MT90880/1/2/3

TDM to Packet Processors

Data Sheet

January 2003

Features

- WAN interface, consisting of 32 input and output streams at 2.048 or 8.192 Mbit/s
- Up to 1024 bi-directional 64 Kbit/s channels
- N * 64 Kbit/s trunking of channels across any stream and channel
- 1K by 1K non-blocking TDM switch
- Local TDM interface, with 32 streams at 2.048, 4.096 and 8.192 Mbit/s
- Flexible, multi-protocol packet encapsulation
- Dual 100Mbit/s MII interfaces for redundancy or for load balancing
- Quality of service features, including weighted fair queuing, strict priority and queue size limit thresholds
- High performance 33MHz / 66MHz 32 bit PCI bus
- Integral Stratum 4E PLL for synchronisation to the TDM domain
- Power consumption of less than 0.75 W

Applications

- Packet backplane interconnection
- Circuit Emulation over packet domain
- Internet Off-load
- Remote Access Concentrators
- H.100/H.110 extension and expansion

Ordering Information

MT90880B/IG/BP1N	456 ball PBGA
MT90881A/IG/BP1N	456 ball PBGA
MT90882B/IG/BP1N	456 ball PBGA
MT90883A/IG/BP1N	456 ball PBGA

-40 to 85 ^{0}C

Description

The MT9088x is a family of highly functional TDM to Packet bridging devices. It provides a bridge between a WAN environment based on constant bit rate TDM streams and a packet domain based on Ethernet technology.

It is capable of assembling user-defined packets of TDM traffic from the WAN Access Interface and transmitting them from the Ethernet interfaces using a variety of protocols. If external processing is required (e.g. HDLC or modem termination) the traffic can be switched to a local interface using the internal TDM switch.

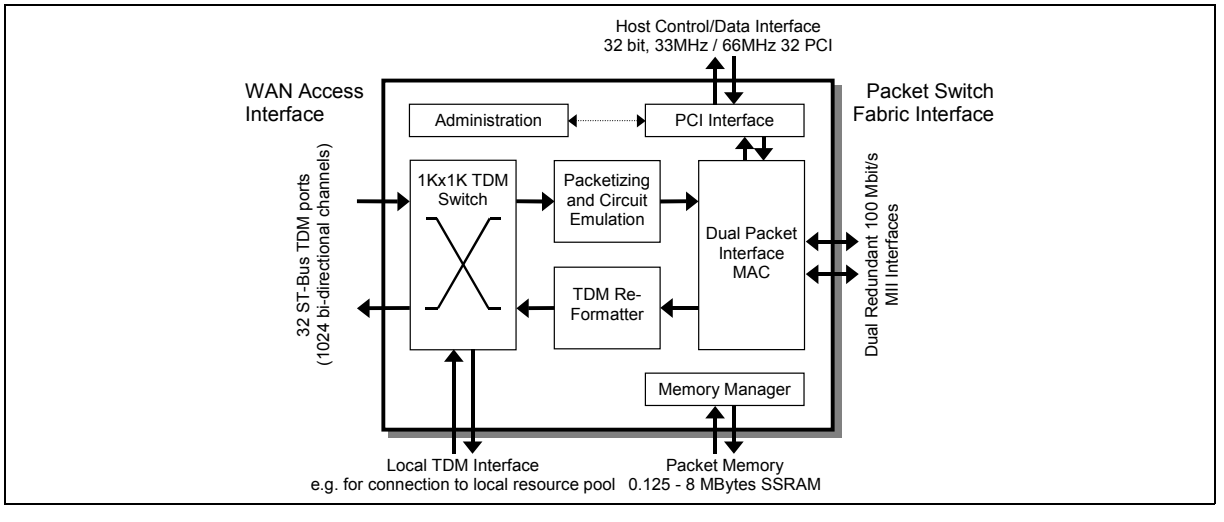

Figure 1 - MT90880 High Level Overview

Packets received from the Ethernet interfaces are parsed to determine the egress destination, and are appropriately queued either to the WAN Access Interface or to the PCI interface. An integrated DMA controller is used to transfer packets to and from the PCI interface with a minimum of CPU intervention.

Variants

There are four device variants in the MT9088x family:

MT90880 1024 bi-directional channels, integral TDM switch

MT90881 1024 bi-directional channels, no TDM switch

MT90882 256 bi-directional channels, integral TDM switch

MT90883 256 bi-directional channels, no TDM switch

Feature	MT90880	MT90881	MT90882	MT90883
No. of WAN streams	32	32	8	8
Local Port	Yes	No	Yes	No
No. of Available Channels	1024	1024	256	256
TDM Switch Availability	Master mode only	Not Available	Master mode only	Not Available
PCI Interface	33 / 66MHz	33 / 66MHz	33 / 66MHz	33 / 66MHz
MII Interface	Yes	Yes	Yes	Yes
RMII Interface	Yes	Yes	Yes	Yes
JTAG	Yes	Yes	Yes	Yes

Table 1 - Variant Options

Related Documents

This data sheet should be read in conjunction with the following related documents and application notes:

Title	Author	Document Number	Issue / Date
1. MT9088x Programmers Model	Zarlink	DM5708	1.0, Aug. 2002
2. MT9088x API User Guide	Zarlink	DM5805	1.0, Sept. 2002
3. MSAN-198 - Performing Clock Recovery for Circuit Emulation when using the MT90880	Zarlink	AN5789	1, Aug. 2002
4. MSAN-199 - Unstructured Circuit Emulation Using the MT90880	Zarlink	AN5790	1, Aug. 2002
5. MSAN-200 - MT90880 TDM Replacement Packet Backplane	Zarlink	AN5791	1, Aug. 2002.

Table 2 - Related Documents

The following external documents and standards are referenced in this data sheet:

Title	Author	Document Number	Issue / Date
1. Local and Metropolitan Area Networks, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications	IEEE	IEEE 802.3u	1995
2. PCI Local Bus Specification	PCI SIG		2.2
3. RMII TM Specification	RMII consortium		Rev 1.2, March 1998
4. Test Access Port and Boundary Scan Architecture	IEEE	IEEE 1149.1	1990
5. Circuit Emulation Service Interoperability Specification	ATM Forum	af-vtoa-0078	Ver 2.0, Jan. 1997
6. Specifications of (DBCES) Dynamic Bandwidth Utilization - in 64KBPS Time Slot Trunking over ATM - Using CES	ATM Forum	af-vtoa-0085	July 1997
7. ST-BUS Generic Device Specification	Mitel	MSAN-126	Rev. B, June 1995
8. H.110 Hardware Compatibility Specification: CT Bus	ECTF		Rev. 1.0, 1997
9. H-MVIP Standard	GO-MVIP		Rel. 1.1a, Jan. '97
10. Clocks for the Synchronized Network: Common Generic Criteria	Telcordia	GR-1244-CORE	Iss. 2, Dec. 2000
11. The Control of Jitter and Wander within digital networks which are based on the 2048 Kbit/s hierarchy	ITU-T	G.823	2000
12. The Control of Jitter and Wander within digital networks which are based on the 1544 Kbit/s hierarchy	ITU-T	G.824	2000

Table 3 - Referenced Documents

Table of Contents

List of Figures

List of Tables

1.0 Typical Applications

1.1 Packet Backplane Interconnection

The MT9088x family can be used to entirely replace the TDM backplane infrastructure in a conventional computer telephony system with a packet backplane. This has several advantages: it is easily scalable, eliminates the timing problems in passing large TDM buses around the system, uses readily available and low-cost network hardware, and simplifies the provision of new services to customers. This type of structure can be used in applications as diverse as telephone switches, multi-service access platforms and voice over IP gateways. Figure 2 shows a multi-service access platform based on an Ethernet backplane using the MT90880.

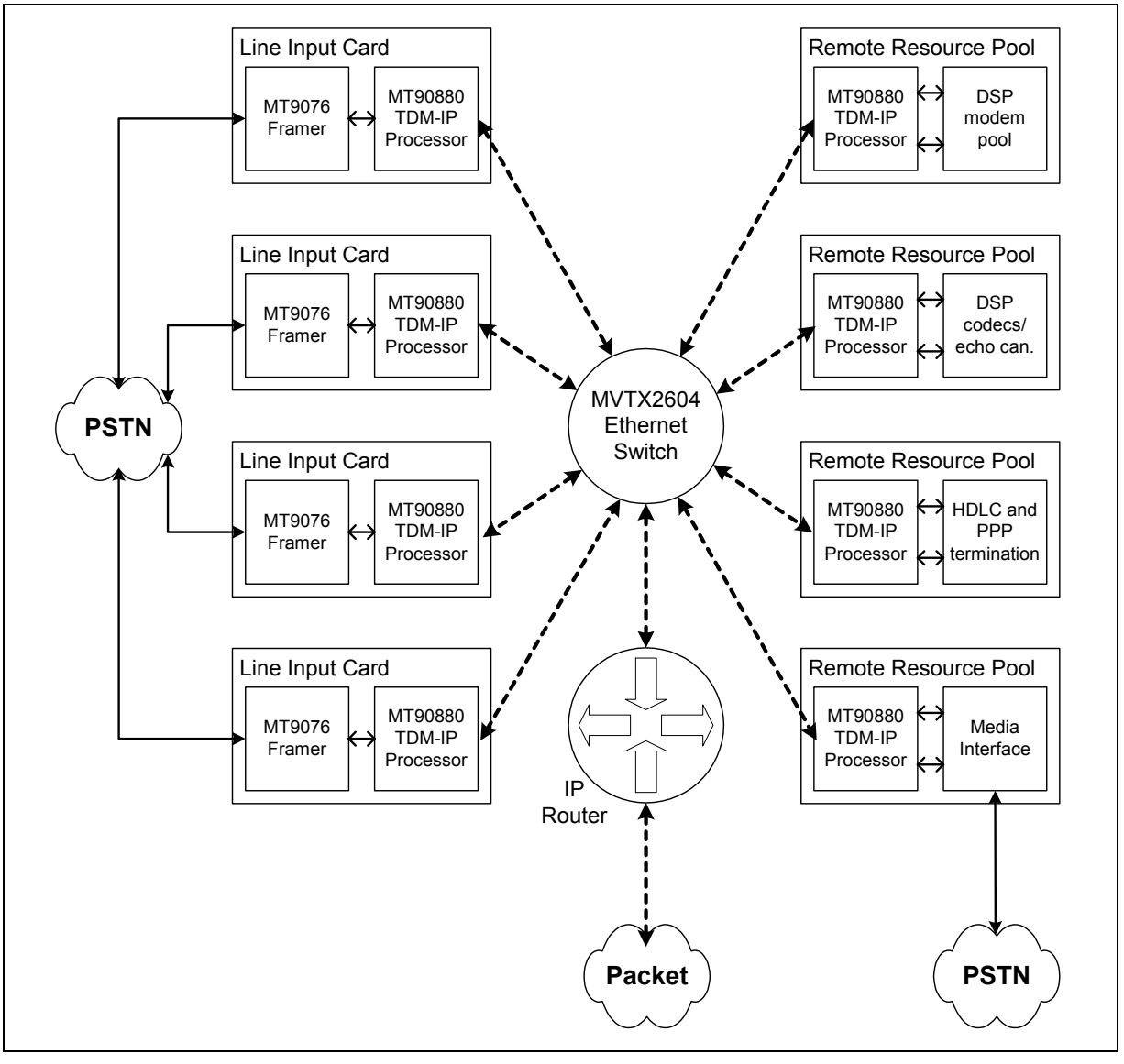

Figure 2 - Multiservice Access Platform using the MT90880

1.2 Circuit Emulation Services

The MT9088x family can be used to transport TDM links across the packet domain and transparently reconstruct the links at the far end (Figure 3). This is similar to the circuit emulation services defined by the ATM Forum, with the MT9088x device providing the circuit emulation inter-working function.

The part is capable of handling circuit emulation of both structured and unstructured DS1 (T1) and E1 links. The use of an Ethernet packet infrastructure for this facilitates the extension of TDM services between locations using easy to manage and readily available network hardware.

The part supports dynamic bandwidth management, by providing the ability to add and delete timeslots from the packet stream as required. A cross-connect service can also be provided by using an Ethernet switch to route packet streams to different destinations.

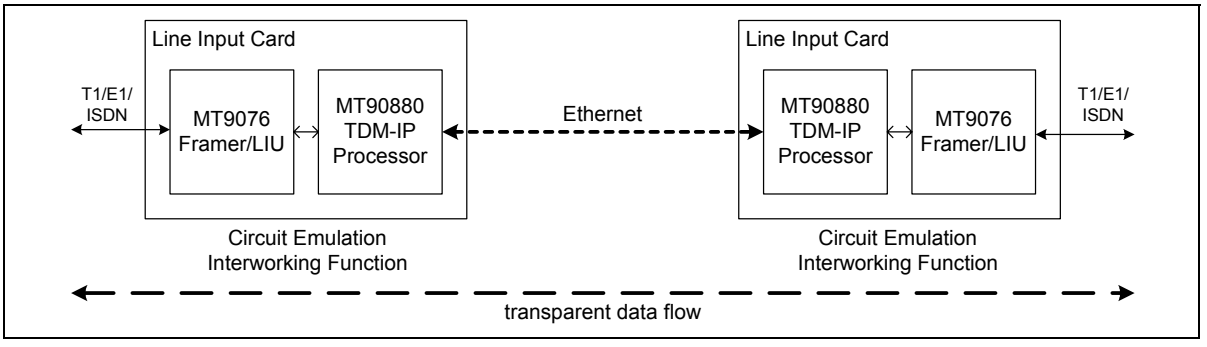

Figure 3 - Circuit Emulation Services over the packet network

1.3 Internet Off-load

Internet off-load is a means of relieving the congestion in circuit-switching equipment caused by excessive dial-up modem traffic with long call duration. Modem traffic is directed to the packet-switched network, bypassing traditional voice switches. This reduces the capacity of the required circuit switches, and hence the cost of a typical installation.

The MT90880 is an ideal device for implementing internet off-load, allowing voice traffic to be routed through the TDM switch to the voice switching infrastructure, and modem traffic to be sent over the packet network to the Remote Access Concentrator (Figure 4). The RAC could be local or remote, with traffic routed over the data networks.

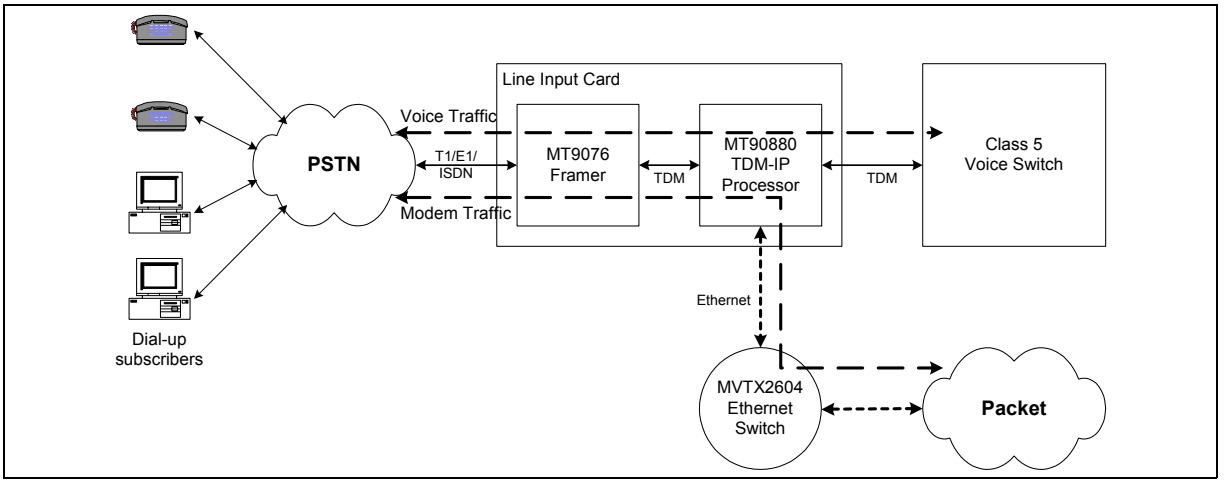

Figure 4 - Internet Off-load solution using the MT90880

1.4 Remote Access Concentration

In a similar manner to the internet off-load solution outlined above, the MT9088x devices can also be used to route traffic to a remote resource pool. This allows resources to be centralized, with multiple line cards accessing the resource pool via the Ethernet switching fabric. A typical application for this is a remote access concentrator for dial-up modem traffic, and Figure 5 shows an example of this.

The MT90880 on the line card forms the modem data into packets, which are transmitted via the Ethernet switch to a remote DSP pool. The MT90880 on the remote card converts the packets back to TDM and forwards them to the DSP pool over the TDM link. The demodulated data is then transmitted back over the packet network to the router via the MT90880.

This packet-based architecture can be easily extended to support large numbers of modem calls by simply adding more Ethernet switch interfaces. The architecture can also support ISDN PPP traffic. The DSP pool strips off the HDLC framing and forwards the packets to the MT90880 for transmission over the IP packet network.

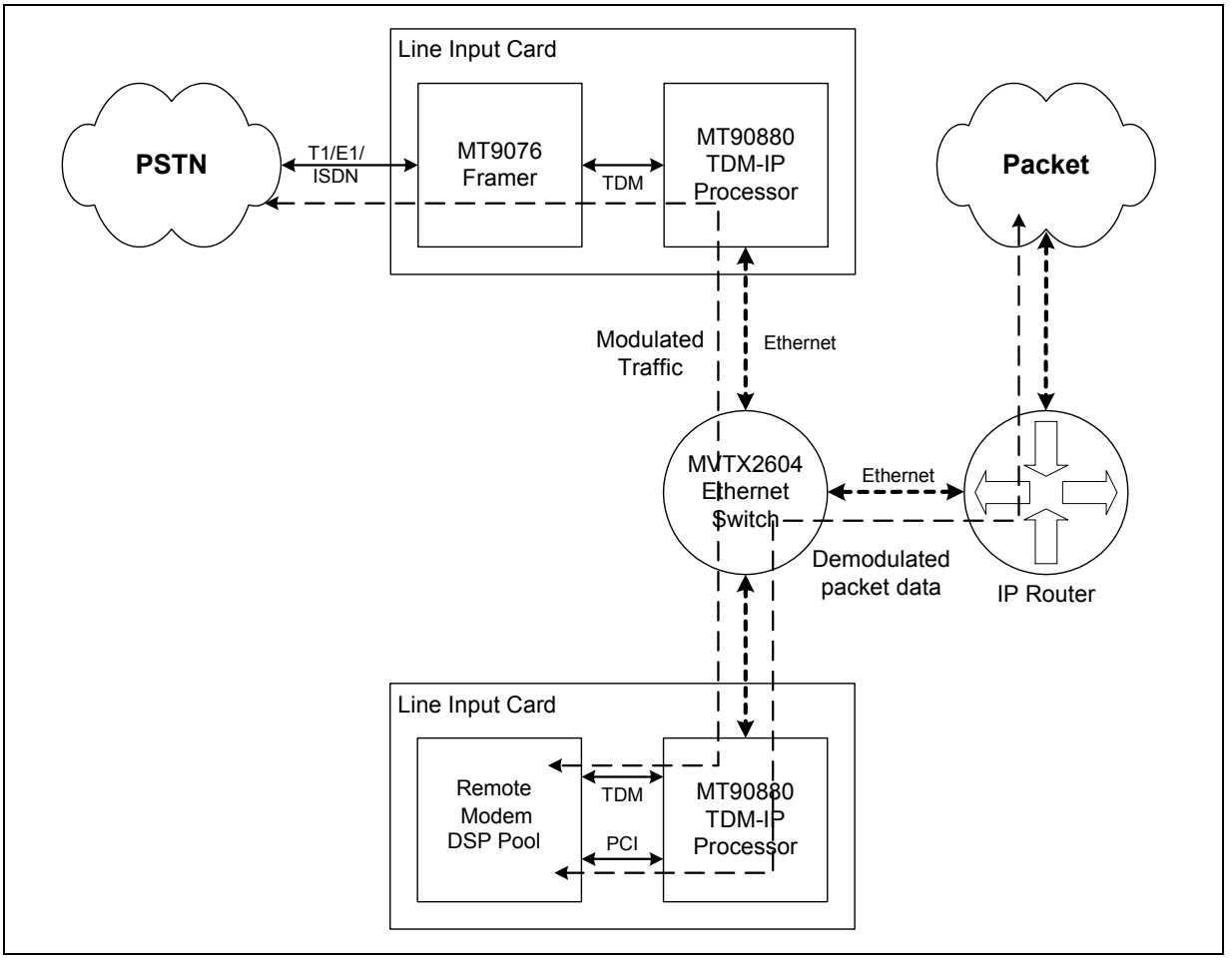

Figure 5 - Remote Access Concentrator using the MT90880

1.5 Local Resource Pool Example

For smaller systems, the built-in TDM switch allows traffic to be switched out to a local resource pool on the line card. The provision of a high bandwidth, 33 MHz PCI interface allows the processed data to be returned to the MT90880 ready for transmission over the network (Figure 6). Alternatively, the Local TDM interface can be used to return data to the MT90880 for transmission across the packet network.

The PCI interface on the MT90880 contains a DMA controller to simplify packet handling over the PCI bus. This can read packets out of the system memory on the PCI bus and place them on the appropriate queue for transmission.

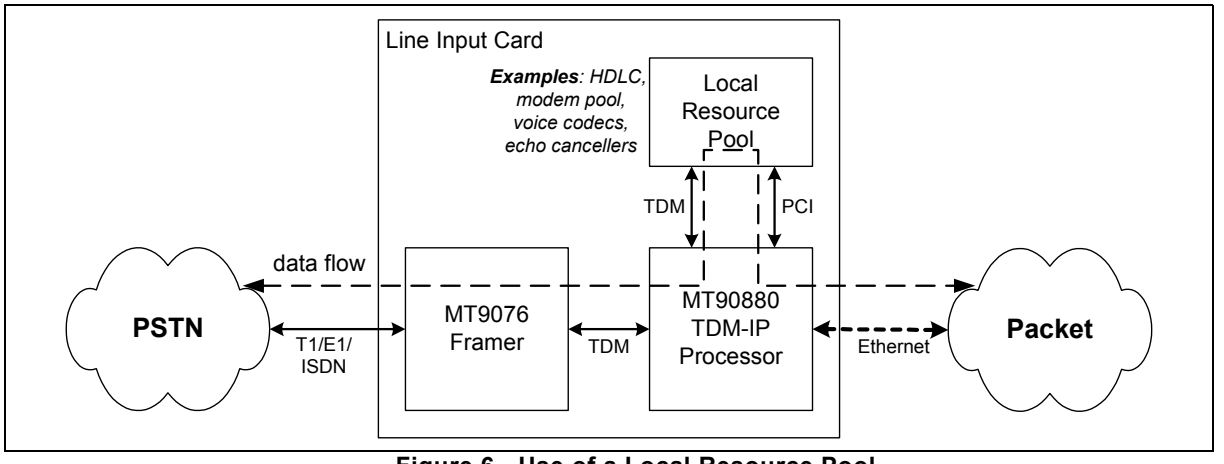

Figure 6 - Use of a Local Resource Pool

1.6 H.100/H.110 Extension

The H.100/H.110 TDM bus commonly used in today's computer telephony systems is based on a physical backplane the width of a single telecom rack. Extending the reach of the bus is expensive using traditional TDM infrastructure. Such links are not easily scalable, and require accurate and stable clock generation.

An MT9088x device enables the bus to be simply and easily extended beyond confines of the rack using an Ethernet connection over simple twisted pair cable. This enables the entire bus to be replicated in another physical location using a low cost but flexible and easily managed connection medium (see Figure 7).

Figure 7 shows the MT9088x device connected to a TDM backplane via a TDM switch device (e.g. the Zarlink MT90866). This is used to concentrate the backplane onto the TDM interface of the MT9088x. The configuration allows any of the TDM channels on the original backplaneto be switched onto any channel on the extension backplane, and vice-versa.

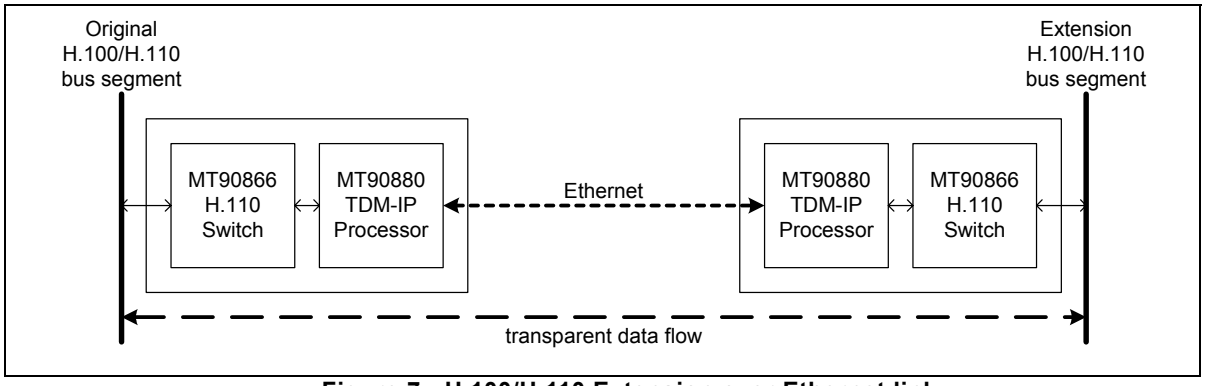

Figure 7 - H.100/H.110 Extension over Ethernet link

1.7 H.100/H.110 Expansion

One of the issues faced by medium and high-end telecommunication systems is scalability. The H.100/H.110 TDM bus is limited to 4096 concurrent timeslots, or 2048 full duplex links. The MT9088x family can be used to expand the capacity of a system by switching timeslots between multiple separate H.100/H.110 bus segments.

This application is shown in Figure 8. Unlike parts based on expensive or proprietary infrastructure, the use of an Ethernet switch fabric enables common, readily available network hardware to be employed, reducing both installation and operational costs.

As in Figure 7, the MT90866 TDM switch is used to concentrate the TDM backplane onto the MT9088x TDM interface. This allows any channel on a bus segment to be switched onto any channel on any other bus segment.

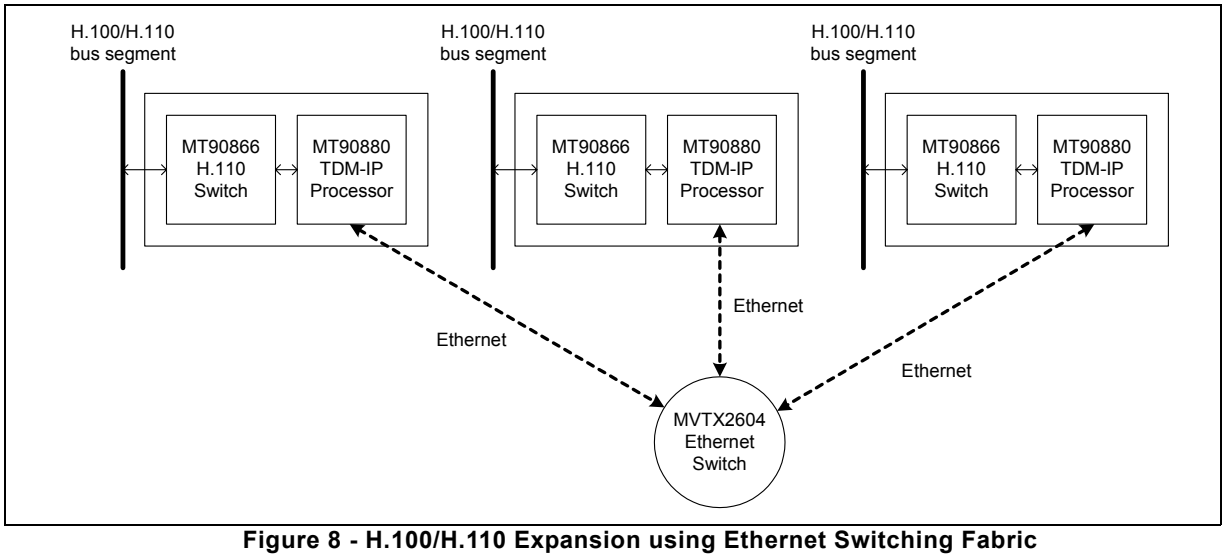

Figure 8 - H.100/H.110 Expansion using Ethernet Switching Fabric

2.0 Functional Operation

2.1 Overview

The MT9088x family provides the data-plane processing to enable constant bit rate TDM services to be carried over a packet network, such as an Ethernet or IP network. The device segments the TDM data into user-defined packets, and passes it transparently over the packet network to be reconstructed at the far end. This has a number of applications, including packet backplanes for TDM-based equipment, and emulation of TDM circuits (similar to the circuit emulation services for ATM, see references 5 and 6).

The advantages of using a packet medium for transport and distribution of TDM services include:

Scalability	TDM-based systems are hard to scale in capacity for two main reasons: • TDM clock frequency increases in proportion to the number of input channels, and this high frequency clock must be distributed around the system while maintaining accurate phase alignment • Physical limitations - the form factor of hard backplane systems effectively restrict the TDM backplane to a single rack, limiting the size of the system that can be connected
Efficiency	TDM equipment dedicates capacity to all channels, whether in use or not. Use of a packet-based system allows the available bandwidth to be statistically multiplexed, taking advantage of the fact that not all channels are active all the time.
Service provisioning	It is easier to add new services by adding new line input cards or resource cards. These are not constrained to be in the same physical rack, as with a TDM backplane system.
Cost	Network hardware is readily available at lower cost than equivalent TDM equipment, reducing the overall cost of ownership.
Physical distribution	Equipment may be physically distributed over a wider area, connected either by simple twisted pair network cable, or by fibre. This allows the footprint for co-located equipment (for example) to be small, with the main resources located elsewhere on the network.

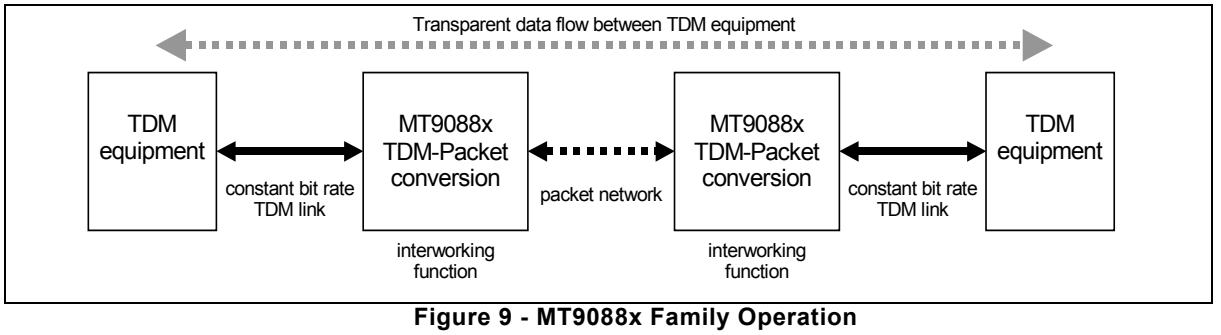

Figure 9 - MT9088x Family Operation

The MT9088x family provides structured, synchronous TDM service, operating at the 64 Kbit/s channel (DS0) level. The devices exhibit the following general service characteristics:

1. N × 64 Kbit/s trunking of any group of timeslots into a single “virtual channel connection” or "context"; these timeslots need neither be contiguous nor from the same stream of data
2. Ability to add or delete timeslots from a context in order to support dynamic bandwidth utilization.
3. Provision of a cross-connect service (i.e. contexts coming from several MT9088x devices are capable of being terminated by a single MT9088x device at the far end).
4. Ability to operate in "master mode" (supplying the timing to the TDM interface), or in "slave mode" (receiving the timing from the TDM interface).
5. Local TDM interface, for connection of external resources, e.g. echo cancellers, modem termination, HDLC controllers
6. Low latency connection, with end to end delay of less than 0.5 ms, depending on user configuration

2.2 Basic Operation

A diagram of the MT90880 device is given in Figure 10, which shows the major data and control flows between functional components.

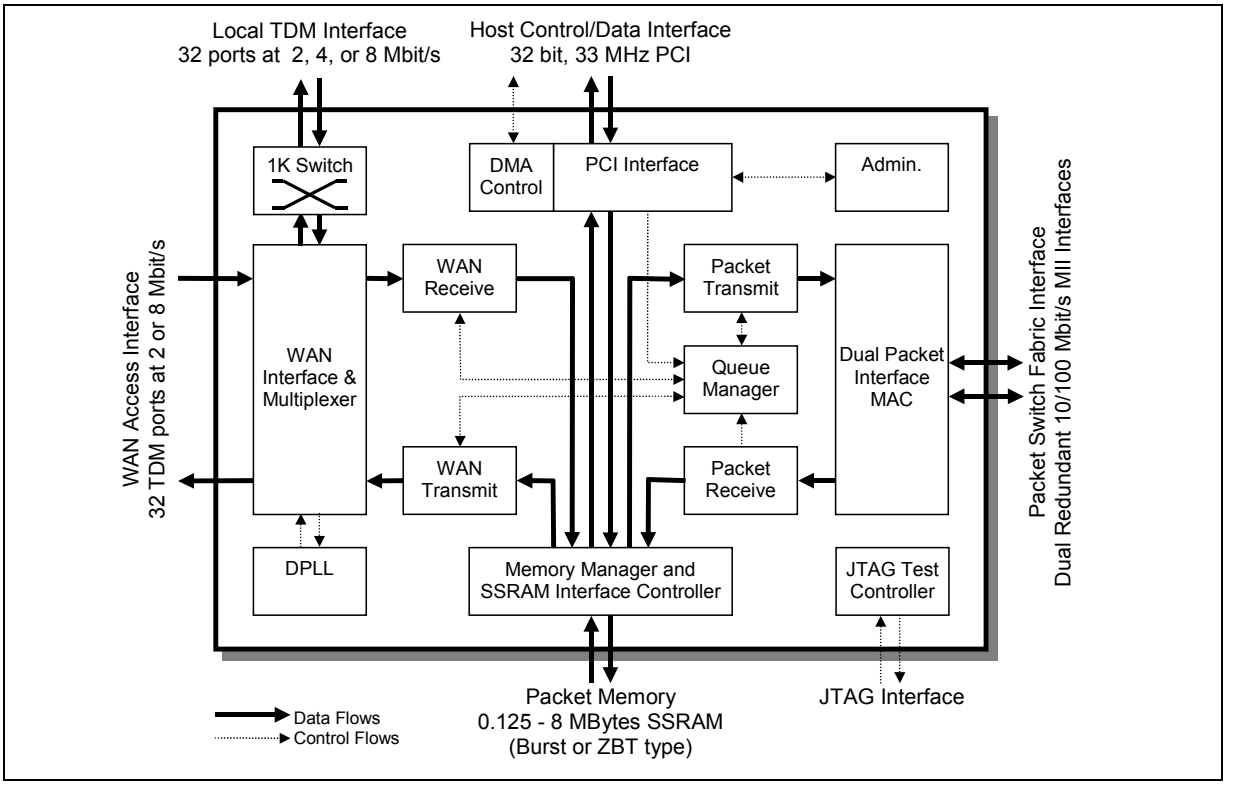

Figure 10 - MT90880 Data and Control Flows

2.2.1 WAN Access Interface

The WAN Access Interface consists of up to 32 ports, each with an input an output data stream operating at 2.048 Mbit/s. Alternatively, it can be configured as 8 ports operating at 8.192 Mbit/s. All 32 ports can operate using a common clock (synchronous mode) or using an independent clock for each port (asynchronous mode). When operating synchronously, the device can either operate as a slave, accepting an external clock, or as a master, supplying the clock to the devices on the WAN Access Interface from its internal Stratum 4E DPLL. The master clock can be locked to any of the incoming 32 frame references.

2.2.2 TDM Packet Assembly

Data traffic received on the WAN Access Interface is sampled in the WAN Interface block. It is then forwarded either to the WAN Receive block for packet assembly, or out to the TDM switch. The switch can be used both for re-ordering timeslots before packet assembly, or to divert traffic out of the local TDM interface for processing in a local resource pool (e.g. a DSP or other data processing unit).

The WAN Receive block can handle up to 128 active virtual channels or “contexts” simultaneously. A context may contain any number of timeslots, from 1 to 1024. This is known as “N x 64 Kbit/s” trunking. Timeslots may be added or deleted dynamically from a context to optimize network bandwidth utilization.

Trunking is not limited to contiguous timeslots. Each context is capable of carrying any combination of timeslots, taken from any stream. However, packets are assembled sequentially, with data placed into the packet as it arrives, maintaining timeslot and stream order.

2.2.3 Packet Transmission

Packets ready for transmission are queued to the switch fabric interface by the Queue Manager. Four classes of service are provided, allowing some packet streams to be prioritized over others. For example, to keep the latency of the TDM traffic low, TDM contexts can be prioritized over traffic from the CPU. Similarly, context control traffic can in turn be prioritized over maintenance traffic. Two types of prioritization are allowed: *weighted fair queuing*, where a traffic class is allocated a fixed proportion of the bandwidth, and *strict priority*, where all higher priority traffic is sent before any lower priority traffic.

On transmission, the Packet Transmitter appends a packet header, which has been set up in advance by the control processor. This allows the higher level protocols to be user-defined. The header is entirely user programmable, allowing any protocol to be used provided that the field contents remain static (i.e. do not change from packet to packet). Supported protocols include Ethernet, VLAN, IPv4, UDP (without using the checksum) and CDP (Zarlink's “Context Descriptor Protocol” on page 18).

2.2.4 CPU Packet Generation

The control processor can generate packets directly, allowing it to use the network for out-of-band communications. This can be used for out-of-band transmission of control data or network setup information, e.g. routing information. The PCI interface can also be used by a local resource for network transmission of processed data. Each of these types of traffic can be allocated a different class of service, allowing the bandwidth of the link to be carefully managed.

A DMA controller is built into the PCI interface to allow pre-formatted packets to be read directly from PCI memory and queued to the appropriate packet interface.

2.2.5 Packet Reception

Incoming data traffic on the packet interface is received by the MACs, which determines if the packets are intended for the device. If they are, the packets are forwarded to a packet classifier to determine the destination. This can either be the TDM domain (either WAN or Local interfaces) or the PCI interface. WAN traffic is then further classified to determine the context it is intended for.

Each TDM context has an individual queue, and the TDM re-formatting process re-creates the TDM streams from the incoming packet streams. Upon context setup, the TDM output is delayed by a programmable number of TDM frames, creating a jitter buffer. This is required to smooth out the variation in delay between individual packets.

There are four separate queues to the PCI interface, allowing different traffic flows to be classified separately. For instance control information could be directed to one queue, with traffic for a local resource pool directed to another queue. This simplifies access to the device by multiple resources or applications. Again the DMA controller can be used to retrieve packet data and write it out into external system memory on the PCI bus.

2.2.6 Call Setup and Control

Context setup must be negotiated between transmitting and receiving devices. This is handled by out-of-band signalling between the control processors at each end. Once the context has been established, the first data packet is transmitted. Each data packet contains a field in the header identifying the context to which it applies. This allows the receiving end to decode the packet and forward the data to the correct TDM channels.

Context modification (e.g. addition or deletion of timeslots) must also be set up in advance before the actual modification takes place. Changes are signalled out of band by the transmit end, which waits for confirmation from the receive end. This indicates that the amendments have been made, and it is ready to handle modified packets. Once confirmation has been received, the changes are activated at the transmit end. A synchronization flag in the header marks the first modified packet. This tells the receiver to switch over to the amended context information.

Removal of a context is signalled in the same way, with an acknowledgement from the receiver. Following “tear-down”, the internal queues are cleared, and any packets received for that context are discarded.

2.3 Data and Control Flows

The major data flows within the device are as follows:

1. WAN Access Interface to Packet Interface
2. Packet Interface to WAN Access Interface
3. WAN Access Interface to Packet Interface via a Local Resource Pool
4. Packet Interface to WAN Access Interface via a Local Resource Pool
5. PCI Interface to Packet Interface
6. Packet Interface to PCI Interface

WAN Access Interface to Packet Interface

Data traffic received on the WAN interface is divided into packets by the WAN Receiver. The packets are stored in the external packet memory. A pointer to the packet is passed to the queue manager, which appends completed packets to the appropriate transmission queue. On transmission the packet is retrieved from memory by the Packet Formatter, a pre-defined header added, and then passed to the MAC for transmission.

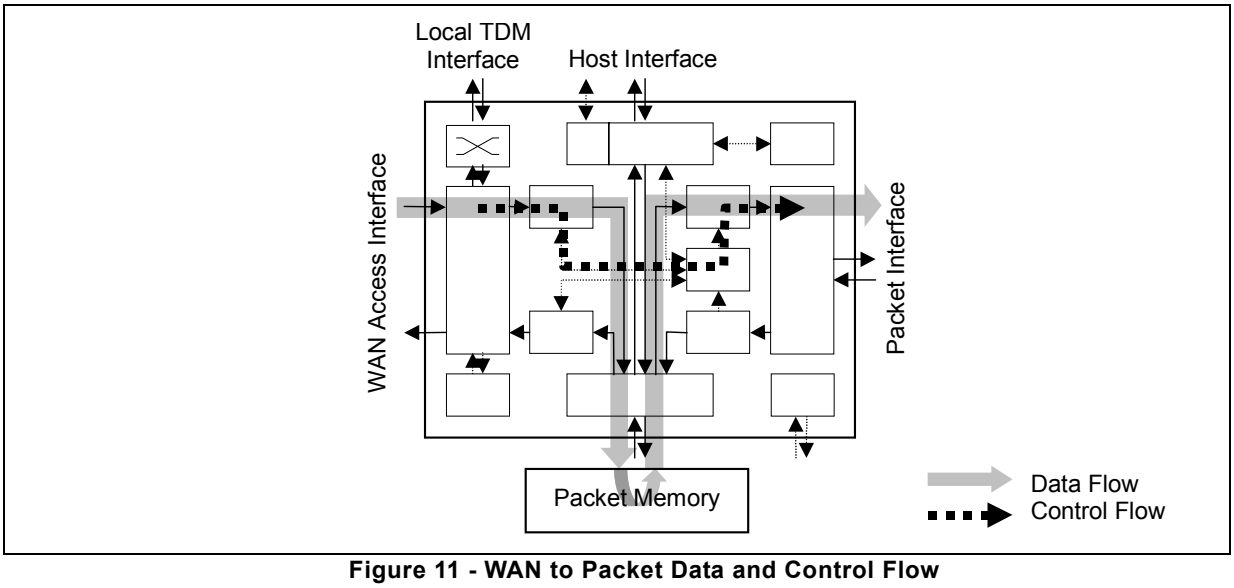

Figure 11 - WAN to Packet Data and Control Flow

Packet Interface to WAN Access Interface

Incoming data is received by the MAC, and its destination address is checked. Packets intended for this device are passed to the packet receive block for placing in external memory, while the header is classified to determine the appropriate destination. A pointer to the packet is passed to the queue manager to be placed on the correct queue as indicated by the classification result. The WAN Transmit block retrieves the data from packet memory, and directs it towards the appropriate timeslots on the WAN interface.

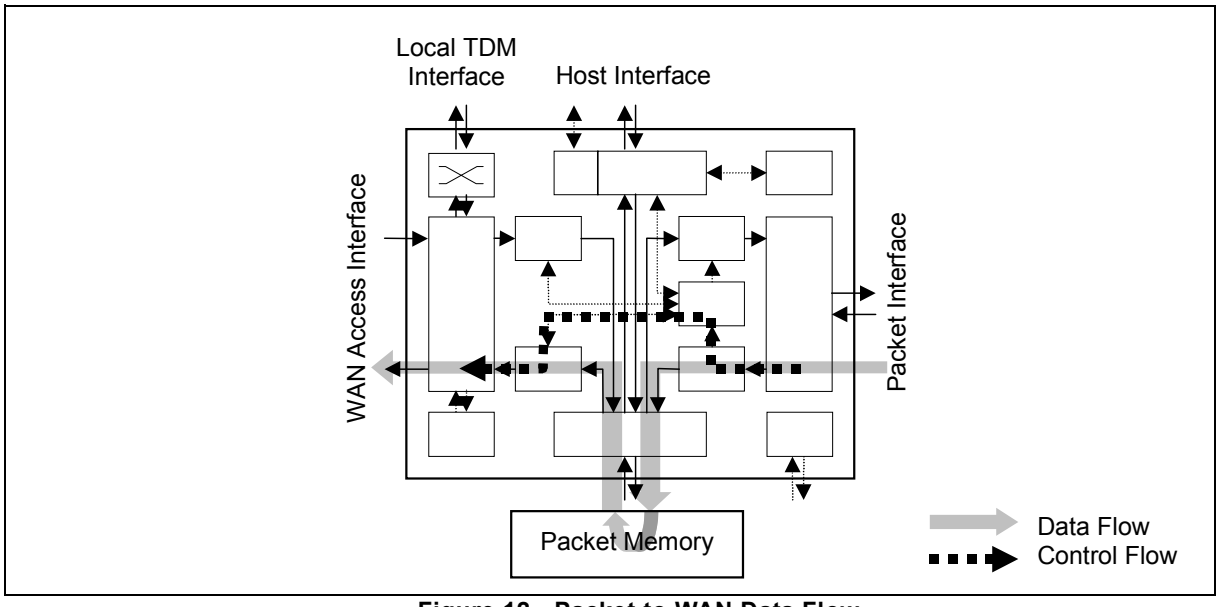

Figure 12 - Packet to WAN Data Flow

WAN Access Interface to Packet Interface via a Local Resource Pool

Data traffic received on the WAN interface is diverted to the TDM switch and out of the local TDM interface for processing in a local resource pool (e.g. a DSP or other data processing unit). The processed data can re-enter the device in two ways. Where the data is still constant bit rate, it can re-enter the device through the local TDM interface, and be routed by the TDM switch to the WAN Receive block for packetisation.

Alternatively, if the processed data has been converted to a variable bit rate or packet format, it can be moved into the device as through the PCI bus using the DMA controller to automatically fetch packets from system memory. The Queue manager appends the packets to the appropriate transmission queue. On transmission the packet is retrieved from memory and passed to the MAC for transmission.

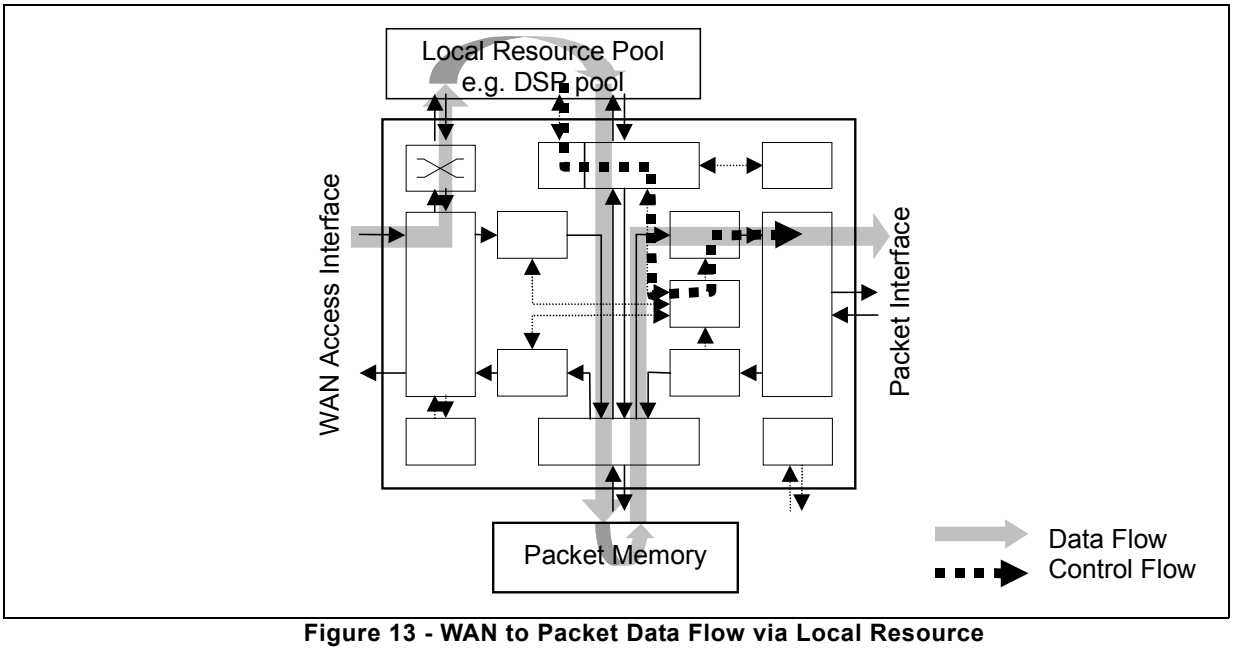

Figure 13 - WAN to Packet Data Flow via Local Resource

Packet Interface to WAN Access Interface via a Local Resource Pool

Incoming data is received by the MAC, and its destination address is checked. Packets intended for this device are passed to the packet receive block for placing in external memory, while the header is classified to determine the appropriate destination. A pointer to the packet is passed to the queue manager to be placed on the correct queue as indicated by the classification result.

If the destination is the PCI interface, the DMA controller is used to write the packets directly into system memory. After local processing, the data is re-directed to the WAN interface by going through the Local TDM interface and the TDM switch.

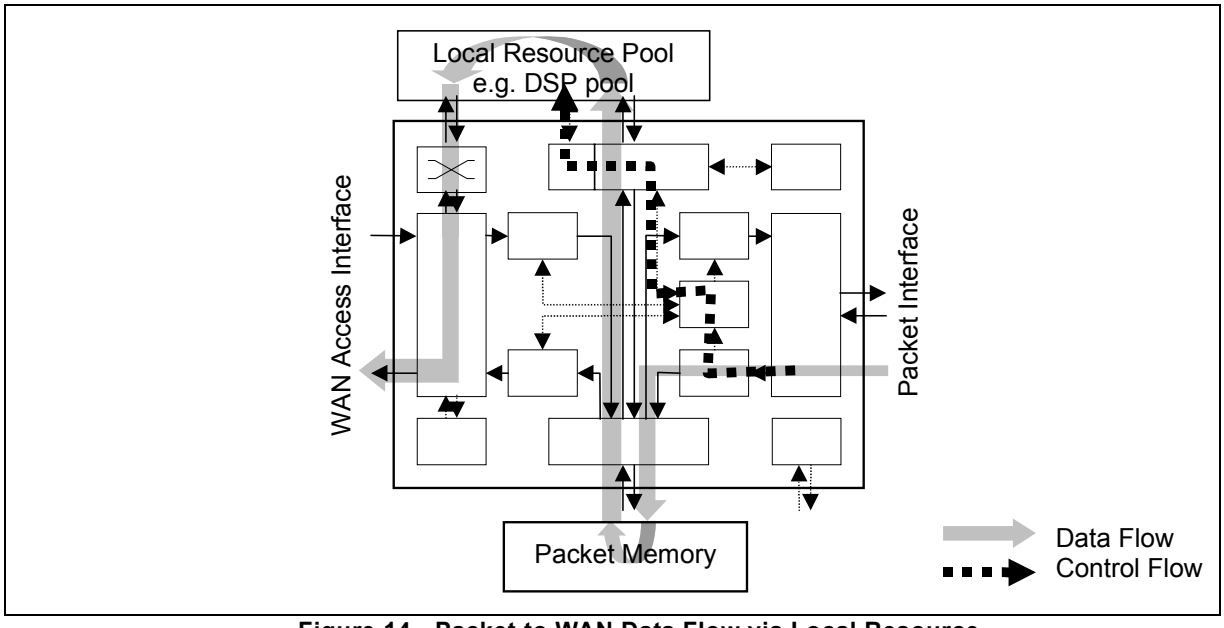

Figure 14 - Packet to WAN Data Flow via Local Resource

PCI Interface to Packet Interface

The host CPU or other devices can use the MT9088x device to send data via the packet network. The data to be sent is formatted into packets with the appropriate protocols in system memory. The DMA controller in the MT9088x is programmed to fetch this information, and place it into the appropriate packet queue.

On transmission the packet is retrieved from memory by the Packet Formatter in the normal way, and passed to the MAC for transmission.

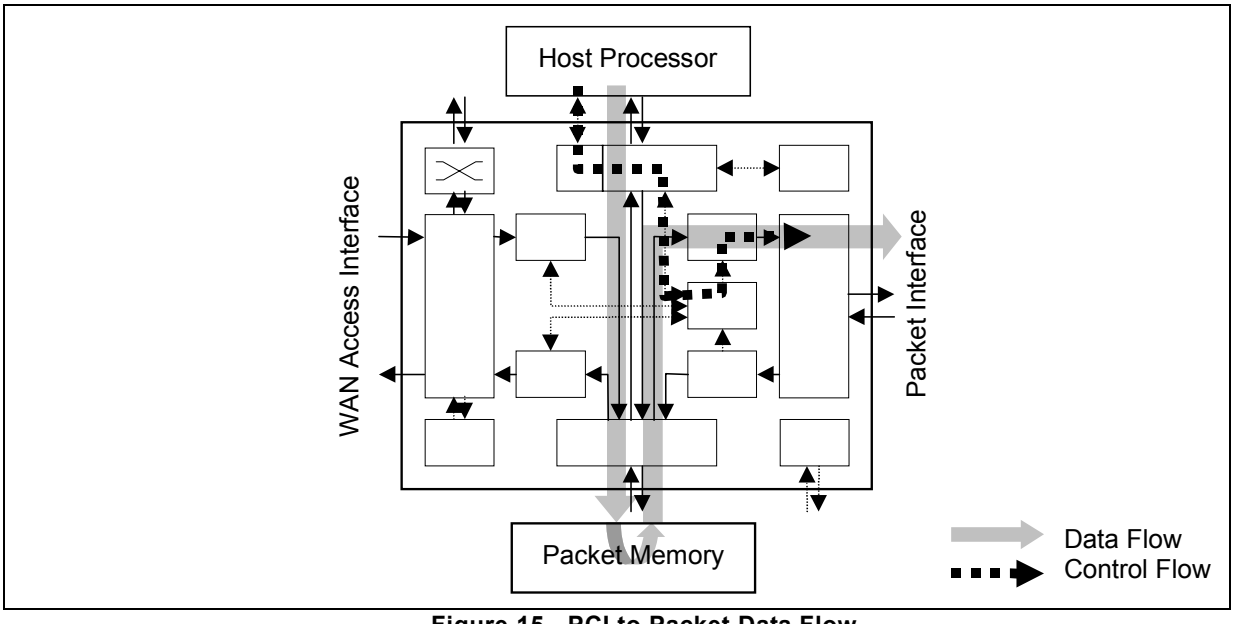

Figure 15 - PCI to Packet Data Flow

Packet Interface to PCI Interface

Similarly, packets destined for the host CPU or other PCI devices can be sent over the packet network to the PCI device. The headers of incoming packets are parsed by the Packet Classifier, and can be directed to one of four queues pointing towards the PCI interface. The classifier can be set-up so that each queue is dedicated to a specific device or application flow.

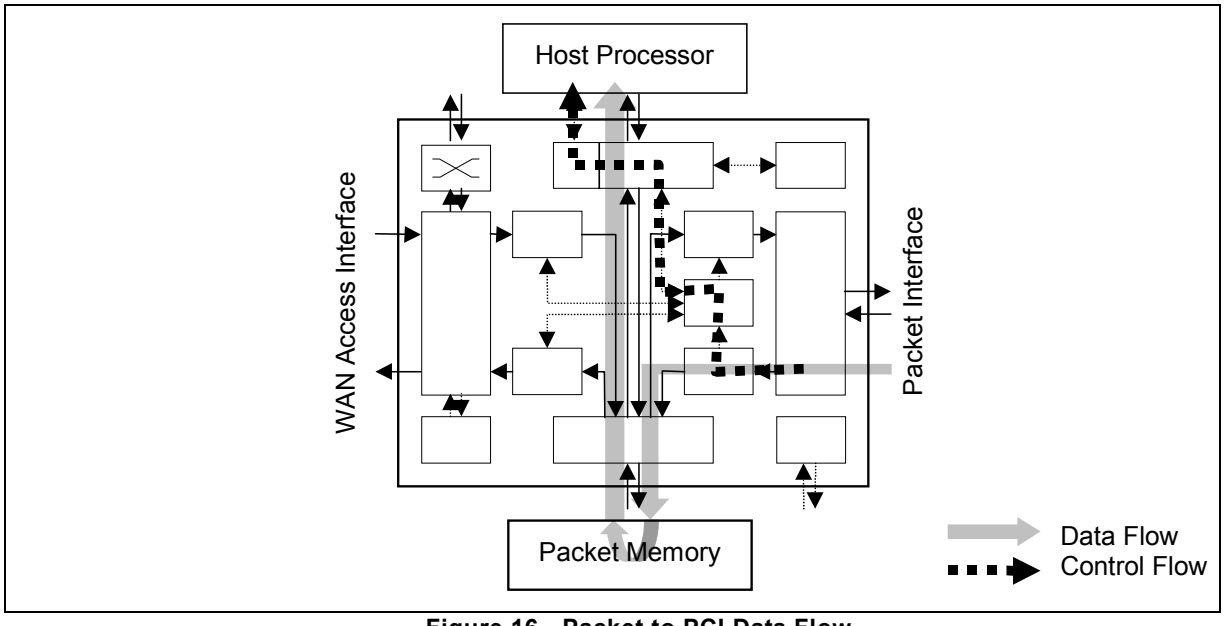

Figure 16 - Packet to PCI Data Flow

2.4 Packet Assembly

The incoming TDM data streams are assembled into packet payloads by the WAN Receive block. This can handle up to 128 active contexts at a time, where each context represents a “virtual channel connection” in CES terms. Each context generates a single stream of packets, identified by a label in the packet header known as the "context ID".

A context may contain any number of 64 Kbit/s channels. These channels need not be contiguous, and in synchronous mode they can be selected from any input stream. In asynchronous mode each stream is independently clocked, which can result in phase and frequency differences between streams. Therefore, in this mode contexts may only contain channels from a single stream.

Channels may be added or deleted dynamically from a context. This feature can be used to optimize bandwidth utilization. Modifications to the context are synchronized with the start of a new packet.

2.4.1 Payload Order

Packets are assembled sequentially, with each channel placed into the packet as it arrives at the WAN Access Interface. A fixed order of streams and channels is maintained (see Figure 17), with channel 0, stream 0 placed before channel 0, stream 1, which is placed before channel 1, stream 0. It is this order that allows the packet to be correctly disassembled at the far end.

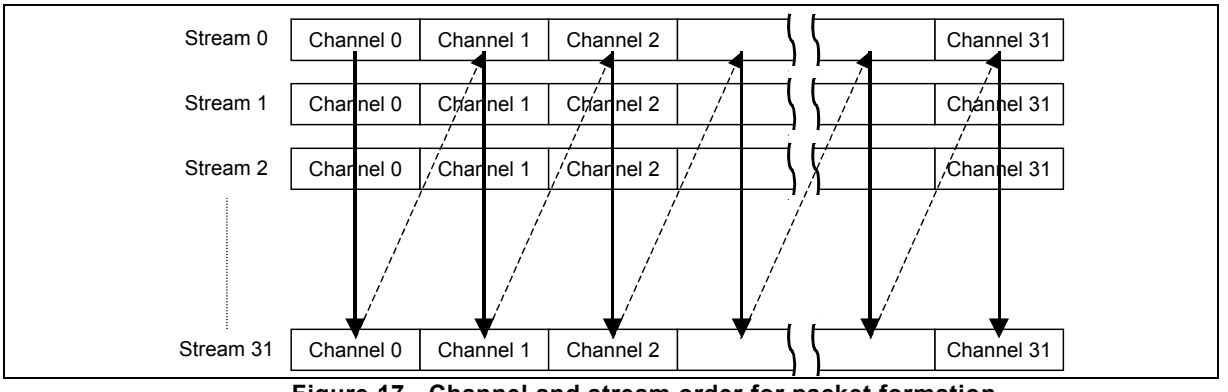

Figure 17 - Channel and stream order for packet formation

Each packet contains one or more complete TDM frames of data, in sequential order. This groups all the channels for the first frame, followed by the same set of channels for the subsequent frame, and so on. Partial TDM frames are not permitted, and each packet starts at the beginning of a new frame. Figure 18 shows the structure of a packet containing ***x*** 64Kbit/s channels, and ***n*** TDM frames.

2.4.2 Packet Structure

The fixed header at the start of each packet is added by the Packet Transmit block. This consists of up to 64 bytes, containing the Ethernet header, any upper layer protocol headers, and the two byte context descriptor field (see section below). The header is entirely user programmable, enabling the use of any protocol provided there are no dynamic fields (i.e. fields that change from packet to packet).

The Packet Transmit block also maintains a "shadow" header for each context. This is used for context modification, for example addition of deletion of channels. Any header changes required by a context modification are programmed into the shadow header in advance. When the changes are complete, the "shadow" header is swapped with the existing header for the first packet containing the new context payload.

Packet Size

The payload and header size must be chosen such that the overall packet size is not less than the Ethernet standard minimum packet size of 64 bytes. Where this is likely to be the case, the header must be padded (as shown in Figure 18) to ensure that the packet is large enough. The padding is required in front of the TDM payload, to ensure that the WAN Transmit block reads packets correctly on reception. This block is given a pointer to the start of the TDM payload, and then continues to read data until the end of the packet, as indicated by the packet length.

In applications where large payloads are being used, the payload size must be chosen such that the overall packet size does not exceed the maximum Ethernet packet size of 1518 bytes (1522 bytes with VLAN tags).

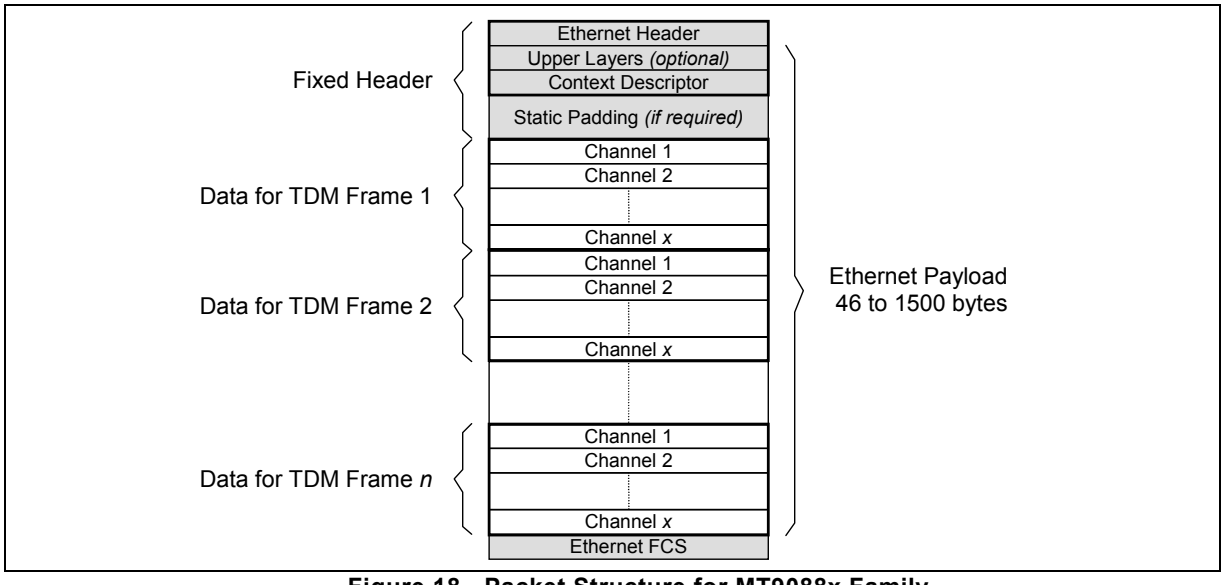

Figure 18 - Packet Structure for MT9088x Family

2.4.3 Context Descriptor Protocol

The MT9088x family uses a two-byte field within the packet header to convey which context the packet belongs to. The format of the field is as follows:

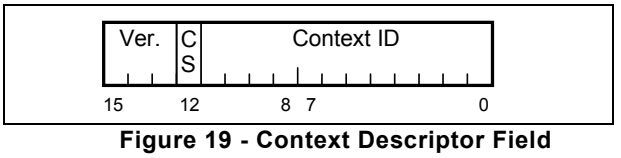

Figure 19 - Context Descriptor Field

Field definitions

Context ID A unique number identifying the context to the receiving device. The Context ID is assigned by the receiving device during context establishment.

The MT9088x only supports values in the range 0-127, and hence bits 7-11 must be set to zero.

CS Context Switch Bit. This bit toggles when there is a change to the context, e.g. addition or deletion of new timeslots from the packet. The toggle action ensures that the change to the context is still signalled even if the first packet indicating the change is lost.

Ver Protocol version number.

For the MT9088x family, the version number is always zero.

2.5 Context Negotiation and Establishment

2.5.1 Overview

The establishment of a context to carry TDM data across a packet network is a negotiated process. The call parameters (e.g. the context ID label, number of channels to be included, number of frames) must be agreed between the host CPU's at either end of the packet network. Similarly, subsequent changes to a context must also be agreed and synchronized between the two ends. The remote end is signalled to expect a modified context, and its host sets up the new context information. This new setup is activated on receipt of the first packet containing the new channel configuration.

This section describes the negotiation process that takes place at context establishment and modification. The examples assume two MT9088x devices connected via a packet network. One device, the *Packet Transmit End*, receives bytes from the WAN, assembles them into packets and transmits them over the network. The second device, the *Packet Receive End*, receives packets from the network, deconstructs them into bytes and transmits them in their proper timeslots toward the destination. Each MT9088x device has its own host, denoted "PTX Host" and "PRX Host".

The tables describe the process of context negotiation, rather than the protocol. The actual protocol can be chosen to suit the overall application and customer requirements.

2.5.2 New Context Establishment

Table 4 describes the process for setting up a new context.

	Packet Transmit End (PTX)	Packet Receive End (PRX)
1	PTX Host sets up a new transmit context, assigning a local context ID. This local context ID is used to identify the context within the packet transmit device.	
2	PTX Host signals Packet Receive host to expect a new context, with details of the channels involved and the destination channels.	
3		PRX Host sets up new receive context, assigning a local context ID (this may not be the same as the PTX's transmit context ID).
4		PRX host sends an acknowledgement back to the PTX that the context is set up and ready to receive packets.The PRX acknowledgement includes the local receive context ID.
5	PTX host completes context setup by placing the PRX receive context ID into the packet header.	
6	PTX host activates new context.	
7	At the start of the next TDM frame, the MT9088x commences formation of the first packet for the new context. This packet is prepended with the new packet header.	
8		First packet received for the new context, identified by the context ID label. Upon reception of packet the MT9088x waits for a predetermined number of TDM frames to allow for delay variation in the LAN, then automatically activates new context. The value of the context switch bit is recorded for reference.

Table 4 - New Context Establishment

2.5.3 Context Modification (Addition or deletion of physical channels)

Context modification must be an atomic operation. Therefore, if the host modifies the context setup, it must wait until that change has been carried out before attempting further modifications. This will take a bounded but indeterminate time, since the context will not actually be modified until the start of the next new packet. The context state machine will set a flag to indicate to the host that it is in the process of modifying the context, and that the host should refrain from further updates.

However, distinct contexts can be modified simultaneously, since each context is completely independent from any other. As a consequence, the MT9088x family supports a cross-connect service, where each context (or virtual channel) can be sent over the packet network to a different terminating device. The context modification process is shown in Table 4.

	Packet Transmit End (PTX)	Packet Receive End (PRX)
1	PTX Host modifies an existing context to add or delete channels.	
2	PTX Host signals Packet Receive host to expect a modified context, with details of the channels to be added or deleted, and their destination channels.	
3		PRX Host modifies receive context concerned.
4		PRX host sends an acknowledgement back to the PTX that the context has been modified and is ready to receive packets.
5	PTX host activates the previously programmed context change.	
6	At the start of the next new packet, the MT9088x commences formation of the first packet for the modified context. This packet is prepended with the new packet header, with the context switch bit toggled.	
7		First packet received for the modified context, identified by the toggling of the context switch bit. This automatically activates the previously programmed context modifications. The new value of the context switch bit is recorded for reference.

Table 5 - Context Modification Process

2.5.4 Context Removal

Context removal is required when all active channels are removed from a context. Normally, channel deletion is signalled in the first packet of the modified context. However, when the last channel is deleted from a context, no packet will be transmitted since there is no payload. Therefore the receiver will not be signalled that the channel deletion has taken place, and the context will underrun, waiting for further packets to arrive.

Therefore, it is not permitted to tear down a context simply by deleting all of its contents. Table 5 below describes the context removal procedure, known as "context teardown", used to allow contexts to be deleted cleanly.

	Packet Transmit End (PTX)	Packet Receive End (PRX)
1	PTX Host signals PRX Host to expect a context removal or "teardown"	
2		PRX Host programs MT9088x device to expect a context teardown.
3		PRX Host sends an acknowledgement back to the PTX Host.
4	PTX host activates the context teardown.	
5	The next new packet is the last one to be transmitted for this context. The context switch bit in the header is toggled to signal the context removal.	
6		Last packet received for the context to be removed, identified by the toggling of the context switch bit. This automatically activates the previously programmed context teardown.
7		The PRX Queue manager discards any future packets that arrive with the same context ID.

Table 6 - Context Teardown Process

2.5.5 Context Cleardown

A second means of context removal, known as "context cleardown", is provided for situations where a system error has occurred, and the context has not been removed properly. The cleardown mechanism prevents expired contexts from using up system resources and memory.

For example, during context teardown, if the last packet sent by the PTX is not received by the PRX, the context will still exist at the PRX when it has already been deleted by the PTX. In this situation, a timeout informs the PRX Host that the context teardown notified by the PTX Host has not taken place.

The PRX host then needs to take two actions:

1. Instruct the queue manager to flush the queue of packets for this context. Following this, the queue manager will discard any packets that arrive using the same context identifier.
2. Program a context cleardown, which immediately clears any state associated with the context.

3.0 Functional Block Descriptions

3.1 WAN Interface and Multiplexers

The WAN Interface provides the synchronization between the external TDM streams and the internal logic. It also multiplexes the between the cross-connect switch and two TDM processing blocks. The multiplexing allows the switch to be used both for routing data to the local TDM interface and for re-ordering timeslots on the way in and out of the device.

Features include:

- 32 bi-directional TDM ports, providing 1024 full-duplex channels
- Data rate of 2.048 Mbit/s supported over all 32 ports
- Data rate of 8.192 Mbit/s supported over 8 ports (MT90880 and MT90881 only)
- Supports “generic E1” mode, where 2.048 Mbit/s TDM streams are supported using a 2.048 MHz clock
- Individual per port control over the polarity of the frame pulse and clock
- Individual per channel high impedance output control
- The primary and secondary references can be selected from any of the incoming 32 ports (8 ports in 8 Mbit/s mode)
- Three operational modes:
  - *Synchronous master* - the MT9088x provides a common clock and frame pulse to all ports, which may be locked to an incoming frame reference
  - *Synchronous slave* - the MT9088x accepts a common external clock and frame pulse to be used by all ports
  - *Asynchronous mode* - Each port has its own clock and frame pulse input

3.1.1 Port Data Formats

As listed above, the MT9088x family accepts three data formats. A brief summary of the characteristics of these formats is given in Table 6 below. For more information see the ST-BUS specification (reference 7). The overall data format is set for the entire WAN Interface device, rather than on a per port basis. However, there is individual per port control of the polarity of both the incoming clock and frame pulse. There is also control of polarity of the master clock and frame pulse outputs, independent of the chosen data format (used when operating in synchronous master mode).

Data Format	Data Rate	Number of channels per frame	Clock Frequency	Nominal Frame Pulse Width	Frame Pulse Polarity	Frame Boundary Alignment	
						clock	frame pulse
Generic E1	2.048 Mbit/s	32 (streams 0-31)	2.048 MHz	488 ns	Per port control	Per port control	Starts at boundary
ST-bus	2.048 Mbit/s	32 (streams 0-31)	4.096 MHz	244 ns	Per port control	Per port control	Straddles boundary
ST-bus	8.192 Mbit/s ('80, '81 only)	128 (streams 0-7)	16.384 MHz	61 ns	Per port control	Per port control	Straddles boundary

Table 7 - Input Data Formats accepted by the MT9088x family

All WAN Interface inputs (including data in, clocks and frame pulses) have internal pull-down resistors, therefore they can be safely left unconnected if not used.

3.1.2 Operational Modes

The WAN Interface operates in three distinct modes: *synchronous master*, *synchronous slave* and *asynchronous* mode.

Synchronous Master Mode

In *synchronous master mode*, the MT9088x supplies the clock and frame signals to the external WAN infrastructure (e.g. framers or TDM backplane). The internal Stratum 4E DPLL is used to lock onto a primary reference signal. This can be chosen from any one of the 32 incoming frame references. A secondary reference may also be selected, and this is automatically switched in if the primary reference fails. The clock and frame pulse generated by the DPLL are used as the master timing source to the WAN Access Interface, and to clock in and out data on all 32 ports.

Typically, this configuration will be used to connect to framers. The MT9088x can be set to choose a primary and secondary frame reference from any of the 32 ports. The internal DPLL is used to lock onto the chosen reference, and provide a stable, low-jitter master clock to drive the TDM data buses between the framers and the MT9088x device.

An example of this is shown in Figure 20. An MT90880 is shown connected to up to 32 MT9076 T1/E1 framer/LIU devices, using the ST-bus at 2.048 Mbit/s. Each framer generates a frame pulse when a frame boundary is detected on the line, and this is fed into the MT90880 via the WAN interface's frame inputs WAN_FRMI[x]. One of these is selected as a reference for the internal DPLL, which generates the 4.096 MHz clock and the frame pulse required for the ST-bus operation. This is used both to clock data in and out of both the framer and the MT90880. The clki[31:0] inputs should be left unconnected, as shown in the figure below.

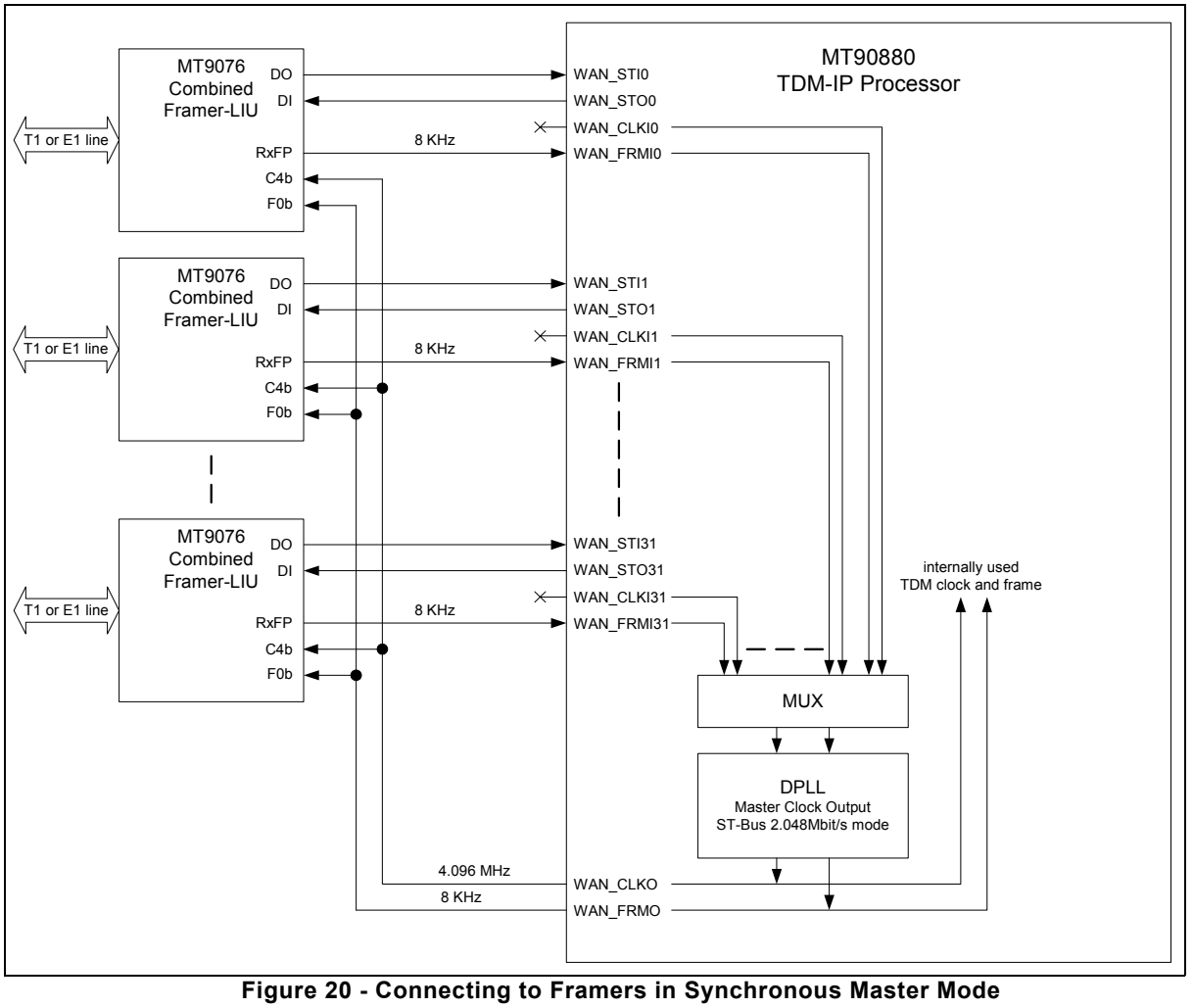

Figure 20 - Connecting to Framers in Synchronous Master Mode

The MT90880 may also be used as a TDM backplane master, such as an H.100/H.110 bus (reference 8), or an H-MVIP backplane (reference 9). The integrated DPLL provides a very stable, low jitter timing reference suitable for use as the master timing source for a backplane, with automatic, low MTIE failover from primary to secondary reference in the event of a reference failure.

Synchronous Slave Mode

In *synchronous slave mode*, the MT9088x accepts a clock from the WAN, rather than mastering the clock. One of the 32 incoming ports is chosen to provide the primary timing reference, and its clock and frame pulse are used directly to clock the data in and out on all 32 streams, bypassing the DPLL output. Again, a secondary reference port may be chosen, and this is automatically switched in if the primary fails. Unused clock and frame inputs may be left unconnected, as all inputs are connected to an internal 100KΩ pull-down resistor to prevent them from floating.

A typical application for this configuration is to connect to a TDM backplane, where the MT9088x is used as a backplane slave device. An example of this is Figure 21, which shows an MT90880 connected as a backplane slave to an H.100/H.110 TDM backplane (reference 8). An MT90866 TDM switch device is shown interfacing the MT90880 to the backplane. Up to three MT90880 devices could be connected directly to a single MT90866 switch. The CT_C8_A and CT_C8_B backplane clocks are used as the primary and secondary master clocks to the slave devices. The MT9088x family can also be connected as slave devices to older TDM backplanes such as MVIP and H-MVIP buses (reference 9), either directly or through a TDM switch.

In synchronous slave mode although the DPLL is not used to sample data from the WAN TDM ports, it is still used to provide the clocks required by the internal TDM switch. The device is able to tolerate jitter on the primary and secondary reference clocks in excess of the G.823 and G.824 standards (references 11 and 12) when the WAN TDM ports are run at 2.048 Mbit/s data rate.

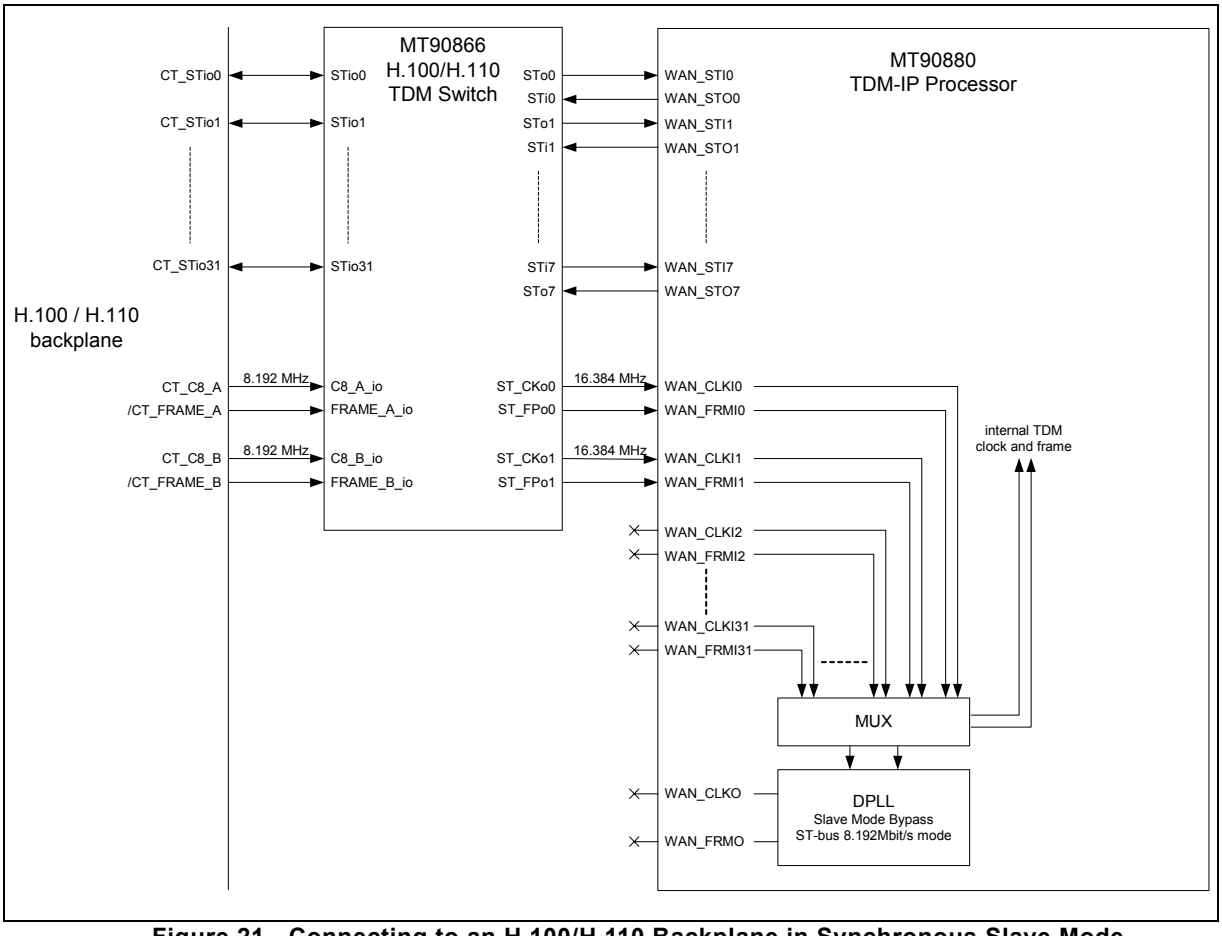

Figure 21 - Connecting to an H.100/H.110 Backplane in Synchronous Slave Mode

Asynchronous Mode

In *asynchronous mode*, each port uses its own clock to sample and drive the data streams. Since each clock can be at a slightly different frequency, trunking of timeslots is restricted to be within the same stream. The TDM cross-connect switch cannot be used in asynchronous mode.

The diagram in Figure 22 shows how the device might be connected to T1/E1 framers in asynchronous mode. Each incoming line is independently timed, bypassing the DPLL in the MT9088x device. The MT9076 contains an integrated PLL to multiply up the line rate to the 4.096 MHz ST-bus clock.

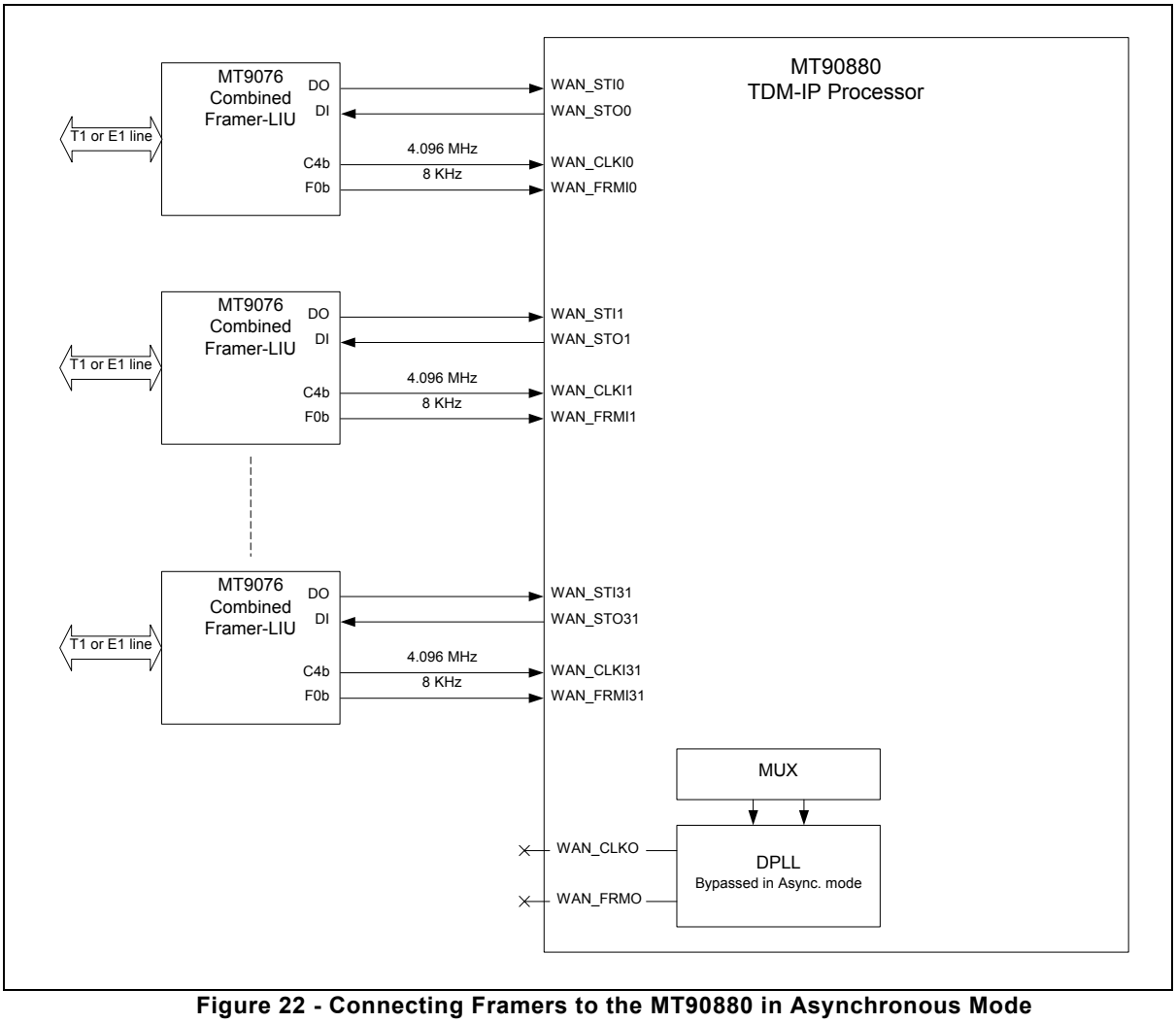

Figure 22 - Connecting Framers to the MT90880 in Asynchronous Mode

3.2 TDM Cross-Connect Switch

The TDM switch is a non-blocking switch with 1024 bi-directional channels. It can be used both for routing WAN traffic out to the local TDM interface, and for re-ordering timeslots on the way in or out of the device. Internally, it provides four independent data directions, from WAN side to local side, local to WAN, WAN to WAN and local to local. A multiplexing arrangement (see Figure 23) controls the access to the TDM switch, permitting the following data flows through the device:

- WAN interface to Packet Interface direct (without going through the switch)
- Packet Interface to WAN interface direct (without going through the switch)
- WAN interface to Packet Interface via the TDM switch (e.g. for re-ordering timeslots on input)
- Packet Interface to WAN interface via the TDM switch (e.g. for re-ordering timeslots on output)
- WAN interface to Local interface, via the TDM switch
- Local interface to WAN interface, via the TDM switch
- Local interface to Packet Interface, via the TDM switch
- Packet Interface to Local interface, via the TDM switch
- The following flows are also possible, facilitating test modes in the system
- WAN interface to WAN interface, looped back through the TDM switch
- Local interface to Local interface, looped back through the TDM switch
- Packet Interface to Packet Interface, looped back through the TDM switch

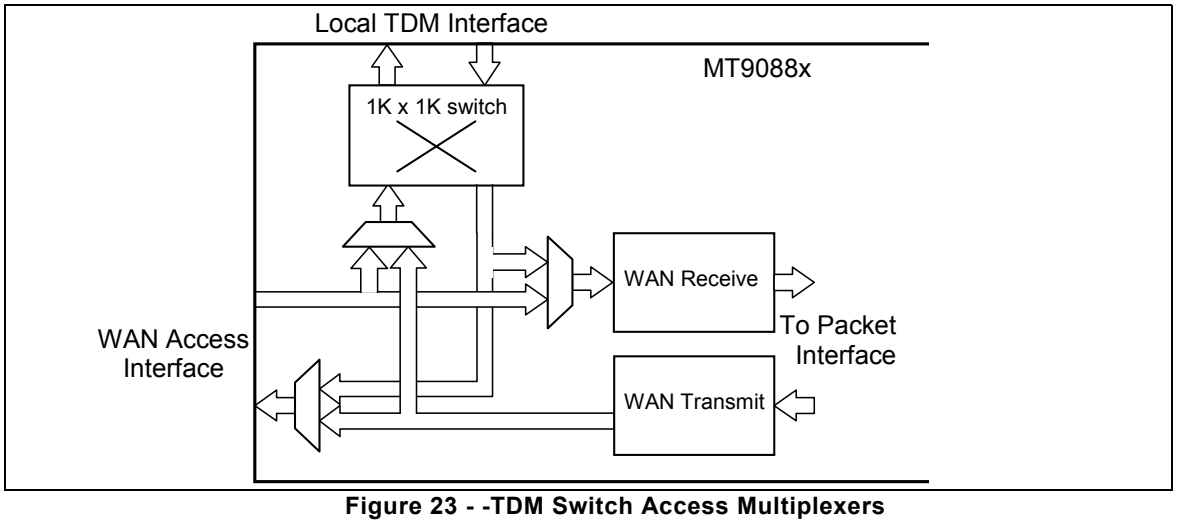

Figure 23 - -TDM Switch Access Multiplexers

Features include:

- Fully bi-directional, 1,024-channel x 1,024-channel non-blocking switch
- Accepts data rates of 2.048 Mbit/s, or 8.192 Mbit/s on the WAN interface side
- Accepts data rates of 2.048 Mbit/s, 4.096 Mbit/s or 8.192 Mbit/s on the local interface side
- Rate conversion between WAN and local streams
- Per-stream bit delay for the local input streams
- Per-stream bit advancement for the local output streams
- Per-channel constant throughput delay
- Per-channel high impedance output control for local and WAN streams
- Per-channel message mode for local and WAN output streams
- PRBS pattern generation and testing
- Block memory programming for fast device initialisation
- Channel duplication facility for broadcast and conference applications

3.2.1 Multiplexing and blocking

Although the switch unit itself is non-blocking, care must be taken with the multiplexing arrangement, since it is possible to set up a blocking configuration outside of the switch. This is because there are two paths at each multiplexer, so when one path has been used for a given timeslot, the other path is blocked. It is also possible to block the output from the switch to a given output timeslot on the WAN side of the switch.

For example, consider the following flow from WAN Access Interface to the Packet Interface, re-ordering using the switch. The input to the WAN Access Interface at stream 0 channel 0 enters the switch, and is re-mapped onto stream 1 channel 0 before entering the WAN Receive block.

This ties up the following resources:

• Switch input mux. at stream 0, channel 0	prevents the WAN transmit output on stream 0, channel 0 being routed via the switch (e.g. for re-ordering or directing to the Local interface.
• WAN Receive input mux. at stream 1, channel 0	prevents the direct connection of WAN Access Interface stream 1, channel 0 to WAN Receive block.
• Switch output for stream 1, channel 0	prevents any connection from the Local interface to stream 1, channel 0 on the WAN Access Interface.

Similar considerations apply to other routes through the device. In general, restricting the use of re-ordering to use either on input to or output from the device reduces the potential for blocking. Similarly, loopback from WAN to WAN also sets up significant blocking potential (two muxes and the switch output), and should be avoided in normal operation where possible.

3.2.2 Re-ordering timeslots

One application of the TDM switch is to re-order channels either on the input or output of the device. This may sometimes be necessary to provide a completely flexible channel mapping, since the packet assembly/disassembly process cannot vary the order of channels within a context. Hence a channel arriving at the packet assembly before another will always exit the packet disassembly process at the receiving end before the second channel. Where it is required to map the channels differently, the TDM switch can be used to handle the re-ordering.

For example, consider the following channel mappings:

- Stream 0, channel 0 is mapped to stream 0, channel 1
- Stream 0, channel 1 is mapped to stream 0, channel 0

Although individually channels can be mapped to any other channel at the packet receive end, if both these channels are contained in the same context, the packet assembly/disassembly process cannot provide this mapping, since the order of channels is changed.

Therefore there are three options to obtain the desired mapping:

1. use two separate contexts for the two channels
2. re-order the channels on input using the switch
3. re-order the channels on output using the switch

Use of a separate context may not be acceptable, since the number of contexts on the MT9088x family is limited to 128. Therefore the only option is to use the switch to re-order the channels. As described above, it is best to choose up front whether to re-order channels on the input or output side, to avoid potential blocking conditions in the multiplexed paths.

It should also be noted that channels routed via the switch (e.g. for re-ordering) incur a three-frame delay for the switch operation. This is not the case for channels routed directly (e.g. WAN Access Interface to WAN Receive block), which incur no delay. This may be a consideration for applications where end to end latency is an issue.

3.2.3 Channel Broadcast

The switch can also be used to broadcast a single input channel to multiple output channels. This is achieved by programming the connection memory to map multiple output channels to the same input channel.

Care must be taken when using this feature, since it does create blocking both within the switch itself, and through the multiplexing arrangement. This is because the number of available output channels is effectively reduced, meaning that not all input channels to the switch will be able to be mapped to an output. Similarly, the output multiplexer to the WAN Access Interface will be assigned to multiple broadcast channels, blocking those channels for output paths from the WAN Transmit block.

3.3 WAN Receive and Transmit Functions

The payload assembly process creates programmable length packet payloads from the TDM input streams. The assembly process is extremely flexible; packets can contain any number of 64 Kbit/s timeslots. These channels need not be contiguous, and in synchronous mode they can be selected from any input port. In asynchronous mode each stream is independently clocked, which can result in phase and frequency differences between streams. Therefore, in this mode payloads may only contain timeslots from a single port. Payloads may contain any integer number of TDM frames. The structure of a packet is shown in Figure 18, “Packet Structure for MT9088x Family,” on page 18.

Each stream of packets created by the packet assembler is known as a context. The packet assembler is capable of handling up to 128 different contexts simultaneously. Each context is allocated an identifier that is agreed in advance between the transmitting and receiving device, and this identifier is placed in each packet header to indicate the contents of the packet.

Timeslots can be dynamically added to or deleted from a context. This helps to conserve bandwidth on the packet network, by only transmitting the active timeslots. Context modifications must be indicated in advance to the receiving end, and acknowledged before the change can be implemented, in order to allow the TDM data to be correctly re-constructed.

At the receiving end of the packet network, the original TDM data must be re-constructed from the packets received. This is known as re-formatting, and follows the reverse process. A programmable jitter buffer is provided to iron out the packet delay variation across the network. The size of the jitter buffer can be programmed in units of TDM frames (i.e. steps of 125 μs).

Features include:

- Supports up to 128 simultaneous packet streams or “contexts”.
- Each context can handle any combination of timeslots, from any TDM stream.
- Supports dynamic addition and deletion of timeslots.
- Maintains timeslot order and TDM frame within each context.
- Supports circuit emulation of structured T1 or E1 services.
- Programmable number of TDM frames per packet.
- Supports payload sizes from 1 to 1500 bytes.
- Allows a user-defined static header to be attached to each packet.
- Programmable size jitter buffer size

3.4 Operation of the WAN Receive Block

The diagram in Figure 24 shows a simplified, conceptual diagram of the WAN Receive Controller, demonstrating the principles of its operation.

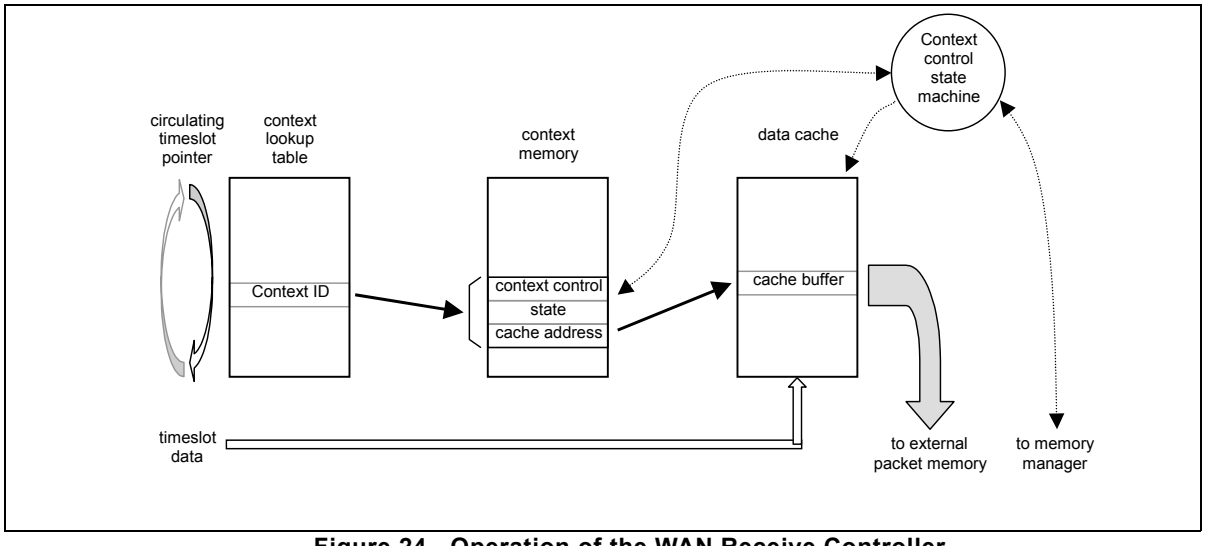

Figure 24 - Operation of the WAN Receive Controller

The WAN Receive controller receives constant bit rate TDM traffic from the WAN Interface. This interface continually presents new timeslot data at the input, together with an indication of the timeslot and stream the data is taken from. The *timeslot pointer* is used to look up the identity of the context that the timeslot is assigned to in the *context lookup table*. This table is programmed by the use to contain the mapping of timeslots into contexts.

The *context ID* is then used to look up information about the context from the *context memory.* This includes a pointer to the *data cache* where the timeslot data for the context is assembled prior to being stored in the external packet memory. The context memory also includes a number of user programmable fields controlling the context operation, such as packet payload length, context update and context teardown.

The *context control state machine* controls the allocation of the data cache and other information related to the context. It also handles the writing of the data cache contents into the external packet memory, requesting memory access from the Memory Management Unit.

The WAN Transmit Controller functions in a similar manner, but with the data flow in reverse. The control flow remains driven by the constant bit rate of the WAN Interface.

3.4.1 Context Control in the WAN Receive/Transmit Controllers

The Context Look-Up Table is the key to the control of contexts in both the WAN Receive and WAN Transmit Controllers. This contains the mapping of individual channels into contexts, and also provides the capability to program modifications to contexts without affecting existing channels. The fields within the context lookup table are shown in Figure 25.

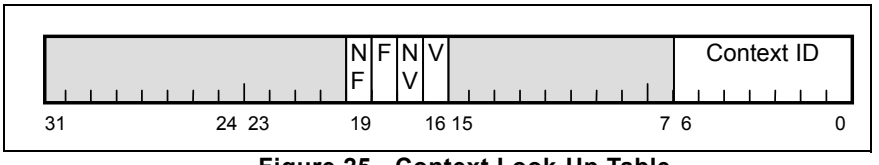

Figure 25 - Context Look-Up Table

Context Look-Up Table field definitions

Context ID The identity of the context that the channel is a member of. Value is in the range 0-127.

V *Valid Channel* bit. A '1' indicates that the channel is an active member of the context, and to be included in the packet payload. A '0' indicates that the channel is inactive.

NV *New Valid Channel* bit. A '1' indicates that the channel will become active in the next context update. A '0' indicates that the channel will be removed in the next context update.

F *First Channel* bit. A '1' indicates that this is the first channel of the indicated context (as indicated by the sequence shown in Figure 17). Each context has one and only one "first channel".

NF *New First Channel* bit. A '1' indicates that this will become the first channel of the indicated context at the next context update.

Context Setup

To setup a new context, the context lookup table is programmed to enter the new context ID against all the members of the context. The New Valid Channel bit is set for all members, and the New First Channel bit is set to indicate the first channel in the sequence. The Valid Channel and First Channel bits are cleared.

When context setup is complete, the Update bit in the Context Memory is set. This bit initiates the operation of the context. The timeslot pointer cycles round until it finds the first channel of the given context. At this point the state machine copies the New Valid Channel and New First Channel bits across into the Valid Channel and First Channel bits. The timeslot data is read and placed into the data cache.

Each time the timeslot pointer finds more channels that are valid members of the context, the bits are copied across in the same way, and the data read into the cache. When the pointer has circulated back round to the first channel, the context setup is complete, and the Update bit in context memory is cleared.

The first channel bit is required because the CPU cannot exactly time the programming of the Update bit to the start of a frame. Therefore, the first channel bit is used as a marker to indicate that this is the first channel in the context, following a frame boundary. This provides the necessary synchronisation to the start of a frame.

Note that context setup is not an isolated operation, and will normally need to be co-ordinated between the WAN Receiver and the remote WAN Transmit Controller. The sequence for co-ordinating context setup is shown in Table 4 on page 19. In general, the application should ensure that the receiving context in the WAN Transmit Controller is updated and ready to receive the packets prior to setting the update bit in the WAN Receiver.

Context Modification

The WAN Receive Controller supports dynamic addition and deletion of channels into contexts without interruption to the existing channels within that context. This is achieved by use of the New Valid Channel and New First Channel bits to program the alteration in advance of the actual update where the changes take place. This allows the CPU time to setup the changes and inform the receiving device before actually initiating the modification.

Note that such changes will normally need to be applied at both the local WAN Receiver and the remote WAN Transmit Controller, and that it is necessary to observe the correct sequence order to avoid mis-operation (see Table 5 on page 20). The application should ensure that following the lookup table modifications, the receiving context in the remote WAN Transmit Controller is updated and ready to receive the modified packets prior to applying the changes to the local WAN Receiver.

To add new channels to an existing context, the context lookup table is programmed to enter the context ID against all the new members of the context. As with context setup, the New Valid Channel bit is set for all new members, and the New First Channel bit is set to indicate the first channel in the sequence (if this is to change). The Valid Channel and First Channel bits are cleared.

Similarly, to delete channels from a context, the New Valid Channel bit (and New First Channel bit if this was the original first channel) is cleared. For these channels the Valid Channel and First Channel bits are left alone.

When the programming of the context modification is complete, the Update bit in the Context Memory is set. This bit initiates the actual modification of the context. On completion of the current packet formation the update cycle starts. The timeslot pointer cycles round until it finds the first channel of the given context. At this point the state machine copies the New Valid Channel and New First Channel bits across into the Valid Channel and First Channel bits. The timeslot data is read and placed into the data cache.

Each time the timeslot pointer finds more channels that are valid members of the context, the bits are copied across in the same way, and the data read into the cache. When the pointer has circulated back round to the first channel, the context setup is complete, and the Update bit in context memory is cleared.

The result at the end of the cycle is that all new channels, as indicated by the New Valid Channel bit are now valid members of the context, and that all channels where the New Valid Channel bit was cleared have now been deleted from the context.

The first packet built by the WAN Receive Controller is indicated to the receiving device by the toggling of the context switch bit in the Context Descriptor (see Figure 19). This triggers a context update cycle at the WAN Transmit Controller, ensuring the context update is correctly synchronized between the two devices.

Context Removal

Context removal is controlled by the use of the "teardown" control bit in the context memory. This causes the internal state machine to send one last packet, while automatically clearing all the flags in the Context Look-Up Table. As with context setup and modification, context removal has to be co-ordinated with that of the WAN Transmit Controller in the receiving device. This is described in Table 6 on page 21.

3.4.2 Jitter buffer Operation

TDM packets received from the network are forwarded into a set of queues, while waiting for processing by the WAN Transmit Controller. Each context has a separate queue, used to buffer the constant bit rate TDM interfaces from the variable delays experienced in the packet network. The queues are therefore known as "jitter buffers". The WAN Transmit block reads packets from the jitter buffer at a regular rate, before re-formatting the data for transmission over the WAN Access Interface or the Local TDM interface.

The size of jitter buffer translates directly into latency, since packets wait in the queue until the time to comes to be serviced. Most telecommunications applications are quite sensitive to latency, and attempt to minimise the overall end to end latency. However, if the jitter buffer is not big enough to cover the full variation in packet delay through the network (PDV), the buffer will either overflow or empty completely (underrun). Both of these conditions will result in data corruption.

The initial size of the buffer is programmable in units of TDM frame periods (125 μs). This is done by delaying the WAN Transmit Controller from reading the first packet received on a new context for the specified number of frames. This has the effect of setting the latency through the jitter buffer at the programmed value.

However, the buffer can grow or shrink with time. When the variation in packet delay through the network is greater than the programmed buffer latency, occasional packets will arrive too late, and the TDM interface will be starved of data.

Jitter Buffer Underrun

The jitter buffer will "underrun", i.e. empty completely under the following circumstances:

- If the TDM data is played out of the TDM interfaces faster than the original TDM interface
- If the packet delay variation in the network is larger than the programmed jitter buffer
- If a large number of packets are lost in the network

Either condition will have the effect that packets will arrive too late to be played out at the correct time, with the TDM interface being starved of data. When an underrun occurs, the TDM interface plays out "underrun data" for an integer number of TDM frames. Underrun data can either be the last value played out on that timeslot, or a pre-programmed value (e.g. 0xFF). When the late packet arrives, it is played out as normal.

The net effect depends on the cause of the underrun. If the interface is underrunning because of a difference in frequency between the TDM clocks at either end of the network, the underrun introduces a frame slip every few frames, depending on the size of the frequency discrepancy. For example, if the frequency difference was 50 ppm, the frame slip will occur every 20,000 frames. The overall latency of the system will stay the same.

If the underrun is being caused by a large PDV in the network, the depth of the jitter buffer (and hence the overall latency) for the underrunning context is increased by the number of frames that the packet is late. Over time, the buffer depth will increase to match the peak PDV in the network.

The context descriptor protocol used by the MT9088x family has no sequencing information, so the device cannot tell when packets are lost in the network. When a packet is lost, it makes the next arriving packet look like a "late" packet. The next arriving packet is played out as soon as it arrives. This has the effect of reducing the jitter buffer depth. If enough packets are lost, this will start to cause underruns in the device, until the buffer depth is built back up to match the network PDV again.

Jitter buffer depth limitation

Packet delay variation in the network varies with time. As described above, the jitter buffer in the MT9088x family will track the peak level of PDV in the network. However, it will only track the PDV upwards. If there is a transient burst of activity in the network, the PDV may rise for a short period, and then settle back down again, but the buffer depth will remain at the level required to handle the peak PDV.

For instance, if there was a link failure somewhere in the network, it could result in a large backlog of packets being stored up at a network node. When the link is restored, these packets may be released suddenly, creating a rush of very late packets. These packets will build up a large jitter buffer and a consequent increase in end to end latency. This may be undesirable in applications that are sensitive to the increased latency.

The MT9088x provides a "drop threshold" limiting the maximum size of the queue to each context. When the queue depth reaches the drop threshold, any further packets arriving are discarded, preventing the size of the buffer and hence the end to end latency from increasing further.

Different applications have different requirements for where to set the drop thresholds. Applications such as packet backplanes generally operate with tightly controlled networks with only a small variation packet delay. Therefore the drop thresholds for this type of application need to be small, and the MT9088x allows the drop thresholds to be set in increments of one packet, between 0 and 15 packets.

However, for applications such as circuit emulation, while end to end latency is still important, the network is much larger and the packet delay variation cannot be as tightly controlled. Therefore the MT9088x provides a second range, where the thresholds can be set to increment by steps of 16 packets, between 15 and 255 packets. Where a packet contains a single TDM frame of data, this represents a step size of 2 ms. The two ranges are known as "backplane mode" and "network mode".

Jitter buffer overrun

The jitter buffer will "overrun", i.e. fill up completely under the following circumstances:

- If the TDM data is played out of the TDM interfaces slower than the original TDM interface
- If the packet delay variation in the network is larger than the programmed jitter buffer

The maximum size of the jitter buffer is governed by the drop threshold described above. When the drop threshold is reached, any further packets arriving are discarded until the buffer depth has reduced by being played out of the TDM interface.

If the interface is overunning because of a difference in frequency between the TDM clocks at either end of the network, the overrun causes packets to be discarded at a constant interval, depending on the size of the frequency discrepancy. For example, if the frequency difference was 50 ppm, a packet will be discarded every 20,000 packets. However, since packets may contain multiple TDM frames, this may cause a more noticeable disturbance in the data stream than in the underrun case, where only a single frame slip occurs.

If the underrun is being caused by a large PDV in the network, the overrun cause the depth of the jitter buffer to be limited to the maximum size, as described above.

3.5 Queue Manager

The Queue Manager is responsible for admission control, linking/dropping, and scheduling incoming packets either from WAN side, packet interfaces or from the PCI Interface.

The Queue Manager provides a full spectrum of QoS (Quality of Service) features. For traffic queued to the packet interface, the Queue Manager provides four different priority levels. These can be managed either on a strict priority basis, or using Weighted Fair Queuing (WFQ).

Similarly, the packet classifier can be used to direct traffic to four separate PCI interface queues. The host processor is responsible for managing the relative priority of each of these queues.

For traffic queued towards the WAN interface, each separate packet stream or context is managed in a separate queue. The Queue Manager sets a programmable limit on the maximum number of packets in an individual context queue. This provides a means of detecting when the egress port has become broken, leading to the queues starting to fill up. Beyond the limit, incoming packets are dropped to avoid the memory becoming full, and therefore causing problems to the remaining contexts.

Features include:

- Four separate queues to each port on the packet interface
- Queue disciplines on packet interface are:
  - Weighted Fair Queuing with programmable weights
  - Strict Priority
- Separate queue for each WAN context
- Programmable drop threshold when queues get too big
- Four traffic classes on incoming packets
- Four separate queues to the PCI interface (host is responsible for any priority scheme)

3.5.1 Queues to the Packet Interface

The MT9088x family has four queues to each port of the packet interface. Each queue is used to provide a different class of service. TDM traffic from the WAN or Local interfaces can be allocated to a particular queue and port on a per-context basis. Similarly, packets from the CPU interface can be directed to any one of the queues and ports.

Classes of Service

The MT9088X uses two methods of queue control to provide for different classes of service: Strict Priority (SP), and Weighted Fair Queuing (WFQ). The queue disciplines can be allocated flexibly across the four different queues, as shown in Table 7.

Options	Class of Service			
	Class 3	**Class 2**	**Class 1**	**Class 0**
Option 1 (default)	SP	SP	SP	WFQ
Option 2	SP	SP	WFQ	WFQ
Option 3	SP	WFQ	WFQ	WFQ
Option 4	WFQ	WFQ	WFQ	WFQ

Table 8 - Configuration of Packet Queues

Queues defined as strict priority take precedence over queues designated as using weighted fair queuing. Hence, in option 3, Class 3 gets all the bandwidth it needs, and the other classes are allocated a proportion of the remaining bandwidth. In effect, option 1 defaults to strict priority across all four queues, since Class 0 only gets serviced when there are no packets in the remaining SP queues.

Strict Priority

The Strict Priority queue system ensures that the queue with the highest priority is empty before transmitting any data from the next highest priority. For example in option 1, if the three queues for Classes 3, 2 and 1 each contained packets to transmit, all the packets in the Class 3 queue would be transmitted before packets from any other classes. The packets in the Class 2 queue will then be transmitted before any packets from Class 1. If more packets arrive in Class 3 while Class 2 or Class 1 packets are being transmitted, the new packet in Class 3 is automatically sent once the current packet has completed transmission. Any packets still in the Class 2 or Class 1 queues must wait until the Class 3 queue is empty again before they can be transmitted.

Weighted Fair Queuing

Weighted Fair Queuing allocates a portion of the available link bandwidth to each class. This prevents higher priority queues from "hogging" the link, since once they have used their allocated bandwidth, the other queues are given access to the link.

In the MT9088x family, each class assigned to WFQ by the option settings is given a weight between 1 and 64. The bandwidth of each LAN port is 100 Mbit/s, and the weights assigned to each class represent a proportion of the bandwidth assigned to the queue. The sum of the weights must equal 64 for correct functionality.

For example, if option 4 is selected the following weights might be programmed to set the appropriate bandwidth as shown in Table 9:

Class	Weight	% Bandwidth allocated
Class 3	32	50.0 Mbit/s
Class 2	24	37.5 Mbit/s
Class 1	06	9.4 Mbit/s
Class 0	02	3.1 Mbit/s
TOTAL	64	100 Mbit/s

Table 9 - Example of Weighted Fair Queuing

Weighted Fair Queuing doesn't set a maximum figure on the amount of bandwidth a queue gets allocated. If there are no other queues requiring service, a queue will continue to be serviced even if its bandwidth allocation is exceeded. The bandwidth allocation is only used to guarantee a minimum allocation to a particular traffic class, should it be required.

In the absence of a priority requirement, the weights should be set to the same value to allow equal access to the bandwidth. For example, in option 4 all the weights should be set to 16, allocating 25 Mbit/s per queue.

Drop Thresholds

The MT9088x has the ability to restrict the amount of memory used by each queue. When the memory utilisation reaches the limit, new packets are dropped. This limitation prevents a queue from increasing in size to the point where the device runs out of memory, causing the whole device to crash and lose data. For instance, this could potentially happen if an Ethernet link failed, causing a backlog of packets to build up in the queues to that link.

The drop thresholds are programmed in units of granules, where a memory granule is a 128-byte block of data (see the section “Granule Structure” on page 45). For the queues to the packet interface, the thresholds can be set to between 0 and 1023 granules per queue. In addition, it is possible to set an overall threshold for the maximum number of granules allowed in all the queues to the packet interface at any one time.

If required, it is possible to disable the packet dropping when the drop thresholds are exceeded, although this is not normally recommended. Exceeding the drop threshold causes an interrupt to the control CPU to inform it of the queue size violation, since this normally indicates some kind of error condition (e.g. an Ethernet link failure, as described above).

3.5.2 Queues to PCI Interface

The MT9088x maintains four separate queues of packets waiting for transfer to the CPU. These are for packets arriving over the network that either do not contain TDM data, or contain TDM data that needs processing by some external resource on the PCI bus (e.g. a DSP). The packet classification process is capable of identifying up to four different traffic classes, and can direct each of these traffic classes to a different queue. Unmatched traffic is always sent to CPU queue 0.

Priority of CPU Queues

The internal DMA controller is used to transfer packets from the MT9088x's packet memory into the system memory on the PCI bus (see the “DMA Controller” on page 51). While the CPU queues have no inherent priority levels associated with them, in that the CPU can choose to service which queue it wants to first, there is a priority level associated with the DMA transfer into system memory. The device uses strict priority order to determine which queue gets serviced first, with queue 3 having the highest priority and queue 0 the lowest priority. Therefore the DMA will only transfer packets in a given queue if all the higher priority queues are empty or stalled (queues may be stalled if there is no free space in system memory for them to be transferred into).

Drop Thresholds

As with the queues to the packet interface, there are also drop thresholds associated with the queues to the PCI interface. The drop thresholds are programmed in units of granules, and can be set to between 0 and 1023 granules per queue. Again, it is possible to disable packet dropping, although this is not normally recommended. Exceeding the drop threshold causes an interrupt to the control CPU to inform it of the queue size violation.

3.5.3 Queues to WAN Interface

The packet classifier identifies the packets arriving over the network that contain TDM data, and retrieves the context ID from the two-byte context descriptor. A separate queue is provided for each context, and the classifier forwards the packets to the appropriate queue. The WAN Transmit block reads packets from the end of the queue at a regular rate, and re-formats the data for transmission over the WAN Access Interface or the Local TDM interface. No priority is associated with these queues, since each WAN context is independent, and does not compete for resource against the other contexts.

Drop Thresholds

The context queues are used to buffer the constant bit rate TDM interfaces from the variable delays experienced in the packet network (see the section on "Jitter buffer Operation", on page 28). The initial size of this jitter buffer can be pre-programmed, but in cases where packets arrive very late across the network, the jitter buffer can grow. For instance, if there was a link failure somewhere in the network, it could result in a large backlog of packets being stored up at a network node. When the link is restored, these packets may be released suddenly, creating a rush of very late packets. These packets will build up a large jitter buffer and a consequent increase in end to end latency.

Drop thresholds can be used to limit the size of this jitter buffer to a maximum value. Each of the queues to the TDM interfaces has a drop threshold that can be programmed in units of the number of packet in the queue. Packets arriving once this threshold has been reached are dropped, preventing the size of the buffer and hence the end to end latency from increasing further.

Different applications have different requirements for where to set the drop thresholds. Applications such as packet backplanes generally operate with tightly controlled networks with small packet delay variations. Therefore the drop thresholds for this type of application need to be small, and the MT9088x allows the drop thresholds to be set in increments of one packet, between 0 and 15 packets.

However, for applications such as circuit emulation, while end to end latency is still important, the network is much larger and the packet delay variation cannot be as tightly controlled. Therefore the MT9088x provides a second range, where the thresholds can be set to increment by steps of 16 packets, between 15 and 255 packets. Where a packet contains a single TDM frame of data, this represents a step size of 2 ms. The two ranges are known as "backplane mode" and "network mode", and are programmed by a bit in the queue manager control register.

An overall threshold can also be set for the total amount of memory used by the queues to the TDM interfaces. This is set in units of granules, and has a maximum size of 65535 granules.

3.6 Packet Transmit

On transmission, a user defined static header is added to the packet payload. This contains the Ethernet header, the context descriptor, and the header required for the higher level protocols to be used (if any), e.g. IP, UDP. Since the header is completely user-defined, any high level protocol can be used provided the header does not change from packet to packet (e.g. contains a sequence number or a checksum across the payload).

3.6.1 Protocol Stacks

The MT9088x family can be used with a number of different protocol stacks. The common element is the use of the context descriptor for determination of the TDM context that the packet relates to. Examples of the most common expected stacks are (CDP = Context Descriptor Protocol):

- Ethernet - CDP

  This is the most basic expected stack. Where the TDM traffic is only passed over a switched network, no upper layer headers are required. The Ethernet destination address is used to switch the packet to the correct destination, and the context descriptor used to determine the context.

  An Ethertype needs to be allocated to CDP to enable the traffic type to be determined in the packet classification process.

- Ethernet - VLAN - CDP

  It is also possible to use a VLAN tag to switch the packets to the correct destination in the network. Again, an Ethertype needs to be allocated to CDP to enable the traffic type to be determined in the packet classification process.

- Ethernet - IPv4 - CDP

  Where packets have to sent across a routed network, rather than a simple switched Ethernet, IP can be used to provide the network layer. CDP can sit directly on top of the IP layer, but in this case a protocol number will need to be assigned.

- Ethernet - IPv4 - UDP - CDP

  Where packets have to sent across a routed network, rather than a simple switched Ethernet, IP can be used to provide the network layer. The use of UDP as the transport layer enables TDM traffic from several different devices to be identified by use of different port numbers, and also removes the need for a protocol number to be assigned to CDP.

- Ethernet - MPLS - CDP

  An alternative to IP for a routed network is to use MPLS. An MPLS tunnel label in the header is used to route the packet across the network. An optional inner label could be used to identify different traffic classes, with the CDP fulfilling the role of a "Martini" style encapsulation layer.

3.6.2 Shadow Headers

The Packet Transmit block also maintains a "shadow" header for each context. This is used for context modification, for example addition of deletion of channels. Any header changes required by a context modification are programmed into the shadow header in advance. When the changes are complete, the "shadow" header is swapped with the existing header for the first packet containing the new context payload.

3.7 Ethernet MAC

The MT9088x family of devices contain two separate, IEEE standard 802.3 compliant, 10/100 Mbit/s Ethernet MACs (see reference 1, Table 2). Each MAC is connected to a Physical Layer (PHY) device via a Media Independent Interface (MII) or Reduced Media Independent Interface (RMII) (reference 3, Table 2). The MAC is responsible for data encapsulation/decapsulation. This includes frame alignment and synchronization, and detection of physical medium transmission errors. The MAC is capable of both full and half-duplex operation. In half-duplex mode it manages the collision avoidance and contention resolution process. In the event of a collision, the MAC will back off and attempt to re-send the packet up to 16 times.

Packets for transmission are forwarded to the MAC by the Packet Transmit block. The MAC appends the frame check sequence, and generates the preamble and start of frame delimiter before transmitting out of the MII or RMII port.

During packet reception, the MAC receive section verifies that the frame check sequence is correct, and that the packet is a valid length. Packets with an invalid frame check sequence, and data packets longer than 1518 bytes (1522 with VLAN tag) and shorter than 64 bytes are dropped. The MAC also checks the destination field to determine if the packet is intended for the device. If the packet is accepted, it is forwarded on for packet classification, and to be entered into the appropriate destination queue. Illegal packets, or packets intended for a different destination are discarded.

The MAC also collects statistics on the different types of packets transmitted and received on the Ethernet. The statistics collected are sufficient to enable the CPU to support the Interfaces sections of some common MIBs.

Features include:

- IEEE 802.3 compliant operation at 10 and 100 Mbit/s
- Industry-standard MII and RMII interfaces to the physical layer devices
- Full and half-duplex operation
- Generates preamble, start-of-frame delimiter and frame check sequence
- Collision avoidance and contention resolution in half-duplex mode
- Verifies frame check sequence and frame length, discarding frames that contain errors
- Statistics collection for common MIB support:
  - RFC 1213 MIB II.
  - RFC 1757Remote Network Monitoring MIB (for SMIv1)
  - RFC 2819Remote Network Monitoring MIB (for SMIv2)
  - RFC 2863Interfaces Group MIB

3.8 Packet Classification

This is used to determine the destination of incoming traffic on the packet interface from information contained in the packet header. Up to four separate traffic classes can be identified, and each class can be forwarded independently. Examples of traffic classes include TDM traffic for forwarding to the WAN interface, CPU control traffic, and traffic destined for a DSP or other off-chip processing element. Unidentified traffic is sent to the host processor via the lowest priority queue.

Traffic classes are identified by matching the header against a pre-programmed pattern. If a pattern match is detected, the traffic can be directed to one of four separate queues to the host interface, or to the WAN interface. All WAN traffic must contain a two-byte adaptation layer (known as the context descriptor) in a known position within the first 64 bytes. This contains the context or packet stream to which the packet belongs, and other information required to synchronize to any changes to the contents of a particular context.

Features include:

- Four separate traffic classes
- Match on any field of combination of fields within the first 64 bytes
- Host traffic can be directed at one of four different queue
- Context Descriptor can be placed anywhere in the first 64 bytes (for TDM data packets)
- Can interface to devices placing the context descriptor in different places

Pattern Matching

Packet classification in the MT9088x is a three-stage process, as shown in Figure 26. Firstly, the packet header is masked, to concentrate the classification on the header fields of interest. Four separate masks are applied in parallel across the first 64 bytes of the packet. Next the masked header is compared against up to four possible patterns. Again, the comparison function is across the first 64 bytes of the packet. This depth of search allows fields from layer three and four headers to be included in the classification (e.g. IP and UDP headers).

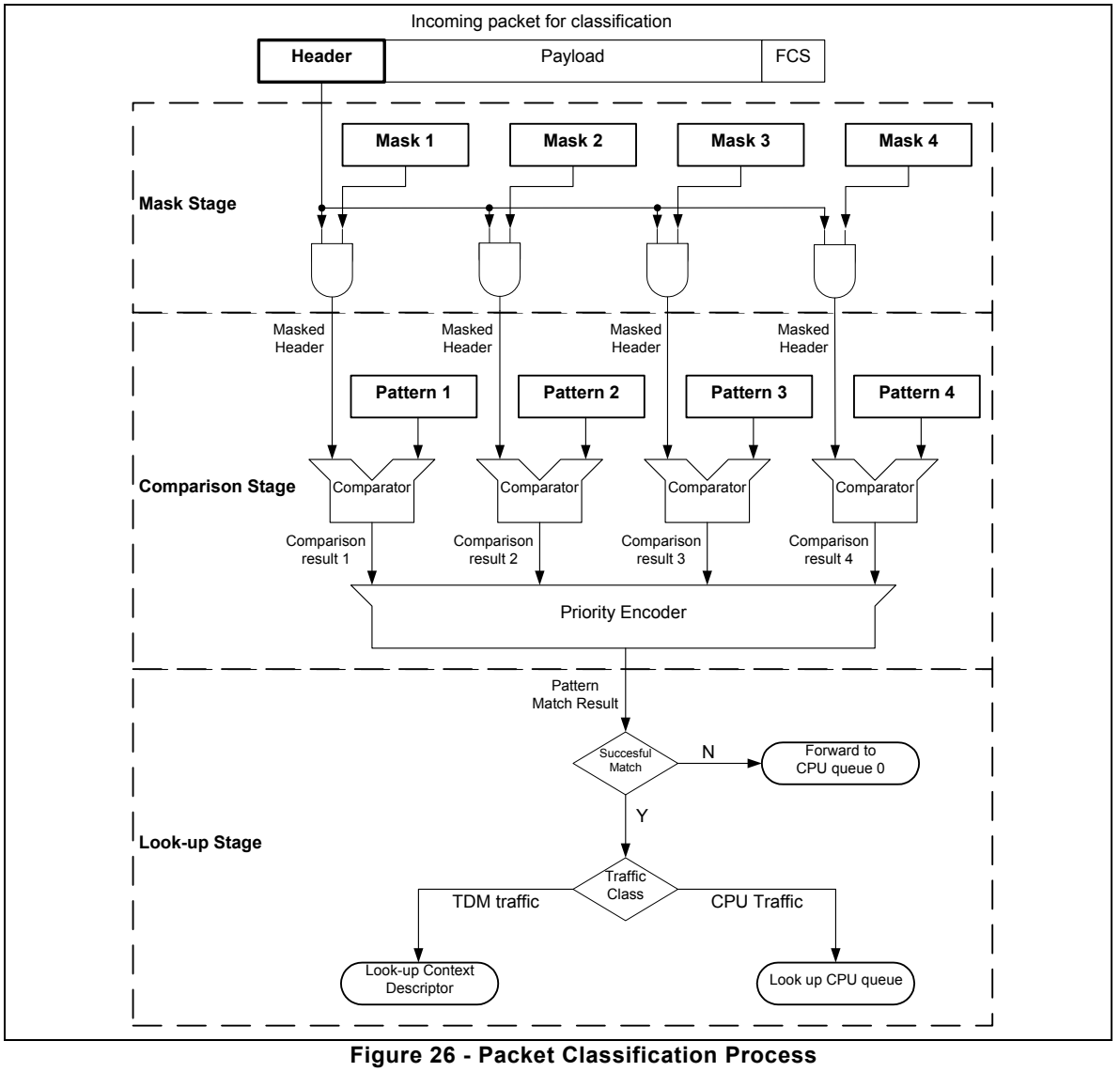

Figure 26 - Packet Classification Process

Once a successful match is established, the characteristics of each traffic class may be determined by a look-up process. If the matched class contains TDM traffic, the context descriptor is retrieved from a pre-determined position. This indicates the identity of the context to which the packet belongs, which is used to place the packet into the appropriate queue. More than one traffic class can be designated as TDM traffic, allowing the context descriptor to be retrieved from a different position for each class. This may be useful when the MT9088x device received traffic from multiple sources, each of which may be using a different protocol stack.

CPU traffic can be allocated to one of four queues waiting for DMA transfer up to the host. Therefore the classifier can be used to pre-sort the traffic destined for the CPU. This can be used either to direct traffic to

different destination devices (e.g. packets for the control CPU, and packets for a DSP or local resource pool), or to separate high priority control traffic from lower priority maintenance information.

Unmatched traffic is also forwarded to the CPU using queue 0. This allows the CPU to analyse the traffic to work out what should be done with it. Alternatively, if the drop threshold on queue 0 is set very small, unmatched traffic can be automatically discarded.

3.8.1 Example Classification Scheme

The following example shows how to set up the packet classification for a typical expected scenario. The MT9088x is expecting the following four classes of traffic:

1. TDM traffic using the protocol stack Ethernet - VLAN - CDP
2. TDM traffic using the protocol stack Ethernet - IPv4 - UDP - CDP
3. Data traffic to an external DSP using the protocol stack Ethernet - IPv4 - xxx
4. CPU control traffic using the protocol stack Ethernet - IPv4 - TCP - xxx
5. All other traffic - routed by default to CPU queue 0

Traffic Class 1: Ethernet - VLAN - CDP

Configure the mask and match registers in the packet engine to direct TDM traffic to the WAN interface.

Protocol Field	Mask	Match / Comment
Ethernet/VLAN		
Destination MAC address	Mask	If the MAC block is programmed into promiscuous mode then the destination MAC address must be matched,
Source MAC address	Mask	
Tag Type	Allow match	Must be 0x8100
VLAN Tag	Allow match	Must be set to the appropriate value for TDM traffic.
Length / Type field	Allow match	Must be set to the appropriate type for TDM traffic. *Will require allocation of an Ethertype by the IEEE.*
Context Descriptor		
Version	Allow match	Must be 0b000
Context Switch	Mask	
Context ID[11:7]	Allow match	Must be 0b00000
Context ID[6:0]	Mask	
Remainder of the header	Mask	

Table 10 - Pattern Matching for Example Traffic Class 1

Fields from Packet Engine Control Register:

Control Register Field	Value	Comment
CPU_SEL1	set to 0	TDM traffic
CPU_PRI1	don't care	Not CPU traffic
Byte offset to Context Descriptor	18	18 bytes in header before the CD

Table 11 - Control Register Fields for Example Traffic Class 1

Traffic Class 2: Ethernet - IPv4 - UDP - CDP

Configure the mask and match registers in the packet engine to direct TDM traffic to the WAN interface.

Protocol Field	Mask	Match / Comment
Ethernet		
Destination MAC address	Mask	If the MAC is programmed into promiscuous mode then the destination MAC address must be matched,
Source MAC address	Mask	
Length / Type field	Allow match	0x0800 (IP)
IPv4		
Version	Allow match	0b0100
Internet Header Length (IHL)	Allow match	0d20
Type of Service (TOS)	Mask	
Total Length	Mask	
Identification	Mask	
Flags	Allow match	0b010 - Ensure set to don't fragment
Fragment Offset	Mask	
Time to Live (TTL)	Mask	
Protocol	Allow Match	0d17 (UDP)
Header Checksum	Mask	
Source IP address	Mask	
Destination IP address	Allow Match	Check the packet has the right IP address for TDM data.
UDP		
Source Port	Allow match	Must be set to the appropriate value for TDM traffic.
Destination Port	Allow match	Must be set to the appropriate value for TDM traffic.
Length	Mask	
Checksum	Mask	

Table 12 - Pattern Matching for Example Traffic Class 2

Protocol Field	Mask	Match / Comment
Context Descriptor		
Version	Allow match	Must be 0b000
Context Switch	Mask	
Context ID[11:7]	Allow match	Must be 0b00000
Context ID[6:0]	Mask	
Remainder of the header	Mask	

Table 12 - Pattern Matching for Example Traffic Class 2 (continued)

Fields from Packet Engine Control Register:

Control Register Field	Value	Comment
CPU_SEL2	set to 0	TDM traffic
CPU_PRI2	don't care	Not CPU traffic
Byte offset to Context Descriptor	0d42	42 bytes in header before the CD

Table 13 - Control Register Fields for Example Traffic Class 2

Traffic Class 3: Ethernet - IPv4 - xxx

Configure the mask and match registers in the packet engine to direct data traffic to an external DSP processor on CPU DMA queue 2.

Protocol Field	Mask	Match / Comment
Ethernet		
Destination MAC address	Mask	If the MAC is programmed into promiscuous mode then the destination MAC address must be matched,
Source MAC address	Mask	
Length / Type field	Allow match	0x0800 (IP)
IPv4		
Version	Allow match	0b0100
Internet Header Length (IHL)	Allow match	0d20
Type of Service (TOS)	Mask	
Total Length	Mask	
Identification	Mask	
Flags	Mask	
Fragment Offset	Mask	
Time to Live (TTL)	Mask	

Protocol Field	Mask	Match / Comment
Protocol	Mask	
Header Checksum	Mask	
Source IP address	Mask	
Destination IP address	Allow Match	Check the packet has the right IP address for external DSP traffic.
Remainder of the header	Mask	

Table 14 - Pattern Matching for Example Traffic Class 3

Fields from Packet Engine Control Register:

Control Register Field	Value	Comment
CPU_SEL3	set to 0b1	Send packets to the CPU.
CPU_PRI3	set to 0b10	Send to queue 2.
Byte offset to Context Descriptor	Don't care	Not TDM traffic

Table 15 - Control Register Fields for Example Traffic Class 3

Traffic Class 4: Ethernet - IPv4 - TCP - xxx

Configure the mask and match registers in the packet engine to control traffic to the CPU.

Protocol Field	Mask	Match / Comment
Ethernet		
Destination MAC address	Mask	If the MAC is programmed into promiscuous mode then the destination MAC address must be matched,
Source MAC address	Mask	
Length / Type field	Allow match	0x0800 (IP)
IPv4		
Version	Allow match	0b0100
Internet Header Length (IHL)	Allow match	0d20
Type of Service (TOS)	Mask	
Total Length	Mask	
Identification	Mask	
Flags	Mask	
Fragment Offset	Mask	
Time to Live (TTL)	Mask	

Table 16 - Pattern Matching for Example Traffic Class 4

Protocol Field	Mask	Match / Comment
Protocol	Allow Match	0d6 (TCP)
Header Checksum	Mask	
Source IP address	Mask	
Destination IP address	Allow Match	Check the packet has the right IP address for CPU control traffic.
Remainder of the header	Mask	

Table 16 - Pattern Matching for Example Traffic Class 4 (continued)

Fields from Packet Engine Control Register:

Control Register Field	Value	Comment
CPU_SEL4	set to 0b1	Send packets to the CPU.
CPU_PRI4	set to 0b11	Send to queue 3.
Byte offset to Context Descriptor	Don't care	Not TDM traffic

Table 17 - Control Register Fields for Example Traffic Class 4

3.9 Memory Management Unit

The Memory Management Unit handles all access to the external packet memory, arbitrating between the different modules requiring access. Efficient use of external memory is maintained by allocating memory in small blocks or “granules”.

Features include:

- Interfaces to industry standard PBSRAM and ZBT SSRAM
- Operates at 66 MHz
- 32 bit wide data path
- Supports one to four equal sized memory banks
- Supports a total of between 0.125 and 8 Mbytes of memory

3.9.1 External Memory Requirements

The majority of the memory is used by the jitter buffer, which compensates for delay variation (PDV) in the packet network. The amount of memory required to compensate for a given PDV depends on the mean size of the packet payload. This is because for small packets, a large proportion of the packet is actually header. The graph in Figure 27 shows the amount of packet delay variation that can be handled given the mean packet payload size and the size of the external packet memory.

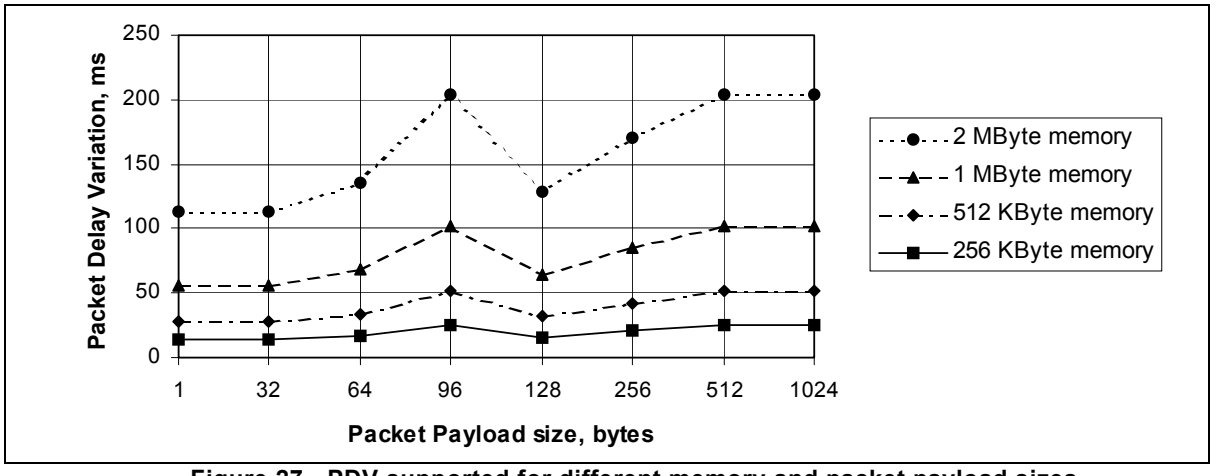

Figure 27 - PDV supported for different memory and packet payload sizes

3.9.2 Granule Structure

Memory on the MT9088x family is dynamically allocated in 128 byte blocks, known as "granules". Each granule contains a 16-byte header section, used primarily for linking granules together into chains, and 112 bytes of data. The granular organisation of memory is responsible for the discontinuities in the graph shown in Figure 27, as the packet size grows to consume another granule.

The granule header contains information passed between the various internal blocks indicating how to handle the information contained within the granule. Most importantly, it contains a pointer field indicating the location of the next granule. This is used to chain granules together, initially to create large packets, and subsequently to link packets together into queues.

To simplify flow of packets between functional blocks within the device, pointers are passed between blocks indicating the location of the packet in memory. These pointers actually contain the location of the first granule in a packet. Subsequent granules are indicated by the pointer field in the granule header.

Free granules are held centrally in a pool managed by a granule manager block. When granules are required (e.g. during payload assembly or packet reception), the relevant block makes a request to the granule manager, which returns a pointer to a free granule. Similarly, when a granule (or chain of granules) is no longer required (e.g. the packet has been sent out of one of the output interfaces), it is released back to the free pool by sending a granule release request to the granule manager.

3.9.3 Connecting the MT9088x to external memory

The MT9088x is capable of interfacing to two types of external memory device, pipelined burst SRAM (such as Micron™'s "Syncburst™" family) and to the more bandwidth efficient types (such as Micron™'s "ZBT®" family), which remove the turnaround cycles required when changing from a read to a write cycle. For most applications, one external device is sufficient, as shown in the graph in Figure 27. All four memory capacities shown in this graph can be implemented using one device (either 4, 8 or 16 Mbit). However, for more demanding applications, the MT9088x can connect to up to four external memory devices, with a total capacity of up to 8 Mbytes.

Connecting to a single external memory devices

The diagram in Figure 28 shows how to connect the MT90880 up to a single PBSRAM device. The example chosen is a Micron™ MT58L256L32P, an 8 Mbit "Syncburst™" family SRAM. The clock to the memory is the 66 MHz system clock, used as the master clock to the MT90880 and the memory system.

The memory is permanently chip enabled, with all chip enable signals (CE#, CE2# and CE2) tied to an active state. Read and write access is controlled by the MT90880 outputs RAM_OE[0]# and RAM_WE[0]#. The MT90880 only performs 32 bit wide accesses to memory, therefore the memory's individual byte write signals (BWa# to BWd#) are tied low. Write access to the full 32-bit word is controlled via the global write enable signal, GW#.

Connection to a ZBT® device, such as the MT55L256L32P, is very similar, substituting the R/W# pin for the Syncburst's GW#, and tying the control pins LBO# and CKE# low.

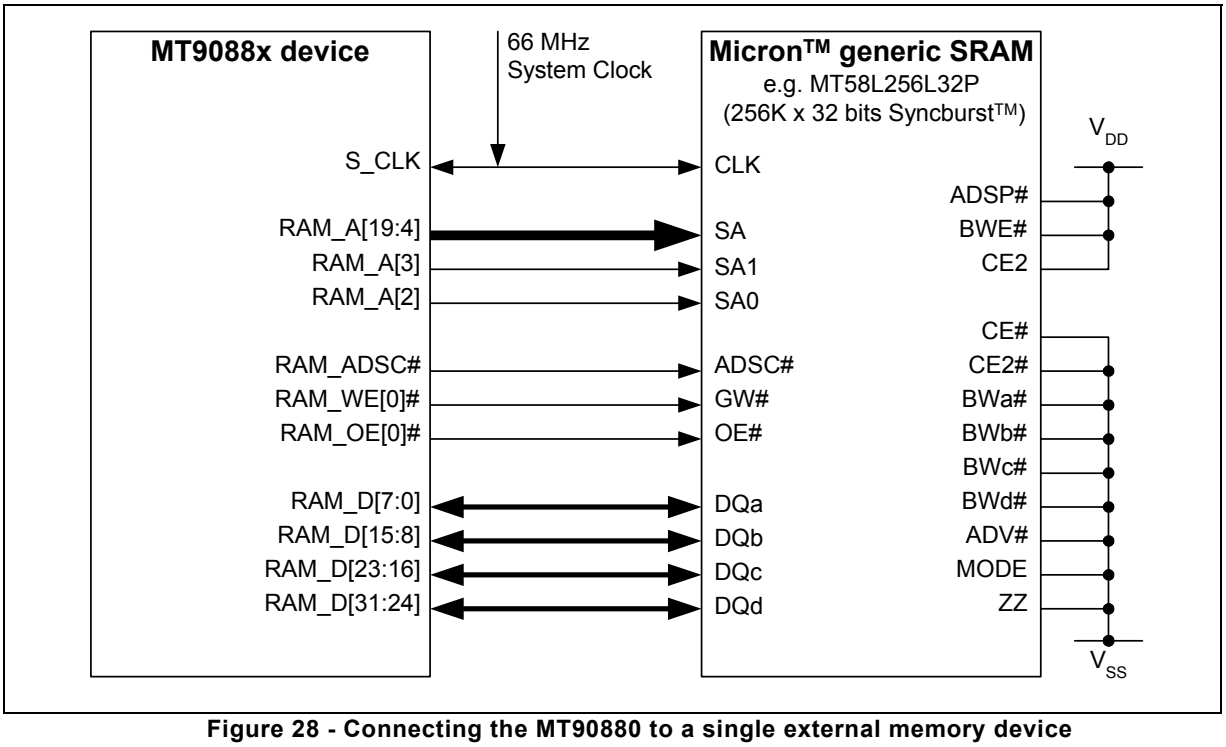

Figure 28 - Connecting the MT90880 to a single external memory device

Connecting to multiple external memory devices

The four devices are connected in banks, and the MT90880 device automatically switches to the next bank when the address exceeds the range of the bank. Therefore the MT90880 needs to be informed during initialisation which memory size is being used. This affects the number of active address signals and the total available memory size, as shown in Table 17. Inactive address signals always output a low state.

Connection to the memory devices is identical to that for a single memory device (see Figure 29), except that the four memory bank write enable signals, and the four memory bank output enable signals are connected to each memory device in turn, e.g. RAM_WE[0]# to device 1, RAM_WE[1]# to device 2, etc. All other connections and tie-offs are the same.

SSRAM size	Total available capacity using 4 devices	Active address signals
16Mbits	8Mbytes	20:2
8Mbits	4Mbytes	19:2
4Mbits	2Mbytes	18:2
2Mbits	1Mbytes	17:2
1Mbits	0.5Mbytes	16:2

Table 18 - Total available memory size, using four external SSRAM devices

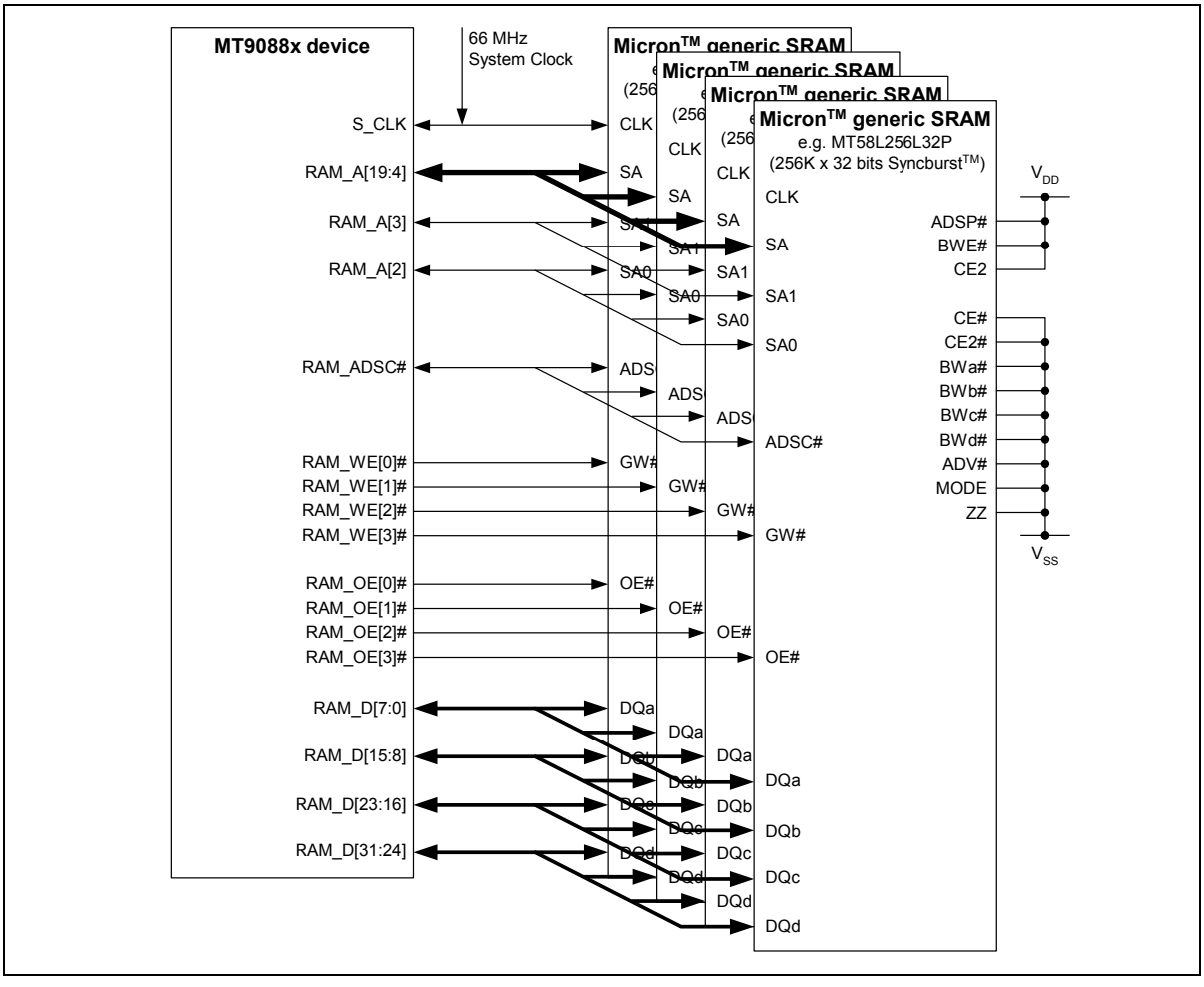

Figure 29 - Connecting the MT90880 to multiple external memory devices

3.9.4 External Memory Interface Timing

The timings for the external memory interface are shown in Table 39. However, calculations to see whether the MT90880 will be capable of operation with a particular memory device also depend on some external factors, notably the skew on the board between the system clock S_CLK at the MT90880 and the memory device.

For example, the diagram in Figure 51 shows the memory write cycle. The MT90880 puts out the data on the bus at time T_{RDV} following the rising edge of S_CLK. This has to meet a setup time to the next rising edge of S_CLK at the RAM (T_{DS}). However, if the clock is skewed on the board such that S_CLK is slightly earlier at the RAM than at the MT90880, the setup time is effectively increased by the amount of skew.

The maximum clock skew between devices can be determined from the following equation:

Maximum allowable clock skew $= T_{S\_CLK} - T_{RDV} - T_{DS}$

For a Micron MT58L256L32P, the minimum RAM data setup time T_{DS} is 1.5 ns when using the 7.5 ns speed grade part. T_{RDV} is 10 ns (from Table 41) and with S_CLK running at 66 MHz, the clock period T_{S_CLK} is 15.15 ns. Therefore,

Maximum allowable clock skew $= 15.15 - 10 - 1.5$

$= 3.65$ ns

This time must be reduced by transmission times of signals across the board between the MT90880 and the memory, e.g. RAM_WE[x]# and the data bus, so in practice, the clock skew must be somewhat lower than this figure suggests.

The read cycle (Figure 49) is not affected by clock skew in the same way. The MT90880 puts out an output enable, RAM_OE[x]#, a time T_{RAV} after S_CLK. In response, the RAM puts the data out onto the bus within a response time (T_{OEQ}), and this data has to meet a setup time T_{RDS} to S_CLK back at the MT90880. Since both the output enable and data setup are timed to the clock at the MT90880, board level clock skew does not influence the read cycle timing.

Therefore the maximum allowable RAM output valid time T_{OEQ} is determined by the equation:

$$\text{Max. allowable RAM output valid time} = T_{S\_CLK} - T_{RAV} - T_{RDS}$$

$$= 15.15 - 7.5 - 3$$

$$= 4.65 \text{ ns}$$

For a 7.5ns speed grade MT58L256L32P, the RAM output valid time T_{OEQ} is 4.2 ns. This part therefore meets the required time with 0.45 ns to spare, although as before, some allowances should also be made for transmission time of the control and data signals across the board.

3.10 PCI Interface

This is a full master/target capable PCI bus interface capable of operation at 33 MHz. Since it has a data width of 32 bits, this can provide up to 1 Gbit/s of data transfer. The PCI interface can be used by an external CPU to provide full access to on-chip registers, and both on-chip memory and the main off-chip packet memory, via the memory management unit. It also contains a DMA controller, which can be used to automatically transfer data between off-chip packet memory and system memory on the PCI bus.

The PCI Core is fully compliant with the PCI Rev 2.2 Specification (reference 2, Table 2) provided by the PCI Special Interest Group (PCI-SIG) responsible for global PCI Standards.

Features include:

- PCI revision 2.2 compliant
- 33 MHz operation
- 32 bit wide data bus
- Master/Target capable
- Scatter/gather DMA controller, capable of following down a linked list of packets for transfer
- Allows target access to all on-chip registers and memory, and to external packet memory

3.10.1 Address and Data Width support

The MT9088x family PCI interface only supports 32-bit PCI addressing and 32-bit wide PCI data transactions. If an 8 or 16 bit data transaction is attempted, the MT9088x responds as though the transaction is 32 bits wide. Hence an 8 or 16 bit read will result in the full 32 bits being placed on the PCI databus, and an 8 or 16 bit write will result in the full 32 bits on the data bus being written into the register or memory location.

The MT9088x requires the master to break 64 bit transactions into two 32 bit accesses, in accordance with the PCI revision 2.2 specification (section 3.2.3).

3.10.2 Target Transaction Support

Supported Transactions

The MT9088x family supports the following PCI target transactions:

- Memory read (single cycle only)
- Memory write (single cycle only)
- Configuration read
- Configuration write

Burst Transactions

Burst transactions are not supported by the MT9088x. The device will respond with a Target Disconnect cycle if a burst access is attempted.

Timeout on Target Read

A PCI Read access timeout is available, set to 32 system clock cycles (485 ns). A timeout will not generate an interrupt or an error, but will result in the data read being 0xFFFFFFFF. The timeout can be disabled.

Fast Back to Back cycles

The MT9088x is capable of receiving fast back to back cycles as a PCI Target device.

PCI Error, Abort and Re-try sources

The following events are among those that will cause a PCI error or abort or re-try event to occur.

- Address Parity error: A Target cycle to the MT9088x with an Address Parity error will cause a PCI System Error to be generated.
- Data Parity error: A Target write cycle to the MT9088x with a Data Parity error will cause a PCI Parity Error to be generated.

3.10.3 Master Support

The MT9088x becomes a PCI Master whenever DMA accesses are invoked to transfer packets from the external packet memory to the Host CPU or vice versa using the DMA controller. One PCI Bus request output and one PCI Bus grant input is provided to allow the MT90880 to request and be granted the PCI bus.

The MT90880 does not generate Fast Back to Back cycles when in Master mode.

Transaction Timeout counters

When the MT9088x is a Master on the PCI bus, two counters are provided to allow the system to recover in the event of the PCI Target behaving abnormally:

The TRDY timeout counter is programmable, and holds the number of cycles the Master will wait before abandoning the cycle. Reasons for timing out include the absence of the TRDY or Stop signals to terminate the cycle normally. This timeout can be disabled.

The Retry timeout counter allows the user to limit the number of re-try cycles that the MT90880 as PCI Master will attempt before abandoning the cycle. This timeout can be disabled.

3.10.4 Configuration and Registers

The MT9088x is configured as a PCI Satellite device. Therefore the PCI interface must be configured by the Host PCI device connected to the bus before the internal register and memory space can be accessed by the Host. This is achieved by programming the device PCI configuration registers.

The PCI configuration registers are used to set up how the MT9088x will fit into the system in which it will be placed. They support the method defined in the PCI Rev 2.2 Specification to determine the size of the memory spaces required by the MT9088x so that the System Host can allocate enough memory. They also contain manufacturer ID information, along with device and revision identifiers, base addresses for the internal register and memory spaces, and various control and status fields. The MT9088x does not include Vital Product Data information or support.

It is not possible to use the MT9088x itself as a PCI Host, and the MT9088x does not provide PCI bus Arbitration capabilities.

The configuration registers are detailed in the document "MT90880 Programmers' Model" (related document 1).

Memory Map

The following diagram (Figure 30) represents the 32-bit PCI bus, and how the address allocation for MT90880 and the External Packet Memory are seen from a system point of view. The register and memory base addresses are programmable, and form part of the MT90880 PCI Configuration registers set.

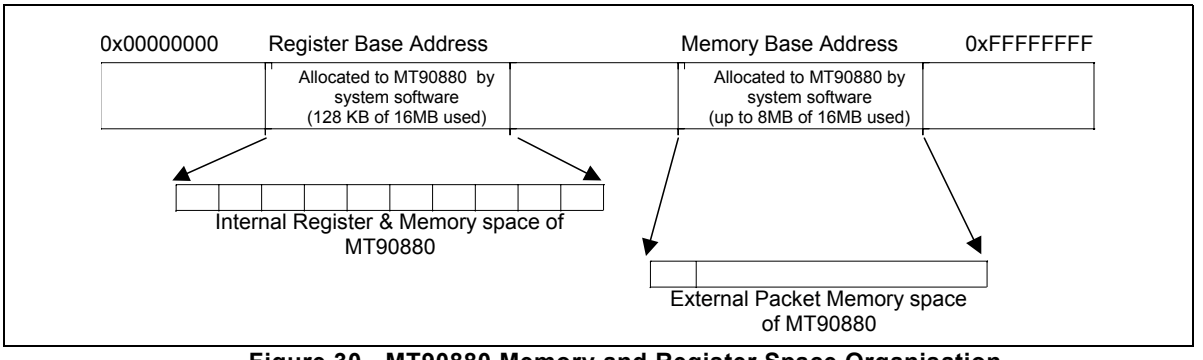

Figure 30 - MT90880 Memory and Register Space Organisation

System Software Reset

The MT90880 has the capability for a software-initiated reset. The PCI interface can be excepted from the general reset if required, by programming a bit in the system control register. This prevents the software reset from destroying the PCI configuration state, allowing the Host CPU to continue to access the device without having to re-program the configuration data.

3.10.5 Signalling environment

The MT9088x PCI Interface is compatible with a 3.3V PCI signalling environment and can be directly connected to other similarly compatible devices in that environment. The 5V PCI signalling environment is not supported as the MT9088x PCI Interface is not 5V tolerant.

3.11 DMA Controller

The MT9088x contains a DMA Controller, automating transfer of data packets between the CPU system memory and the MT9088x's external packet memory (Figure 31). This can be used to send packets from the CPU into the packet network, and to direct packets received from the network to the CPU.

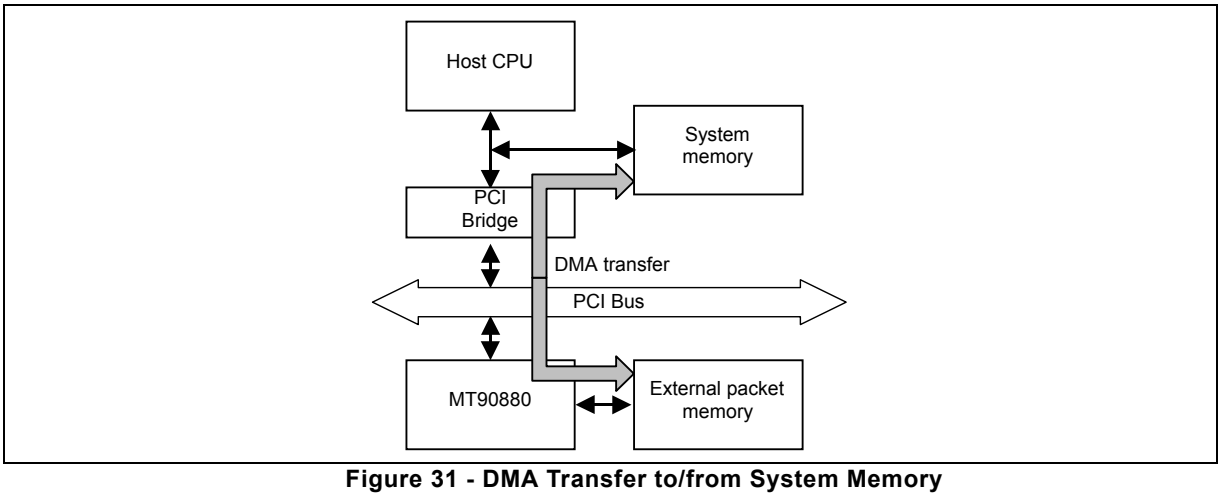

Figure 31 - DMA Transfer to/from System Memory

3.11.1 DMA Descriptor Rings and Lists

Data for transfer to and from the MT90880 is held in a Descriptor Ring or List data structure in system memory (see Figure 32 and Figure 33). Once the CPU has set up the descriptor structure, the DMA engine is simply told where to find the head descriptor. The DMA engine co-ordinates the transfer of packets to and from the data structures with minimal further CPU intervention.

The "ring" data structure (Figure 32) allows a circular buffer to be created. Once the descriptors have been set up, they should never need re-programming, save to change the relevant status and command bits. The ring data structure is useful when a finite size buffer is allocated for packet transfer.

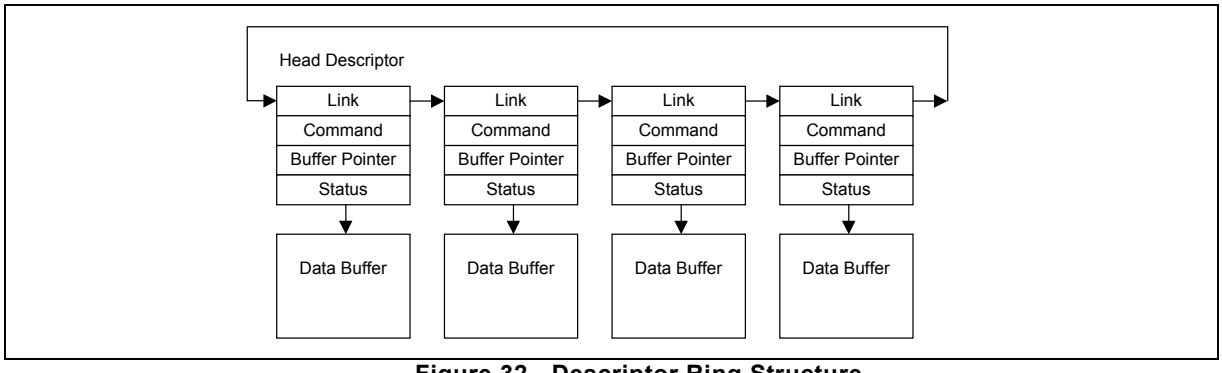

Figure 32 - Descriptor Ring Structure

A list structure (Figure 33) may also be used. This is similar to the ring, but final link pointer does not point back to the head descriptor. This is useful where a more elastic buffer is required, but requires more CPU intervention, since once the list has completed, it is necessary to re-program the DMA to point to the next set of descriptors.

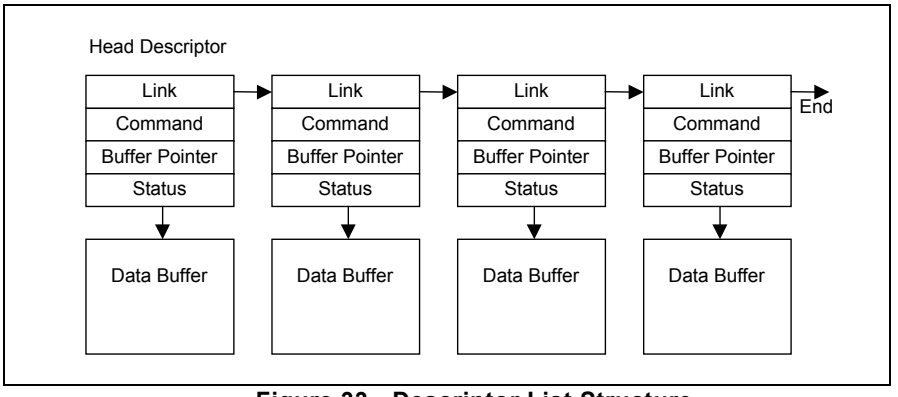

Figure 33 - Descriptor List Structure

Full details of the DMA descriptor data structures, and the fields within the descriptor, are provided in the "MT90880 Programmers' Model " (related document 1).

3.11.2 Data Transfer from CPU to MT90880

For data transfer from the CPU to the MT90880 (denoted "P2L", or Processor to LAN traffic), the DMA engine transfers packets until it reaches the end of the populated data structure. An interrupt can be generated each time a packet is transferred, or alternatively, it can be programmed to occur once the whole populated structure has been transferred. Data transfer is resumed on CPU command once there are more packets to transfer in system memory.

Packet and Descriptor Queues

The MT9088x family devices contain four separate queues to each port in the packet interface. These four queues are assigned different priorities, enabling different classes of service to be defined. Fields within the descriptor command direct the packet to the appropriate queue and port.

In addition, the device supports the use of two entirely separate descriptor structures for P2L traffic. One of these, P2L queue 0, is given higher priority than the other, P2L queue 1. This enables high priority traffic from the CPU to be given preferential access to the DMA. The ratio of DMA bandwidth allocated to each queue can be adjusted from 2:1 in favour of P2L queue 0, to 8:1. Alternatively, strict priority may be used, where packets in P2L queue 1 only get transferred if there are no packets in P2L queue 0. This is the default option.

3.11.3 Data Transfer from MT90880 to CPU

For data transfer from the MT90880 to the CPU (denoted "L2P", or LAN to Processor traffic), the DMA automatically transfers packets as they arrive from the network. The data is placed into the data buffers in system memory until either the data structure is full or there are no more packets to transfer. When subsequent packets arrive, data transfer is resumed automatically, provided that the structure is not full.

An interrupt may be generated each time a packet is transferred to inform the CPU that there is a packet ready to be read, alternatively the CPU can periodically poll the descriptor status to check for new packets. The CPU can process the incoming packet stream at any rate it chooses because the DMA will suspend the transfer when there are no empty descriptors available.

CPU Queues

The MT90880 maintains four separate queues of packets waiting for transfer to the CPU. Each queue has a corresponding descriptor list or ring held in System Memory (see Figure 34). Packets are classified as they are received by the device to determine their destination. The classification process can identify up to four separate traffic types, and each type placed into one of the four queues or directed to the TDM domain. Unmatched traffic is always sent to CPU queue 0.

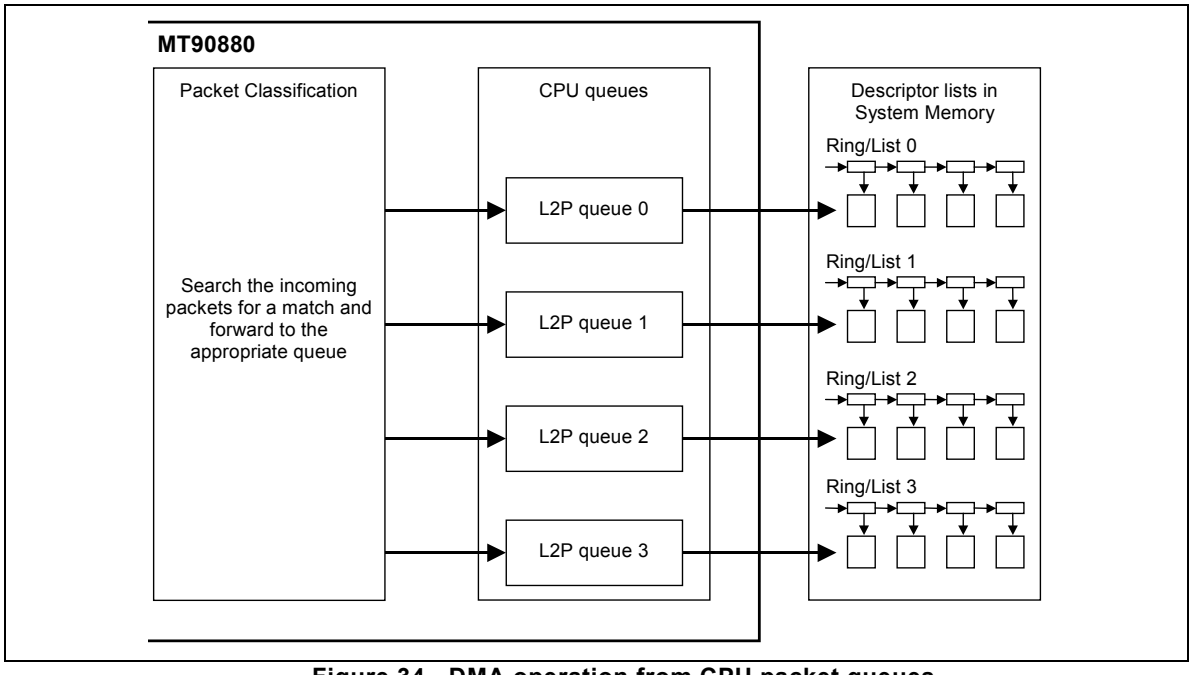

Figure 34 - DMA operation from CPU packet queues

Queue Priority

While the CPU queues have no inherent priority levels associated with them, in that the CPU can choose to service which queue it wants to first, there is a priority level associated with the DMA transfer into system memory. The MT90880 uses a prioritised round robin methodology to determine which queue gets serviced first, with L2P queue 3 having the highest priority and L2P queue 0 the lowest priority. Therefore the DMA will only transfer packets in a given queue if all the higher priority queues are empty.

However, this is only true if there is room in the corresponding data structure in system memory for the DMA to transfer into. If there are no empty data buffers for a particular queue then that queue will be stalled, irrespective of its priority. The DMA engine will then continue to service the lower priority queues. Therefore the CPU can also control the priority by how fast it processes each queue.

3.12 Board Level Test Features

3.12.1 JTAG Support

The JTAG port is used to access the boundary scan logic for board level production testing. A NAND tree test mode is also provided on the MT9088x family. The MT9088x JTAG interface conforms to the IEEE standard 1149.1 (reference 4, Table 2). This standard specifies a design-for-testability technique called "Boundary-Scan Test" (BST). An external Test Access Port (TAP) Controller controls the operation of the boundary scan circuitry.

Details of the contents of the scan register are contained in the BSDL file.

3.12.2 Test Access Port (TAP)

The Test Access Port (TAP) accesses the MT9088x test functions. It consists of four input pins and one output pin as follows:

Test Clock Input (TCK)

TCK provides the clock for the test logic. The TCK does not interfere with any on-chip clock and thus remains independent. The TCK permits shifting of test data into or out of the Boundary-Scan register cells concurrently with the operation of the device and without interfering with the on-chip logic.

Test Mode Select Input (TMS)

The TAP Controller uses the logic signals received at the TMS input to control test operations. The TMS signals are sampled at the rising edge of the TCK pulse. This pin is internally pulled to V_{DD} when it is not driven from an external source.

Test Data Input (TDI)

Serial input data applied to this interface is fed either into the instruction register or into a test data register, depending on the sequence previously applied to the TMS input. Both registers are described in a subsequent section. The received input data is sampled at the rising edge of TCK pulses. This pin is internally pulled to V_{DD} when it is not driven from an external source.

Test Data Output (TDO)

Depending on the sequence previously applied to the TMS input, the contents of either the instruction register or data register are serially shifted out towards the TDO. The data out of the TDO is clocked on the falling edge of the TCK pulses. When no data is shifted through the boundary scan cells, the TDO driver is set to a high impedance state.

Test Reset (TRST)

Reset the JTAG scan structure. This pin is internally pulled to VDD.

3.12.3 Test Access Registers

Instruction Register

The MT9088x uses the public instructions defined in the IEEE 1149.1 standard. The JTAG Interface contains a two-bit instruction register. Instructions are serially loaded into the instruction register from the TDI when the TAP Controller is in its shifted-IR state. These instructions are subsequently de-coded to achieve two basic functions: to select the test data register that may operate while the instruction is current; and, to define the serial test data register path that is used to shift data between TDI and DO during data register scanning.

Test Data Register

As specified in IEEE 1149.1, the MT9088x JTAG Interface contains three test data registers:

Boundary-Scan Register

The Boundary-Scan register consists of a series of Boundary-Scan cells arranged to form a scan path around the boundary of the MT9088x core logic.

Bypass Register

The Bypass register is a single stage shift register that provides a one-bit path from TDI to its TDO.

Device Identification Register

The device identification register is a 32-bit register. The register contents are included in the BSDL file for the chip.

3.13 DPLL Specification

The MT9088x contains an internal Digital Phase Locked Loop (DPLL) which exceeds the requirements of Stratum 4E. This is provided both to synchronise to external references and generate the internal clocks required by the device when operating in synchronous mode.

The DPLL accepts two references, primary and secondary, selected from the incoming ports on the WAN Access Interface. Failure of a reference is automatically detected, and the DPLL can switch between references without introducing bit errors. Reference switching can be performed under automatic or manual control.

The DPLL operates in two main modes, "Master" and "Slave". In Master mode, the DPLL accepts an incoming 8KHz frame reference, and generates a stable, low jitter clock and frame pulse for the selected data format.

Master mode is generally used where the MT9088x is supplying the clocks to the TDM infrastructure, for example, T1/E1 framers, or a TDM backplane such as H.110 or H-MVIP.

In Slave mode, both the incoming and outgoing streams are timed using the external reference, bypassing the DPLL. The DPLL locks to both the clock and the frame pulse, and generates the internal clocking for the device. This mode is used where the master clock and frame references are provided externally.

Features include:

- Exceeds Stratum 4E standard
- Three fundamental modes of operation:
  - "Master" clock-generator mode - locks to incoming 8KHz frame reference
  - "Slave" clock-follower mode - locks to incoming clock and frame reference
  - "Free-run" mode allows clock generation without an external reference
- Primary and secondary references
- Automatic detection of reference failure
- Holdover operation, maintaining last locked reference frequency
- Error-free reference switching, meeting Telcordia GR-1244-CORE (reference 10, Table 3)

3.13.1 Master Mode

In master mode, the DPLL locks to an incoming 8KHz frame reference. It generates the clock and frame pulses required by the selected data format on the WAN Access Interface (see Table 7 on page 22). From a device reset condition or after a reference switch, the DPLL will take up to 50 seconds to phase lock the output signals to the selected input reference signal.

Primary and Secondary References

The DPLL can operate with two references, a primary and a secondary, selected from any of input frame references. The health of both selected references is continuously monitored. When an incoming reference fails, the DPLL can either automatically switch to the alternate reference, or enter holdover operation. The reference switch can then be manually controlled. Should the failed reference return, the device will re-lock to the reference. The DPLL provides bit error free reference switching, meeting the phase slope and MTIE requirements defined by the Telcordia GR-1244-CORE standard (reference 10, Table 3).

Holdover Operation

In Holdover, the DPLL maintains the clock frequency at the value recorded before the reference was judged to have failed. Holdover is typically used for short durations while network synchronization is temporarily disrupted. The initial accuracy of the held frequency is $\pm$0.06 ppm, which translates in the worst case to 42 frame (125 μs) slips in 24 hours.

Two factors affect the frequency stability while in Holdover. The first factor is the drift on the master clock (S_CLK) frequency. Any drift in master clock frequency translates directly into drift on the holdover frequency. Note that the absolute master clock accuracy does not affect the stability of the held frequency, only changes in the master clock frequency while in holdover mode. For example, a 32 ppm master clock may have a temperature coefficient of 0.1 ppm/^{0}C. So, a 10^{0}C change in temperature may result in an additional frequency offset of 1 ppm, over and above the intrinsic accuracy of 0.06 ppm.

The second factor affecting stability is any large frequency jitter on the reference input prior to the reference failure. This could cause the measured frequency to be inaccurate, resulting in an incorrect holdover frequency.

Jitter Transfer

In master mode, jitter on the incoming reference is attenuated by both a Phase Slope Limiter and the internal Loop Filter. The Phase Slope Limiter limits the output phase slope to 7 ns per 125 μs. Therefore, even if the phase slope of the input signal exceeds this rate, such as for low frequency input jitter with a very large amplitude, the maximum output phase slope will be limited to 7 ns per 125 μs. The internal loop filter is a single

pole low-pass filter with a cutoff frequency of 1.25 Hz. This progressively attenuates all jitter above the cutoff frequency at a rate of 20 dB/decade.

Figure 35 shows the DPLL jitter transfer function diagram across a wide range of frequencies, while Figure 36 is the portion of the diagram from Figure 35 around 0dB of the jitter transfer amplitude. From this diagram it is possible to see that the DPLL is a second order, type 2 PLL.

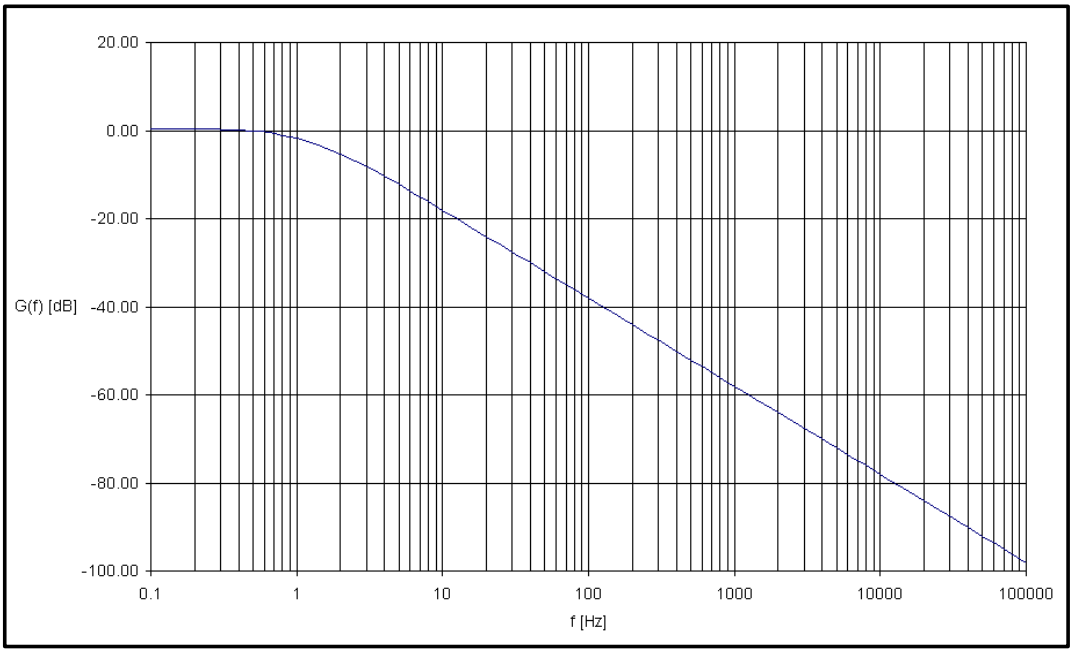

Figure 35 - DPLL Jitter Transfer Function Diagram across a wide range of frequencies.

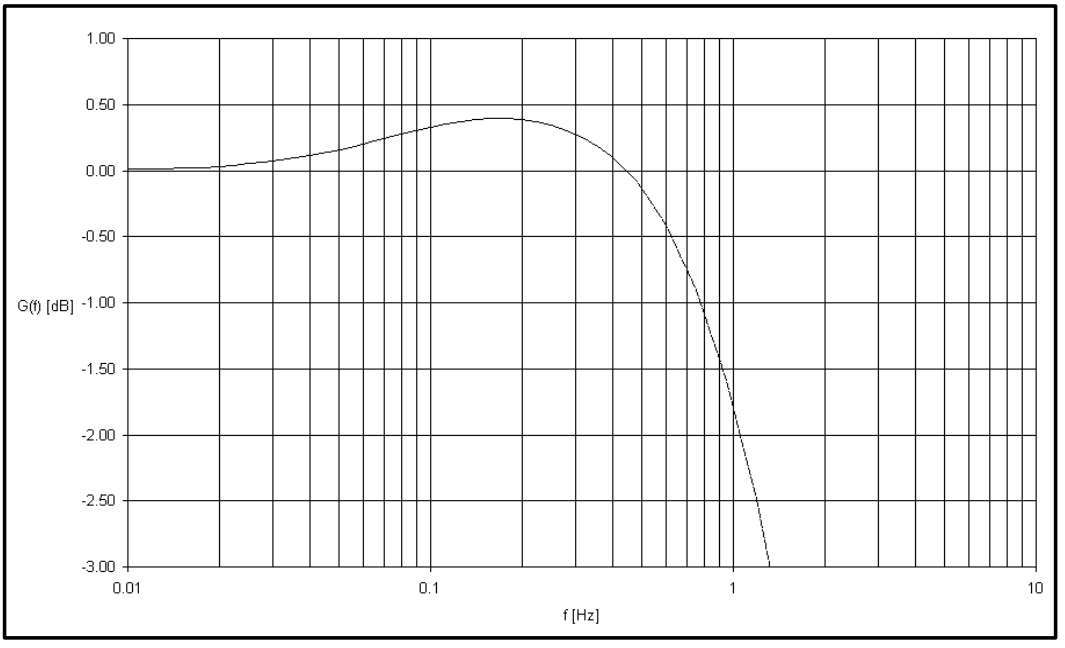

Figure 36 - Detailed DPLL Jitter Transfer Function Diagram

All outputs are derived from the same signal, therefore these diagrams apply to all outputs. Using the method mentioned above, the jitter attenuation can be calculated for all combinations of inputs and outputs.

Frame alignment

When operating in master mode, the incoming frame reference is used solely as a frequency reference. There is no requirement on the width of the frame pulse. The output frame pulse must therefore be used as the master for the devices connected to the MT9088x's WAN Access Interface. There is no guaranteed phase relationship between the input frame pulse and the output master frame pulse.

3.13.2 Slave Mode

Slave mode is used where the master clock and frame references are provided externally. Both the incoming and outgoing streams are timed using the external reference, bypassing the DPLL. The DPLL locks to the incoming clock and frame pulse, and is used solely to generate the internal clocks required by the device.

The input frequency combinations and frame pulse widths for correct slave mode operation are given in Table 7 on page 22. These correspond to the data formats selected for the WAN Access Interface.

When operating in slave mode, the MT9088x is capable of accepting timing references with jitter meeting ITU-T standards G.823 and G.824 when operating in either Generic E1 or ST-Bus formats at 2.048 Mbit/s. When operating in ST-bus format at 8.192 Mbit/s, the maximum allowable input jitter above 10 KHz is 61 ns (1 UI at the input clock rate of 16.384 MHz). This translates to 0.125 UI at 2.048 MHz, or just outside the G.824 specification of 0.2 UI.

3.13.3 Free-run Mode

In the Free-run Mode, the DPLL provides timing and synchronization signals, which are based on the frequency of the master clock (S_CLK) only, and are not synchronized to a reference input. The DPLL outputs have a frequency accuracy of ± 0.005 ppm plus the accuracy of the master clock. For example, clock output C8OB will have a frequency of 8.192 MHz ± S_CLK accuracy ± 0.005 ppm).

Free-run Mode is typically used when a master clock source is required, or immediately following system power-up before network synchronization is achieved.

3.13.4 DPLL Performance Parameters

Table 19 lists the key performance parameters of the DPLL.

Parameter	Min	Max	Units	Comment
Intrinsic jitter		± 3.8	ns	Note 1
Lock range		± 245	ppm	Note 2
Lock time		50	s	
Free-run frequency accuracy		± 0.005	ppm	Note 3
Master Mode Parameters (the following parameters are only applicable in master mode):				
Phase slope output		56	ppm	Note 4
Holdover frequency accuracy		± 0.06	ppm	Note 5
MTIE during a reference switch		20	ns	
Slave Mode Parameters (the following parameters are only applicable in slave mode):				
Input clock jitter tolerance (above 10 KHz)		1	UI	Note 6

Table 19 - DPLL Performance Parameters

Note 1: For the DPLL, the intrinsic output jitter is determined by the frequency of the master clock input (S_CLK). The edges of the DPLL output clocks are synchronous to both edges of the master clock, giving a resolution of 7.6 ns for a 66 MHz clock. In addition, any jitter present on the master clock is transferred without attenuation to the output clocks.

Note 2: Note that the locking range is related to the master clock frequency (S_CLK). For example, if the master clock is -100 ppm, the whole locking range also shifts -100 ppm downwards from -345ppm to 145ppm.

Note 3: Assumes ideal master clock (S_CLK) frequency of 66.0 MHz. Any deviation of master clock frequency from 66 MHz directly translates into a degradation of free-run frequency accuracy.

Note 4: This is equivalent to 7ns per 125 μs, or better than specified in GR-1244-CORE (reference 10) which states that the phase slope may not exceed 81 ns/1.327 ms (61 ppm).

Note 5: Assumes no drift on the master clock (S_CLK) frequency. Any drift in master clock frequency after holdover is entered will translate directly into a holdover frequency error.

Note 6: Maximum jitter is 1 UI (1 bit period) of input clock frequency. With input clock frequencies of 2.048 MHz and 4.096 MHz, this comfortably exceeds ITU-T standards G.823 and G.824 (references 11 and 12). With a 16.384 MHz input clock, this translates to 61 ns, or 0.125 UI at 2.048 MHz. This is just outside the G.823 specification of 0.2 UI.

4.0 Memory Map and Register definitions

Details of the memory map and register definitions are included in the document "MT90880 Programmers' Model" (related document 1).

5.0 Special Note to Users: MT9088x handling of received corrupted Ethernet packets and dealing with the next valid packet

Issue

Whenever the MT9088x device receives Ethernet packets which are corrupted (CRC error, too short or too long etc.) the context related to the error packet will have incorrect data. The next valid packet following the error packet will be lost and this may belong to a different context.

Recommendation

This device should not be used in a half-duplex network application. The final link to the device should be as short as possible, (for example from an Ethernet Switch), in order to minimise the probability of corrupted packets on the link.

6.0 Physical Specification

The device is contained in a 456-ball plastic ball grid array (456 PBGA) package:

- Body Size: 27mm x 27mm
- Ball Count: 456
- Ball Pitch: 1.0mm
- Ball Matrix: 26 x 26 (partially populated with a 6 x 6 GND matrix in the centre)
- Ball Diameter 0.63mm
- Total Package Thickness 2.03mm

Package is viewed from the top side (i.e. through top of the package). Note ball A1 is non-chamfered corner.

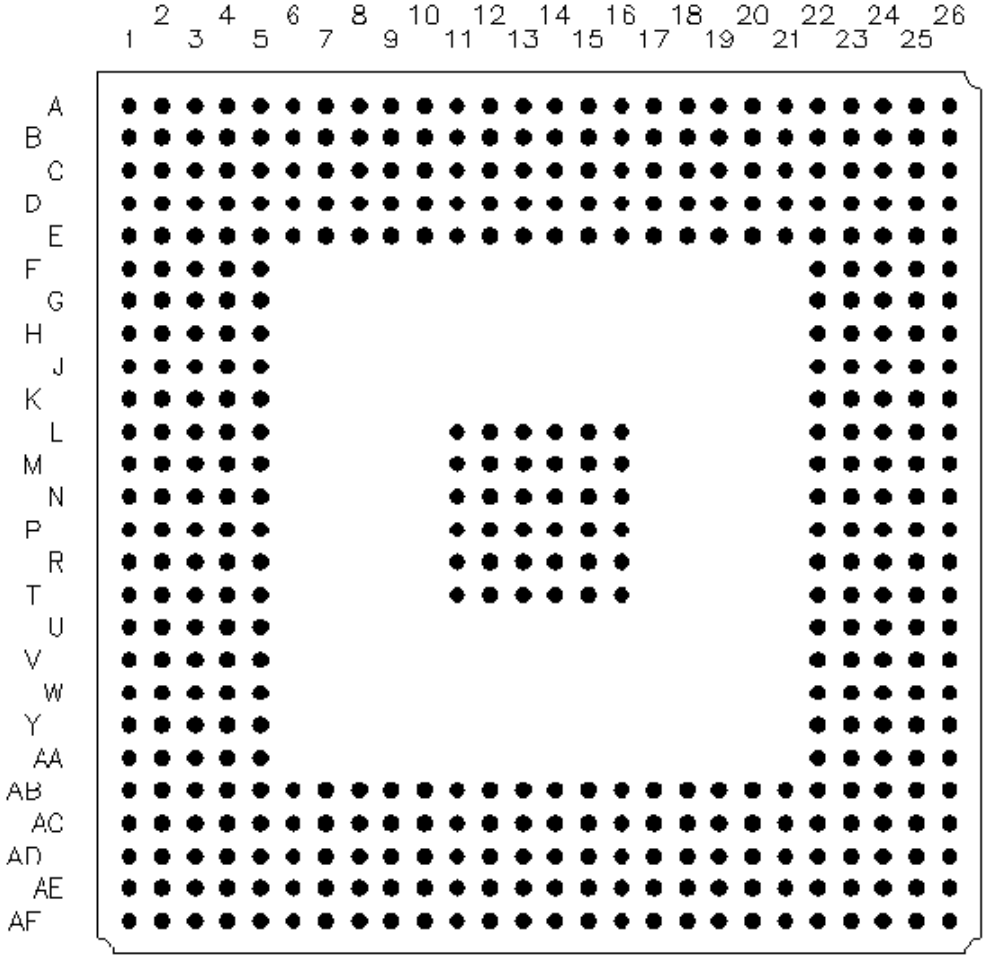

Figure 37 - Package View and Ball Positions

7.0 External Interface Description

The following key applies to all tables:
O Output
I Input
D internal 100KΩ pull-down resistor present
U internal 100KΩ pull-up resistor present

7.1 WAN Access Interface

All WAN Access Interface signals are 5V tolerant.

All WAN Access Interface outputs are high impedance while S_RST is low.

Signal	I/O	Package Balls	Description
WAN_STO[31:24]	O	T22 [31], U26 [30], R26 [29], P26 [28], N25 [27], M24 [26], L25 [25], K24 [24]	WAN Interface serial output streams Operate at 2.048 Mbit/s All variants
WAN_STO[23:8]	O	J25 [23], G26 [22], G25 [21], G24 [20], D26 [19], D22 [18], B24 [17], B23 [16], B22 [15], A22 [14], E18 [13], E17 [12], C18 [11], E16 [10], A17 [9], A16 [8]	WAN Interface serial output streams Operate at 2.048 Mbit/s MT90880 and MT90881 variants only.
WAN_STO[7:0]	O	B15 [7], A14 [6], B13 [5], B12 [4], E11 [3], A10 [2], E10 [1], E9 [0]	WAN Interface serial output streams Operate at 2.048 and 8.192 Mbit/s MT90880 and MT90881 variants only.
WAN_STI[31:24]	I D	W26 [31], U25 [30], R25 [29], M23 [28], L23 [27], L22 [26], K22 [25], K25 [24]	WAN Interface serial input streams Operate at 2.048 Mbit/s All variants.
WAN_STI[23:8]	I D	J22 [23], H25 [22], G23 [21], F25 [20], E25 [19], C24 [18], D21 [17], E20 [16], D20 [15], D19 [14], A21 [13], C19 [12], A19 [11], A18 [10], B17 [9], E15 [8]	WAN Interface serial input streams Operate at 2.048 Mbit/s MT90880 and MT90881 variants only
WAN_STI[7:0]	I D	D14 [7], B14 [6], D12 [5], A12 [4], A11 [3], C11 [2], A9 [1], C9 [0]	WAN Interface serial input streams Operate at 2.048 and 8.192 Mbit/s MT90880 and MT90881 variants only
WAN_FRMI[31:24]	I D	W25 [31], U24 [30], R24 [29], P25 [28], N26 [27], M25 [26], L26 [25], K26 [24]	WAN Interface frame pulse inputs. Programmable as active high or low. All variants
WAN_FRMI[23:0]	I D	J26 [23], H26 [22], H24 [21], F26 [20], F23 [19], C26 [18], A25 [17], C23 [16], C22 [15], C21 [14], C20 [13], D18 [12], B19 [11], D17 [10], D16 [9], B16 [8], C15 [7], C14 [6], A13 [5], C13 [4], D11 [3], D10 [2], C10 [1], D9 [0]	WAN Interface frame pulse inputs. Programmable as active high or low. MT90880 and MT90881 variants only.
WAN_CLKI[31:24]	I D	U22 [31], V26 [30], N23 [29], P24 [28], M22 [27], N24 [26], K23 [25], L24 [24]	WAN Interface clock inputs. Programmable as active high or low. All variants

Table 20 - WAN Access Interface

Signal	I/O	Package Balls	Description
WAN_CLKI[23:0]	I D	J23 [23], J24 [22], H22 [21], G22 [20], E26 [19], E24 [18], A26 [17], A24 [16], A23 [15], E19 [14], B21 [13], B20 [12], A20 [11], B18 [10], C17 [9], C16 [8], D15 [7], A15 [6], D13 [5], E12 [4], C12 [3], B11 [2], B10 [1], B9 [0]	WAN Interface clock inputs. Programmable as active high or low. MT90880 and MT90881 variants only.
WAN_FRMO	O	V23	WAN Interface master frame pulse output
WAN_CLKO	O	V22	WAN Interface master clock output

Table 20 - WAN Access Interface (continued)

7.2 Local TDM Interface

All Local TDM Interface signals are 5V tolerant.

All Local TDM Interface outputs are high impedance while S_RST is low.

Signal	I/O	Package Balls	Description
LOC_STI [31:24]	I D	B6 [31], C7 [30], A5 [29], C5 [28], A4 [27], D6 [26], A3 [25], B3 [24]	Local TDM interface serial input streams. Operate at 2.048 Mbit/s MT90880 and MT90882 variants only.
LOC_STI [23:16]	I D	D5 [23], D4 [22], B2 [21], F4 [20], D3 [19], G5 [18], E3 [17], G4 [16]	Local TDM interface serial input streams. Operate at 2.048 Mbit/s MT90880 variant only.
LOC_STI [15:8]	I D	H5 [15], F3 [14], F2 [13], J5 [12], J4 [11], K5 [10], H2 [9], H1 [8]	Local TDM interface serial input streams Operate at 2.048 and 4.096 Mbit/s MT90880 variant only
LOC_STI [7:0]	I D	J2 [7], J1 [6], K3 [5], K1 [4], M4 [3], N4 [2], M3 [1], M1 [0]	Local TDM interface serial input streams Operate at 2.048, 4.096 and 8.192 Mbit/s MT90880 variant only
LOC_STO [31:24]	O	E8 [31], D7 [30], C6 [29], B5 [28], E7 [27], B4 [26], C4 [25], A2 [24]	Local TDM interface serial output streams Operate at 2.048 Mbit/s MT90880 and MT90882 variants only
LOC_STO [23:16]	O	C3 [23], E4 [22], A1 [21], B1 [20], C2 [19], C1 [18], D2 [17], D1 [16]	Local TDM interface serial output streams Operate at 2.048 Mbit/s MT90880 variant only
LOC_STO [15:8]	O	H4 [15], E1 [14], G3 [13], F1 [12], G2 [11], H3 [10], G1 [9], K4 [8]	Local TDM interface serial output streams Operate at 2.048 and 4.096 Mbit/s MT90880 variant only

Table 21 - Local TDM Interface

Signal	I/O	Package Balls	Description
LOC_STO [7:0]	O	L5 [7], L4 [6], K2 [5], L3 [4], L2 [3], P4 [2], M2 [1], N3 [0]	Local TDM interface serial output streams Operate at 2.048, 4.096 and 8.192 Mbit/s MT90880 variant only
c4ob	O	B8	TDM interface 4.096 MHz clock All variants
fp4ob	O	C8	8 KHz frame pulse for C4OB clock All variants
c8ob	O	A7	TDM interface 8.192 MHz clock All variants
fp8ob	O	A6	8 KHz frame pulse for C8OB clock. All variants
c16ob	O	B7	TDM interface 16.384 MHz clock All variants
fp16ob	O	D8	8 KHz frame pulse for C16OB clock. All variants
ode	I D	A8	WAN and TDM serial output enable. High to enable outputs, low for high impedance. All variants

Table 21 - Local TDM Interface (continued)

7.3 Packet Interfaces

7.3.1 MII Interfaces

Data for the MII packet switching is based on Specification IEEE Std 802.3u – 1995. The MII has separate transmit and receive circuits, and consequently the characteristics are shown in the relevant tables.

All MII signals are 5V tolerant.

All MII outputs are high impedance while S_RST is low.

Signal	I/O	Package Balls	Description
m_mdc	O	AF9	MII management data clock. Common for both MII ports.
m_mdio	IOU	AC10	MII management data I/O Common for both MII ports; 2.5MHz
MII Port A			
m_mint0	I U	AF6	MII management interrupt for port A
m0_txd[3:0]	O U	AC8 [3], AF5 [2], AE6 [1], AD7 [0]	Transmit data
m0_txen	O U	AD6	Transmit enable
m0_txclk	I U	AB9	Transmit clock
m0_rxd[3:0]	I U	AF8 [3], AE8 [2], AE9 [1], AD9 [0]	Receive data
m0_rxdv	I D	AF7	Receive data valid
m0_rxclk	I U	AB10	Receive clock
m0_rxer	I D	AD8	Receive error
m0_crs	I D	AC9	Carrier sense
m0_col	I D	AE7	Collision detect
MII Port B			
m_mint1	I U	AE11	MII management interrupt for port B
m1_txd[3:0]	O U	AC11 [3], AE10 [2], AF10 [1], AD11 [0]	Transmit data
m1_txen	O U	AD10	Transmit enable
m1_txclk	I U	AB12	Transmit clock
m1_rxd[3:0]	I U	AF12 [3], AD13 [2], AE13 [1], AF13 [0]	Receive data
m1_rxdv	I D	AE12	Receive data valid
m1_rxclk	I U	AC13	Receive clock
m1_rxer	I D	AD12	Receive error
m1_crs	I D	AC12	Carrier sense
m1_col	I D	AF11	Collision detect

Table 22 - MII Interfaces

7.3.2 RMII Interfaces

The RMII is a low pin count MII interface. It has no defined standard, but the RMII Consortium publishes details (reference 3, Table 2). The MT90880 RMII Interface shares the same pins as the MII interface with the exception of the REF_CLK.

The RMII comprises a low pin count Reduced Media Independent Interface™ (RMII™) specification intended for use between Ethernet PHYs and Switch ASICs. Under IEEE 802.3u [2] an MII, comprised of 16 pins for data and control, is defined. In devices incorporating many MACs or PHY interfaces such as switches, the number of pins can add significant cost as the port counts increase. The purpose of this interface is to provide a low cost alternative to the IEEE 802.3u [2] MII. Architecturally, the RMII specification provides for an additional reconciliation layer on either side of the MII but can be implemented in the absence of an MII. The management interface (MDIO/MDC) is assumed to be identical to that defined in IEEE 802.3u [2].

All RMII signals are 5V tolerant.

All RMII outputs are high impedance while S_RST is low.

Signal	I/O	Package Balls	Description
ref_clk	I U	AB11	Synchronous clock reference for receive, transmit and control interface. Common to both RMII ports.
RMII Port A			
rm0_rxd[1:0]	I U	AE9 [1], AD9 [0]	Receive data
rm0_rxdv/ crs	I D	AF7	Carrier sense & receive data valid
rm0_txen	O U	AD6	Transmit enable
rm0_txd[1:0]	O U	AE6 [1], AD7 [0]	Transmit data
RMII Port B			
rm1_rxd[1:0]	I U	AE13 [1], AF13 [0]	Receive data
rm1_rxdv/ crs	I D	AE12	Carrier sense & receive data valid
rm1_txen	O U	AD10	Transmit enable
rm1_txd[1:0]	O U	AF10 [1], AD11 [0]	Transmit data

Table 23 - RMII Interface

7.4 PCI Interface

The PCI signals are **NOT** 5V tolerant.

All PCI outputs are high impedance while S_RST is low.

Active low signals are designated by a # suffix, in accordance with the convention used in the PCI specification.

Signal	I/O	Package Balls	Description
pci_clk	I	AF20	PCI clock
pci_rst#	I	AE19	PCI reset
pci_req#	O	AC19	PCI request (open drain)
pci_gnt#	I	AD19	PCI grant
pci_ad[31:0]	I O	AE20 [31], AB19 [30], AF21 [29], AC20 [28], AD20 [27], AE21 [26], AF22 [25], AD21 [24], AF23 [23], AD22 [22], AE23 [21], AC21 [20], AE24 [19], AC22 [18], AD24 [17], AC23 [16], AC26 [15], AB25 [14], W23 [13], AA24 [12], AB26 [11], AA25 [10], Y24 [9], AA26 [8], V24 [7], U23 [6], V25 [5], T23 [4] T24 [3], R23 [2], T25 [1], T26 [0]	PCI address / data
pci_cbe[3:0]#	I O	AE22 [3], AB23 [2], Y23 [1], W24 [0]	PCI control / byte enable
pci_par	I O	AC25	PCI parity
pci_frame#	I O	AE25	PCI frame
pci_trdy#	I O	AA23	PCI target ready
pci_irdy#	I O	AF26	PCI initiator ready
pci_stop#	I O	AC24	PCI stop
pci_devsel#	I O	AE26	PCI device select
pci_idsel	I	AB20	PCI ID select
pci_perr#	I O	AD26	PCI parity error
pci_serr#	I O	AB24	PCI system error (open drain)
pci_lock#	I O	Y22	PCI lock. Refer to PCI spec section 2.2.3
pci_inta#	I O	AF19	PCI interrupt (open drain)
PCI_M66EN	I O	Y25	33MHz /66MHz select

Table 24 - PCI Interface

7.5 External Memory Interface

The external memory signals are **NOT** 5V tolerant.

Active low signals are designated by a # suffix, in accordance with the convention used in common memory data sheets

Signal	I/O	Package Balls	Description
ram_d[31:0]	IOU	N2 [31], R4 [30], N1 [29], R5 [28], P1 [27], T4 [26], P2 [25], P3 [24], R1 [23], AB4 [22], AC2 [21], AD1 [20], AC3 [19], AD2 [18], AC4 [17], AE1 [16], AE2 [15], AD3 [14], AC5 [13], AF1 [12], AF2 [11], AC6 [10], AE3 [9], AF3 [8], AD4 [7], AB7 [6], AE4 [5], AF4 [4], AD5 [3], AC7 [2], AE5 [1], AB8 [0]	Buffer memory data
ram_a[22:2]	O	T5 [22], R3 [21], U4 [20], T1 [19], U5 [18], T2 [17], T3 [16], U1 [15], U2 [14], U3 [13], V4 [12], V1 [11], V5 [10], V2 [9], W4 [8], W1 [7], W2 [6], Y1 [5], W5 [4], W3 [3], Y4 [2]	Buffer memory address (up to 8 MBytes)
ram_adsc#	O U	AB3	Buffer memory address status control
ram_we[3:0]#	O U	Y5 [3], AA1 [2], Y3 [1], AA2 [0]	Buffer memory write chip select
ram_oe[3:0]#	O U	AB1 [3], AA4 [2], AB2 [1], AC1 [0]	Buffer memory read chip select

Table 25 - External Memory Interface

7.6 System Control Interface

The system signals are 5V tolerant.

RESOUT# is low while S_RST# is low.

The core of the chip will be held in reset for 16348 S_CLK cycles after S_RST# has gone high and 16348 PCI_CLK cycles after PCI_RST# have gone high to allow the PLLs to lock.

Signal	I/O	Package Balls	Description
s_clk	I	B25	System clock
s_rst#	I	D24	Reset input
resout#	O	C25	Reset PHY

Table 26 - System Control Interface

7.7 Test Facilities

7.7.1 JTAG Interface

All JTAG signals are 5V tolerant.

Signal	I/O	Package Balls	Description
jtag_trst#	I	AB17	JTAG reset
jtag_tck	I	AB18	JTAG test clock
jtag_tms	I	AC18	JTAG test mode select
jtag_tdi	I	AD18	JTAG test data input
jtag_tdo	O	AE18	JTAG test data output

Table 27 - JTAG Interface

7.7.2 Test Facility

All test signals are 5V tolerant except for IDDQ, SCLK_AT1 and PCLK_AT1.

Signal	I/O	Width	Description
t_mode[1:0]	IOU	AF14 [1], AC14 [0]	Test – Set Mode upon Reset 00 - PLL test mode 01 - Scan test mode 10 - Board level test mode 11 - Normal operation
t_d[15:0]	O U	AC15 [15], AE14 [14], AB15 [13], AD14 [12], AF15 [11], AE15 [10], AD15 [9], AF16 [8], AB16 [7], AE16 [6], AC17 [5], AD16 [4], AF17 [3], AE17 [2], AD17 [1], AF18 [0]	Test MUX output. Also used as bootstrap pins to configure the device following a reset.
iddq	I	AD23	IDDQ test enable. Set low for normal operation.
sclk_AT1	O	B26	Analog test output for system clock PLL. Leave unconnected in normal operation.
pclk_AT1	O	AF24	Analog test output for PCI clock PLL. Leave unconnected in normal operation.
ram_clk	O	Y2	66MHz memory clock output. This is used for test purposes only. Leave unconnected in normal operation.

Table 28 - Test Facilities

7.7.3 Test Operating Modes

1) Normal operating mode (T_MODE[1:0] = '11')

The device samples the T_D pins to determine the required operating mode shortly after reset is released. These pins are known as the "bootstrap controls". The device has internal pull-up (100KΩ ±30%) resistors connected to the T_D bus pins. To pull-low connect a 10KΩ pull-down resistor to ground. Bootstrap controls are as follows:

T_D	Function	Notes
15	Reserved	
14	Reserved	Pull low for normal operation.
13:11	Reserved	
10:8	Selects delay of RAM_CLK output relative to S_CLK.	Leave unconnected in normal operation.
7	Pull low to bypass PCI_CLK PLL.	Leave unconnected in normal operation.
6	Pull low to bypass the S_CLK PLL.	Leave unconnected in normal operation.
5	Controls m_mint0 and m_mint1 polarity - pull low for active low and leave unconnected for active high.	See Table 21, page 60.
4	Pull low to enable RAM_CLK output (otherwise it will be high impedance).	Leave unconnected in normal operation.
3	Pull low for freeze enable.	Leave unconnected in normal operation.
2	Pull low for simulation speed-up. This reduces the number of cycles that the chip is held in reset after S_RST# and PCI_RST# have gone high from 16348 to 256. This should not be used during normal operation, since the PLLs will not have time to lock.	Leave unconnected in normal operation.
0:1	Reserved	

Table 29 - Bootstrap Pin Functions

2) Board level continuity test mode (T_MODE[1:0] = '10')

In continuity test mode the core of the chip is held in reset. Pin T_D[10] selects the board level test mode.

3) Scan test mode (T_MODE[1:0] = '01')

System clock and PCI clock PLLs are put into bypass mode. The core PCI and system resets are driven directly from the pins i.e. the logic which holds the resets asserted until the PLLs have had time to lock is disabled.

4) PLL test mode (T_MODE[1:0] = '00')

The core of the chip is held in reset. Test bus pins are configured for direct test access to the system clock and PCI clock PLLs.

7.8 Power and Ground Connections

Signal	Package Balls	Description
VDD33	AA5, AA22, AB5, AB6, AB13, AB14, AB21, AB22, E5, E6, E13, E14, E21, E22, F5, F22, M5, N5, N22, P5, P22, R22	3.3V VDD power supply
VDD18	AA3, E2, F24, H23, J3, L1, M26, P23, R2, V3, W22, Y26	1.8V VDD power supply
GROUND	L11, L12, L13, L14, L15, L16, M11, M12, M13, M14, M15, M16, N11, N12, N13, N14, N15, N16, P11, P12, P13, P14, P15, P16, R11, R12, R13, R14, R15, R16, T11, T12, T13, T14, T15, T16, AD25, E23	0V power supply
A1VDD	D25	1.8V VDD power supply
A2VDD	AF25	1.8V VDD power supply
no connection	D23, AC16	

Table 30 - Power and Ground Connections

8.0 DC Characteristics

Absolute Maximum Ratings - Voltage measurements are with respect to ground (V_{SS}) unless otherwise stated.

Parameter	Symbol	Min	Max	Units
I/O Supply Voltage	V_{DD_IO}	-0.5	5.0	V
Core Supply Voltage	V_{DD_CORE}	-0.5	2.5	V
PLL Supply Voltage	V_{DD_PLL}	-0.5	2.5	V
Input Voltage	V_I	-0.5	V_{DD} + 0.5	V
Input Voltage (5V tolerant inputs)	V_{I_5V}	-0.5	7.0	V
Continuous current at digital inputs	I_{IN}	-	±10	mA
Continuous current at digital outputs	I_O	-	±15	mA
Package power dissipation	PD	-	4	W
Storage Temperature	TS	-55	+125	°C

Table 31 - Absolute Maximum Ratings

Exceeding these figures may cause permanent damage. Functional operation under these conditions is not guaranteed.

The core and PLL supply voltages must never be allowed to exceed the I/O supply voltage by more than 0.5V during power-up. Failure to observe this rule could lead to a high-current latch-up state, possibly leading to chip failure, if sufficient cross-supply current is available. To be safe ensure the I/O supply voltage supply always rises earlier than the core and PLL supply voltages.

8.1 Recommended Operating Conditions - Voltage measurements are with respect to ground (V_{SS}) unless otherwise stated.

Characteristics	Symbol	Min	Typ	Max	Units	Test Condition
Operating Temperature	T_{OP}	-40	25	+85	°C	
Junction temperature	T_J	-40		125	°C	
Package thermal resistance to ambient	θ_{JA}		33		°C/w	
Positive Supply Voltage, I/O	V_{DD_IO}	3.0	3.3	3.6	V	
Positive Supply Voltage, Core	V_{DD_CORE}	1.65	1.8	1.95	V	
Positive Supply Voltage, Core	V_{DD_PLL}	1.65	1.8	1.95	V	
Input Voltage Low - all inputs	V_{IL}			0.8	V	
Input Voltage High	V_{IH}	2.0		V_{DD_IO}	V	
Input Voltage High, 5V tolerant inputs	V_{IH_5V}	2.0		5.5	V	

Table 32 - Recommended Operating Conditions

Typical figures are at 25°C and are for design aid only.

8.2 DC Characteristics

Characteristics	Symbol	Min	Typ	Max	Units	Test Condition
Input Leakage	I_{LEIP}			±1	mA	No pull up/down V_{DD} = 3.6V
Output (High impedance) Leakage	I_{LEOP}			1	mA	No pull up/down V_{DD} = 3.6V
Input Capacitance	C_{IP}		2		pF	
Output Capacitance	C_{OP}		4		pF	
Pullup Current	I_{PU}		-33		mA	Input at 0V
Pulldown Current	I_{PD}		33		mA	Input at V_{DD}
Core 1.8V supply current	I_{DD_CORE}		220		mA	
PLL 1.8V supply current	I_{DD_PLL}			1.25	mA	Note 1
I/O 3.3V supply current	I_{DD_IO}		100		mA	

Table 33 - DC Characteristics

Typical characteristics are at 1.8V core, 3.3V I/O, 27°C and typical processing. The min and max values are defined over all process conditions, from -40 to 125°C junction temperature, core voltage 1.65 to 1.95 and I/O voltage 3.0 and 3.6V unless otherwise stated.

Note 1: There are two analog PLLs with separate 1.8V supplies. Each supply consumes current of I_{DD_PLL}.

8.2.1 Input Levels

Characteristics	Symbol	Min	Typ	Max	Units	Test Condition
Input Low Voltage	V_{IL}			0.8	V	
Input High Voltage	V_{IH}	2.0			V	
Positive Schmitt Threshold	V_{T+}		1.6		V	
Negative Schmitt Threshold	V_{T-}		1.2		V	

Table 34 - Input Levels

8.2.2 Output Levels

Characteristics	Symbol	Min	Typ	Max	Units	Test Condition
Output Low Voltage	V_{OL}			0.4	V	
Output High Voltage	V_{OH}	2.4			V	
Output Low Current	I_{OL}		1.6		mA	PCI Bus
Output High Current	I_{OH}		1.2		mA	PCI Bus
Output Low Current	I_{OL}		1.6		mA	Note 1
Output High Current	I_{OH}		1.2		mA	Note 1
Output Low Current	I_{OL}		1.6		mA	Note 2
Output High Current	I_{OH}		1.2		mA	Note 2

Table 35 - Output Levels

Note 1: WAN and Local TDM busses. External RAM busses, except for RAM_CLK. T_D bus.
Note 2: RAM_CLK. MII and RMII busses except M1_TXEN, M0_TXEN and M_MDC

9.0 AC Characteristics

9.1 WAN Access Interface

9.1.1 Slave Clock Mode

Data format	Parameter	Symbol	Min	Typ	Max	Units	Notes
ST-bus 8.192 Mbit/s mode	WAN_CLKI frequency	f_{C16P}		16.384		MHz	Note 1
	WAN_CLKI High Width	t_{C16H}		30		ns	
	WAN_CLKI Low Width	t_{C16L}		30		ns	
ST-bus 2.048 Mbit/s mode	WAN_CLKI frequency	f_{C4P}		4.096		MHz	Note 1
	WAN_CLKI High Width	t_{C4H}		122		ns	
	WAN_CLKI Low Width	t_{C4L}		122		ns	
Generic E1 2.048 Mbit/s mode	WAN_CLKI frequency	f_{C2P}		2.048		MHz	Note 1
	WAN_CLKI High Width	t_{C2H}		244		ns	
	WAN_CLKI Low Width	t_{C2L}		244		ns	
All modes	WAN_FRMI Setup Time	t_{FOIS}	5			ns	
	WAN_FRMI Hold Time	t_{FOIH}	5			ns	
	WAN_STO Delay	t_{STOD}	1		20	ns	
	WAN_STI Setup Time	t_{STIS}	5			ns	
	WAN_STI Hold Time	t_{STIH}	5			ns	

Table 36 - WAN Access Interface Timing - Slave Mode

Note 1: In synchronous mode the clock must be within the locking range of the DPLL to function correctly (±245 ppm). In asynchronous mode, the clock may be any frequency, provided there are 32 clocks between frame pulses (128 in 8.192 Mbit/s mode)

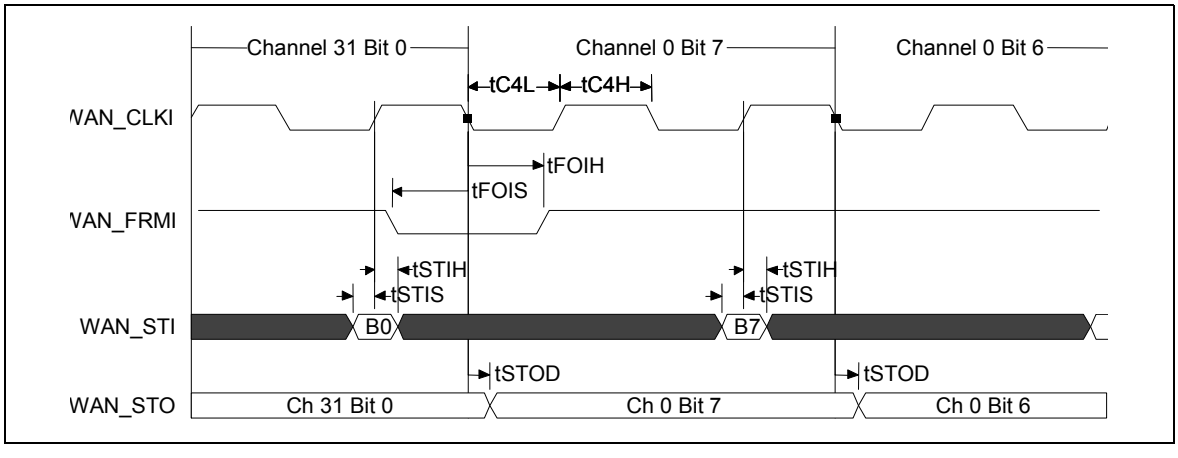

Figure 38 - WAN Bus slave mode timing at 2.048Mbits/s

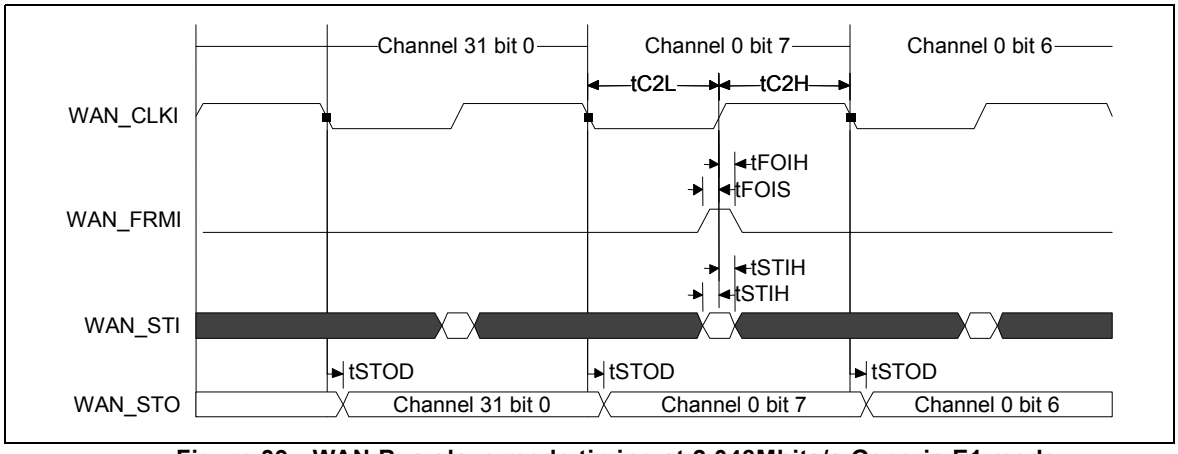

Figure 39 - WAN Bus slave mode timing at 2.048Mbits/s Generic E1 mode

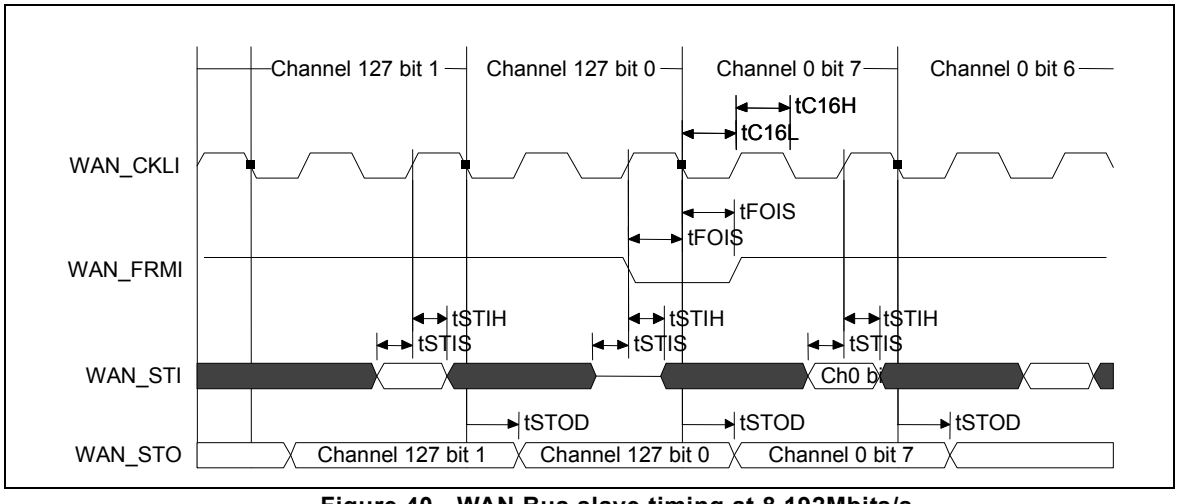

Figure 40 - WAN Bus slave timing at 8.192Mbits/s

9.1.2 Clock Master Mode

Data format	Parameter	Symbol	Min	Typ	Max	Units	Notes
ST-bus 8.192 Mbit/s mode	WAN_CLKO frequency	f_{C16P}		16.384		MHz	
	WAN_CLKO High Width	t_{C16H}		30		ns	
	WAN_CLKO Low Width	t_{C16L}		30		ns	
ST-bus 2.048 Mbit/s mode	WAN_CLKO frequency	f_{C4P}		4.096		MHz	
	WAN_CLKO High Width	t_{C4H}		122		ns	
	WAN_CLKO Low Width	t_{C4L}		122		ns	
Generic E1 2.048 Mbit/s mode	WAN_CLKO frequency	f_{C2P}		2.048		MHz	
	WAN_CLKO High Width	t_{C2H}		244		ns	
	WAN_CLKO Low Width	t_{C2L}		244		ns	
All modes	WAN_FRMO Output Delay	t_{FOD}			25	ns	
	WAN_STO Delay	t_{STOD}			25	ns	
	WAN_STI Setup Time	t_{STIS}	5			ns	
	WAN_STI Hold Time	t_{STIH}	5			ns	

Table 37 - WAN Access Interface Timing - Master Mode

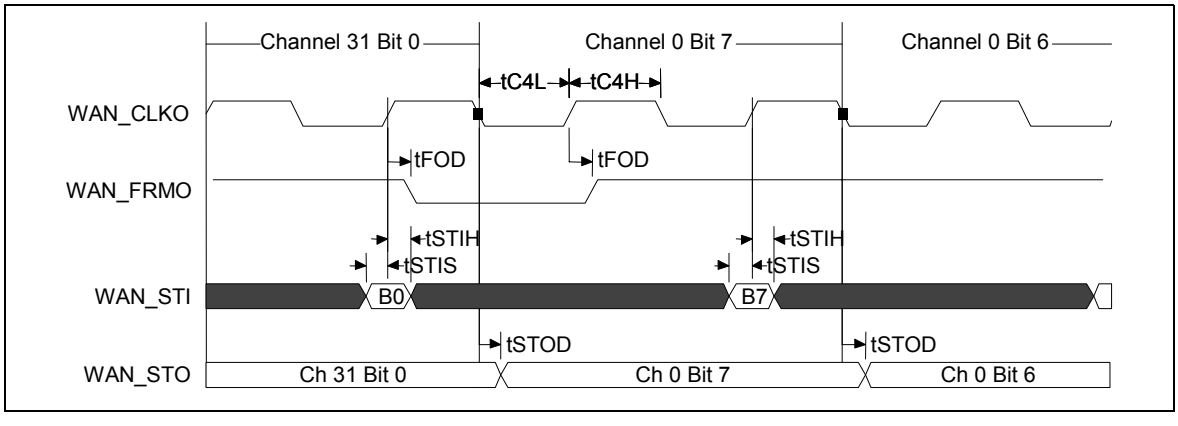

Figure 41 - WAN Bus clock master mode timing at 2.048Mbits/s

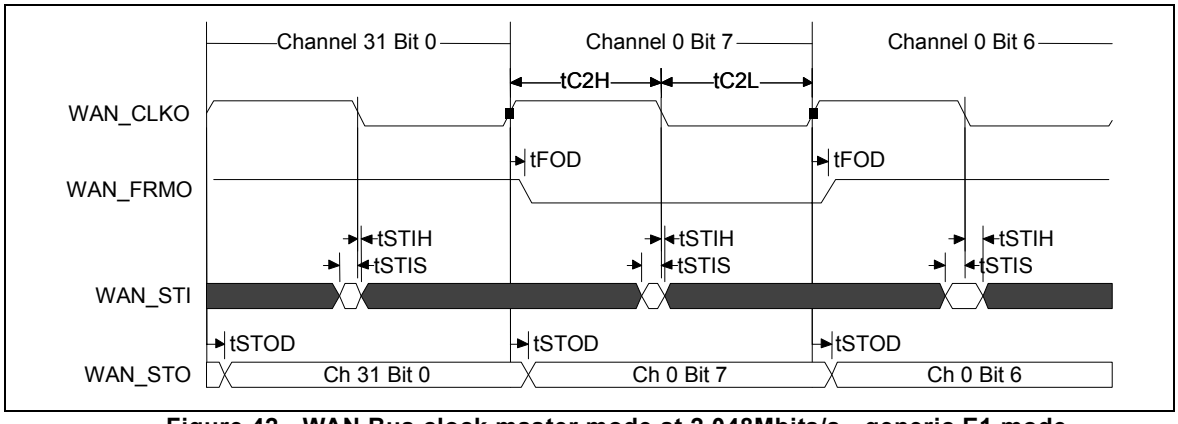

Figure 42 - WAN Bus clock master mode at 2.048Mbits/s - generic E1 mode

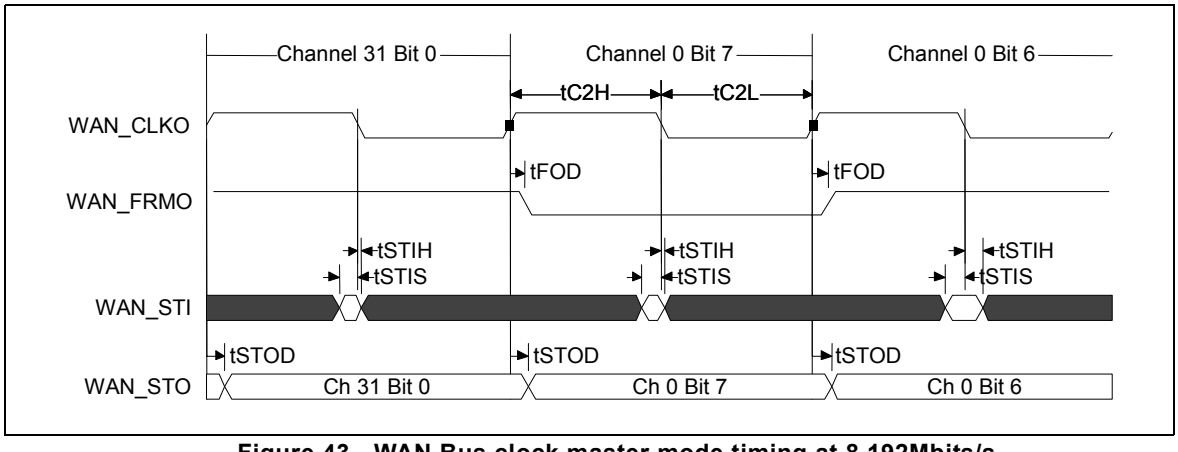

Figure 43 - WAN Bus clock master mode timing at 8.192Mbits/s

9.2 Local TDM Interface

Parameter	Symbol	Min	Typ	Max	Units	Notes
Clock C16OB frequency	f_{C16F}		16.384		MHz	
Clock C16OB High Width	t_{C16H}		30		ns	
Clock C16OB Low Width	t_{C16L}		30		ns	
Clock C8OB frequency	f_{C8F}		8.192		MHz	
Clock C8OB High Width	t_{C8H}		61		ns	
Clock C8OB Low Width	t_{C8L}		61		ns	
Clock C4OB frequency	f_{C4F}		4.096		MHz	
Clock C4OB High Width	t_{C4H}		122		ns	
Clock C4OB Low Width	t_{C4L}		122		ns	
C*OB to FP*OB delay	t_{FOD}			25	ns	
WAN_STO Delay	t_{STOD}			25	ns	
LOC_STI Setup Time	t_{STIS}	5			ns	
LOC_STI Hold Time	t_{STIH}	5			ns	

Table 38 - Local TDM Interface Timing

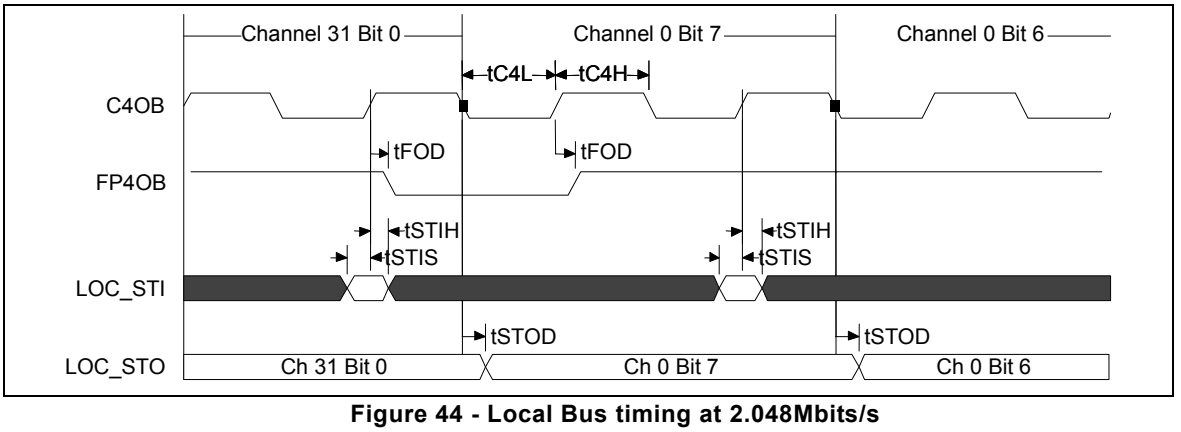

Figure 44 - Local Bus timing at 2.048Mbits/s

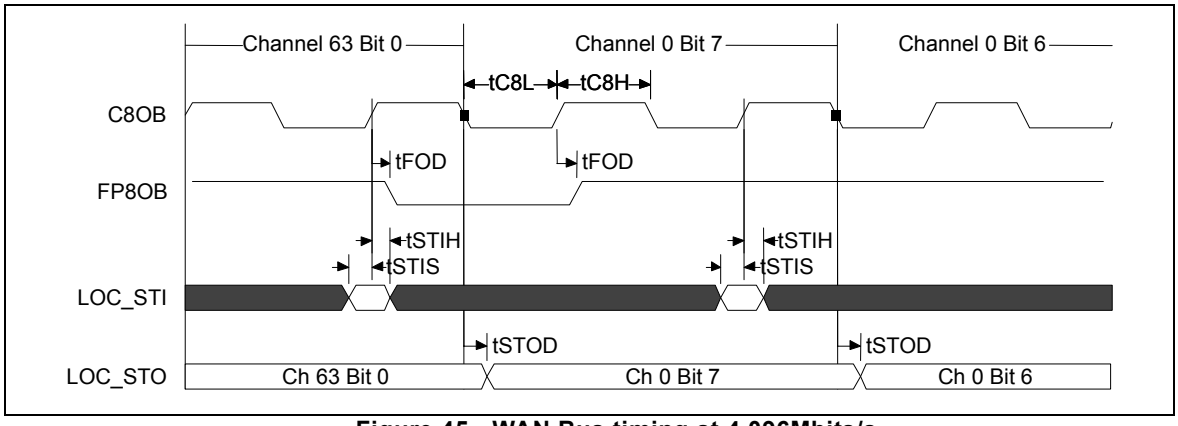

Figure 45 - WAN Bus timing at 4.096Mbits/s

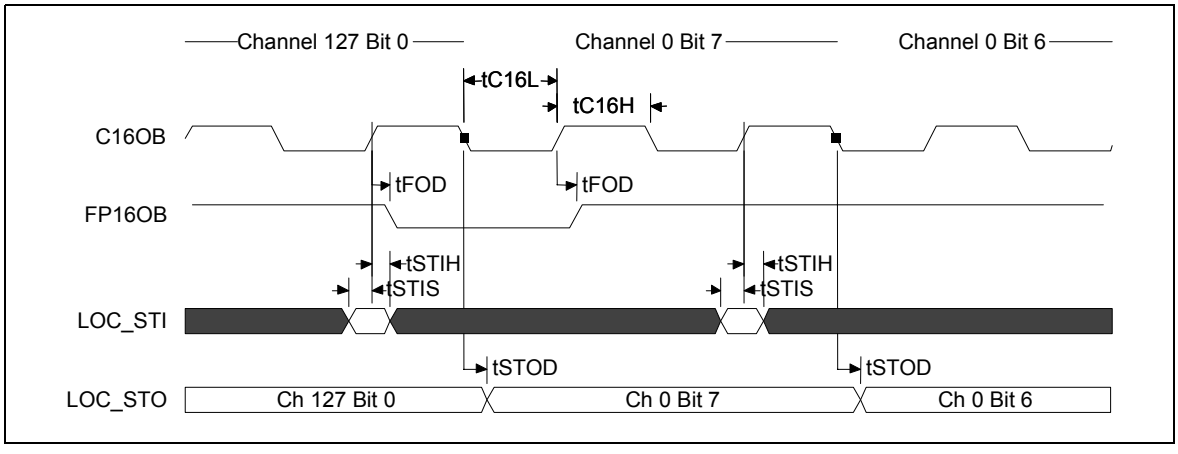

Figure 46 - WAN Bus timing at 8.192Mbits/s

9.3 Packet Interface

9.3.1 MII Transmit

Data for the MII packet switching is based on Specification IEEE Std 802.3u – 1995.

Parameter	Symbol	10 Mb/s			100Mb/s			Units
		Min	Typ	Max	Min	Typ	Max	
TXCLK period	t_{CC}	-	400	-	-	40	-	ns
TXCLK high time	t_{CHI}	140	-	260	14	-	26	ns
TXCLK low time	t_{CLO}	140	-	260	14	-	26	ns
TXCLK rise time	t_{CR}	-	-	5	-	-	5	ns
TXCLK fall time	t_{CF}	-	-	5	-	-	5	ns
TXCLK rise to TXD active delay	t_{DV}	0	-	25	0	-	25	ns
TXCLK to TXEN active delay	t_{EV}	0	-	25	0	-	25	ns

Table 39 - Packet Interface Timing - MII Transmit

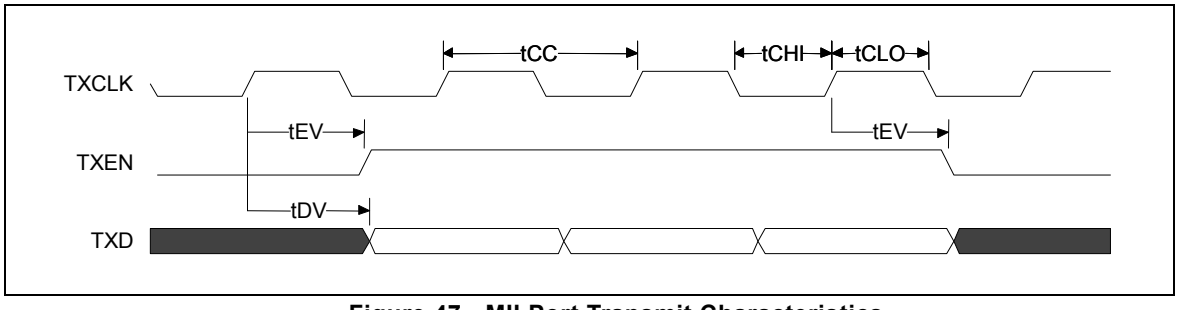

Figure 47 - MII Port Transmit Characteristics

9.3.2 MII Receive

Parameter	Symbol	10 Mb/s			100Mb/s			Units
		Min	Typ	Max	Min	Typ	Max	
Data input setup time	t_{DS}	10	-	-	10	-	-	ns
Data input hold time	t_{DH}	5	-	-	5	-	-	ns
Data valid input setup time	t_{DVS}	10	-	-	10	-	-	ns
Data valid input hold time	t_{DVH}	5	-	-	5	-	-	ns
RXCLK high wide time	t_{CH}	140	200	260	14	20	26	ns
RXCLK low wide time	t_{CL}	140	200	260	14	20	26	ns
RXCLK rise time	t_{CR}	-	-	5	-	-	5	ns
RXCLK fall time	t_{CF}	-	-	5	-	-	5	ns
RXCLK period	t_{CC}	-	400	-	-	40	-	ns

Table 40 - Packet Interface Timing - MII Receive

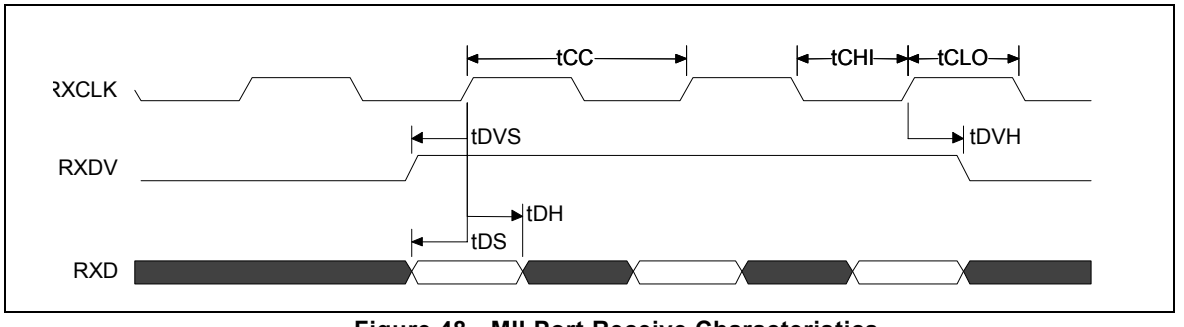

Figure 48 - MII Port Receive Characteristics

9.3.3 RMII Interface

Parameter	Symbol	Min	Typ	Max	Units	Notes
Clock Frequency			50		MHz	± 50 ppm
Clock Duty Cycle		35		65	%	
Setup Time to Clock Rising Edge RXD[1:0], CRS_DV	t_S	4			ns	
Hold Time from Clock Rising Edge RXD[1:0], CRS_DV	t_H	2			ns	
TXD[1:0] TX_EN Output delay	t_{DV}			14	ns	

Table 41 - Packet Interface Timing - RMII Interface

9.4 External Memory Interface

Parameter	Symbol	Min	Typ	Max	Units	Notes
S_CLK to Data Out Valid Delay (RAM_D)	T_{RDV}			10	ns	
S_CLK to Signal Valid Delay (RAM_A, RAM_ADSC#, RAM_RW, RAM_OE)	T_{RAV}			7.5	ns	
RAM_D setup time before S_CLK rising edge	T_{RDS}	3			ns	
RAM_D hold time after S_CLK rising	T_{RDH}	0			ns	

Table 42 - External Memory Timing

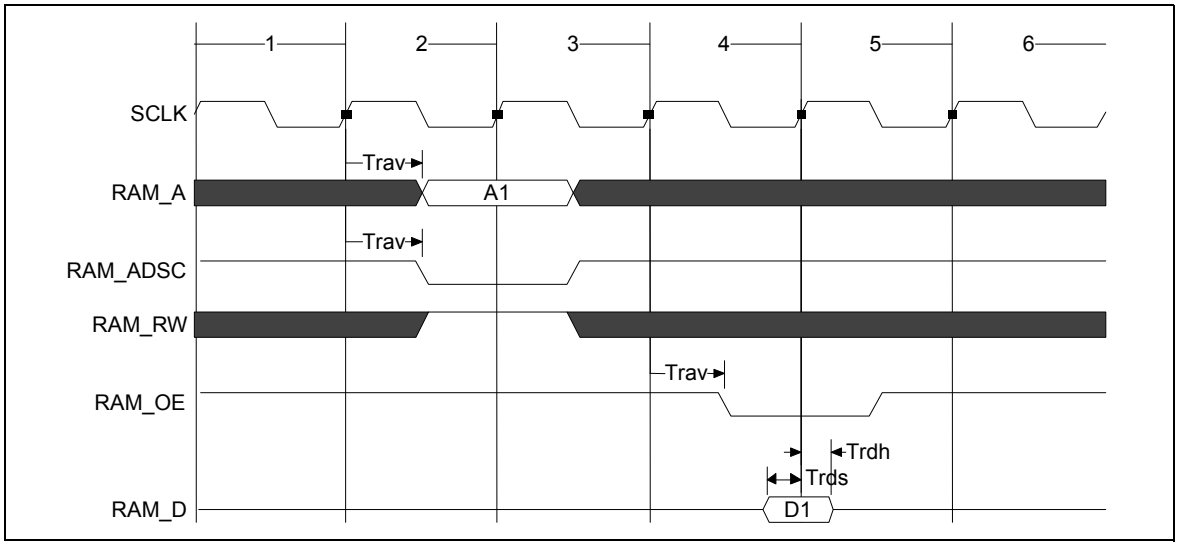

Figure 49 - External RAM single Cycle Read

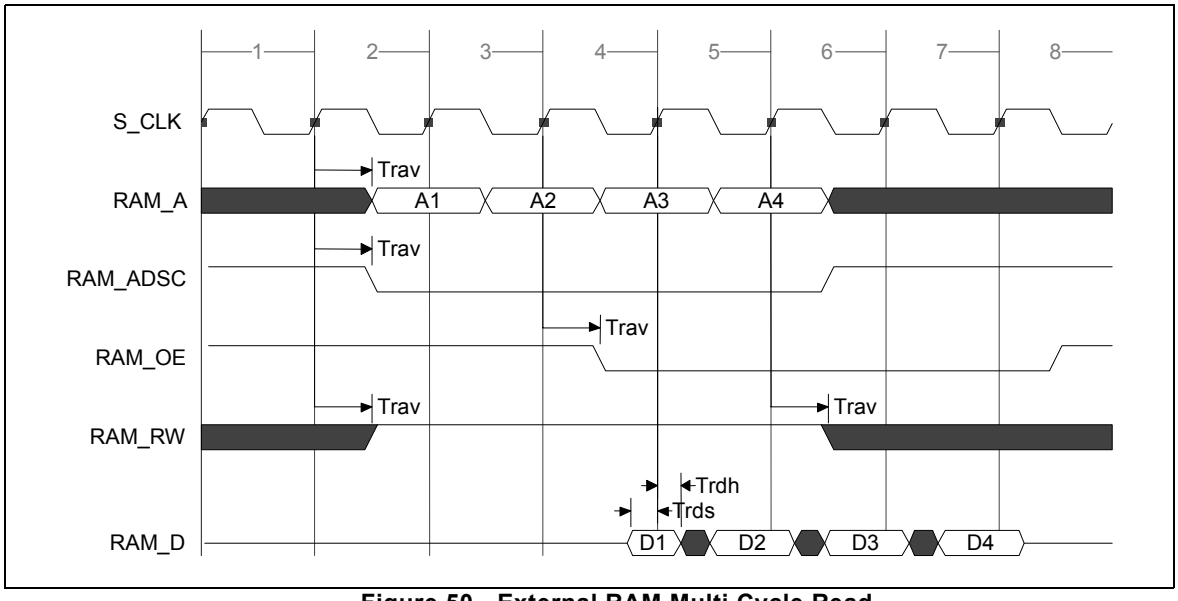

Figure 50 - External RAM Multi Cycle Read

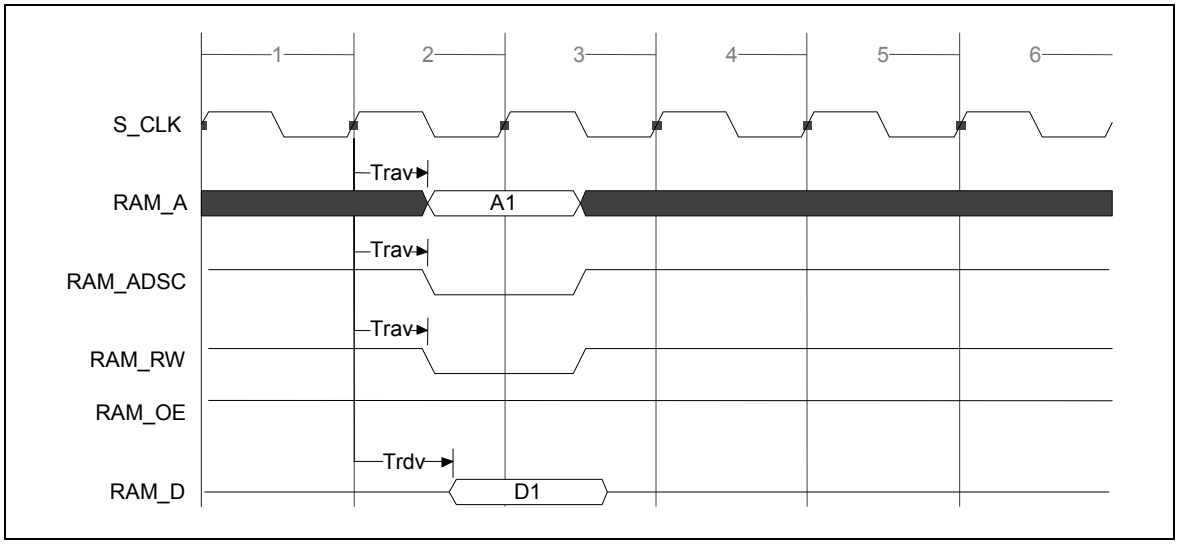

Figure 51 - External RAM Single Cycle Write

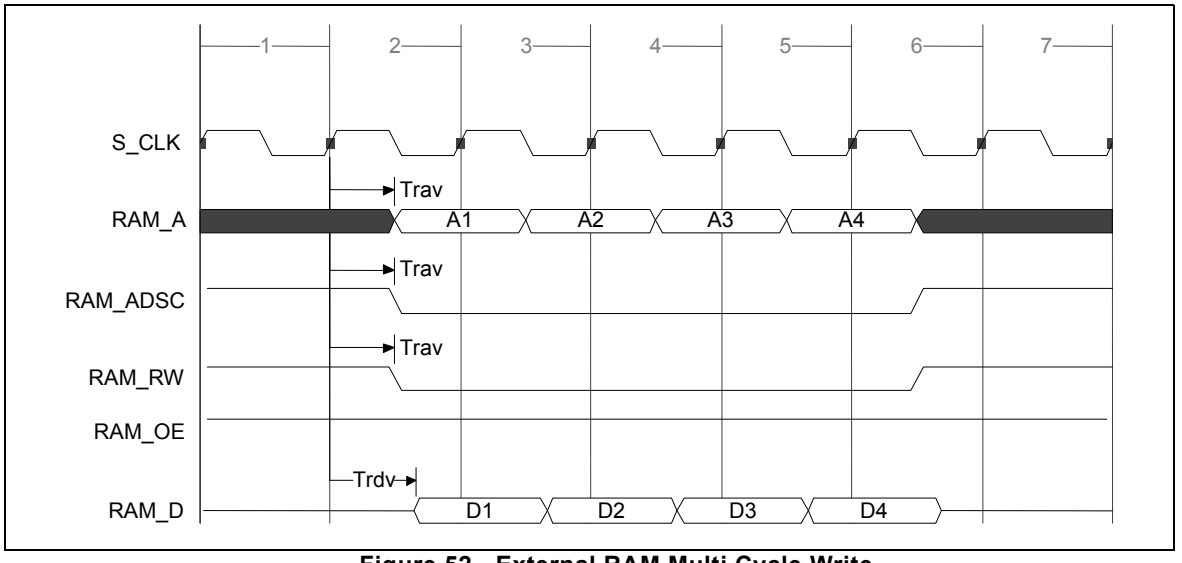

Figure 52 - External RAM Multi Cycle Write

9.5 System Control Port

Parameter	Symbol	Min	Typ	Max	Units	Notes
System Clock Frequency	CLK_{FR}		66		MHz	Notes 1 and 2
System Clock Accuracy (synchronous mode)	CLK_{ACS}			±32	ppm	Note 3
System Clock Accuracy (asynchronous mode)	CLK_{ACA}			±200	ppm	Note 4

Table 43 - System Clock

Note 1: The System clock frequency stability affects the holdover-operating mode of the DPLL. Holdover Mode is typically used for short durations while network synchronisation is temporarily disrupted. Drift on the system clock directly affects the Holdover Mode accuracy. Note that the absolute system clock accuracy does not affect the Holdover accuracy, only the change in the system clock (S_CLK) accuracy while in Holdover. For example, if the system clock oscillator has a temperature coefficient of 0.1 ppm/^{0}C, a 10^{0}C change in temperature while the DPLL is in the Holdover Mode will result in a frequency accuracy offset of 1 ppm. The intrinsic frequency accuracy of the DPLL Holdover Mode is 0.06 ppm, excluding the system clock drift.

Note 2: The system clock frequency affects the operation of the DPLL in free-run mode. In this mode, the DPLL provides timing and synchronisation signals which are based on the frequency of the master clock (S_CLK) only. The free-run frequency accuracy of the DPLL is ± 0.005 ppm plus the accuracy of the master clock (i.e. frequency of clock output C8OB equals 8.192 MHz ± S_CLK_accuracy ± 0.005 ppm).

Note 3: The absolute S_CLK accuracy must be controlled to ± 32 ppm in synchronous mode to enable the internal DPLL to function correctly.

Note 4: In asynchronous mode the DPLL is not used. Therefore the tolerance on S_CLK may be relaxed slightly.

9.6 JTAG Interface

Parameter	Symbol	Min	Typ	Max	Units	Notes
TCK Frequency of Operation		0	10	25	MHz	
TCK Cycle time	t_{CYC}	40		-	ns	
TCK Clock pulse width	t_{LOW}, t_{HIGH}	20		-	ns	
TCK rise and fall time		0		3	ns	
TRST Setup time to TCK falling edge	t_{RSTSU}	10		-	ns	Note 1
TRST Assert time	t_{RST}	10		-	ns	
Input data setup time	t_{JSU}	5		-	ns	Note 2
Input data hold time	t_{JH}	15		-	ns	Note 2
TCK to Output data valid	t_{JDV}	0		30	ns	Note 3
TCK to Output data high impedance	t_{JZ}	0		30	ns	Note 3
TMS, TDI Data setup time	t_{TPSU}	5		-	ns	
TMS, TDI Data hold time	t_{TPH}	15		-	ns	
TCK to TDO data valid	t_{TPODV}	0		15	ns	
TCK to TDO High impedance	t_{TPZ}	0		15	ns	

Table 44 - JTAG Interface

Note 1: TRST is an asynchronous signal. The setup time is for test purposes only
Note 2: Non Test (other than TDI and TMS) signal input timing with respect to TCLK
Note 3: Non Test (other than TDO) signal output timing with respect to TCLK

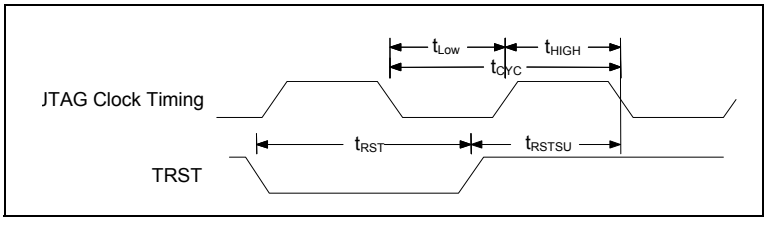

Figure 53 - JTAG Clock and Reset Timing

10.0 Glossary

ATM	Asynchronous Transfer Mode
CBR	Constant Bit Rate
CDP	Context Descriptor Protocol (used to indicate the contents of a packet stream or "context")
CES	Circuit Emulation Services
DBCES	Dynamic Bandwidth Circuit Emulation Services
DMA	Direct Memory Access
DPLL	Digital Phase Locked Loop
DSP	Digital Signal Processor
H.100/H.110	High capacity TDM backplane standards
HDLC	High-Level Data Link Control
H-MVIP	High-performance Multi-Vendor Integration Protocol (a TDM backplane standard)
IP	Internet Protocol
JTAG	Joint Test Algorithms Group (generally used to refer to a standard way of providing a board-level test facility)
LAN	Local Area Network
MAC	Media Access Control
MII	Media Independent Interface
MIB	Management Information Base
PBSRAM	Pipelined Burst SRAM (a type of synchronous SRAM)
PCI	Peripheral Control Interconnect
PDV	Packet Delay Variation
PLL	Phase Locked Loop
PPP	Point to Point Protocol
PRBS	Pseudo-Random Bit Sequence
PSTN	Public Switched Telephone Circuit
QoS	Quality of Service
RMII	Reduced Media Independent Interface
SSRAM	Synchronous Static Random Access Memory
ST BUS	Standard Telecom Bus, a standard interface for TDM data streams
TDM	Time Division Multiplexing
WAN	Wide Area Network
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Discard
ZBT	Zero Bus Turnaround, a type of synchronous SRAM
0x	Denotes Hexadecimal notation
0b	Denotes binary notation

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

www.zarlink.com