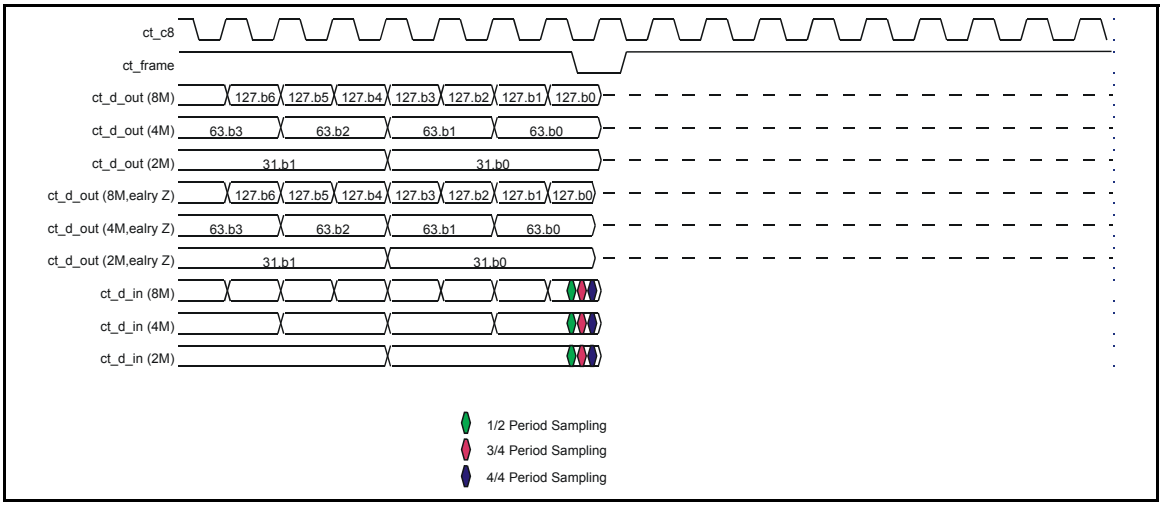


Figure 74 - TDM Bus Timing - ct_d

10.1 Slave Mode

When operating as a slave, the interface has the choice between clocking on A, clocking on B, clocking on A with B as backup or clocking on B with A as backup. When set to perform automatic switchover, the interface monitors the current bus master to see if its **ct_c8** and **ct_frame** signals are still valid. The **ct_c8** signal is checked to see if its clock edges are within ± 35 ns of where they are supposed to be (122 ns apart). The **ct_frame** signal is checked to make sure that it occurs exactly once every 1024 **ct_c8** clock cycles. If either of these two errors is reported about a given pair of bus master signals, the pair is considered invalid and the slave will switch to the backup master if any has been programmed to do so. The MT92210 will always monitor these signals and report errors on either of the two bus masters, even if it does not act on these errors.

10.2 Bus Master Mode

When acting as a bus master, the MT92210 can choose to be a bus master on A, master on B, backup on A or backup on B. When acting as a bus backup, the MT92210 uses the same error signals described above to determine if the current bus master is still valid or if it should take over the bus. Note that the bus mastership can be overridden in registers by ensuring that the chip cannot drive the H.110 clock and frame signals: this will ensure that it remains a passive slave on the bus. If the chip is a backup on the bus and the primary master fails, it will stop synchronizing itself on the master and track the local reference.

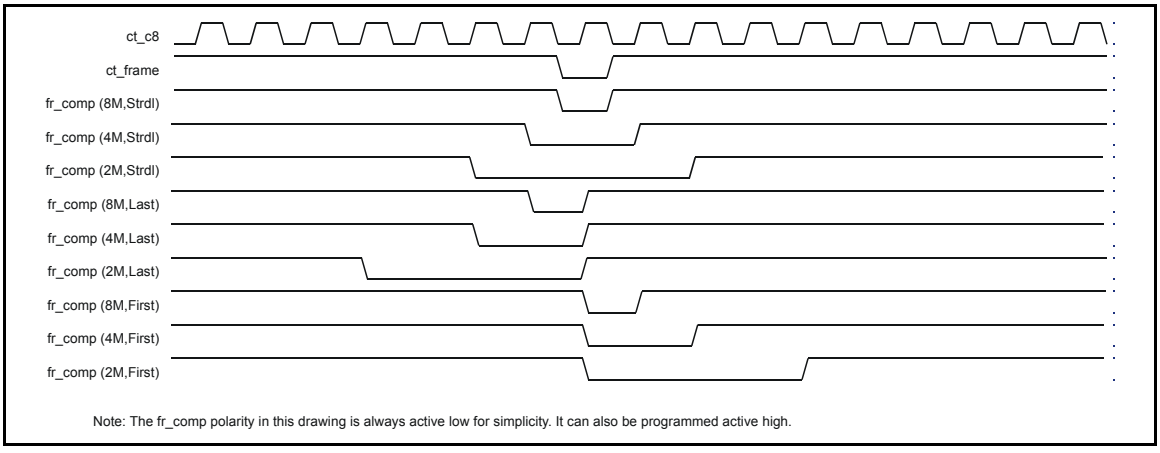


Figure 75 - TDM Bus Timing - fr_comp Generation

The chip can also control all the compatibility clocks that must be generated on H.110. There are a good number of these signals and their generation is independent of the mastership on either A or B: the chip can choose to generate all of these, or not, whether or not it is bus master or backup. Because these compatibility signals are, by definition, used to meet the specific requirements of an older bus standard, their generation is not programmable for the most part.

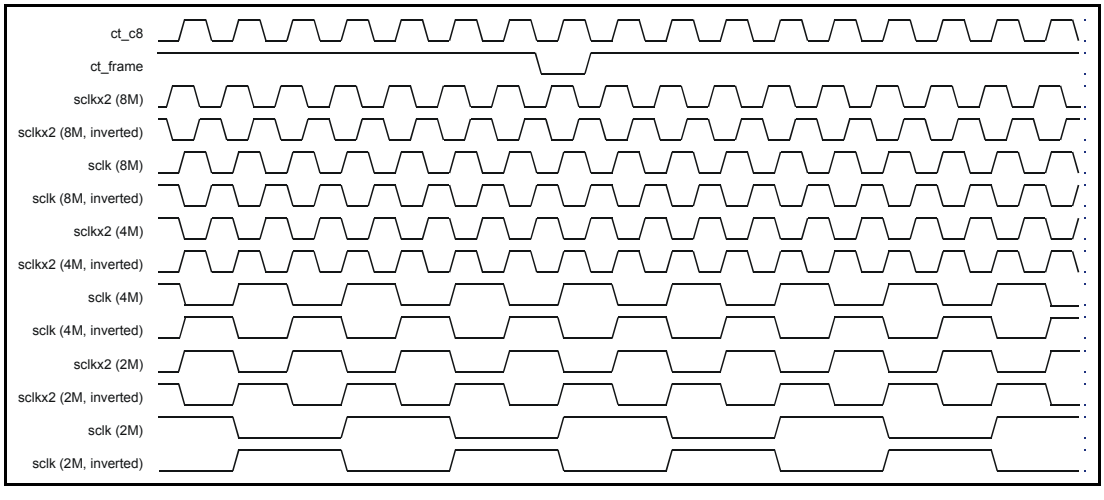


Figure 76 - TDM Bus Timing - sclkx2 Generation

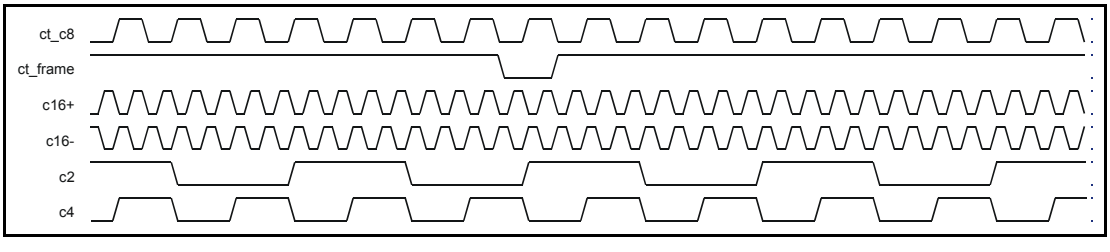


Figure 77 - TDM Bus Timing - Compatibility Clock Generation (other than sclk, sclkx2)

10.3 Polarities

There are a few exceptions, however. The **sclk** and **sclkx2** signals can be programmed to have their polarities identical to those defined in the H.110 specification, or they can have the opposite polarity. In some cases their rising edges will coincide with the rising edge of **ct_c8**, and in some cases will coincide with falling edges. In addition, the frequency of **sclk** can be programmed to 2, 4 or 8 MHz, with **sclkx2** being generated in function of this (note that when **sclk** is 8 MHz, **sclkx2** is also 8 MHz but off-shifted by a quarter-period. For more details on expected polarity, see H.110 specification). The **fr_comp** signal can also be generated active-high or active-low, can be generated as coinciding with a clock edge or as straddling it, and can be generated as relating to a clock at 2 MHz, 4 MHz or 8 MHz.

As mentioned earlier, H.110 requires that the lower 16 streams on the bus be capable of running at 2 or 4 MHz, with a single clock frequency determined for each group of 4 streams. The MT92210 allows every group of 4 streams on the bus to operate at 2, 4 or 8 MHz, allowing all 32 streams to be used in backwards compatibility with another non-H.110 agent.

In addition, the MT92210, instead of supporting the full bandwidth of H.110, can be configured to only interface with 4, 8 or 16 of the streams on the bus. This allows the chip to be run at a much lower frequency than 60 MHz.

11.0 Clocking

The MT92210 has 2 main clock domains: **mem_clk_net** and **mem_clk_sar**. They are used for the Network Interface and the SAR portion of the device accordingly. These clocks can be sourced externally or created internally using PLLs.

11.1 Programming the mem_clk_xxx PLL

The clock multiplication PLLs in the MT92210 are used to take a low frequency **upclk**, always provided on the **upclk** pin, and multiply it up to the **fast_clk** frequency. There, it is divided down to the **mem_clk_xxx** frequency.

The X, Y and Z divider can be programmed to any legal value through registers 164h or 166h (as defined in the tables below). **upclk** can be fed into the device with a frequency ranging from 25 MHz to 66 MHz. Only frequencies between ***50 MHz and 53.3 MHz*** are not supported by the PLL. The X and Y divisor tables indicate what values to program in the X,Y and Z registers depending on the input value of **upclk**. The Z divisor table indicates the range of **mem_clk_xxx** that can be achieved with typical values of Z. Note that **mem_clk_xxx** cannot be programmed above 100 MHz or below 40 MHz. **fast_clk** is used by the TDM interface and must operate between 160 MHz to 200 MHz.

The **mem_clk_xxx** PLL drives the output **mem_clk_xxx** pins. These pins provide both TTL and PECL interfaces for **mem_clk_xxx** input and output. For both types, the output pins for **mem_clk_xxx** is always driven. However, when the output pins are not being used, the register bits that control the toggling of these 2 pins should be disabled to reduce power consumption.

The user must configure the MT92210 to select the desired **mem_clk_xxx** input type, either PECL or TTL. **mem_clk_xxx** serves as the main clock for the chip and therefore must be present for the MT92210 to function. It is absolutely necessary for **mem_clk_xxx** to be present and one of the inputs to be selected. The **mem_clk_xxx** outputs however, are simply a convenience for the user and do not require to be connected. These outputs eliminate the need for a second, high-speed oscillator. The user need only generate **upclk**.

The clock that is connected to the **mem_clk_xxx** inputs on the MT92210, whether it is TTL or PECL, must be in phase with the clock connected to SSRAM used with the chip. The maximum skew allowed is ± 0.5 ns.

Div X	Div Y	upclk (MHz)	fast_clk (MHz)
-	-	0 to 30	-
1	6	30 to 33.33	160 to 200
1	5	33.33 to 40	166.66 to 200
2	8	40 to 50	160 to 200
-	-	50 to 53.33	-
2	6	53.33 to 66.66	160 to 200

Table 48 - Clock Divisor X and Y

Div Z	Mem_clk_xxx
2	80 to 100
3	53.3 to 66.6
4	40 to 50
5*	32 to 40
6*	26.6 to 33.3
7*	22.8 to 28.6
8*	20 to 25
9*	17.8 to 22.2
10*	16 to 20
12*	13.3 to 16.6
14*	11.4 to 14.3
16*	10 to 12.5
* These values of Z should not be used.	

Table 49 - Clock Divisor Z

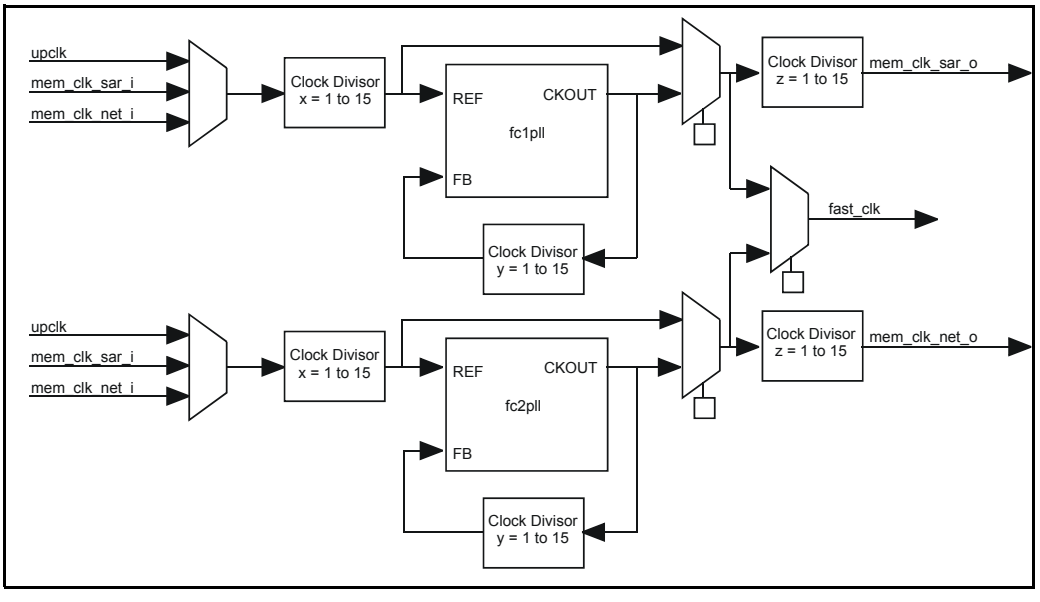


Figure 78 - Clock Synthesis

11.2 Clock Recovery

Since the MT92210 interfaces with a TDM bus, it is necessary that it incorporate a clock recovery circuit that will allow it to generate a precise TDM clock. As such, the MT92210 contains data gathering hardware to accumulate information concerning timing VCs or channels, as well as clock generation mechanisms.

The first portion of the clock recovery is a point gathering system that feeds software with all the information it needs to obtain a very precise measurement of the clock it needs to generate. The module receives pulses obtained either from the UTOPIA module (timing reference VCs) or from the RTP RX SAR (timing reference packets). These pulses can then be filtered, so that only a single pulse out of N is actually kept and treated by the hardware: this artificially “slows down” the timing reference signal by a factor of N.

In parallel, free running counters of **mem_clk** and of the **adap_ref** signal are kept, giving a very good idea of the time at which each pulse is received. When a pulse is kept by the filtering algorithm, a clock recovery event structure is written to external memory, with the 32-bit **mem_clk** counter, the 32-bit **adap_ref** counter and a 16-bit fraction of the **adap_ref** counter, as well as the LI and UUI of the received mini-packet. These event structures are written to a buffer in external memory, and the clock recovery module will generate an interrupt when the buffer is more than half full. Because clock recovery events arrive at a fixed rate, the size of the buffer chosen will completely determine the rate at which it needs to be serviced.

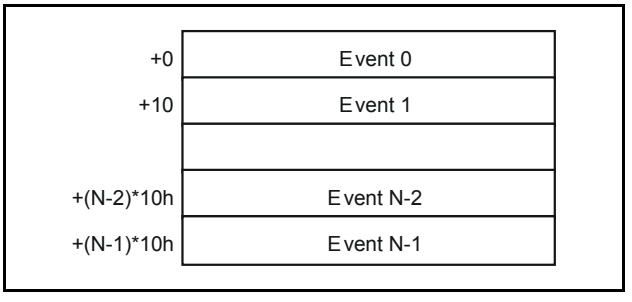


Figure 79 - Adaptive Clock Recovery Event Queue

The Adaptive Clock Recovery Event Queue is a circular buffer in SSRAM A used to report packet reception events to the host processor. It can be programmed to sizes of 4K bytes to 128K bytes in steps of 2^k. It must be mapped on a base address of its size. The position and size of this queue can be programmed at register address 0xB20 and 0xB28, for queue A and B respectively.

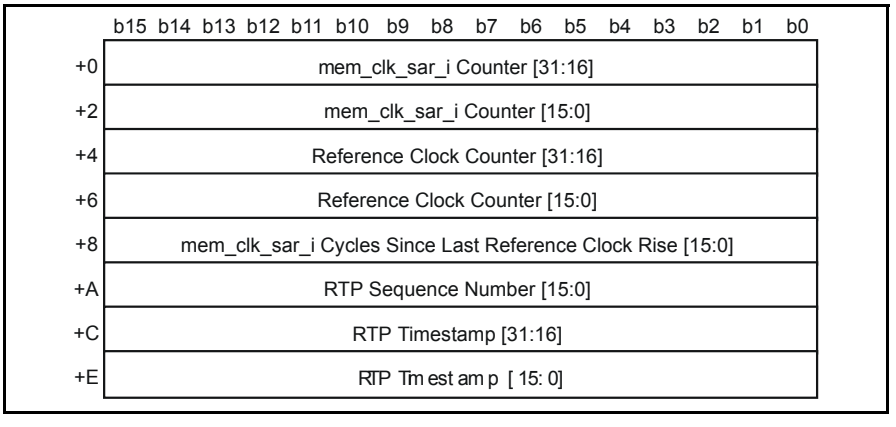


Figure 80 - Adaptive Clock Recovery RTP Event Structure

Field	Description
mem_clk_sar_i Counter	When an Adaptive Clock Recovery RTP Event Structure is generated, the mem_clk_sar_i free-running counter (the same one as in registers 908h and 90Ah) is sampled and written in this structure.
Reference Clock Counter	When an Adaptive Clock Recovery RTP Event Structure is generated, the free-running counter that counts the rising edges of the clock generator A module will be sampled and written in this structure.
mem_clk_sar_i Cycles Since Last Reference Clock Rise	When the Reference Clock Counter increments infrequently (i.e. the generate clock is low in frequency) this field can be used to approximate the fraction of a Reference Clock Cycle that can be appended to the Reference Clock Counter. This field in defined as the number of mem_clk_sar_i cycles that elapsed since the last increment of the Reference Clock Counter.
RTP Sequence Number	Sequence number of the packet that caused this Adaptive Clock Recovery Event Structure to be written.
RTP Timestamp	Timestamp of the packet that caused this Adaptive Clock Recovery Event Structure to be written.

Table 50 - Fields and Description

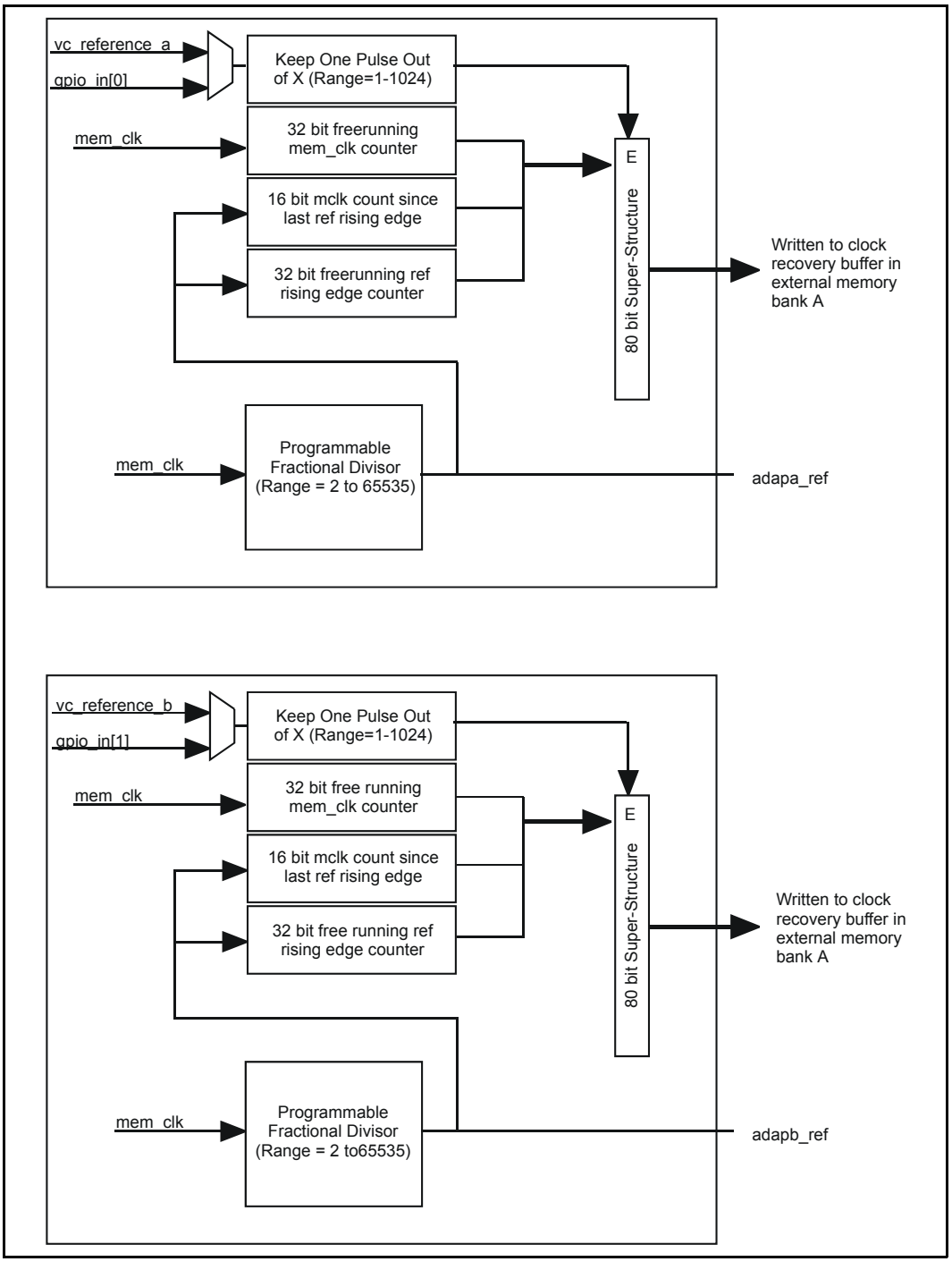


Figure 81 - Adaptive Clock Recovery Modules

The second portion of the clock recovery circuit is concerned with actually generating the **adap_ref** signal. When software reads the information placed in the buffer, it can calculate what is the rate of the clock that needs to be generated. The generated clock is always **mem_clk** divided by a 16-bit integer and 16-bit fraction, which allows a very high precision on the division (with **mem_clk** running at 50 MHz and a target clock speed of 8 kHz, it gives a precision of 0.4 ppb). Software can program the integer and fraction by which it desires **mem_clk** to be divided by, and the division will be performed in hardware. The **adap_ref** signal can then be output onto any of a number of GPIOs on the chip, or to one of the **ct_netref**s: among other uses, it can be routed to an external PLL used to multiply it up from 8 kHz to 16 MHz and then rerouted into the chip on the **pll_clk** pin.

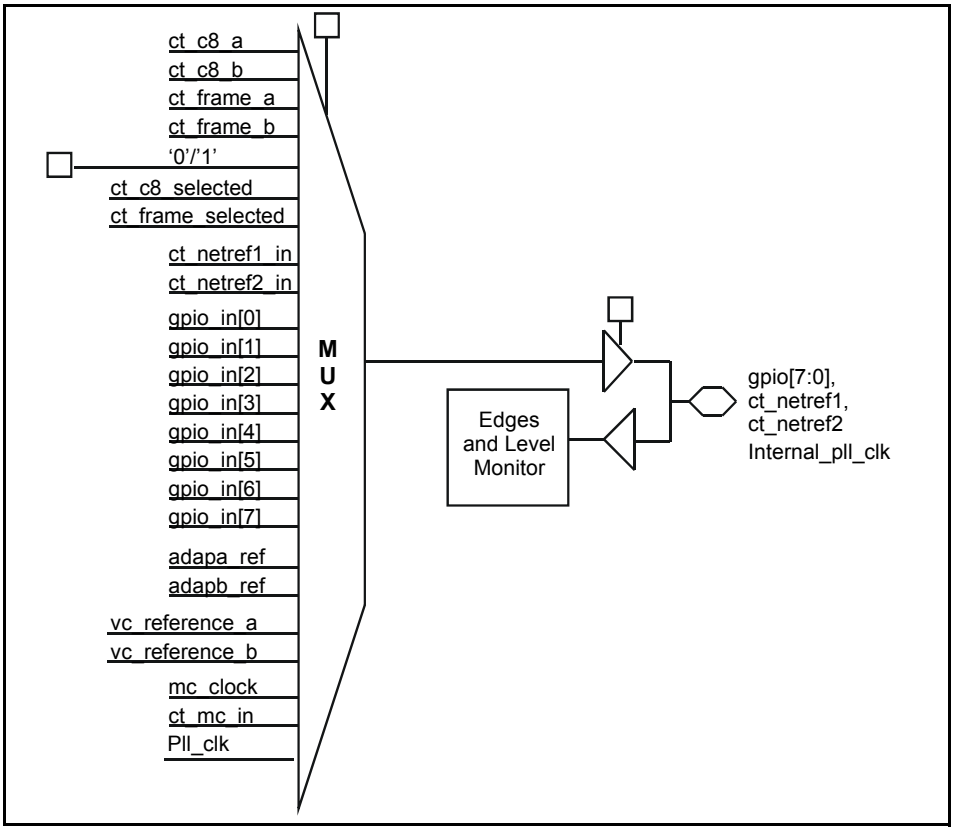


Figure 82 - GPIO Functionality

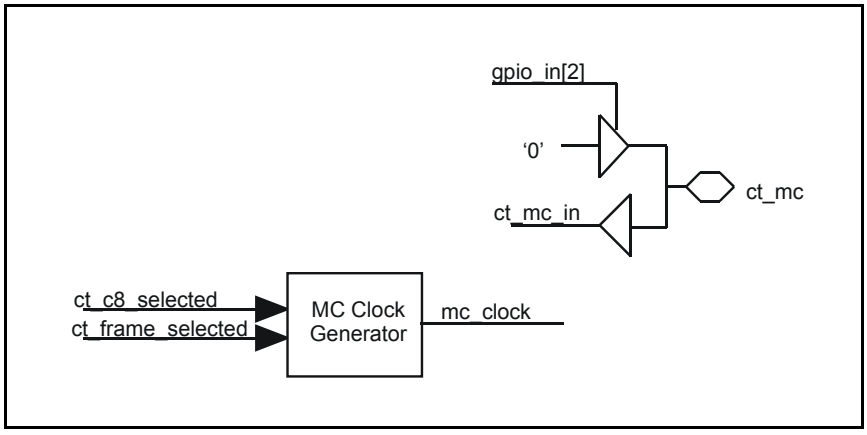


Figure 83 - Message Channel Circuit

11.3 Memory Controllers

The MT92210 uses 3 separate memory banks, each with its own memory controller. Memory bank A is used for structures in the TX direction. Memory bank B is used for structures in the RX direction. Memory bank C is used by the device's network interface.

Each memory bank, A, B or C, can connect up to 4 external SSRAM chips, each ranging in size from 128K to 1M bytes. However, the total size of SSRAM chips on each bank is limited to 2M bytes. Memory bank C can also connect to one external SDRAM of either 16M or 32M bytes. Memory banks A and B are 16 bits wide, while bank C is 32 bits wide. Because SDRAM devices are much more commonly found in 16-bit configuration, 2 devices can be placed side by side. On memory bank C, the address and data pins are shared between the SSRAM and SDRAM devices.

SSRAM must be pipelined or flow-through ZBT (Zero-Bus Turnaround) type memory.

To multiplex the accesses that all agents require of these memories, memory controllers are used. Each memory controller grants the memory bus to the various agents within the chip, using a priority algorithm to make sure that the agents that need the memory most urgently get it, and transforming these memory accesses into the correct pin signals depending on the configuration of the memories.

The memory controller is responsible for generating even parity on the parity pins of the memories and detecting that the parity is correctly received when data is read from the memory. To do so, it calculates even parity on all the address bits and data bits used to generate each access. When reading from the memory, it performs the same calculation in the opposite direction. Any errors in parity are reported to registers. To render parity generation and detection more powerful, masks can be used, causing the memory controller to only calculate parity on some bits. These masks are programmed in registers 230h to 234h.

Parity is calculated on all locations in memory except for the reception circular buffers, in which the parity bits are used for underrun information. It is possible to override this and use parity even on these circular buffers through control bits in registers.

The controller also makes sure that the SDRAMs used are refreshed often enough so as to ensure that data in them is never corrupted. The refresh period is indicated in registers, and a limit value is placed on how far behind in its refreshing the chip can afford to be. If the refresh mechanism falls behind by more than this number, a status error will be reported. This is programmed in registers 398h and 39Ah.

Finally, the memory controller allows manual accesses to the SDRAM to be performed through registers, allowing CPU accesses to perform the initialization sequence to the SDRAM.

12.0 Pin-out

The following list describes the pin-out of the MT92210 device. Some pins have multiple functions, however, each pin bears the name of its principal function.

All outputs are 3.3 V outputs only.

Pins identified with (F) in **Type** tolerate 5V for inputs and outputs.

All pins are tested with 50 pf of load, unless otherwise specified.

Pin Description Table

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
A14	upclk	CPU			I			LVTTL (F)	
A19	cpu_mode[0]	CPU			I			LVTTL	
C20	cpu_mode[1]	CPU			I			LVTTL	
B18	cpu_mode[2]	CPU			I			LVTTL	
D19	cpu_mode[3]	CPU			I			LVTTL	
T1	cpu_a[0]	CPU			I			LVTTL (F)	
P1	cpu_a[1]	CPU			I			LVTTL (F)	
N1	cpu_a[2]	CPU			I			LVTTL (F)	
P2	cpu_a[3]	CPU			I			LVTTL (F)	
P5	cpu_a[4]	CPU			I			LVTTL (F)	
M1	cpu_a[5]	CPU			I			LVTTL (F)	
M3	cpu_a[6]	CPU			I			LVTTL (F)	
N2	cpu_a[7]	CPU			I			LVTTL (F)	
N5	cpu_a[8]	CPU			I			LVTTL (F)	
N4	cpu_a[9]	CPU			I			LVTTL (F)	
K4	cpu_a[10]	CPU			I			LVTTL (F)	
M2	cpu_a[11]	CPU			I			LVTTL (F)	
K1	cpu_a[12]	CPU			I			LVTTL (F)	
L3	cpu_a[13]	CPU			I			LVTTL (F)	
J4	cpu_a[14]	CPU			I			LVTTL (F)	
R3	cpu_wr_rw	CPU			I			LVTTL (F)	
R4	cpu_rd_ds	CPU			I			LVTTL (F)	
R2	cpu_ale	CPU			I			LVTTL (F)	
R1	cpu_a_das	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
T2	cpu_cs	CPU	12.5	25	I/O			LVTTL 4 mA (F)	
AA4	cpu_d[0]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
Y2	cpu_d[1]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
AB1	cpu_d[2]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
W5	cpu_d[3]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
W3	cpu_d[4]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
W2	cpu_d[5]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
V4	cpu_d[6]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
U5	cpu_d[7]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
V5	cpu_d[8]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
V2	cpu_d[9]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
W1	cpu_d[10]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
V1	cpu_d[11]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
T3	cpu_d[12]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
T4	cpu_d[13]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
U1	cpu_d[14]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
U2	cpu_d[15]	CPU	12.5	25	I/O	Z		LVTTL 4 mA (F)	
T5	cpu_rdy_ndtack	CPU	12.5	25	I/O	Z	U	LVTTL 8 mA (F)	
Y5	cpu_int[0]	CPU			O	Z	U	LVTTL 4 mA (F)	
W4	cpu_int[1]	CPU			O	Z	U	LVTTL 4 mA (F)	
AD2	ct_c8_a	H110	8	8	I/O	Z		Schmitt 12 mA (F)	200 pf load
AC1	ct_c8_b	H110	8	8	I/O	Z		Schmitt 12 mA (F)	200 pf load
AG2	ct_frame_a	H110	1	1	I/O	Z		LVTTL 12 mA (F)	200 pf load
AG1	ct_frame_b	H110	1	1	I/O	Z		LVTTL 12 mA (F)	200 pf load
AD4	ct_netref1	H110	2	2	I/O	Z		Schmitt 12 mA (F)	200 pf load
AD5	ct_netref2	H110	2	2	I/O	Z		Schmitt 12 mA (F)	200 pf load
AH2	ct_mc	H110	1	2	I/O	Z		LVTTL 12 mA (F)	If this pin is connected to the H110 bus, gpio[2] must be used to drive it. When reset is deasserted and ct_mc output enabled, gpio[2] low will cause ct_mc to be driven low. Otherwise it will be tri-stated.
AJ14	ct_d[0]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ13	ct_d[1]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AG13	ct_d[2]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK13	ct_d[3]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF14	ct_d[4]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ12	ct_d[5]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AG10	ct_d[6]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK12	ct_d[7]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF13	ct_d[8]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK11	ct_d[9]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ10	ct_d[10]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ11	ct_d[11]	H110	1	4	I/O	Z		PCI (F)	200 pf load

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
AF10	ct_d[12]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF12	ct_d[13]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF11	ct_d[14]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ9	ct_d[15]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK10	ct_d[16]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK9	ct_d[17]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AH8	ct_d[18]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF8	ct_d[19]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AG7	ct_d[20]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AG5	ct_d[21]	H110	1	4	I/O	Z		PCI (F)	200 pf load
Ak7	ct_d[22]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ6	ct_d[23]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF9	ct_d[24]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK6	ct_d[25]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ5	ct_d[26]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK8	ct_d[27]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AF7	ct_d[28]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ8	ct_d[29]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AK5	ct_d[30]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AJ7	ct_d[31]	H110	1	4	I/O	Z		PCI (F)	200 pf load
AE1	sclk	H110	8	8	O	Z		LVTTL 12 mA (F)	200 pf load
AF2	sclkx2	H110	8	8	O	Z		LVTTL 12 mA (F)	200 pf load
AB2	c16p	H110	16	16	O	Z		LVTTL 12 mA (F)	200 pf load
AE2	c16n	H110	16	16	O	Z		LVTTL 12 mA (F)	200 pf load
AF5	c2	H110	2	2	O	Z		LVTTL 12 mA (F)	200 pf load
AH1	c4	H110	4	4	O	Z		LVTTL 12 mA (F)	200 pf load
AF4	frcomp	H110	1	1	O	Z		LVTTL 12 mA (F)	200 pf load
T28	nreset	MISC			I			LVTTL	
AB4	pll_clk	MISC			I			LVTTL (F)	
AC4	gpio[0]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AA2	gpio[1]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AC3	gpio[2]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AA1	gpio[3]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AA5	gpio[4]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AC2	gpio[5]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
AB5	gpio[6]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
Y4	gpio[7]	MISC	16	16	I/O	Z		LVTTL 4 mA (F)	
R28	trst	Test			I		D	LVTTL	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
AF1	tck	Test			I			LVTTL (F)	
AD1	tdi	Test			I			LVTTL (F)	
AF15	tms	Test			I			LVTTL	
AE4	tdo	Test			O	X		LVTTL 4 mA (F)	
AH16	iddtn	Test			I		D	LVTTL	
K30	Icptn	Test			I		U	LVTTL	
AG3	Manufacturing_tm	Test			I		D	LVTTL (F)	
AK4	proc_out	Test			O	X		LVTTL 4 mA	
AE5	txa_led	PORTA			I/O			LVTTL 8 mA (F)	
F29	rxa_led	PORTA			I/O			LVTTL 8 mA	
D18	rxa_alarm	PORTA			I			LVTTL (F)	
L29	mem_clk_sar_o	MEM	80	80	O	0		LVTTL 4 mA	
AJ24	mem_clk_net_o	MEM	80	80	O	0		LVTTL 4 mA	
L26	mem_clk_sar_i	MEM			I			LVTTL	
AJ23	mem_clk_net_i	MEM			I			LVTTL	
Y30	mem_oe	MEM			O	1		LVTTL 4 mA	
G27	mema_cs[0]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
F30	mema_cs[1]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
J30	mema_cs[2]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
B26	mema_cs[3]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
H27	mema_rw	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
J26	mema_bws[0]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
E27	mema_bws[1]	MEMA	18.75	37.5	O	1		LVTTL 4 mA	
C27	mema_a[0]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
A24	mema_a[1]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
B25	mema_a[2]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
D25	mema_a[3]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
C28	mema_a[4]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
D26	mema_a[5]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
E30	mema_a[6]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
D28	mema_a[7]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
F27	mema_a[8]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
K29	mema_a[9]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
C24	mema_a[10]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
E23	mema_a[11]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
B21	mema_a[12]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
D24	mema_a[13]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
C23	mema_a[14]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
A22	mema_a[15]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
B20	mema_a[16]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
H30	mema_a[17]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
G28	mema_a[18]	MEMA	18.75	37.5	O	X		LVTTL 4 mA	
E22	mema_d[0]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
E24	mema_d[1]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
A26	mema_d[2]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
B24	mema_d[3]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
D30	mema_d[4]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
E29	mema_d[5]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
H29	mema_d[6]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
F26	mema_d[7]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
E26	mema_d[8]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
G26	mema_d[9]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
C30	mema_d[10]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
G30	mema_d[11]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
B23	mema_d[12]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
A23	mema_d[13]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
B28	mema_d[14]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
A25	mema_d[15]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
G29	mema_p[0]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
A28	mema_p[1]	MEMA	9.375	37.5	I/O	Z		LVTTL 4 mA	
N30	memb_d[0]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
M27	memb_d[1]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
P30	memb_d[2]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
R27	memb_d[3]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
T29	memb_d[4]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
T27	memb_d[5]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
U30	memb_d[6]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
W27	memb_d[7]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
U29	memb_d[8]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
U26	memb_d[9]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
T30	memb_d[10]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
T26	memb_d[11]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
R30	memb_d[12]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
P29	memb_d[13]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
R29	memb_d[14]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
M30	memb_d[15]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
V30	memb_p[0]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
R26	memb_p[1]	MEMB	9.375	37.5	I/O	Z		LVTTL 4 mA	
W26	memb_cs[0]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
AB27	memb_cs[1]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
Y28	memb_cs[2]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
Y26	memb_cs[3]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
V27	memb_rw	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
AA27	memb_bws[0]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
W29	memb_bws[1]	MEMB	18.75	37.5	O	1		LVTTL 4 mA	
N27	memb_a[0]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
K27	memb_a[1]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
P26	memb_a[2]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
N29	memb_a[3]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
L30	memb_a[4]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
M28	memb_a[5]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
Y27	memb_a[6]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
AA26	memb_a[7]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
W30	memb_a[8]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
V29	memb_a[9]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
M29	memb_a[10]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
N26	memb_a[11]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
H28	memb_a[12]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
L28	memb_a[13]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
K26	memb_a[14]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
M26	memb_a[15]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
J29	memb_a[16]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
W28	memb_a[17]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
V26	memb_a[18]	MEMB	18.75	37.5	O	X		LVTTL 4 mA	
AH24	memc_d[0]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AK24	memc_d[1]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AK22	memc_d[2]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AH23	memc_d[3]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AG24	memc_d[4]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AJ21	memc_d[5]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AK21	memc_d[6]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AJ22	memc_d[7]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AG23	memc_d[8]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
AJ20	memc_d[9]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AF19	memc_d[10]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AF21	memc_d[11]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AK20	memc_d[12]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AJ19	memc_d[13]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AF20	memc_d[14]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AH20	memc_d[15]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AD29	memc_d[16]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AE26	memc_d[17]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AC29	memc_d[18]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AH29	memc_d[19]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AD26	memc_d[20]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AC30	memc_d[21]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AH30	memc_d[22]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AE30	memc_d[23]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AB26	memc_d[24]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AE27	memc_d[25]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AE29	memc_d[26]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AD30	memc_d[27]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AG28	memc_d[28]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AD28	memc_d[29]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AC26	memc_d[30]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AA29	memc_d[31]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AF23	memc_p[0]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AG22	memc_p[1]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AK26	memc_p[2]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AC28	memc_p[3]	MEMC	9.375	37.5	I/O	Z		LVTTL 4 mA	
AG26	memc_cs[0]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AK27	memc_cs[1]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AB29	memc_cs[2]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AA30	memc_cs[3]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AF22	memc_rw	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AH28	memc_bws[0]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AG25	memc_bws[1]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AJ25	memc_bws[2]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AK23	memc_bws[3]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AG16	memc_cas[0]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
Y29	memc_cas[1]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
AB30	memc_ras	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AG19	memc_we[0]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AC27	memc_we[1]	MEMC	18.75	37.5	O	1		LVTTL 4 mA	
AG21	memc_a[0]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AJ17	memc_a[1]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK18	memc_a[2]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AF18	memc_a[3]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AH19	memc_a[4]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK16	memc_a[5]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK17	memc_a[6]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AF17	memc_a[7]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK15	memc_a[8]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AJ16	memc_a[9]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AG18	memc_a[10]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AF16	memc_a[11]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK19	memc_a[12]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AJ18	memc_a[13]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AF27	memc_a[14]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AF24	memc_a[15]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AK25	memc_a[16]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
AG30	memc_a[17]	MEMC	18.75	37.5	O	X		LVTTL 4 mA	
E13	txb_clk	UTOPIA B	25	25	I/O	Z		LVTTL 8 mA (F)	
B13	1. txb_enb 2. txb_clav	UTOPIA B	6.25	12.5	1. O 2. O	Z	1. U 2. D	LVTTL 4 mA (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIA PHY Layer
E12	1. txb_clav 2. txb_enb	UTOPIA B			1. I 2. I	Z	1. D 2. U	LVTTL (F)	1. MT92210 is a UTOPIA ATM Laye1. 2. MT92210 is a UTOPIA PHY Layer
A13	txb_soc	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
D10	txb_prty	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
A11	txb_d[0]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
E10	txb_d[1]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
C11	txb_d[2]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
C12	txb_d[3]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
A12	txb_d[4]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
B12	txb_d[5]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
D13	txb_d[6]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
D12	txb_d[7]	UTOPIA B	6.25	12.5	O	Z		LVTTL 4 mA (F)	
D11	rxb_clk	UTOPIA B	25	25	I/O	Z		LVTTL 8 mA (F)	
E11	1. rxb_enb 2. rxb_clav	UTOPIA B	6.25	12.5	1. O 2. O	Z	1. U 2. D	LVTTL 4 mA (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIA PHY Layer
B9	1. rxb_cla 2. rxb_enb	UTOPIA B			1. I 2. I	Z	1. D 2. U	LVTTL (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIAPHY Layer
D9	rxb_soc	UTOPIA B			I	Z		LVTTL (F)	
A10	rxb_prty	UTOPIA B			I	Z		LVTTL (F)	
A9	rxb_d[0]	UTOPIA B			I	Z		LVTTL (F)	
E7	rxb_d[1]	UTOPIA B			I	Z		LVTTL (F)	
E9	rxb_d[2]	UTOPIA B			I	Z		LVTTL (F)	
C8	rxb_d[3]	UTOPIA B			I	Z		LVTTL (F)	
D7	rxb_d[4]	UTOPIA B			I	Z		LVTTL (F)	
B10	rxb_d[5]	UTOPIA B			I	Z		LVTTL (F)	
A8	rxb_d[6]	UTOPIA B			I	Z		LVTTL (F)	
D8	rxb_d[7]	UTOPIA B			I	Z		LVTTL (F)	
D5	txa_clk	NETA	25	25	I/O	Z		LVTTL 8 mA (F)	
H2	txa_d[0]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
F2	txa_d[1]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
G2	txa_d[2]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
E2	txa_d[3]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
D3	txa_d[4]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
A4	txa_d[5]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
D4	txa_d[6]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
A7	txa_d[7]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B3	txa_d[8]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
E8	txa_d[9]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
E6	txa_d[10]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
C4	txa_d[11]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B8	txa_d[12]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B6	txa_d[13]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
A3	txa_d[14]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B4	txa_d[15]	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B7	txa_prty	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	
B15	1. txa_enb 2. txa_clav 3. txa_enb	NETA	6.25	12.5	O	Z	1. U 2. D 3. U	LVTTL 8 mA (F)	1. MT92210 is UTOPIA ATM Layer 2.MT92210 is UTOPIA PHY Layer 3. MT92210 is POS-PHY Link Layer
A5	1. txa_soc 2. txa_sop	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	1. MT92210 is UTOPIA ATM/PHY Layer 2. MT92210 is a POS-PHY Link Layer
C2	1. txa_clav 2. txa_enb 3. txa_ptpa	NETA			I		1. D 2. U 3. D	LVTTL (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIA PHY Layer 3. MT92210 is a POS-PHY Link Layer
D22	txa_addr[0]	NETA			I			LVTTL	
E20	txa_addr[1]	NETA			I			LVTTL	
B19	txa_addr[2]	NETA			I			LVTTL	MT92210 is UTOPIA PHY Layer
E18	1. txa_addr[3] 2. txa_mod	NETA	6.25	12.5	1. I 2. O	Z		LVTTL 8 mA	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer
A21	1. txa_addr[4] 2. txa_eop	NETA	6.25	12.5	1. I 2. O	Z		LVTTL 8 mA	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer
L1	rxa_d[0]	NETA			I			LVTTL (F)	
K5	rxa_d[1]	NETA			I			LVTTL (F)	
M5	rxa_d[2]	NETA			I			LVTTL (F)	
L5	rxa_d[3]	NETA			I			LVTTL (F)	

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
J2	rxa_d[4]	NETA			I			LVTTL (F)	
H4	rxa_d[5]	NETA			I			LVTTL (F)	
L2	rxa_d[6]	NETA			I			LVTTL (F)	
J1	rxa_d[7]	NETA			I			LVTTL (F)	
H3	rxa_d[8]	NETA			I			LVTTL (F)	
G4	rxa_d[9]	NETA			I			LVTTL (F)	
K2	rxa_d[10]	NETA			I			LVTTL (F)	
G1	rxa_d[11]	NETA			I			LVTTL (F)	
G3	rxa_d[12]	NETA			I			LVTTL (F)	
D1	rxa_d[13]	NETA			I			LVTTL (F)	
H5	rxa_d[14]	NETA			I			LVTTL (F)	
F1	rxa_d[15]	NETA			I			LVTTL (F)	
J5	rxa_prty	NETA			I			LVTTL (F)	
H1	1. rxa_soc 2. rxa_sop	NETA			I			LVTTL (F)	1. MT92210 is UTOPIA ATM/PHY Layer 2. MT92210 is POS-PHY Link Layer
G5	1. rxa_enb 2. rxa_clav 3. rxa_enb	NETA	6.25	12.5	O	Z		LVTTL 8 mA (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIA PHY Layer 3. MT92210 is a POS-PHY Link Layer
C1	1. rxa_clav 2. rxa_enb 3. rxa_prpa	NETA			I			LVTTL (F)	1. MT92210 is a UTOPIA ATM Layer 2. MT92210 is a UTOPIA PHY Layer 3. MT92210 is a POS-PHY Link Layer
E4	rxa_clk	NETA	25	25	I/O	Z		LVTTL 8 mA (F)	
A20	rxa_addr[0]	NETA			I			LVTTL	
D20	1. rxa_addr[1] 2. rxa_val	NETA			I			LVTTL	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer
E21	1. rxa_addr[2] 2. rxa_err	NETA			I			LVTTL	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer
E19	1. rxa_addr[3] 2. rxa_mod	NETA			I			LVTTL	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
D23	1. rxa_addr[4] 2. rxa_eop	NETA			I			LVTTL	1. MT92210 is UTOPIA PHY Layer 2. MT92210 is POS-PHY Link Layer
B14	etha_tx_clk	NETA			I			LVTTL (F)	
A18	etha_tx_en	NETA	6.25	12.5	O	0		LVTTL 4 mA (F)	
B16	etha_tx_d[0]	NETA	6.25	12.5	O	X		LVTTL 4 mA (F)	
E16	etha_tx_d[1]	NETA	6.25	12.5	O	X		LVTTL 4 mA (F)	
C16	etha_tx_d[2]	NETA	6.25	12.5	O	X		LVTTL 4 mA (F)	
D16	etha_tx_d[3]	NETA	6.25	12.5	O	X		LVTTL 4 mA (F)	
D15	etha_col	NETA			I			LVTTL (F)	
C15	etha_crs	NETA			I			LVTTL (F)	
E15	etha_rx_clk	NETA			I			LVTTL (F)	
A16	etha_rx_d[0]	NETA			I			LVTTL (F)	
A17	etha_rx_d[1]	NETA			I			LVTTL (F)	
E17	etha_rx_d[2]	NETA			I			LVTTL (F)	
B17	etha_rx_d[3]	NETA			I			LVTTL (F)	
C19	etha_rx_er	NETA			I			LVTTL (F)	
E14	etha_rx_dv	NETA			I			LVTTL (F)	
A2	vss_ring	Power							0V Power Supply used for core and I/Os.
A29	vss_ring	Power							
AA3	vss_ring	Power							
AA28	vss_ring	Power							
AB3	vss_ring	Power							
AB28	vss_ring	Power							
AE3	vss_ring	Power							
AE28	vss_ring	Power							
AF3	vss_ring	Power							
AF28	vss_ring	Power							
AG4	vss_ring	Power							
AG27	vss_ring	Power							
AH05	vss_ring	Power							
AH06	vss_ring	Power							
AH09	vss_ring	Power							
AH10	vss_ring	Power							
AH13	vss_ring	Power							
AH14	vss_ring	Power							
AH17	vss_ring	Power							

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
AH18	vss_ring	Power							
AH21	vss_ring	Power							
AH22	vss_ring	Power							
AH25	vss_ring	Power							
AH26	vss_ring	Power							
AJ1	vss_ring	Power							
AJ2	vss_ring	Power							
AJ29	vss_ring	Power							
AJ30	vss_ring	Power							
AK2	vss_ring	Power							
AK29	vss_ring	Power							
B1	vss_ring	Power							
B2	vss_ring	Power							
B29	vss_ring	Power							
B30	vss_ring	Power							
C3	vss_ring	Power							
C5	vss_ring	Power							
C6	vss_ring	Power							
C9	vss_ring	Power							
C10	vss_ring	Power							
C13	vss_ring	Power							
C14	vss_ring	Power							
C17	vss_ring	Power							
C18	vss_ring	Power							
C21	vss_ring	Power							
C22	vss_ring	Power							
C25	vss_ring	Power							
C26	vss_ring	Power							
D27	vss_ring	Power							
E3	vss_ring	Power							
E28	vss_ring	Power							
F3	vss_ring	Power							
F28	vss_ring	Power							
J3	vss_ring	Power							
J28	vss_ring	Power							
K3	vss_ring	Power							
K28	vss_ring	Power							
N3	vss_ring	Power							

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
N13	vss_ring	Power							
N14	vss_ring	Power							
N15	vss_ring	Power							
N16	vss_ring	Power							
N17	vss_ring	Power							
N18	vss_ring	Power							
N28	vss_ring	Power							
P3	vss_ring	Power							
P13	vss_ring	Power							
P14	vss_ring	Power							
P15	vss_ring	Power							
P16	vss_ring	Power							
P17	vss_ring	Power							
P18	vss_ring	Power							
P28	vss_ring	Power							
R13	vss_ring	Power							
R14	vss_ring	Power							
R15	vss_ring	Power							
R16	vss_ring	Power							
R17	vss_ring	Power							
R18	vss_ring	Power							
T13	vss_ring	Power							
T14	vss_ring	Power							
T15	vss_ring	Power							
T16	vss_ring	Power							
T17	vss_ring	Power							
T18	vss_ring	Power							
U3	vss_ring	Power							
U13	vss_ring	Power							
U14	vss_ring	Power							
U15	vss_ring	Power							
U16	vss_ring	Power							
U17	vss_ring	Power							
U18	vss_ring	Power							
U28	vss_ring	Power							
V3	vss_ring	Power							
V13	vss_ring	Power							
V14	vss_ring	Power							

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
V15	vss_ring	Power							
V16	vss_ring	Power							
V17	vss_ring	Power							
V18	vss_ring	Power							
V28	vss_ring	Power							
AA25	vdd33_ring	Power							3.3V Power Supply used for core and I/Os.
AB6	vdd33_ring	Power							
AC6	vdd33_ring	Power							
AD25	vdd33_ring	Power							
AE8	vdd33_ring	Power							
AE9	vdd33_ring	Power							
AE12	vdd33_ring	Power							
AE13	vdd33_ring	Power							
AE16	vdd33_ring	Power							
AE17	vdd33_ring	Power							
AE20	vdd33_ring	Power							
AE21	vdd33_ring	Power							
AE24	vdd33_ring	Power							
AE25	vdd33_ring	Power							
F6	vdd33_ring	Power							
F7	vdd33_ring	Power							
F10	vdd33_ring	Power							
F11	vdd33_ring	Power							
F14	vdd33_ring	Power							
F15	vdd33_ring	Power							
F18	vdd33_ring	Power							
F19	vdd33_ring	Power							
F22	vdd33_ring	Power							
F23	vdd33_ring	Power							
G6	vdd33_ring	Power							
H25	vdd33_ring	Power							
J25	vdd33_ring	Power							
K6	vdd33_ring	Power							
L6	vdd33_ring	Power							
M25	vdd33_ring	Power							
N25	vdd33_ring	Power							
P6	vdd33_ring	Power							

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
R6	vdd33_ring	Power							
T25	vdd33_ring	Power							
U25	vdd33_ring	Power							
V6	vdd33_ring	Power							
W6	vdd33_ring	Power							
Y25	vdd33_ring	Power							
AA6	vdd33_ring	Power							
AD6	vdd33_ring	Power							
AE6	vdd33_ring	Power							
H06	vdd33_ring	Power							
J6	vdd33_ring	Power							
M6	vdd33_ring	Power							
N6	vdd33_ring	Power							
T6	vdd33_ring	Power							
U6	vdd33_ring	Power							
Y6	vdd33_ring	Power							
AE07	vdd33_ring	Power							
AE10	vdd33_ring	Power							
AE11	vdd33_ring	Power							
AE14	vdd33_ring	Power							
AE15	vdd33_ring	Power							
AE18	vdd33_ring	Power							
AE19	vdd33_ring	Power							
AE22	vdd33_ring	Power							
AE23	vdd33_ring	Power							
AB25	vdd33_ring	Power							
AC25	vdd33_ring	Power							
F25	vdd33_ring	Power							
G25	vdd33_ring	Power							
K25	vdd33_ring	Power							
L25	vdd33_ring	Power							
P25	vdd33_ring	Power							
R25	vdd33_ring	Power							
V25	vdd33_ring	Power							
W25	vdd33_ring	Power							
F8	vdd33_ring	Power							
F9	vdd33_ring	Power							
F12	vdd33_ring	Power							

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
F13	vdd33_ring	Power							
F16	vdd33_ring	Power							
F17	vdd33_ring	Power							
F20	vdd33_ring	Power							
F21	vdd33_ring	Power							
F24	vdd33_ring	Power							
AK14	Vss_0	Power							Additional 0V power supply pins.
AJ15	vss_1	Power							Additional 0V power supply pins.
A6	vdd5_0	Power							5V Power Supply used for 5V tolerance. May be connected to 3.3Vpower supply if all devices connected to the MT92210 are 3.3V only.
B11	vdd5_1	Power							See vdd5_0.
D17	vdd5_2	Power							See vdd5_0.
F4	vdd5_3	Power							See vdd5_0.
M4	vdd5_4	Power							See vdd5_0.
Y1	vdd5_5	Power							See vdd5_0.
AC5	vdd5_6	Power							See vdd5_0.
AG6	vdd5_7	Power							See vdd5_0.
AG9	vdd5_8	Power							See vdd5_0.
AH11	vdd5_9	Power							See vdd5_0.
AH15	vdd5_10	Power							See vdd5_0.
AJ3	fc1pll_pllvss	Power							0V PLL power supply. See Figure PLL noise reduction circuit for reference design.
AK3	fc1pll_pllvdd	Power							2.5V PLL power supply. See Figure PLL noise reduction circuit for reference design.
AJ26	fc2pll_pllvdd	Power							2.5V PLL power supply. See Figure PLL noise reduction circuit for reference design.
AF25	fc2pll_pllvss	Power							0V PLL power supply. See Figure PLL noise reduction circuit for reference design.

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
C29	h110pll_pllvss	Power							0V PLL power supply. See Figure PLL noise reduction circuit for reference design.
D29	h110pll_pllvdd	Power							2.5V PLL power supply. See Figure PLL noise reduction circuit for reference design.
A15	vdd25_0	Power							2.5V Core power supply.
B5	vdd25_1	Power							See vdd25_0
B22	vdd25_2	Power							See vdd25_0
C7	vdd25_3	Power							See vdd25_0
D14	vdd25_4	Power							See vdd25_0
D21	vdd25_5	Power							See vdd25_0
F5	vdd25_6	Power							See vdd25_0
H26	vdd25_7	Power							See vdd25_0
J27	vdd25_8	Power							See vdd25_0
L4	vdd25_9	Power							See vdd25_0
P4	vdd25_10	Power							See vdd25_0
P27	vdd25_11	Power							See vdd25_0
U4	vdd25_12	Power							See vdd25_0
U27	vdd25_13	Power							See vdd25_0
Y3	vdd25_14	Power							See vdd25_0
AD3	vdd25_15	Power							See vdd25_0
AD27	vdd25_16	Power							See vdd25_0
AG11	vdd25_17	Power							See vdd25_0
AG15	vdd25_18	Power							See vdd25_0
AG17	vdd25_19	Power							See vdd25_0
AG20	vdd25_20	Power							See vdd25_0
AG29	vdd25_21	Power							See vdd25_0
AH3	vdd25_22	Power							See vdd25_0
AH27	vdd25_23	Power							See vdd25_0
A27	NC								Leave these “No Connect” pins floating.
B27	NC								
D2	NC								
D6	NC								
E1	NC								
E5	NC								
E25	NC								

Pin Description Table (continued)

Pin	SIGNAL Name	Interface	Nom Switch (MHz)	WC Switch (MHz)	I/O	Reset	Pull	Type	Description
L27	NC								
R5	NC								
AF6	NC								
AF26	NC								
AF29	NC								
AF30	NC								
AG8	NC								
AG12	NC								
AG14	NC								
AH4	NC								
AH7	NC								
AH12	NC								
AJ4	NC								
AJ27	NC								
AJ28	NC								
AK28	NC								

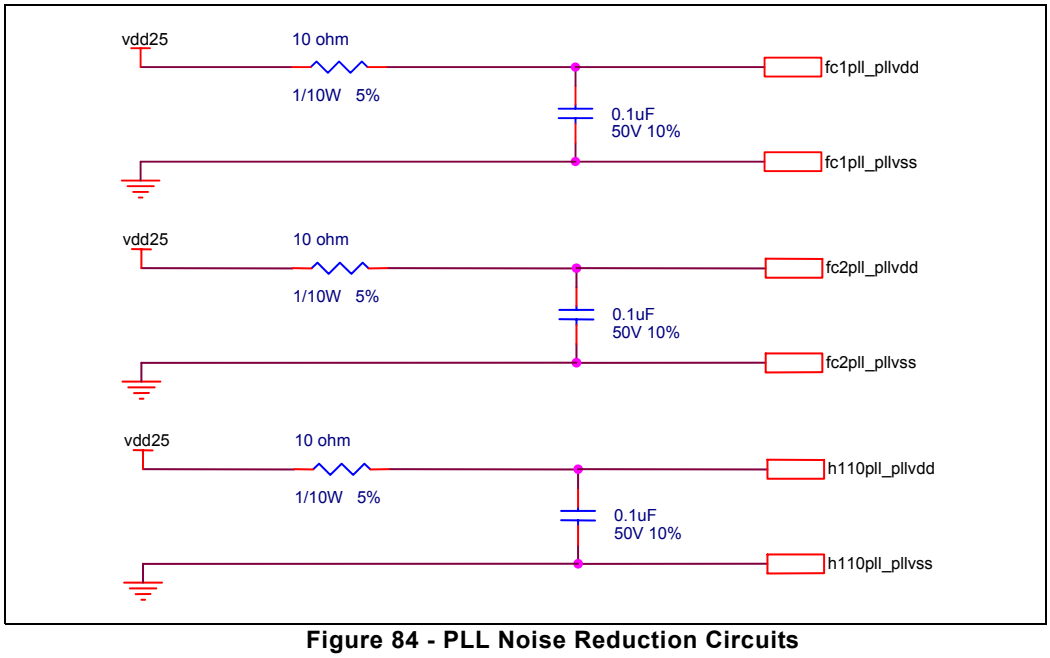


Figure 84 - PLL Noise Reduction Circuits

13.0 Electrical Characteristics

13.1 Absolute Maximum Conditions

Absolute Maximum Conditions

Parameter	Symbol	Limits	Unit
DC Supply Voltage[2]	VDD	-0.3 to + 3.1	V
3.3 V DC Supply Voltage	VDD3.3	-0.3 to + 3.9	V
LVTTL Input Voltage	$V_{IN2.5}$	-1.0 to VDD + 0.3	V
3.3 V Drive Input Voltage	$V_{IN3.3}$	-1.0 to VDD3.3 + 0.0	V
5 V Compatible Input Voltage	V_{IN5}	-1.0 to 6.5	V
DC Input Current	I_{IN}	+10	uA
Storage Temperature Range (Ceramic)	TSTG	-65 to + 150	°C
Storage Temperature Range (Plastic)	TSTG	-40 to + 125	°C
ESD Per MIL-STD-883 Test Method 3015, Notice 8, Spec 2001V Latchup Over/undershoot: + 150 mA, 125 °C		VDD Overstress: 2x VDD (7.2 V)	V

Note 1: Operation beyond the limits specified in this table may cause permanent device damage.
Note 2: Internal cells (core) and standard I/Os operate at 2.5 V.
Note 3: Voltage are referenced to ground (VSS) unless otherwise stated

13.2 Recommended Operating Conditions

Recommended Operating Conditions

Parameter	Symbol	Limits[1]	Unit
DC Supply Voltage [2]	VDD	+ 2.25 to 2.75	V
Operating Junction Temperature Range Industrial Commercial	 TJ TJ	 -40 to + 125 0 to + 85	 °C °C

Note 1: For normal device operation, adhere to the limits in this table. Sustained operation of a device at conditions exceeding these values, even if they are within the absolute maximum rating limits, may result in permanent device damage or impaired device reliability. Device functionality to stated DC and AC limits is not guaranteed if conditions exceed recommended operating conditions.
Note 2: Core supply voltage (2.5 V nominal).
Note 3: Voltage are reference to ground (Vss) unless otherwise stated

13.3 DC Characteristics

Standard 2.5V LVTLL Buffers

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage	VDD	_	2.25	2.5	2.75	V
Power Dissipation	P_D	1023 RTP Connections, all output pins loaded			2.33	W
Input Low Voltage	V_{IL}	_	Vss-0.5	_	0.7	V

Standard 2.5V LVTLL Buffers (continued)

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Input High Voltage	V_{IH}	LVTTL Comm/Ind Temp Range	1.7	–	VDD +0.3	V
Switching Threshold	V_T	–	–	1.35	2.0	V
Schmitt Trigger, Positive going Threshold	V_{T+}	–	–	1.7	2.0	V
Schmitt Trigger, Negative going Threshold	V_{T-}	–	0.8	1.0	–	V
Schmitt Trigger, Hysteresis		–	0.6	0.7	–	V
Input Current Inputs with Pull-down Resistors Inputs with Pull-up Resistors	I_{IN}	V_{IN} = VDD or VSS V_{IN} = VDD V_{IN} = VSS	-10 80 -28	11 200 -185	10 390 -393	uA uA uA

Note 1: Industrial junction temperature range: - 40 to + 125 °C, + 5% power supply, commercial junction temperature range: 0 to 85 °C, + 5% power supply.
Note 2: Output using single buffer structure (excluding package).

Standard 3.3V LVTLL Buffers and 5V Compatible Buffers

Parameter	Symbol	Condition[1]	Min	Typ	Max	Unit
Supply Voltage	VDD3.3	_	3.0	3.3	3.6	V
Input Low Voltage	V_{IL}	_	VDD3.3-0.5	_	0.8	V
Input High Voltage	V_{IH}	LVTTL Comm/Ind Temp Range	2.0	_	VDD3.3 + 0.3	V
		5-Volt Compatible	2.0	_	5.5	V
Switching Threshold	V_T	–	–	1.4	2.0	V
Schmitt Trigger, Positive going Threshold	V_{T+}	–	–	1.7	2.0	V
Schmitt Trigger, Negative going Threshold	V_{T-}	–	0.8	1.0	–	V
Schmitt Trigger, Hysteresis		–	0.6	0.7	–	V
Input Current Inputs with Pull-down Resistors Inputs with Pull-up Resistors	I_{IN}	V_{IN} = VDD or VSS V_{IN} = VDD V_{IN} = VSS	-10 35 -35	11 115 -115	10 222 -214	uA uA uA
Output High Voltage	V_{OH}	_	2.4	_	VDD3.3	V
Output Low Voltage	V_{OL}	_	_	0.2	0.4	V

Standard 3.3V LVTLL Buffers and 5V Compatible Buffers (continued)

Parameter	Symbol	Condition[1]	Min	Typ	Max	Unit
3-state Output Leakage Current	I_{OZ}	V_{OH}=VSS or VDD3.3	-10	11	10	uA
Input Capacitance2	C_{IN}	Input and Bidirectional Buffers	4.0			pF
		5-volt Compatible	4.6			pF
Output Capacitance	C_{OUT}	Output Buffer	4.0			pF
		5-volt Compatible	4.6			pF

Note 1: Industrial junction temperature range: - 40 to + 125 °C, + 5% power supply, commercial junction temperature range: 0 to 85 °C, + 5% power supply.
Note 2: Output using single buffer structure (excluding package).

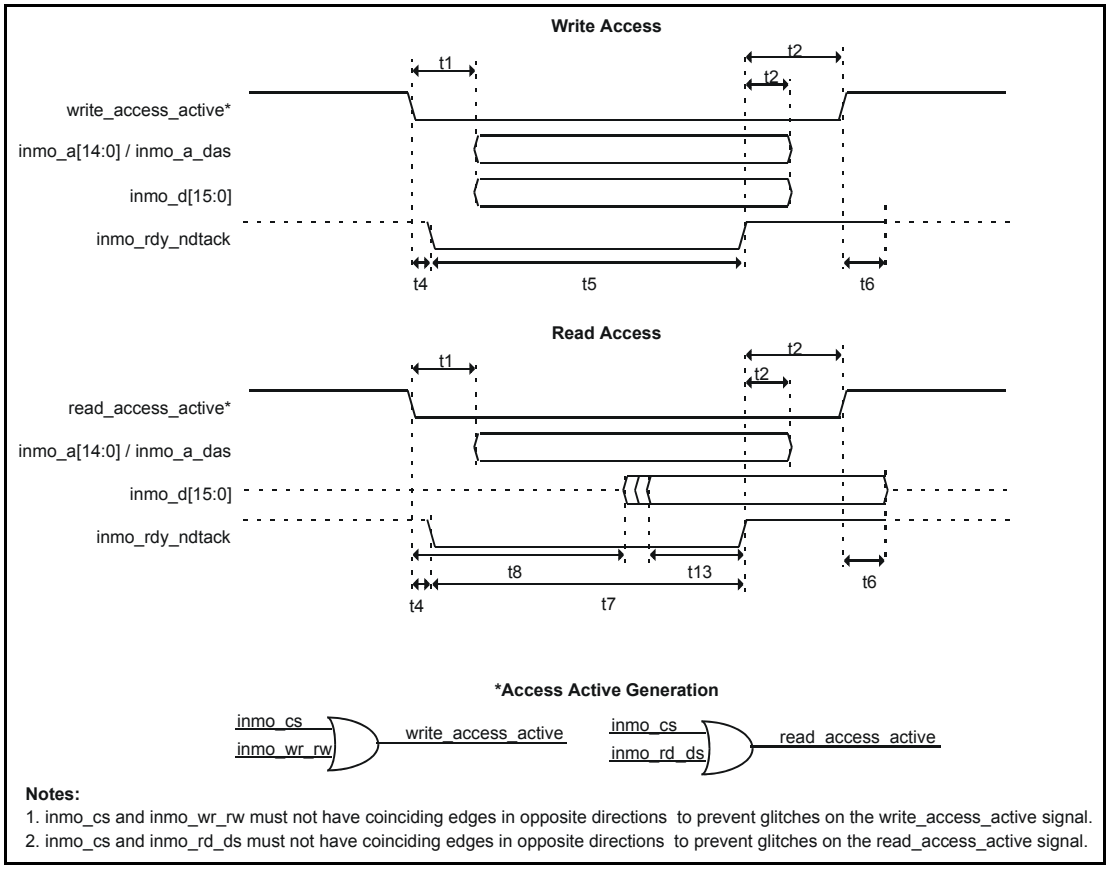
13.4 Clock Signals

Clock Signals

Clock Networks							
Clock Name	Minimum Frequency	Typical Frequency	Maximum Frequency	Required For Device Operation	Minimum Duty Cycle	Maximum Duty Cycle	Accepted Jitter (cycle-to-cycle)
mem_clk_sar_i	10 MHz	60 MHz	100 MHz	Yes	40%	60%	0 ps
mem_clk_net_i	10 MHz	80 MHz	100 MHz	Yes	40%	60%	400 ps
upclk	1 MHz	40 MHz	66 MHz	Yes	40%	60%	400 ps
rxa_clk	1 MHz	25 MHz	50 MHz	No	40%	60%	400 ps
rxb_clk	1 MHz	25 MHz	50 MHz	No	40%	60%	400 ps
txa_clk	1 MHz	25 MHz	50 MHz	No	40%	60%	400 ps
txb_clk	1 MHz	25 MHz	50 MHz	No	40%	60%	400 ps
etha_tx_clk	2.5 MHz	25 MHz	25 MHz	No	40%	60%	400 ps
etha_rx_clk	2.5 MHz	25 MHz	25 MHz	No	40%	60%	400 ps
pll_clk	8.192 MHz		65.536 MHz	No	40%	60%	400 ps
ct_c8_a/b	8.192 MHz		8.192 MHz	No	40%	60%	50 ns

13.5 AC Characteristics

13.5.1 Intel/Motorola CPU Interface



Notes:
1. inmo_cs and inmo_wr_rw must not have coinciding edges in opposite directions to prevent glitches on the write_access_active signal.
2. inmo_cs and inmo_rd_ds must not have coinciding edges in opposite directions to prevent glitches on the read_access_active signal.

Figure 85 - Non-multiplexed CPU Interface - Intel Mode

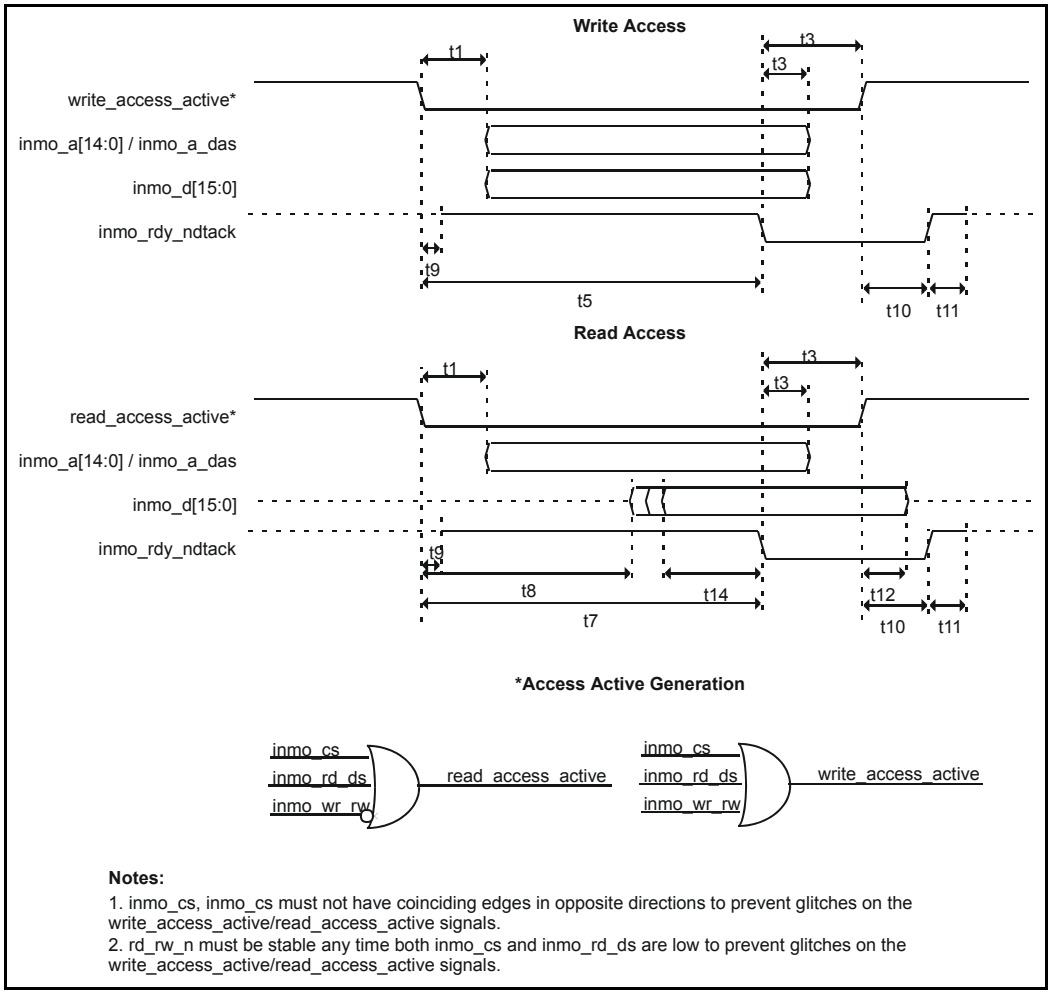
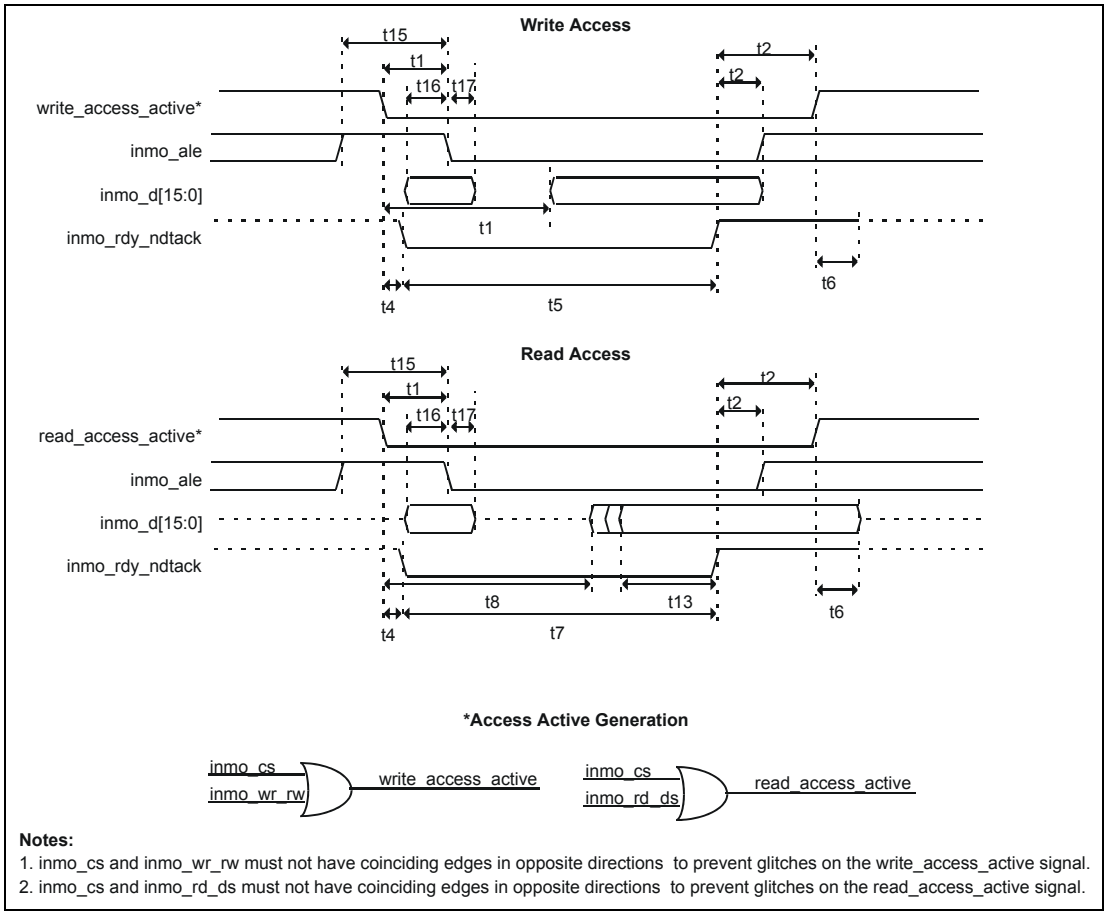


Figure 86 - Non-multiplexed CPU Interface - Motorola Mode



Notes:
1. inmo_cs and inmo_wr_rw must not have coinciding edges in opposite directions to prevent glitches on the write_access_active signal.
2. inmo_cs and inmo_rd_ds must not have coinciding edges in opposite directions to prevent glitches on the read_access_active signal.

Figure 87 - Multiplexed CPU Interface - Intel Mode

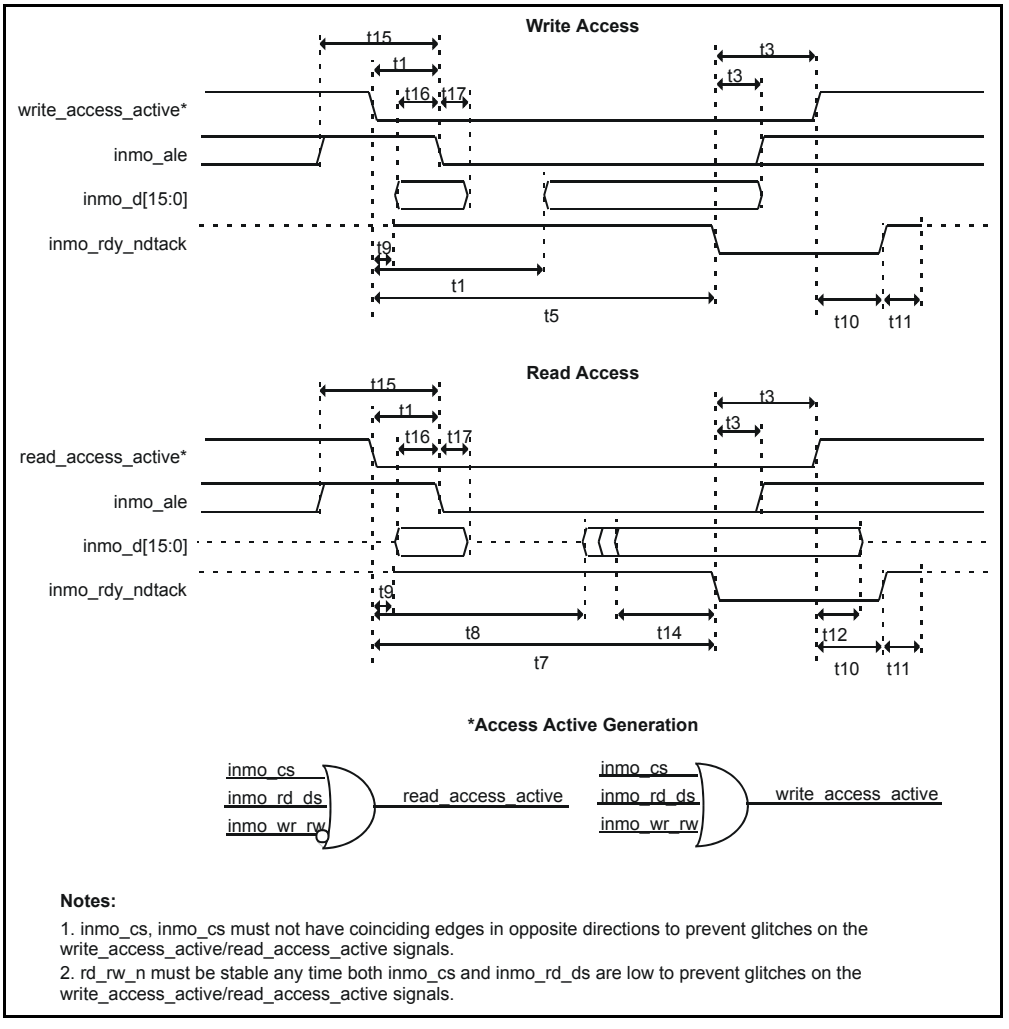


Figure 88 - Multiplexed CPU Interface - Motorola Mode

AC Characteristics - CPU Interface

Symbol	Description	Min	Typical	Max	Unit	Notes
t1	write_access_active falling edge / read_access_active falling edge to inmo_a valid (non-multiplexed) inmo_a_das valid (non-multiplexed) inmo_d valid (writes) inmo_ale fall (multiplexed)			2 * upclkp - 4	ns	

AC Characteristics - CPU Interface (continued)

Symbol	Description	Min	Typical	Max	Unit	Notes
t2	inmo_rdy_ndtack rising edge to write_access_active rising edge (writes) read_access_active rising edge (reads) inmo_ale rising edge (multiplexed) inmo_d invalid (writes) inmo_a invalid (non-multiplexed) inmo_a_das invalid (non-multiplexed)	0			ns	
t3	inmo_rdy_ndtack falling edge to write_access_active rising edge (writes) read_access_active rising edge (reads) inmo_ale rising edge (multiplexed) inmo_d invalid (writes) inmo_a invalid (non-multiplexed) inmo_a_das invalid (non-multiplexed)	0			ns	
t4	write_access_active falling edge / read_access_active falling edge to inmo_rdy_ndtack falling edge	0		8	ns	
t5	Write Access Time (all writes)		6 * upclkp	see Table 51	ns	
t6	write_access_active rising edge / read_access_active rising edge to inmo_rdy_ndtack tri-state	0		15	ns	
t7	Read Access Time (to cpureg)			6 * upclkp	ns	
t7	Read Access Time (to internal reg/mem)		see Table 52	see Table 52	ns	
t7	Read Access Time (to external memory)		see Table 52	see Table 52	ns	
t8	read_access_active falling edge to inmo_d driven	3 * upclkp - 4			ns	

AC Characteristics - CPU Interface (continued)

Symbol	Description	Min	Typical	Max	Unit	Notes
t9	read_access_active falling edge / write_access_active falling edge to inmo_rdy_ndtack driven high	0		8		
t10	write_access_active rising edge / read_access_active rising edge to inmo_rdy_ndtack rising edge	0		9		
t11	inmo_rdy_ndtack rising edge to inmo_rdy_ndtack tri-state	1.5		6		
t12	read_access_active rising edge to inmo_d tri-state	0		10	ns	
t13	inmo_d valid to inmo_rdy_ndtack rising edge	upclkp - 4			ns	
t14	inmo_d valid to inmo_rdy_ndtack falling edge	upclkp - 4			ns	
t15	inmo_ale high pulse width	5			ns	
t16	inmo_d / inmo_a /inmo_a_das valid to inmo_ale falling edge	5			ns	
t17	inmo_ale falling edge to inmo_d / inmo_a / inmo_a_das invalid	5			ns	

Symbol	Description	Max.	Unit	Test Conditions
t5	Register and Internal Memory	640	ns	Note 1
		680	ns	Note 2
t5	SSRAM	720	ns	Note 1
		760	ns	Note 2
t5	SDRAM	760	ns	Note 1
		820	ns	Note 2

Table 51 - t5 Write Access Time

Note 1: Note 1: MCLK_SAR = MCLK_NET = 100 MHz, and Upclk = 32.768 MHz
Note 2: Note 2: MCLK_SAR = MCLK_NET = 82 MHz, and Upclk = 32.768 MHz

Symbol	Description	Burst Length	Max.	Unit	Test Conditions
t7	Register	1	640	ns	Note 1
			680	ns	Note 2
		2	680	ns	Note 1
			700	ns	Note 2
		64	800	ns	Note 1
			860	ns	Note 2
		128	800	ns	Note 1
			860	ns	Note 2
t7	Internal Memory	1	640	ns	Note 1
			680	ns	Note 2
		2	700	ns	Note 1
			740	ns	Note 2
		64	1.26	us	Note 1
			1.4	us	Note 2
		128	1.26	us	Note 1
			1.4	us	Note 2
t7	MEMC_SSRAM	1	720	ns	Note 1
			760	ns	Note 2
		2	740	ns	Note 1
			860	ns	Note 2
		64	860	ns	Note 1
			980	ns	Note 2
		128	860	ns	Note 1
			1.1	us	Note 2
t7	MEMA_SSRAM	1	720	ns	Note 1
			760	ns	Note 2
		2	740	ns	Note 1
			760	ns	Note 2
		64	860	ns	Note 1
			1.26	us	Note 2
		128	860	ns	Note 1
			1.4	us	Note 2

Table 52 - t7 Read Access Time

Symbol	Description	Burst Length	Max.	Unit	Test Conditions
t7	MEMB_SSRAM	1	700	ns	Note 1
			760	ns	Note 2
		2	740	ns	Note 1
			920	ns	Note 2
		64	860	ns	Note 1
			960	ns	Note 2
		128	860	ns	Note 1
			960	ns	Note 2
t7	MEMC_SDRAM	1	760	ns	Note 1
			820	ns	Note 2
		2	800	ns	Note 1
			840	ns	Note 2
		64	940	ns	Note 1
			1.04	us	Note 2
		128	940	ns	Note 1
			1.2	us	Note 2

Table 52 - t7 Read Access Time (continued)

Note 1: MCLK_SAR = MCLK_NET = 100 MHz, and Upclk = 32.768 MHz
Note 2: MCLK_SAR = MCLK_NET = 82 MHz, and Upclk = 32.768 MHz
Note 3: in burst read, all consecutive read access time after the first read is 400 ns/per read.

13.5.2 UTOPIA / POS-PHY / Ethernet Interface

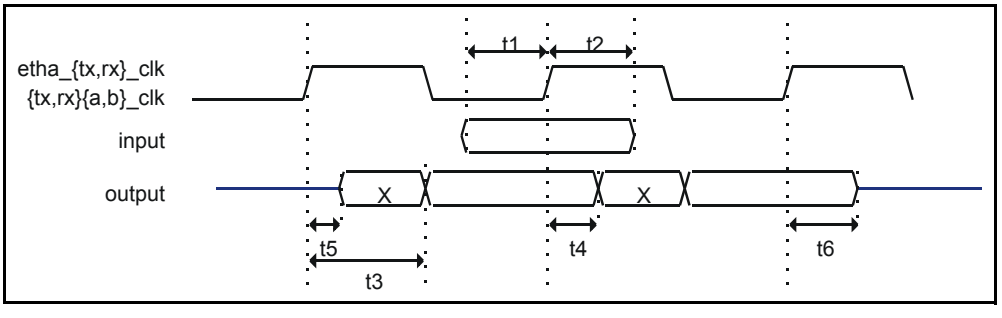


Figure 89 - UTOPIA / POS-PHY / Ethernet Timing

	Characteristics	Symbol	Min	Typ	Max	Units	Test Conditions
t1	Input setup time	t1	4			ns	
t2	Input hold time	t2	0			ns	
t3	Clock to data valid	t3			14	ns	
t4	Clock to data change	t4	2			ns	
t5	Clock rising to signal driven	t5	2			ns	
t6	Clock rising to signal tri-state	t6	1		10	ns	

Table 53 - Fields and Description

13.5.3 H.110 Interface

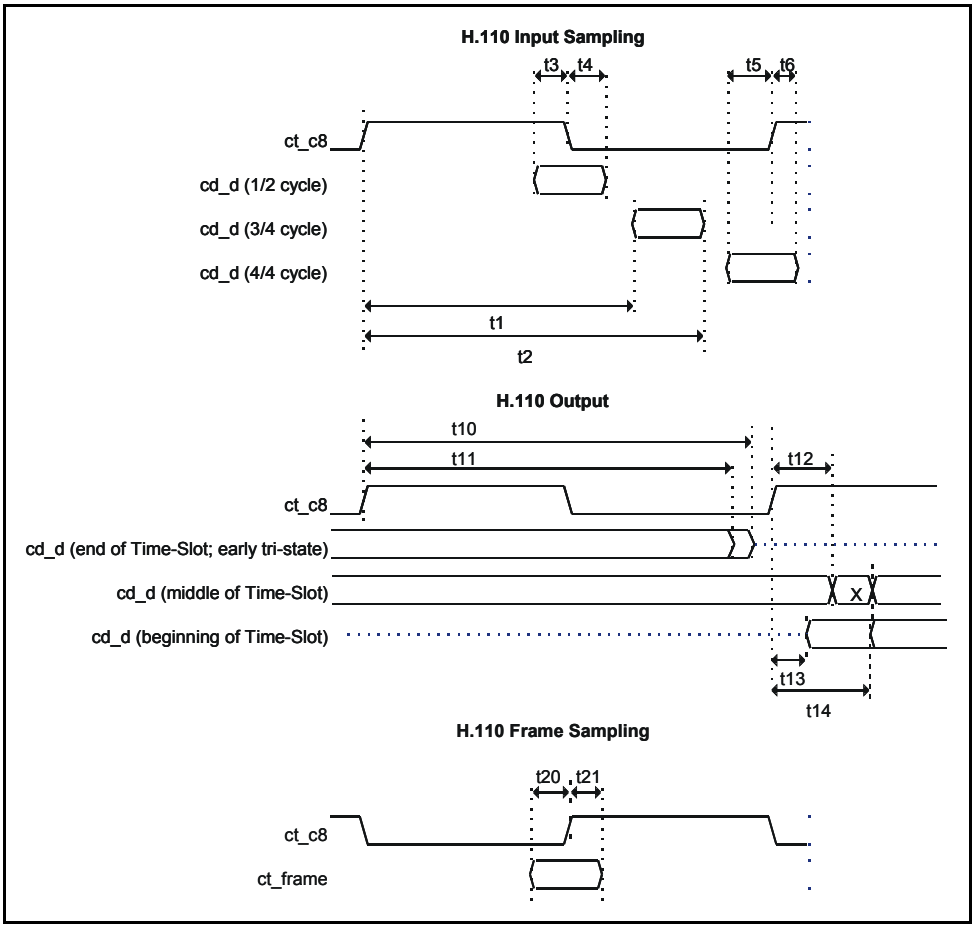


Figure 90 - H.110 Input Output

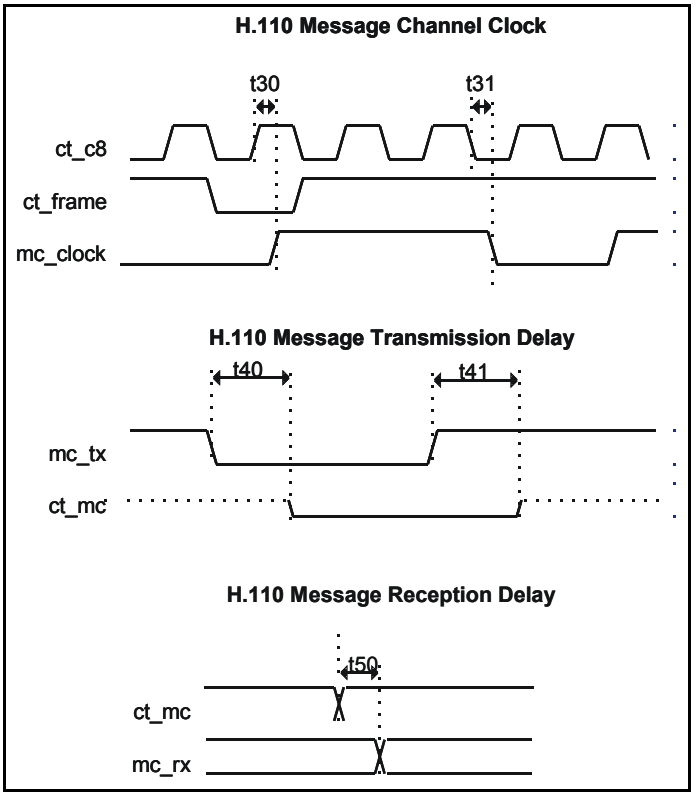


Figure 91 - H.110 Message Handling

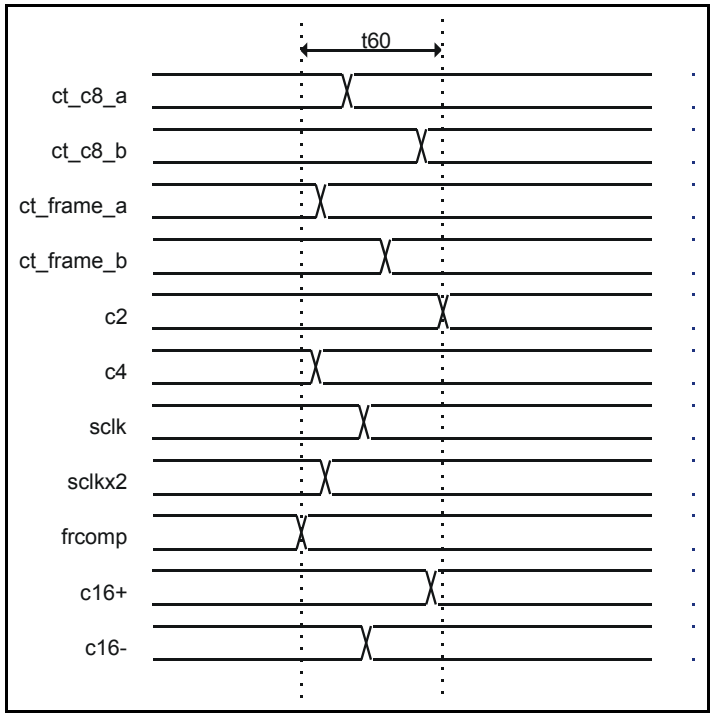


Figure 14.1 - H.110 Clock Skew (when chip is Master)

Symbol	Description	Min	Typical	Max	Unit	Notes
t1	ct_c8 rise to ct_d valid			69	ns	
t2	ct_c8 rise to ct_d invalid	102			ns	
t3	ct_d valid to ct_c8 fall	5			ns	
t4	ct_c8 fall to ct_d invalid	5			ns	
t5	ct_d valid to ct_c8 rise	5			ns	
t6	ct_c8 rise to ct_d invalid	0			ns	
t10	ct_c8 rise to ct_d tri-state			122	ns	200 pf
t11	ct_c8 rise to ct_d invalid	102			ns	200 pf
t12	ct_c8 rise to ct_d invalid	2			ns	200 pf
t13	ct_c8 rise to ct_d driven	2			ns	200 pf
t14	ct_c8 rise to ct_d valid			22	ns	200 pf
t20	ct_frame valid to ct_c8 rise	5			ns	
t21	ct_c8 rise to ct_frame invalid	5			ns	
t30	ct_c8 rise to mc_clock rise			15	ns	
t31	ct_c8 fall to mc_clock fall			15	ns	
t40	mc_tx fall to ct_mc low	3		15	ns	200 pf

Table 54 - Fields and Description

Symbol	Description	Min	Typical	Max	Unit	Notes
t41	mc_tx rise to ct_mc tri-state	3		15	ns	200 pf
t50	ct_mc fall to mc_rx fall	3		15	ns	
t60	ct_c8_a, ct_c8_b, ct_frame_a, ct_frame_b, c2, c4, sclk, sclkx2, frcomp, c16+, c16- maximum skew when generated by the chip			5	ns	200 pf

Table 54 - Fields and Description (continued)

13.5.4 External Memory Interface

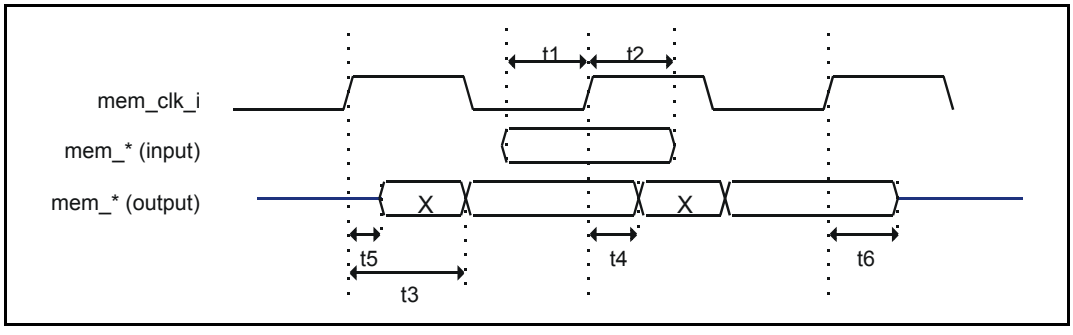


Figure 92 - External Memory Timing (both SSRAM and SDRAM)

Symbol	Description	Min	Typical	Max	Unit	Notes
t1	Input setup time	3.1			ns	
t2	Input hold time	0			ns	
t3	Clock to data valid			8.1	ns	Load = 50 pf
t4	Clock to data change	1.1			ns	Load = 50 pf
t5	Clock rising to signal driven	1.9			ns	
t6	Clock rising to signal tri-state			4	ns	

Table 55 - Fields and Description

Appendix A

Notes

1. The RX Ethernet / RX POS module (eptoatm) does not pad any packets to the end of a cell. Thus a single dword of random data may reside in parts of a packet that would otherwise have contained zero padding. This abnormality has no adverse side-effects.
2. All four memory-controllers cannot limit the throughput of watomic accesses to their respective memories. Thus, the software may be able to “sink” the external memory bandwidth and make the chip fail. To avoid this, any burst of more than 16 consecutive watomic accesses should be isolated from other watomic accesses via 3 other (non-watomic) accesses. These can be three writes to registers (on the proper clock domain). So to break up watomic accesses to memory bank A/B, writes to the mainreg can be performed; to break up watomic accesses to memory bank C, writes to the netreg can be performed.
3. The number of bearers that is programmable in the disassembly structure has a maximum value of 255 rather than 256.
4. When the external SDRAM is configured to be 16 Megabytes rather than 32 Megabytes, the auto-clear function will initialize the links to values in the range 16-32 Megabytes.
5. The SDRAM refresh process will be active during the init of the chip. Therefore, to ensure that the SDRAM init sequence is executed without any refresh instructions being inserted, the **sdram_refresh_cnt** must be changed from a small value (e.g. 0x100) to 0xFFFF. This will create 65000 **mclk_net** cycles during which no refresh instructions will be executed. Once the init sequence is complete, the **sdram_refresh_cnt** must be returned to its valid value. A precise timer must be used to ensure that the sequence was executed within the time budget.
6. The packet disassembly module’s extended PDV monitoring section analyzes the delay of packets in the negative direction instead of the positive direction, which means that a “late” packet will obtain a very small delta, while an “early” packet will obtain a very large delta. Because of this, the exponential section of the delay entries is located in the delay section in which early packet will arrive. In addition, the **time_zero_delta** field must be programmed to the total number of frames of latency with which the latest packet can be expected to arrive (from source to destination).

Appendix B

HDLC Format, Including Zero-Insertion and Extraction

The MT92210 supports 2 types of HDLC over the TDM bus: the first, bit-wise form of HDLC uses a control flag of “01111110” and inserts a '0' after every 5 '1's of payload. When using this form of HDLC, each HDLC packet must begin with a flag and end with a flag, although a single flag may represent both the end of a packet and the beginning of another. If neither flags nor data are being transmitted onto the bus, the idle code must be used: it is simply an endless string of '1's. Note that the idle code must be at least 7 bits long (7 '1's) to be valid

The second form is a byte-wise HDLC format, which also uses “01111110” (7Eh) as a control flag. An actual 7Eh within the payload is replaced by 7Dh - 5Eh, while a 7Dh is replaced by 7Dh - 5Dh. On the SONET interface, a 7Dh - 7Eh control code indicate the abortion of the packet. This form of HDLC is easier in terms of computation (because it looks at each byte individually instead of each bit) and thus is easier for neighboring DSP to use. It does not contain an idle code: instead, the flag character is repeated endlessly until valid data is ready to be transmitted onto the bus.

In both cases, the MT92210 can accept or generate an HDLC header that can contain 0, 1 or 2 address bytes, as well as a possible control byte. There is also an optional 16-bit CRC that may be added at the end of the packet. When using HDLC streams, the low 9 bits of the address are used to select the channel number. Finally, the payload of the mini-packet may range from 1 to 1500 bytes. Note that the payload of the mini-packet may include an RTP header.

In RTP HDLC format, a complete RTP packet including RTP header is encapsulated.

Figure 93 - Supported RTP HDLC Packet Format (after zero extraction)

1. Address bytes can be 0, 1 or 2 byte(s).
2. Control byte is optional.
3. CRC bytes are optional
4. Complete RTP packet starts from D0

Appendix C

Standards & Specifications

The MT92210 is designed to conform to sections of the following standards and specifications. Some of these specifications define requirements that can only be implemented in software. Some of the specifications are “umbrella” specifications to which the MT92210 only complies to portions of.

ECTF

H.100 Revision 1.0 Hardware Compatibility Specification: CT Bus

H.110 Revision 1.0 Hardware Compatibility Specification: CT Bus

IEEE

802.3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications

IETF

RFC768 (also STD6): User Datagram Protocol

RFC791 (also STD5): Internet Protocol Version 4

RFC826 (also STD37): Ethernet Address Resolution Protocol

RFC894 (also STD41): Standard for the transmission of IP datagrams over Ethernet networks

RFC1122 (also STD3): Requirements for Internet Hosts

RFC1123 (also STD3): Requirements for Internet Hosts

RFC1661 (also STD51): The Point-to-Point Protocol

RFC1662 (also STD51): The Point-to-Point Protocol

RFC1889: Real-Time Protocol

RFC1890: Real-Time Protocol

RFC2225: Classical IP and ARP over ATM

RFC2460: Internet Protocol Version 6

RFC2464: Transmission of IPv6 Packets over Ethernet Networks

RFC2684: Multi-protocol Encapsulation over ATM AAL5

Draft (draft-ietf-mpls-label-encaps-07): MPLS Label Stack Encoding

ITU

I.363.5: ATM Adaptation Layer 5

ATM Forum

Af-lane-0021.000: LAN Emulation over ATM version 1.0

Af-lane-0084.000: LAN Emulation over ATM version 2

Af-lane-0112.000: LAN Emulation over ATM version 2

Af-mpoa-0114.000: Multi-protocol over ATM version 1.1

Af-phy-0017.000: UTOPIA Specification Level 1, version 2.01

Af-phy-0039.000: UTOPIA Level 2, version 1.0

PMC-Sierra

POS-PHY: Saturn Compatible Packet over SONET Interface Specification (Level 2)

Appendix D

Glossary of Terms

Standard Terms and Abbreviations

AAL0 (ATM **A**daptation **L**ayer **0):** AAL0 is a straight packaging of 48 bytes of data within an ATM cell. AAL0 is often used to carry signaling ATM cells, which are treated in this device as “raw cells”.

AAL5 (ATM **A**daptation **L**ayer **5):** AAL5 is a protocol used to carry higher-layer datagrams while enhancing the link layer with services available through ATM. Defined in the ITU standard I.363.5, AAL5 is typically used to carry IP datagrams over ATM, but can be used for other higher-layer protocols.

ADPCM (Adaptive **D**ifferential **P**ulse **C**ode **M**odulation**):** ADPCM is a compression standard that allows the encoding of PCM data at rates of 40, 32, 24 and 16 kbps. Defined by the ITU standard G.726. ADPCM running at 32 kbps is often used as the definition of “toll” quality, or quality that is comparable or superior to that of the PSTN today.

Application data: The “data” unit carried by the transport layer. This will typically be the UDP payload in a voice-over-IP implementation, though it could be TCP payload or others for other applications.

ARP (Address **R**esolution **P**rotocol**):** A protocol designed for Ethernet that allows the MAC address associated to a network address (usually an IP address) to be known so the packet can be sent over Ethernet. Defined by IETF RFC826 & STD37.

CLIP (CLassical **IP** Over ATM**):** Encapsulation of IP packets over AAL5 using LLC and SNAP header to identify the network protocol (in this case IP) being used.

CPU (Central **P**rocessing **U**nit**)**

CRC (Cyclic **R**edundancy **C**heck**):** The CRC is a method of error detection and correction that is applied to a certain field of data. CRC is an efficient method of error detection because the odds of erroneously detecting a correct payload are low.

Datagram: A logical entity containing an IP header and the IP message contained within. IP protocols communicate through datagrams, but these are sometimes fragmented on links. Datagrams are sent as packets on the link layer, and a datagram may be sent as one or many packets.

FIFO (First **I**n, **F**irst **O**ut**):** A FIFO memory is one in which the first byte to have been written into the memory is the first one to be read from the read port.

Frame: A unit of data transported on a link layer. In Ethernet, for example, a frame would contain the MAC header, the IP (or other) packet within, and the Ethernet trailer.

H.110: A TDM bus standard developed by ECTF to provide backward compatibility to existing TDM busses with more bandwidth and potential for development. H.100 has a total bandwidth of 256 Mbps. On a compact PCI platform, the name H.110 is used for this bus, which keeps the same logical characteristics but different electrical ones.

HDLC (High-level **D**ata **L**ink **C**ontrol**):** An encapsulation protocol that defines specific bit patterns as delimiters and thus allows transmission of data over a serial link. In the MT92210, HDLC is used to carry variable-length packets on the H.110 bus.

IP (Internet **P**rotocol**):** Designed for use in interconnected systems of packet-switched computer communication networks, IP is the ubiquitous protocol on which the Internet runs. Logically, two machines communicate through IP datagrams, which are then sent over the link layer as packets. Runs on top of link layers like Ethernet, Packet over SONET or ATM. Defined in IETF RFC791 & STD5.

LANE (Local **A**rea **N**etwork **E**mulation**):** LANE is a method for emulating Ethernet behavior over ATM AAL5. It takes over the behavior of the MAC layer in Ethernet networks.

LLC (Logical **L**ink **C**ontrol**):** The LLC method allows multiplexing of multiple protocols over a single ATM VC. LLC headers are 3 bytes.

MAC (Media **A**ccess **C**ontrol**):** The MAC layer is concerned with the control of access to a medium shared between two or more entities. It is a control layer for Ethernet.

OC-3 (Optical **C**arrier level-**3):** A SONET channel that carries a bandwidth of 155.55 Mbps.

Packet: The “data” unit carried on a link layer. A packet, on an IP network, consists of an entire IP datagram or a fragment of a datagram.

PCM (Pulse **C**ode **M**odulation**):** PCM is the basic method of encoding an analog voice signal into digital form using 8-bit samples. Defined by the ITU standard G.711.

PHY (PHYsical layer**)**

PLL (Phase **L**ock **L**oop**):** A phase lock loop is a component that generates an output clock by synchronizing itself to an input clock. PLLs are often used to multiply the frequency of clocks.

POS (Packet **O**ver **SONET):** A means of transporting packets over a SONET link with minimal overhead in a point-to-point connection. POS uses PPP as its link protocol.

POS-PHY: A bus standard for connecting **P**acket **O**ver **SONET** link layer devices to **PHY**sical layer ones. POS-PHY level 2 was based loosely on UTOPIA level 2.

POTS (Plain **O**ld **T**elephone **S**ervice**):** The ubiquitous, 64 kbps phone service widely deployed in today's phone networks.

PPP (Point-to-**P**oint **P**rotocol**):** A link protocol that allows for transport of many network protocols over a point-to-point link. PPP has very little overhead (1 or 2 bytes per packet), making it very attractive for some applications.

RAM (Random **A**ccess **M**emory**):** RAM is the main memory in the computer. It is called “random” because any random address can be accessed in an equal amount of time.

RTCP (Real-**T**ime **C**ontrol **P**rotocol**):** The control protocol for RTP, RTCP is used for control and diagnostic on RTP sessions. Like RTP, RTCP typically runs on top of UDP and is defined in the IETF RFC1889.

RTP (Real-**T**ime **P**rotocol**):** A transport protocol designed to provide end-to-end delivery services for data with real-time characteristics. RTP typically runs on top of UDP. Defined in the IETF RFC1889.

SNAP (Sub **N**etwork **A**ccess **P**rotocol**):** A SNAP header consists of 5 bytes, three bytes of OUI (Organizationally Unique Identifier) and 2 bytes of PID (Protocol IDentifier). The OUI defines which organization administers the PID that follows. The value of 000000h in the OUI means that the PID is defined as an EtherType.

TCP (Transmission **C**ontrol **P**rotocol**):** A transport layer, TCP is a highly reliable host-to-host protocol that guarantees packet delivery, non-duplicated and in order. TCP runs on top of IP. Defined in IETF RFC791 & STD7.

TDM (Time **D**ivision **M**ultiplexing**):** TDM busses carry voice data divided according to frames. In a single 125 us frame, the TDM bus will have carried one byte from each channel it contains.

UDP (User **D**atagram **P**rotocol**):** A transport layer, UDP provides a protocol for applications to communicate with a minimum of overhead. UDP does not guarantee packet delivery or ordering. UDP runs on top of IP. Defined in IETF RFC768 & STD6.

UTOPIA (Universal **T**est and **OP**erations **I**nterface for **A**TM**):** The electrical interface on which ATM cells are passed.

VC (Virtual **C**ircuit**):** VC define a point-to-point connection between two nodes in a network. A single ATM cell carries data that belongs to a single VC.

VCI (Virtual **C**hannel **I**dentifier**):** This is the label given to an ATM VC to identify it and determine its destination. The VCI is a 16-bit number that is included in the header of an ATM cell.

VPI (Virtual **P**ath **I**dentifier**):** A virtual path determines the way an ATM cell should be routed. The VPI is an 8-bit (in UNI) or 12-bit (in NNI) number that is included in the header of an ATM cell.

WATOMIC: An uninterruptable Write operation.

Terms Specific to This Specification

Bearer: A unidirectional xxPCM stream. In PCM A-law or u-law, a single bearer will have a bandwidth of 64 kbps; this bandwidth will be reduced in ADPCM. In this device, a single PCM channel may interleave many bearers.

Channel: A data stream of a given nature mapped over a connection is considered a channel. For example, a PCM data stream mapped over an RTP connection would be a single channel, even if that data stream interleaved more than 1 bearer in each packet. In like manner, any set of payload types that are all destined to the same HDLC endpoint would also be considered a channel.

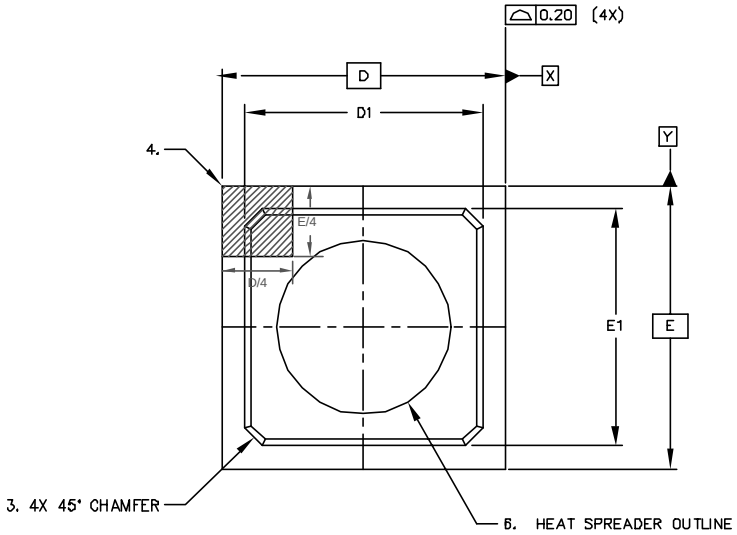
Connection: A link between two end-points that is unique through some look-up method, either through source & destination IP addresses and UDP ports, through those plus RTP SSRC, or through ATM VC number and possibly RTP SSRC on Null-application data VC.

HDLC Stream: A group of HDLC channels that are carried over the same time slots. HDLC mini-packets in streams have an address byte that indicates to which HDLC channel they belong. Usually, HDLC streams carry a series of channels communicating to and from the same agent (e.g. a DSP). HDLC Streams may be carried over one or multiple consecutive time-slots on the H.100 bus. This device supports streams of length 1 to 2046 time slots.

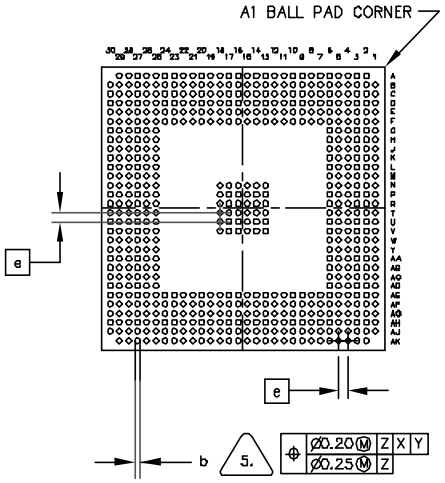
PDV: Packet **D**elay **V**ariation. RTP packets arrive with a certain delay with respect to when they were sent. PDV is a measure of how much that delay varies on an xxPCM channel. PDV measures the peak-to-peak packet delay throughout the network. PDV is only relevant on CBR connections.

Time Slot: In this document, the term time slot is often used to define a combination of a time slot and a stream on the H.100 bus. Thus a time slot would represent a single 8-bit slot every 125 us on the TDM bus. Time slot/Stream numbers are numbered 0 to 4095 according to the following equation: time slot * 32 + stream. On reduced-frequency TDM streams, certain time slots become unusable. For streams running on a 4 MHz clock, time slots are numbered 0 to 63, and the equations to determine TSSTs are the following: in the TX TDM, TSST = (time slot * 2 + 1) * 32 + stream, and in the RX TDM, TSST = (time slot * 2) * 32 + stream. In like manner, for streams running on a 2 MHz clock, time slots are numbered 0 to 31, and the equations are: in the TX TDM, TSST = (time slot * 4 + 3) * 32 + stream, and in the RX TDM, TSST = (time slot * 4) * 32 + stream.

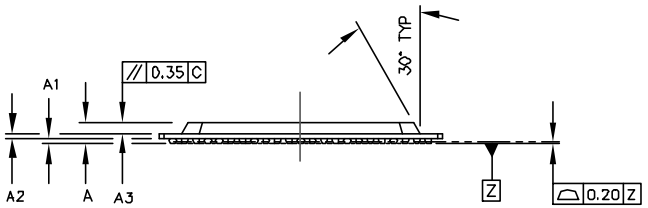
“Null” VC: A Null VC contains data in which some or all of the headers have been encoded into the VC number itself. For example, a Null-application data VC's payload would begin with the application data itself (either RTP or the payload) with no IP or UDP header, and no SNAP/LLC, LANE or other headers. Null encapsulation is referred to in IETF RFC2684 as VC Multiplexing.



TOP VIEW



BOTTOM VIEW



SIDE VIEW

DIMENSION	MIN	NOM	MAX
A	---	---	2.50
A1	0.40	---	---
A2	0.25	---	0.70
A3	---	---	1.25
D		31.00 BSC	
D1	25.00	---	27.00
E		31.00 BSC	
E1	25.00	---	27.00
b	0.50	---	0.70
e		1.00 BSC	
F	20.50	---	22.50
N		608	

Conforms to JEDEC MS - 034 Rev. A

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994.
2. ALLDIMENSIONS ARE IN MM.
3. CORNER DETAIL IS LSI LOGIC CORP OPTION.
4. DETAILS OF PIN 1 IDENTIFIER ARE OPTIONAL, AND MAY CONSIST OF INK, LASER MARK OR METALLIZED MARKING, BUT MUST BE LOCATED WITHIN THE INDICATED ZIONE.
5. PRIMARY DATUM Z AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
6. HEAT SPREADER OUTLINE.

ISSUE	1			
ACN	213960			
DATE	29Jan03			
APPRD.				

ZARLINK SEMICONDUCTOR

Package Code GK

Previous package codes: BH

Package Outline for 608 Ball EPBGA (31x31x2.50mm)

GPD00823

For more information about all Zarlink products visit our Web Site at

www.zarlink.com