A Comparison of StarCore™ DSP Ethernet Controllers

MSC711x Family, MSC8101/MSC8103, and MSC8122/MSC8126

by Marcus Ramirez

This document compares the Ethernet controllers of Freescale DSP devices based on StarCore™ technology. Specifically, it examines the differences and similarities between the Ethernet controllers of three product platforms:

- MSC7113/MSC7116 devices (MSC711x platform)
- MSC8101/MSC8103 devices (MSC810x platform)
- MSC8122/MSC8126 devices (MSC812x platform)

These three platforms target different tier applications. Therefore, the device architecture as well as the Ethernet controllers reflect their intended application ranges. The MSC711x platform targets low-cost general-purpose and packet telephony applications with an independent streamlined Ethernet MAC. The MSC810x devices target mid-tier general-purpose and packet telephony applications. They have two multi-purpose fast communications controllers that can function as Ethernet MACs as well as HDLC and ATM controllers. The MSC812x devices are high performance multi-core DSPs in which the cores share a single Ethernet MAC. The MSC812x Ethernet controller has the most advanced features of the three DSP platforms, such as the ability to sort packets based on pattern matching into different channels for each of its four DSP cores.

All three DSP platforms have IEEE 802.3-compliant 10/100 Mbps Ethernet controllers with similar programming models. Comparing the features and operation of these Ethernet controllers may assist you in developing scalable systems based on different StarCore DSPs. To assist you in evaluating Ethernet controllers, this document addresses Ethernet considerations in general.

CONTENTS

1 Feature Comparison

Table 1 compares the Ethernet features of the MSC711x, MSC810x, and MSC812x platforms.

Table 1. Ethernet Feature Comparison

Features	MSC711x	MSC810x	MSC812x
10/100 Mbps IEEE 802.3 MII	Yes	Yes × 2	Yes
10/100 Mbps RMII	Yes	No	Yes
10/100 Mbps SMII	No	No	Yes
10 Mbps 7-wire interface	No	Yes × 2	No
MDIO PHY management	Yes	No (GPIO only)	Yes
Full and half-duplex support	Yes	Yes	Yes
IEEE 802.3 full-duplex flow control	Yes	Yes	Yes
Support for out-of-sequence transmit queue (for initiating flow-control)	No	Yes	Yes
Programmable maximum frame length up to 1522 bytes	Yes	Yes	Yes
Retransmission from transmit FIFO after a collision	Yes	Yes	Yes
CRC generation and verification of inbound/outbound packets	Yes	Yes	Yes
Broadcast address (accept/reject)	Yes	Yes	Yes
Exact match 48-bit individual (unicast) address	Yes	Yes	Yes
Hash check of individual (unicast) addresses	64-bit hash	64-bit hash	256-bit hash
Hash check of group (multicast) addresses	64-bit hash	64-bit hash	256-bit hash
Promiscuous mode	Yes	Yes	Yes
RMON statistics support	Yes	Yes	Yes
Multi-channel receive filtering based on pattern matching	No	No (FDS only)[2]	Yes (4 rings, 1 per core)
32-byte buffer descriptors for transmit data insertion	No	No (FDS only)[2]	Yes
Interrupt on multiple frame count	Yes (with event counter)	No (FDS only)[2]	No (each core on its own frames only)

NOTES: 1. The MSC810x devices contain two SCCs that are capable of 10 Mbps Ethernet. However, this document does not discuss SCCs because it focuses on 100 Mbps Ethernet.

2. The MSC810x devices support the Fast Data Switching (FDS) CPM microcode (available from Freescale sales office) that greatly enhances the capabilities of the FCC Ethernet controllers. However, FDS is not covered in this document.

2 Interrupt Configurations

This section covers the Ethernet event registers, interrupt routing, and interrupt service routines.

2.1 Ethernet Event Registers

The MSC810x FCCE register and the MSC711x and MSC812x IEVENT registers all function as interrupt pending registers to indicate Ethernet events. These registers trigger interrupts if they are enabled by the corresponding mask (FCCM or IMASK) bit. They are not accessed as normal registers. That is, writing a value of 1 to a bit clears the pending event, and writing a value of 0 has no effect. These Ethernet event registers cascade to the interrupt pending registers in the main interrupt controllers, so events in the main interrupt controller are configured as level-triggered rather than edged-triggered. For example, on the MSC810x devices, if no other interrupts are pending, clearing all Ethernet events in the FCCEx automatically clears the SIC-SIPNR[FCCx] bit, which clears the PIC-IPRB[SIC] bit.

Table 2. Interrupt Event Summary

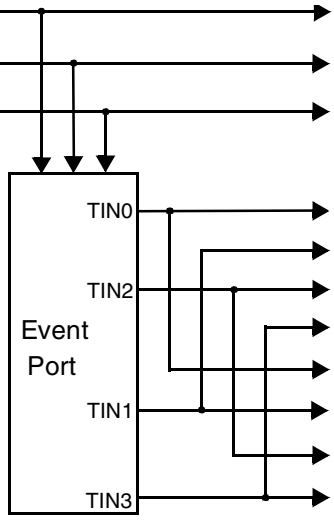
Event	MSC711x IEVENT	MSC810x FCCE	MSC812x IEVENT
Complete Frame Transmission	TFINT*		TXF*
Complete Buffer Transmission	TXB	TXB	TXB
Graceful Stop Complete	GRA	GRA	GTSC
Control Frame Transmitted		TXC	TXC
Transmit Error		TXE	TXE
Late Collision	LC		LC
Collision Retry Limit Reached	CRL		CRL
Transmit FIFO Underrun	XFIFO_UN		XFUN
Heartbeat Error	HBERR		
Babbling Transmitter Error	BABT		BABT
Transmit Insert Error			IE
Complete Frame Reception	RFINT*	RXF	RXF0* RXF1* RXF2* RXF3*
Closure of Non-End-of-Frame Receive Buffer	RXB	RXB	RXB0 RXB1 RXB2 RXB3
Control Frame Received		RXC	RXC
Graceful Rx Stop Complete			GRSC
Busy		BSY	BSY
Babbling Receiver Error	BABR		
MII Transfer Complete	MII		
Ethernet Bus Error			EBERR
MSTAT Register Overflow			MSRO
* Indicates an independent interrupt			

2.2 MSC711x Ethernet Interrupt Routing and ISRs

The MSC711x Ethernet controller has three interrupt controller (IC) channels that can each be independently prioritized relative to all other system interrupt channels. Therefore, each Ethernet interrupt channel can be preempted by higher-priority interrupts, or they can preempt lower-priority interrupts. The two primary channels are interrupted on either a complete Ethernet frame reception IEVENT[RFINT] or a complete Ethernet frame transmission IEVENT[TFINT], as shown in the first two lines of **Table 3**. However, transmit frames (ENTTxF) and receive frames (ENTRxF) share one ISR vector, so the ISR control code must test for which event caused the interrupt. Everything else in the IEVENT register is grouped together in the *Ethernet Summary* (ENTSMRY) event, which shares an ISR vector with the *HDI16 Host Command* event, as shown in the third and fourth rows of **Table 3**.

Table 3. MSC711x Interrupt Routing

Ethernet IEVENT Register	Interrupt Routing	IC Channel	ISR Vector
RFINT	ENTRxF	64	0xA00
TFINT	ENTTxF	65	
TXB, GRA, LC, CRL, XFIFO_UN, HBERR, BABT, RXB, BABR, MII	ENTSMRY	66	0xA40
	HDICMD	67	
	TMR_A0	40	0x700
	TMR_A1	42	0x740
	TMR_A2	44	0x780
	TMR_A3	46	0x7C0
	TMR_B0	98	0xE40
	TMR_B1	99	
	TMR_B2	100	0xE80
	TMR_B3	101	



Each of the three MSC711x interrupt channels can be routed to one of two event port auxiliary events. Ethernet events can therefore be routed to any of four timer input triggers that can count Ethernet events to generate the timer interrupt. Rather than using the Ethernet receive frame interrupt, you can combine multiple interrupts into one timer interrupt, thereby reducing the interrupt context switch overhead when many short back-to-back frames are received. The event port can also route these Ethernet events to the EOnCE counter so that Ethernet events can be debugged along with other system events.

2.3 MSC810x Ethernet Interrupt Routing and ISRs

Ethernet interrupts for each of the two FCCs on the MSC810x devices are routed through the SIU interrupt controller (SIC). **Table 4** shows that all SIU and CPM interrupts share the same ISR vector with FCC1 and FCC2. You can assign any of the seven priority levels to the SIC interrupt, so the SIC ISR code typically uses a software branch table based on the SIVEC register. Even with several options for prioritizing interrupts among all SIC level interrupt sources, no Ethernet interrupts can preempt other SIC interrupts.

Table 4. MSC810x Interrupt Routing

Ethernet Event FCCE Register	Interrupt Routing		PIC $\overline{\text{IRQ}}$	ISR Vector
TXB, GRA, TXC, TXE, RXF, RXB, RXC, BSY	FCCE1	SIC (SIU interrupt controller)	$\overline{\text{IRQ16}}$	0xC00
TXB, GRA, TXC, TXE, RXF, RXB, RXC, BSY	FCCE2			
	CPM interrupts: I^2C, SPI, SMC[1-2], SDMA, FCC3, MCC[1–2], SCC[1–4],			
	All SIU Interrupts: timers, TMCNT, PIT			
	External IRQ: $\overline{\text{IRQ}}$[1, 4–7], PC[4–7, 12–15]			

Since no SIC interrupts can preempt other SIC interrupts, nested interrupts must be emulated in software. The SIC user code must adjust the core SR[IPL] field to a lower relative priority and clear the current pending SIC interrupt. Then, while the SIC interrupt is serviced, another high-priority SIC interrupt, such as an Ethernet interrupt, can immediately be serviced. This technique is complex with high overhead and should not be necessary if interrupt code is kept short.

2.4 MSC812x Ethernet Interrupt Routing and ISRs

Like the MSC711x family, the MSC812x devices have three types of independent Ethernet interrupt events. The completed Ethernet frame transmission IEVENT[TXF] event is directly routed to the PIC of each SC140 core. Since there are four receive RxBD rings (channels), a complete Ethernet frame reception on each corresponding ring has its own interrupt event, IEVENT[RXFn]. Each interrupt event is directly routed to the PIC of each SC140 core, so each core has its own ISRs. Just as the MSC711x devices group everything else into a summary event, the MSC812x groups the remaining IEVENTs together in “Another” event (ETHAE) that is routed to the GIC (see **Table 5**).

Table 5. MSC812x Interrupt Routing

Ethernet Event IEVENT Register	Interrupt Routing		PIC $\overline{\text{IRQ}}$	ISR Vector
RXF0	Ring 0 Receive Frame Event (RFE0)		$\overline{\text{IRQ0}}$	0x800
RXF1	Ring 1 Receive Frame Event (RFE1)		$\overline{\text{IRQ1}}$	0x840
RXF2	Ring 2 Receive Frame Event (RFE2)		$\overline{\text{IRQ2}}$	0x880
RXF3	Ring 3 Receive Frame Event (RFE3)		$\overline{\text{IRQ3}}$	0x8C0
TXF	Transmit Frame Event (TFE)		$\overline{\text{IRQ4}}$	0x900
MIIGSK_IEVENT[IE0–IE7]	Receive Inter Frame Gap Status (RIFGSI)		$\overline{\text{IRQ5}}$	0x940
	IRQ[8–15]	GIC global interrupt	$\overline{\text{IRQ16}}$	0xC00
TXB, GTSC, TXC, TXE, LC, CRL, XFUN, BABT, IE, RXB0, RXB1, RXB2, RXB3, RXC, GRSC, BSY, EBERR, MSRO	GCIER[ETHAE] (another event)			

Since the interrupts listed in **Table 5** are shared by four SC140 cores, the MSC812x Ethernet controller can be controlled in a variety of ways. For example, one configuration assigns an RxBD ring interrupt to each SC140 core while all cores equally share the TXF interrupt. In another configuration, one master core with an operating system

(OS) and protocol stack controls all four SC140 cores, sorting packets by priority rather than by core destination. A master core is needed anyway to recycle used transmit buffers for TXF events and receive broadcast frames and to handle Ethernet errors.

3 Buffer Descriptors (BDs)

Buffer descriptors (BDs) are the primary run-time interface between user code and the Ethernet controller. Each 8-byte BD contains a pointer to a data buffer, a field specifying the length of that buffer, control bits, and status flags. BDs are arranged in 8-byte aligned arrays called rings because the last BD indicates that it should wrap back to the first BD. Each BD ring contains two or more BDs. BDs are unique to the Ethernet controller on the MSC711x devices and the MSC812x devices, but they are commonly used throughout the CPM operation on the MSC810x devices. Only a few status and control bit fields change from controller-to-controller, which not only makes software more flexible across platforms but also makes it easier to describe Ethernet controllers in general terms.

Driver code for DSP Ethernet controllers requires significantly different addressing than on RISC platforms. Ethernet BDs can be located in M1 SRAM, M2 SRAM, or external SDRAM/DDRAM. However, the DSP core can access only the M1 SRAM in a single cycle without stalling the core. Therefore, BDs should be placed into local/M1 SRAM to maximize performance when BDs are serviced. On RISC platforms, drivers access buffers at the physical system address, but applications access buffers at an MMU page-mapped address that often requires intermediate copy operations in systems such as Linux. In contrast, M1 SRAM is mapped to address 0x0 for the DSP core and is immediately available to application code.

Each Ethernet MAC contains a small RISC microcontroller with its own dedicated DMA controller for accessing the system memory to fetch BDs. The RISC microcontroller converts BDs into DMA configurations for data buffer transfers. The DSP core gains access to the local/M1 SRAM from the StarCore bus, yet the Ethernet controller DMA gains access to the local/M1 SRAM from another system bus. Therefore, the addresses programmed in BD parameters should use the system address offset shown in **Table 6**. However, the DSP core should actually service the BDs at the address without the offset so that accesses occur on the StarCore bus rather than attempting to access the platform's particular system bus. This rule applies to all BD address parameters, including all pointers in the general parameters as well as the data buffer pointer in each BD.

Table 6. Core Versus DMA Memory Map

Platform SRAM	SC140 Core Bus Access	Core Address Offset	Ethernet Controller DMA Bus Access	DMA Address Offset
MSC711x M1 SRAM	SC Bus	0x0	AMENT-ASM1	0x01800000
MSC711x M2 SRAM	AMEC-ASM2	0x01000000	AMENT-ASM2	0x01000000
MSC810x Local SRAM	SC Bus	0x0	Local Bus	0x02000000
MSC812x Core 0 M1 SRAM	SC Bus	0x0	Local Bus	0x02080000
MSC812x Core 1 M1 SRAM	SC Bus	0x0	Local Bus	0x020C0000
MSC812x Core 2 M1 SRAM	SC Bus	0x0	Local Bus	0x02100000
MSC812x Core 3 M1 SRAM	SC Bus	0x0	Local Bus	0x02140000
MSC812x M2 SRAM	MQ-Bus	0x01000000	Local Bus	0x02000000

Note: If either BDs or data are placed into the M2 SRAM on the MSC711x devices, the address is the same offset at 0x01000000. However, if either BDs or data are placed into the M2 SRAM on the MSC812x devices, the DSP core requires an MQ-Bus address offset at 0x01000000, and the Ethernet controller DMA requires a local bus offset at 0x02000000.

4 Ethernet Transmit Operations

The Ethernet transmission process is very similar for all platforms. After the Ethernet controller is initialized and enabled, it polls the first TxBD[R] control bit. If the TxBD is ready, the Ethernet controller DMA starts to move the buffer into the transmit FIFO. The data passes from the FIFO through the Media-Independent Interface (MII) (or RMII or SMII) to the Ethernet PHY. Transmission continues to the end of the frame unless a collision or abort condition occurs. A 4-byte cyclic redundancy check (CRC) value is calculated during the entire transmission and is appended to the end of the frame. Each frame in the TxBD ring is processed sequentially, and the number of frames queued depends on how many TxBDs are used.

4.1 Comparison of Transmit Buffer Descriptors (TxBDs)

All the Ethernet controllers compared have very similar TxBD transmit control functions. In all cases, the TxBD data length (TxBD[DL]) is a 16-bit control field designating the length of the data buffer to be sent, and the 32-bit transmit data buffer pointer (TxBD[TXDBPT]) indicates the location of the data buffer. The ready token bit (TxBD[R]) passes control/ownership of the TxBD to-and-from the user and Ethernet controller. The wrap control bit (TxBD[W]) indicates the end of the TxBD ring. The MSC810x and MSC812x TxBD transmit status indicators are similar, but those for MSC711x devices have no transmit status flags.

Table 7. Transmit Buffer Descriptors

TxBD Bit	MSC711x	MSC810x FCC	MSC812x
0	R: Ready		
1	T01: Free use by software	PAD: Padding up to MINFLR register value	PAD/CRC: Padding up to 64 bytes and CRC attachment
2	W: Wrap		
3	T02: Free use by software	I: interrupt	
4	L: Last		
5	TC: Transmit CRC (in last BD of frame)		TC: Transmit CRC (in first BD of frame)
6	ABC: Append Bad CRC	DEF: Defer indication	
7	Reserved	HB: Heartbeat error	T01: Free use by software
8	Reserved	LC: Late Collision	
9	Reserved	RL: Retransmission Limit	
10:13	Reserved	RC: Retry Count	
14	Reserved	UN: Underrun	
15	Reserved	CSL: Carrier Sense Lost	Reserved
16–31	DL: Data Length		
32–63	TXDBPTR: Transmit Data Buffer Pointer		

A special feature of the MSC812x devices is the 32-byte TxBD for a buffer insertion. Insertion allows you either to replace a portion of the Ethernet frame with a different buffer or expand the Ethernet frame to include an extra buffer segment. In addition to the fields of the 8-byte TxBD, the 32-byte TxBD contains fields to describe the insertion request:

- *Tx Insert Buffer Pointer* and *Insert Length.* Describe the size and location of the buffer for insertion.
- *Insert Index.* Specifies the offset into the Ethernet frame where insertion should start.
- *Insertion Type*. Specifies whether the insertion is a replacement or an expansion.
- *Insertion Error*. A status flag that indicates whether an error occurred during insertion.

On the other platforms without this feature, the same effect is achieved by using multiple 8-byte transmit BDs.

4.2 Buffer to Frame Assignment

Because the TxBD structure is so similar across all platforms, some transmission techniques can be used on all of them, and only small TxBD variations require attention. Transmit data buffers do not require a particular alignment, but keeping them aligned to 32-byte addresses prevents the initial memory accesses from being divided into multiple bus transactions, degrading performance when buffers are small.

The most straightforward method for transmitting data is to use a single buffer per Ethernet frame, which simplifies memory management and reduces the BD processing load for the Ethernet controller. The last control bit (TxBD[L]), the transmit CRC control bit (TxBD[TC]), and the pad control bit (TxBD[PAD]) should be set in every TxBD. In **Figure 1**, the Ethernet controller has just sent frame A and it is currently transmitting Frame B. The user has just passed control of Frame D to the Ethernet controller, and the next new frame should be attached to the blank BD. The MSC812x MACCFG2 automatically appends short frame padding and the CRC to every frame. Use of this option is recommended because packet processing is simplified when adding TxBD[PAD,TC] is no longer necessary. The MSC711x devices automatically pad short frames and do not have the TxBD[PAD] field. They have an additional control bit to append a bad CRC for diagnostic purposes

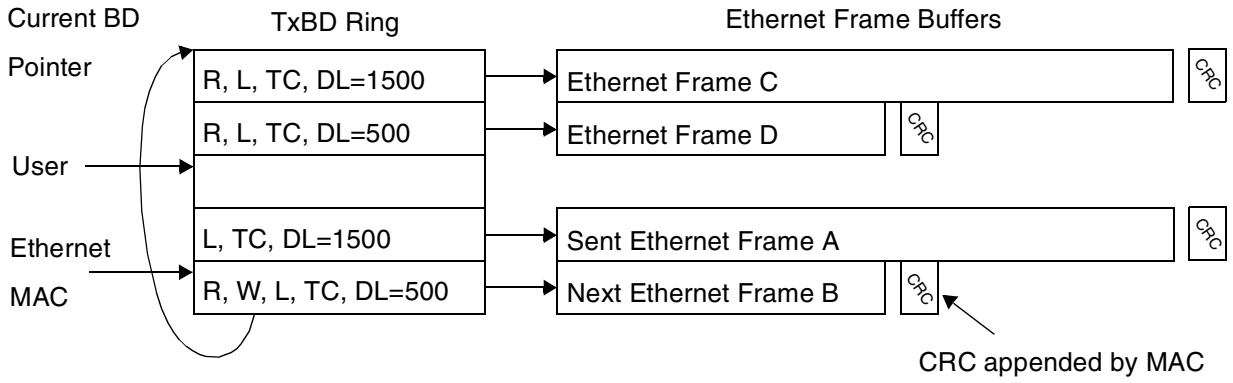


Figure 1. Single Buffer Frame Transmission

Another transmission method is to use multiple buffers per Ethernet frame. Multiple BDs are used in a multi-channel application if the output frame is a concatenation of multiple buffers. In **Figure 2**, buffers A and B make up one Ethernet frame, and buffers C, D, and E make up another complete Ethernet frame. The user code does not need to pre-assemble a complete frame via copies or DMA. The multi-buffer frame technique should be used when all associated buffers are available and ready, so a good practice is to set the Ready bits of each TxBD in reverse order. **Figure 2** shows the order in which buffers are passed to the Ethernet controller: buffer E first, buffer D

second, and buffer C third. The buffer insertion feature on the MSC812x devices allow you to append one buffer using the simple single-buffer per frame method. This method of appending different buffers into the same frame requires disciplined memory management.

In multi-buffer frames, the last control bit (TxBD[L]) indicates that the buffer is the final buffer in an Ethernet frame, and the transmit CRC control bit (TxBD[TC]) should accompany it. **Figure 2** shows that these flags are set on buffers B and E. However, on the MSC812x devices, if the CRC is not automatically generated, the transmit CRC control bit must be indicated in the first buffer of an Ethernet frame.

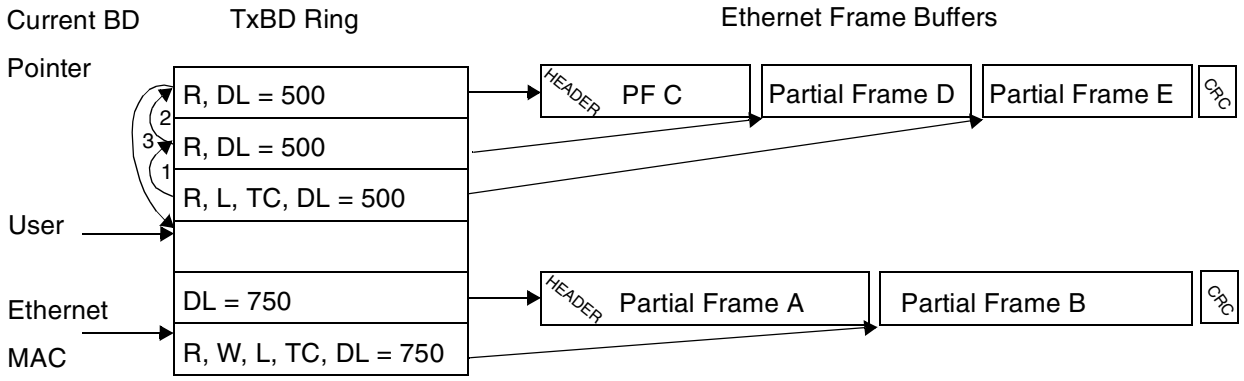


Figure 2. Multi-Buffer Frame Transmission

In a variation on the multi-buffer method, the transmission of a large frame starts before all the data is ready, as shown in **Figure 3**. Each small buffer is sent before the remaining data becomes available. This is useful for a local/proprietary system to reduce inter-packet overhead or system latency in a real-time, deadline critical system. If other higher-priority activities prevent the control software from passing off each partial-frame buffer on time, there is a risk of transmit underrun. Therefore, you should transmit from a synchronous guaranteed on-time arrival source, but definitely not from a non-deterministic source. In **Figure 3**, three channels (A, B, and C) are processed sequentially in separate buffers. Channel A is sent immediately after it is processed and before the other channel data buffers are processed.

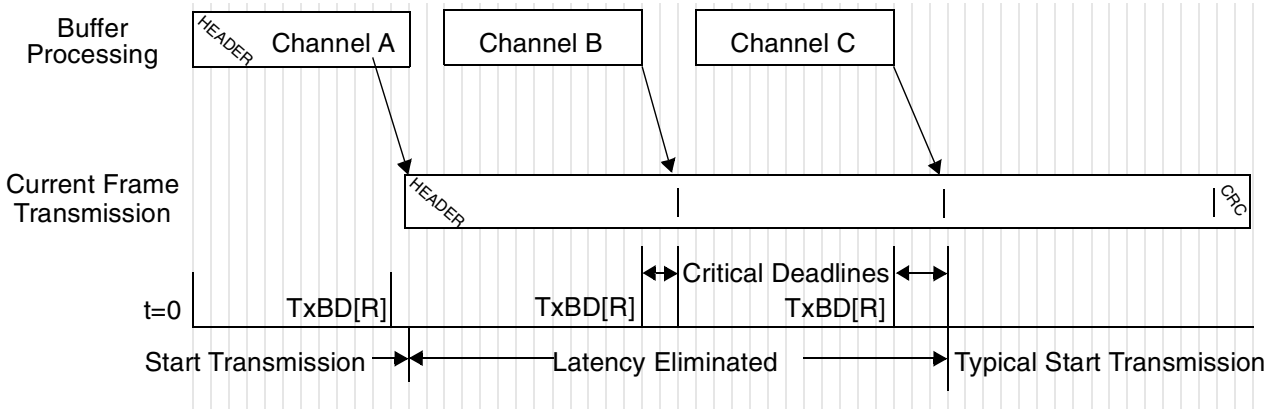


Figure 3. Synchronous Source, Low Latency, Multi-Buffer Frame Transmission

There are constraints on using a protocol stack. For example, a fixed frame size and checksum must be known at the beginning for the protocol header fields, and an extra data word must be appended at the end to correct for the known checksum.

4.3 Ethernet Transmission on the MSC812x

MSC812x Ethernet transmission must either be shared among four DSP cores or dedicated to one master DSP core. If the single TxBD ring control structure is shared among the four SC140 cores, it must be placed into M2 memory. TxBD ring resource ownership is arbitrated by locking a multi-core semaphore. Minimize the ownership of the TxBD ring to the time necessary to configure the BDs and then immediately release it.

Some applications may require only one master DSP core for Ethernet transmission, eliminating the need to arbitrate continually for the Ethernet transmission resource. Rather than share the TxBD ring, the other cores can pass messages to the master core to signal that they have an Ethernet frame ready for transmission. The master core translates these messages into TxBDs to initiate transmission but also tracks ownership of BDs. The TxBD ring is located in M1 memory, which the master core can access more quickly than M2 memory. In a system in which all cores have different functions, only the core at the output end of the processing chain may need to transmit Ethernet frames. In addition to the benefits of sole resource ownership, set-up is simplified because only the master core requires a transmit network protocol stack.

4.4 DMA Polling Ready Transmit Buffers

On MSC711x devices, you must set the Transmit Descriptor Active (TDA) register so that the Ethernet controller DMA can poll the TxBD ring. When the end of the current Ethernet buffer enters the FIFO, the Ethernet controller polls the next TxBD and starts loading the data buffer if it is ready. If the TxBD[R] is not ready, the Ethernet controller clears the TDA bit and stops polling the TxBD ring. Therefore, after setting up each TxBD, you must reset TDA to inform the Ethernet controller that new TxBDs are ready. When back-to-back frames or multi-buffer frames are sent, the TDA is set only once for the group of TxBDs.

The MSC810x devices do not require any command or configuration to poll the TxBD ring. When the end of the current Ethernet buffer enters the FIFO, the CPM polls the next TxBD and starts loading the data buffer, if it is ready. If the TxBD[R] is not ready, the CPM polls it again every 256 serial clocks. With large packets, this is not a problem because the time to transmit the current frame plus the inter-frame gap time leaves time for the second poll to occur with no effect on the traffic. However, inadvertently setting the next TxBD as ready after the first poll can add a polling latency if small back-to-back packets are transmitted:

$$256\ serial\ clocks - (64\ bytes \times 2\ \frac{nibbles}{byte}) - \left(4\ \frac{\frac{96\ bit\ clocks}{bit\ clocks}}{serial\ clocks}\right) = 104\ serial\ clocks \approx 4\ \mu s$$

Polling latency affects transmission throughput by about 50 percent. Therefore, when an Ethernet frame immediately follows frames smaller than 110 bytes, the Transmit on Demand register bit should be set to force an immediate fetch for the new buffer. Force a TxBD poll by writing to the FTOD register after setting the TxBD[R].

The MSC812x devices can operate like either the MSC711x devices or the MSC810x. When Wait or Poll (DMACTRL[WOP]) bit is set, TxBD polling behaves like polling on the MSC711x devices. The Ethernet controller allows two additional reads of a TxBD that is not ready before it enters a halt state. To resume transmission, software must clear the TSTAT[THLT] bit. When the DMACTRL[WOP] bit is cleared, the TxBD polling behaves like polling on the MSC810x devices. The Ethernet controller polls the TxBD periodically according to the DMA_MR[4–5] field, every 512, 256, 128, or 64 serial clocks. (Serial clocks are based on the internal MII clock, not on the RMII or SMII clocks.) Reducing the polling period prevents polling latency on the MSC812x for small packets. When small Ethernet frames are transmitted back-to-back, you can either adjust the polling period to accommodate them or use the Transmit on Demand register. To force an immediate fetch for the new frame buffer, write to the DMACTRL[TOD] bit after setting the TxBD[R] bit.

With flow control, an Ethernet controller can request that other stations pause their transmission. The MSC711x devices construct their own control frame from flow control parameters. However, both the MSC810x and MSC812x use a different method to transmit a flow control frame. The Ethernet controller polls an out-of-sequence TxBD (OSTBD) register between polls for each TxBD. The Ethernet transmitter sends this flow control frame before all others in the TxBD ring. This BD can also be used as a high-priority secondary transmit channel with only a single BD. The OSTBD register can be useful for sending a low-latency high-priority message. Also, the OSTBD register is a separate resource from the TxBD ring, so another core or task can use the OSTBD register without arbitrating for ownership of the TxBD ring resource.

4.5 Transmit Interrupts and Recovering Buffers

The primary purpose of transmit interrupts is to recover transmitted buffers and process errors. Closure of a TxBD is indicated by events in the interrupt event registers shown in **Table 2,** *Interrupt Event Summary,* on page 3. The MSC711x devices and the MSC812x devices have two interrupt events to indicate closure of buffers. There is a TXB register for every buffer and a subset to indicate end-of-frame buffers. For the MSC711x devices, the IEVENT[TFINT, TXB] events are triggered automatically, if enabled. For the MSC812x devices, the IEVENT[TXF, TXB] events are triggered by the interrupt control bit (TxBD[I]) shown in **Table 7,** *Transmit Buffer Descriptors,* on page 7. The MSC810x devices do not distinguish the end of frames, so TxBD[I] triggers FCCE[TXB] bit field on every buffer, and transmission error events are triggered by the FCCE[TXE] bit.

Handling a used transmit buffer is a low-priority task, so the interrupt routine should be short and should pass the work to an RTOS task. To improve system performance, minimize the number of transmit interrupts or eliminate them entirely. To minimize transmit interrupts, coalesce the interrupts so that multiple events are combined into a single interrupt. For the MSC810x and MSC812x devices, you can set the pace of the transmit frame notifications by setting the TxBD[I] bit only on user-selected TxBDs. The MSC711x devices do not have this control bit, but the same effect is achieved by routing the transmit frame interrupt to the event port and then to a timer. This process is described in **Section 2.2**, *MSC711x Ethernet Interrupt Routing and ISRs,* on page 4. In some cases, recycle used transmit buffers during transmission because this is such a low priority.

After a complete Ethernet frame transmission, either return the buffer to the memory manager or retain it for later use. A network stack protocol (TCP/IP) can temporarily hold the frame for possible retransmission until its reception is acknowledged. This task is too complex if the frame contains multiple buffers with multiple TxBDs. Therefore, use multi-buffer frames only in protocols such as Real-Time Protocol (RTP over UDP/IP), which do not require possible retransmission.

If multiple cores share the TxBD ring resource on the MSC812x devices, the SC140 core that initiated a frame transmission should manage its own buffers. However, enabling the Ethernet transmit frame event interrupt on all cores is not reasonable. Instead, the master core receives the interrupt and sends a GIC virtual interrupt to the owner of the frame. During initial transmission, the TxBD ring resource owner tracks frame buffer ownership with a user-defined procedure.

Transmission errors are processed after interrupts are received. Transmit error interrupts are shown in **Table 2,** *Interrupt Event Summary,* on page 3. The MSC711x Ethernet event register reports each transmit error individually, but not in the TxBDs. The MSC810x devices report transmission errors as a single transmit error event that requires a cross-reference with the TxBD error status bits shown in **Table 7,** *Transmit Buffer Descriptors,* on page 7. The MSC812x Ethernet event register reports both the transmit error summary and each individual transmit error in each TxBD status bit.

The MSC810x devices indicate a heartbeat error only in the TxBD. However, MSC711x devices indicate this error only in the IEVENT[HBERR] register field. The TxBDs of both the MSC810x and MSC812x devices indicate late collisions, reaching the collision retry limit, and transmit underruns. However, these events are indicated only in the IEVENT register on MSC711x devices.The MSC812x devices indicate the retry count and a late collision in the first TxBD of a frame rather than in the last TxBD, as on the MSC810x. The Defer status flag indicates that the frame was transmitted late because the line was busy, but on the MSC812x devices, if HAFDUPR[EXCESS_DEFER] = 0, this frame is aborted and not sent.

4.6 Transmit FIFO

The Ethernet transmit FIFO sizes and features differ across platforms, but they exhibit similar behaviors. The FIFO must exceed the start-transmission watermark to start transmitting externally. A small start-transmission watermark reduces the initial latency, and a larger one is more tolerant of internal system delays. When the system is heavily loaded and the start-transmission watermark is too small, it may cause an underrun condition. If it is too large, it may introduce unnecessary system delay. If frames are smaller than the transmit watermark, they are transmitted anyway because the TxBD Last bit indicates that it is ready for immediate transmission. If the FIFO empties before the Ethernet frame completes, an underrun error is reported. All devices have a panic mechanism to prevent underruns.

The MSC711x Ethernet controller has an adjustable FIFO size that is unique to the MSC711x family. The transmit and receive FIFOs share a total 512 byte FIFO size, and the FIFO Receive Start Registers (FRST) adjusts the division between them. The start-transmission watermark is set in the TWMRK register, to 64, 128, or 192 bytes.

The MSC810x FCC transmit FIFO has an effective size of 192 bytes. The 208 byte FIFO has a 176 byte full mark to accommodate SDMA 32-byte burst transfers. The FCC start transmission watermark is fixed at 48 bytes. Underruns are averted when the FIFO enters emergency mode and increases the priority of the FCC relative to the other communications controllers in the CPM.

The MSC812x has a 2 KB FIFO, and the start-transmission watermark is set in the FTXTHR. The default start-transmission watermark is 1 KB, but to reduce unneeded latency, it should be set between 48 and 192 bytes for 100 Mbps operation. Underruns are averted with a starve watermark that is set in the FTXSR. If the number of valid bytes in the FIFO is less than that in the FTXSR, a starve alert is triggered to increase the Ethernet DMA priority to the local bus arbiter.

For all three Ethernet controllers, do not disable transmission without first gracefully stopping, which allows the transmit FIFO to flush properly and maintains proper traffic protocol on the Ethernet network.

Both the MSC711x and MSC810x devices start transmitting after the standard inter-frame gap (IFG). The IEEE 802.3 standard IFG is 96 bit times, which is 960 ns for 100 Mbps where bit times is the period for transferring one data bit. However, the transmit IFG can be programmed on the MSC812x via the IPGIFGR[NBBIPG2] bit. Some systems have a controlled network environment to reduce the IFG time for efficiency. Adjusting both the transmit IFG and the carrier deference (IPGIFGR[NBBIPG1]) speeds up access to the Ethernet network.

5 Ethernet Receive Operations

The Ethernet reception process is very similar for all platforms. After the Ethernet controller is initialized and enabled, it enters Hunt mode and waits for the beginning of the next frame, which is delineated by the RX_DV assertion. The data passes from the Ethernet PHY through the MII (or RMII or SMII) to the receive FIFO. The first six bytes of the Ethernet frame make up the destination address, which the Ethernet controller tests with an address recognition algorithm. If the frame is accepted, the Ethernet controller DMA starts to move the buffer from the

receive FIFO to the available buffer indicated in the first BD. This process continues to the end of the frame unless a collision or abort condition occurs. A 4 byte CRC is calculated during the entire reception and is compared to the CRC at the end of the frame. When an entire frame is received, the Ethernet controller sends an interrupt to the core and the receive buffer is processed. Each buffer in the RxBD ring is filled sequentially, and the number of frames queued without an overrun depends on the size and number of available buffers.

5.1 Comparison of Receive Buffer Descriptors (RxBDs)

The RxBD receive control functions are very similar to the transmit control functions (see **Table 8**). For each platform, the 32-bit Receive Data Buffer Pointer (RxBD[RXDBPT]) indicates the location of an empty buffer. The Empty token bit (RxBD[E]) passes control/ownership of the RxBD back and forth between the Ethernet controller and user software. The wrap control bit (RxBD[W]) indicates the end of the RxBD ring. The RxBD receive status indicators are very similar among the MSC711x, MSC810x, and MSC812x devices.

Table 8. Receive Buffer Descriptors

RxBD Bit	MSC711x	MSC810xFCC	MSC812x
0	E: Empty		
1	R01: Free use by software	Reserved: zero	R01: Free use by software
2	W: Wrap		
3	R02: Free use by software	I: interrupt	
4	L: Last		
5	Reserved	F: First in frame	
6	Reserved	CMR: CAM Match	Reserved
7	M: Miss, destination address did not match (Promiscuous Mode only)		
8	BC: Broadcast Address (0xFFFFFFFFFFFF)		
9	MC: Multicast Address		
10	LG: Receive Frame Length Violation		
11	No: Received Non-Octet Aligned Frame		
12	Reserved	SH: Short Frame	
13	CR: Receive CRC Error		
14	OV: Receive Overrun		
15	TR: Truncate	CL: Collision	TR: Truncate
16–31	DL: Data Length		
32–63	RXDBPTR: Receive Data Buffer Pointer		

The MSC812x also has 32-byte RxBDs that can be used for pattern matching to route frames to one of four RxBD rings. The 32-byte RxBD contains the same fields as the normal 8-byte RxBD, plus fields to report matching status. The matched pattern (RxBD[MP]) bit indicates which pattern is matched and is valid only if the pattern match (RxBD[PM]) bit is set. The multiple match (RxBD[MM]) bit indicates whether two or more patterns are matched. The frame is accepted on the basis of either a pattern match (RxBD[ABPM]) or the destination address. The byte count (RxBD[BCT]) indicates the length of the each data buffer, and the Data Length field indicates the entire frame length on the last buffer.

5.2 Rx Frame and Buffer Interrupts

When small Ethernet frames are received back-to-back, an interrupt on every frame overwhelms the core with frequent context-saving overhead. Therefore, either group Ethernet frame interrupts together or skip multiple frame interrupts. Closure of an RxBD is indicated by two events, depending on whether the buffer is at the end of a complete Ethernet frame. As shown in **Table 2,** *Interrupt Event Summary,* on page 3, the RXF/RFINT bits are set on the reception of every complete frame, and the RXB event is asserted for closed buffers that are not the end of a frame.

The MSC810x and MSC812x RXB event is conditionally asserted based on the interrupt control bit (RxBD[I]). In the MSC812x, the RXF event is also conditionally asserted based on the RxBD[I] bit. Intermittently setting the RxBD[I] bit within a BD ring reduces the interrupt rate. The MSC812x has four IEVENT[RXBn] event bits and four IEVENT[RXFn] event bits, one for each RxBD ring. If packets are evenly distributed among the cores, the interrupt rate is by default reduced to one-fourth.

The MSC711x devices do not have the RxBD[I] feature. Instead, they reduce interrupts by routing the RFINT event to the event port, which can be routed to timer inputs. The timers can be used as frame event counters that are set for an interrupt after a configurable number of Ethernet events.

With either of these techniques, traffic must be constant so that a received frame is followed by enough frames to trigger the interrupt quickly. Otherwise, problems will result. Therefore, group interrupts and poll the next RxBD periodically so that frames are serviced efficiently. Use a periodic timer to schedule polling or use the RTOS tick.

5.3 Destination Address Recognition Algorithms

All three Ethernet controllers have similar destination address checking schemes, as shown in **Figure 4**. They check for a single individual MAC address, the broadcast address of 0xFFFFFFFFFFFF, and the flow-control frame address of 0x0180C2000001. Each has two destination address hash tables: one for individual addresses, the other for group addresses. The RxBD destination address matching flags are the same on all platforms: individual station address miss (RxBD[M]), broadcast address (RxBD[BC]), and multicast address (RxBD[MC]) bit fields. The MSC810x Ethernet controller has the option for post-processing the destination address with a look-up in a CAM, with a match indicated in the RxBD[CMR] status flag.

The hash tables on each platform work exactly the same way. The multicast bit in the destination address is checked to select the individual or group address paths, and one hash table is dedicated to each algorithm path. After the on-going CRC calculation is completed for the destination address, the high-order bits of the current CRC value are saved as a hash table index. Each bit entry of the hash table represents a range of addresses with a partial CRC of the entry's index to promote a semi-random assignment of MAC addresses to each 1-bit hash entry. However, the MSC711x and MSC8101 devices use a 6-bit hash that indexes 64 table entries. The MSC812x devices use an 8-bit hash that indexes 256 entries. With a random distribution of MAC address values, the MSC812x hash table is four times more accurate in filtering unwanted packets. When the packet is processed, verify that the destination address is an exact match. To set the hash entry for a particular address in the MSC711x and MSC812x devices, calculate the CRC and set the bit in the hash table manually. However, the MSC810x device provides a command to calculate and set the hash table automatically, using a temporary MAC address parameter.

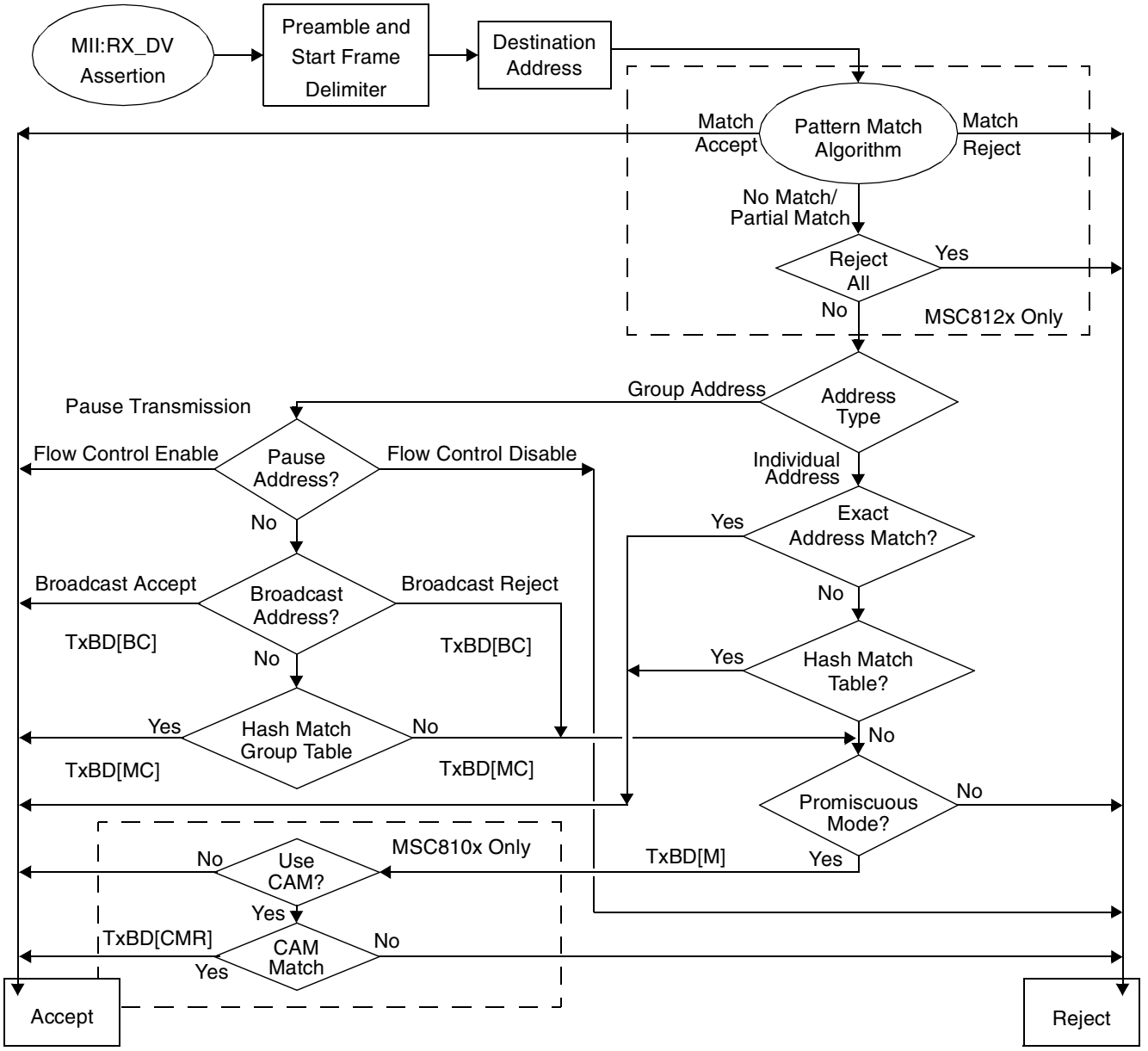


Figure 4. Destination Address Matching Algorithm

The distinguishing feature of the MSC812x Ethernet controller is its ability to perform pattern matching on 4-byte fields within the first 256 bytes of the Ethernet frame. Matches are sent to one of four independent BD rings or are explicitly rejected. Pattern matching is used for destination address sorting or protocol header filtering by features such as frame type, IP address, and UDP port. Pattern recognition can be used with the standard destination address recognition algorithm as a partial-match pre-filter to increase the likelihood of a hash table match. Alternatively, the frames with no matching patterns can be processed with the standard address recognition and sent to the default RxBD ring. Use of the standard 8-byte RxBD is recommended, but the 32-byte RxBD provides additional match status information.

5.4 Buffers to Frame Assignment

On all DSP platforms, there are several techniques for buffer assignment. The most straightforward method of Ethernet frame reception is to use one buffer per Ethernet frame, with all buffers located in local/M1 SRAM. Assign a single frame buffer to each RxBD, which requires simple memory management (see **Figure 5**). The buffer space allocated must be at least 1518 bytes, which is the Ethernet (MTU) of 1500 bytes plus Ethernet encapsulation of a 14-byte header and a 4-byte CRC. The MSC711x and MSC810x devices require 32-byte address alignment for receive data buffers, but the MSC812x devices require 64-byte address alignment. The 16-bit Receive Buffer Descriptor Data Length (RxBD[DL]) indicates the entire length of the frame. The last flag (RxBD[L]) and first flag (RxBD[F]) are always set for each RxBD. MSC711x devices do not have a First flag.

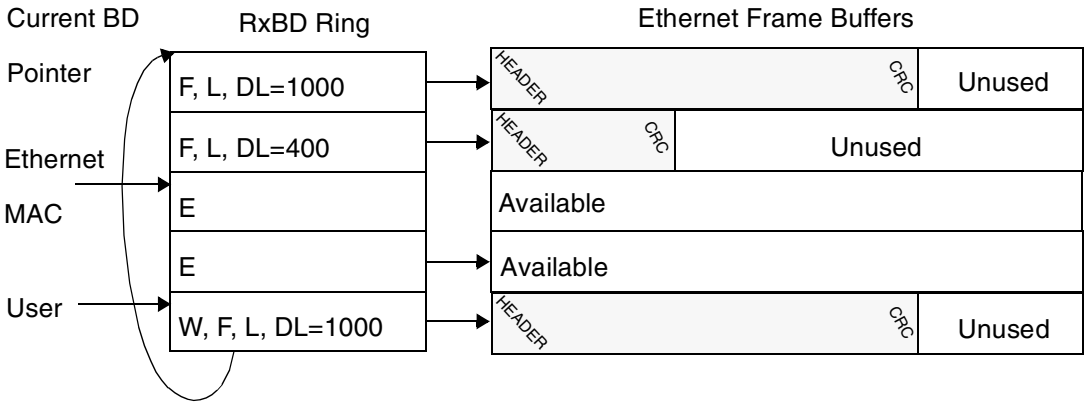


Figure 5. Single-Buffer Frame Reception

However, when the traffic primarily consists of small packets, most of the full MTU buffer space is unused, which results in poor memory usage, as shown in **Figure 6**. On the MSC812x and MSC711x devices, the M1 memory is not intended to be the primary buffer space for all receive packets. This is not a problem if the Ethernet controller puts received buffers into M2 or external memory, and you can configure the system DMA controller to move Ethernet frames into M1 memory when the core needs them. Driver code can hide the intermediate transfer of data into M1 memory. Moreover, only portions of frames that require intense data processing must be transferred to M1. However, this technique introduces processing delay and increases overall bus traffic.

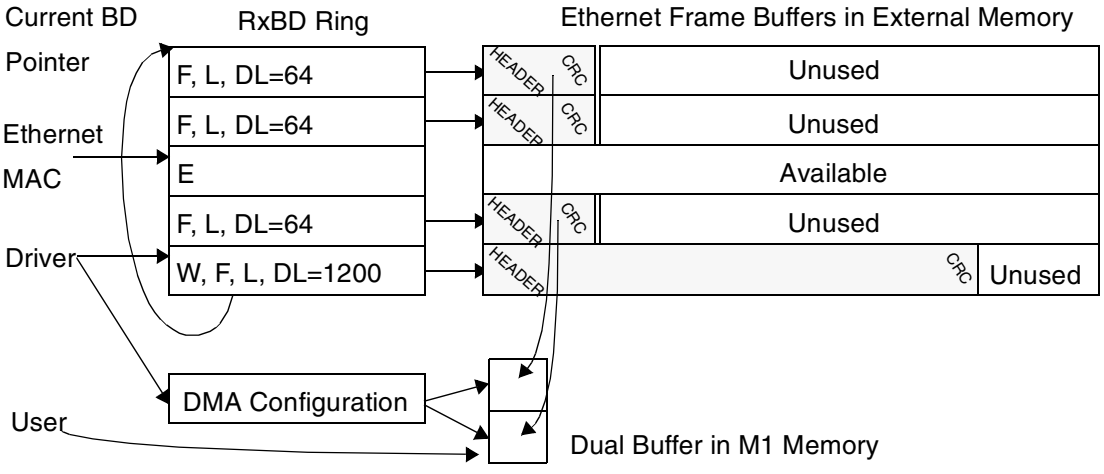


Figure 6. Single-Buffer Frame Reception in External Memory with Intermediate Transfer to M1

Some systems require both low latency and constrained M1 memory usage. To minimize copying and maximize effective memory usage, break large frames into multiple buffers in M1 memory as shown in **Figure 7**. The Ethernet controller closes RxBDs at a specified maximum receive buffer length indicated in the MRBLR and fills required buffers as needed to complete the frame. In **Figure 7**, the MRBLR is 128 bytes. The buffers assigned to each RxBD are at least the maximum receive buffer length, but memory management becomes more complicated because a dynamic memory manager allocates out-of-order buffers in non-contiguous blocks. The Data Length (RxBD[DL]) bit indicates the length of the data buffer just received, unless it is the last buffer in a frame. The last buffer indicates the entire frame length, and its last flag (RxBD[L]) is set. Frames B and E in **Figure 7** show buffer data lengths as 128 bytes with the last RxBD indicating the full frame length.

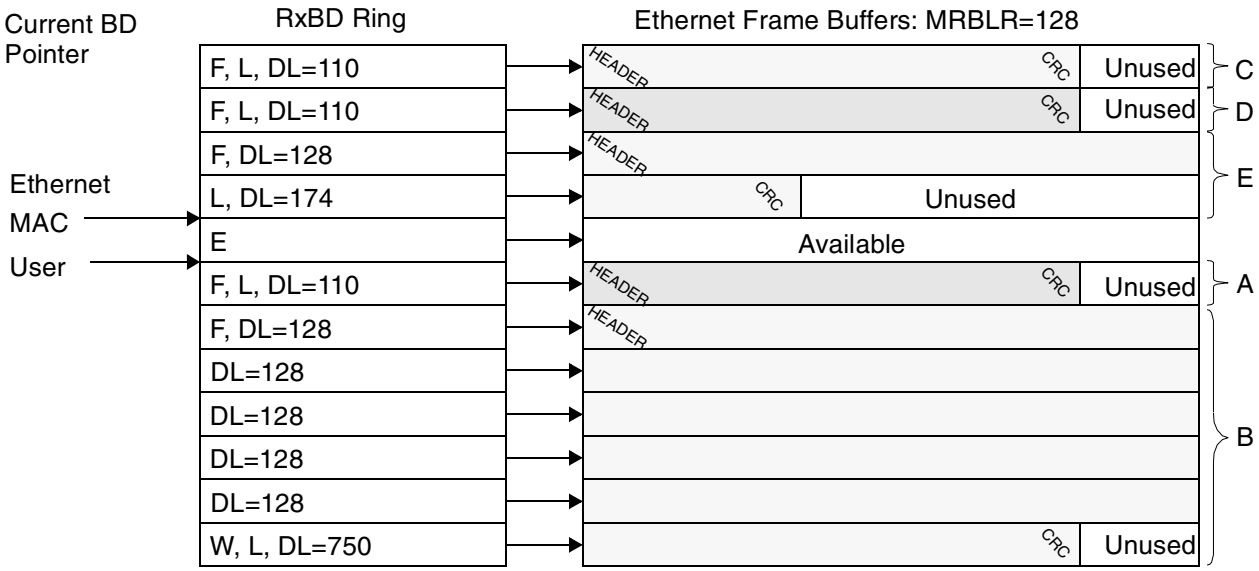


Figure 7. Multi-Buffer Frame Reception

You must either use software that can traverse across non-contiguous buffers or a driver code to hide the re-assembly of broken packets. In addition to increasing user code complexity, this increases the packet processing load for the Ethernet controller. Although 64 bytes is the smallest possible buffer size, the maximum receive buffer length should generally be greater than 128 bytes. In the best case, the entire frame size of typical traffic can be accommodated by a complete buffer so that only the occasional large frames require reassembly. To minimize unnecessary/unexpected burst buffer usage, enforce strict address filtering, which means avoiding hash table partial matches and promiscuous mode

5.5 Ethernet Reception on the MSC812x

Because MSC812x pattern matching and interrupts are so flexible, you can use a variety of methods to handle Ethernet reception. You can assign one of the four RxBD rings to each SC140 core for maximum efficiency. You can distinguish the cores with pattern matching and sort them according to MAC destination address or protocol-specific socket information. Each RxBD ring resides in one of the four M1 SRAMs, and only the associated RXF interrupt is enabled for each core. Buffer processing and memory management are handled independently for each core. If a master core is designated to handle non-matches, broadcast frames and error handling, it can be the only core that requires a full network stack. The frames routed to the other cores can be filtered by pattern matching to very specific sockets/ports or protocols, for instance, RTP over UDP/IP. In this situation, the master core handles non-deterministic traffic while the other cores receive very predictable traffic rates.

Alternatively, the master core can process all four RxBD rings as separate channels. Pattern matching can sort packets by priority based on header information, such as the protocol or socket. Each RXF interrupt is assigned a different interrupt priority level (IPL) so that Ethernet frames can be serviced out of order. If the data streams are managed so that the frame size is predictable for typical traffic, each RxBD ring can have a different maximum receive buffer length. Different buffer sizes can be allocated from heaps of different sizes. Only the master core requires a protocol stack. This configuration is useful when the master core is the front end of a processing chain or if it dynamically assigns processing tasks to the other cores.

5.6 Rx Error Reporting

Table 8, *Receive Buffer Descriptors,* on page 13 and **Table 2,** *Interrupt Event Summary,* on page 3 summarize error flags and error events. All platforms maintain a CRC calculation over the entire Ethernet frame, and at the end, the last four bytes are compared to the current CRC value. If they do not match, the CRC error flag (RxBD[CR]) is set. All platforms verify that the frame ends on a byte boundary, and if it does not, they set the non-octet alignment flag (RxBD[No]). The MSC810x indicates a late collision on the receiver with the collision flag (RxBD[CL]), but there is no late collision indication for the receivers on other platforms. They indicate collisions as CRC errors (RxBD[CR]).

If a frame is less than the minimum frame length of 64 bytes, the Ethernet controller sets the short frame error flag (RxBD[SH]), but the MSC711x devices do not indicate short frames. When a received frame reaches the maximum frame length, all platforms set the length violation flag (RxBD[LG]). However, only the MSC711x devices have a dedicated babbling receiver interrupt (BABR) for this error, and they also cut off reception at the end of the last buffer and set the frame truncation flag (RxBD[TR]).

Receive overruns occur when the receive FIFO is full, additional free buffers are not available, or the system is too busy to make all the necessary transfers on time. The error is reported by setting the receive overrun flag (RxBD[OV]) and the BSY event interrupt in the current RxBD. The MSC810x Ethernet controller does not stop receiving after it reports a busy error condition with a BSY interrupt. However, when the MSC812x Ethernet controller reports a busy error condition, the controller sets the IEVENT[BSY] register field and stops receiving data until the RSTAT[RHLT] field is cleared. The MSC711x devices do not indicate busy receiver (BSY) errors with an interrupt; these errors are indicated in the RxBD[OV] overrun flag in the next frame.

5.7 Receive FIFO

If the core does not process packets at the line rate, but the rate of packet reception is not constant, the receive FIFO and the available buffers must tolerate the worst case burst traffic. On MSC711x devices, the transmit and receive FIFOs share the total 512 byte FIFO size, and the FIFO Receive Start (FRST) register adjusts the division between them. The MSC810x FCC Receive FIFO has an effective size of 192 bytes. The 208-byte FIFO has a 176 byte full mark to accommodate SDMA 32-byte burst transfers. The MSC812x has a 2 KB FIFO that may lock up unless a graceful stop is performed before the Ethernet receiver is disabled.

The IEEE 802.3 standard receive inter-frame gap (IFG) is 96 bit times, which is 960 ns for 100 Mbps where *bit times* is the period for transferring one bit. In asymmetric point-to-point systems such as an Ethernet switch to a DSP, it is acceptable to adjust the transmitter IFG of some stations, minus the standard time, as long as the receiving stations do not reject it. All three Ethernet controllers can receive frames sooner than the IFG as long as there is a deassertion of RX_DV between frames. The minimum time for the MSC711x devices is 28 bit times, which is seven MII serial clocks. The minimum time for the MSC810x devices is 8 bit times, which is two MII serial clocks. On the MSC812x devices, the minimum receive IFG is programmed via the IPGIFGR[MIFGE] bit.

6 Summary

The Ethernet controllers on three StarCore DSP platforms are well suited for their particular application targets. This application note explains their feature differences to assist system designers in planning multi-scale platforms or migrations. This document also discusses the similarities between the Ethernet controllers so that engineers working on projects across these platforms can build on this knowledge.

Awareness of the interrupts and BD fields discussed in this document will prove helpful to programmers planning system control software. It is important to consider the differing interrupt events available on each platform when you are working with multiple platforms because they may require different approaches for real-time scheduling. Transmit and receive BDs across all platforms share many of the same control fields, which helps to generalize the behavior of run-time control software. The addresses of internal memories differ according to the bus on which the access occurs. Therefore, BDs and Ethernet parameters should be made accessible to the Ethernet controller DMA.

It is also helpful to know that the various methods of buffer assignment to BDs are the same for all three platforms. Regardless of the platform on which you are transmitting data, you can choose to send a complete frame. For receive operations, you can always allocate buffers large enough to accommodate a complete frame. These methods help you to keep control software simple and memory management straightforward. Alternatively, you can transmit separate small buffers for the Ethernet controller to concatenate for you, and for receive operations, you can choose a small maximum buffer size for the Ethernet controller to split into separate buffers. These methods reduce memory usage and yield other benefits, but they require more complicated packet processing software and memory management.

Knowledge of the internal Ethernet mechanisms is important for tuning performance parameters. The Ethernet transmission processes on each platform use different methods to fetch transmit buffers, some by periodic polling, others by explicit command. While all the Ethernet receivers use similar destination address recognition algorithms, each platform has its own additional features or operational requirements. The transmit and receive FIFOs are constructed differently on each platform, so the parameters must be tuned differently according to the application requirements.

7 Related Reading

All documents listed here are available at the website shown on the back cover of this document

[1] MSC8101 Reference Manual (MSC8101RM/D

[2] *MSC8103 Reference Manual* (MSC8103RM/D.

[3] *MSC711x Reference Manual* (MSC711xRM).

[4] *MSC8122 Reference Manual* (MSC8122RM).

[5] *MSC8126 Reference Manual* (MSC8126RM).

[6] Philippe Chartier, *Maximizing the Performance of Two Fast Ethernet Links on MSC8101 FCCs* (AN2333/D).

[7] Jay Azurin, *Migrating to PowerQUICC III TSEC from Previous Ethernet Controllers* (AN2652).

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu. Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2781
Rev. 1
7/2004

freescale™ semiconductor