MC92460UM/D
5/2002
REV 2.2

MC92460 Multichannel HDLC User's Manual

Home Page:
www.freescale.com
email:
support@freescale.com
USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com
Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com
Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 2666 8080
support.japan@freescale.com
Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com
For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

freescale™
semiconductor

Contents

Contents

Contents

Contents

Figures

Figures

Tables

Tables

Chapter 1
Overview

The MC92460 is a 40-channel high-level data link controller (HDLC) with an aggregate throughput of up to 1919 Mbps across a synchronous optical network (SONET). It is a peripheral device that connects to the 60x bus of microprocessors such as the MPC8260 or the MPC603e. Figure 1-1 shows a block diagram of the MC92460.

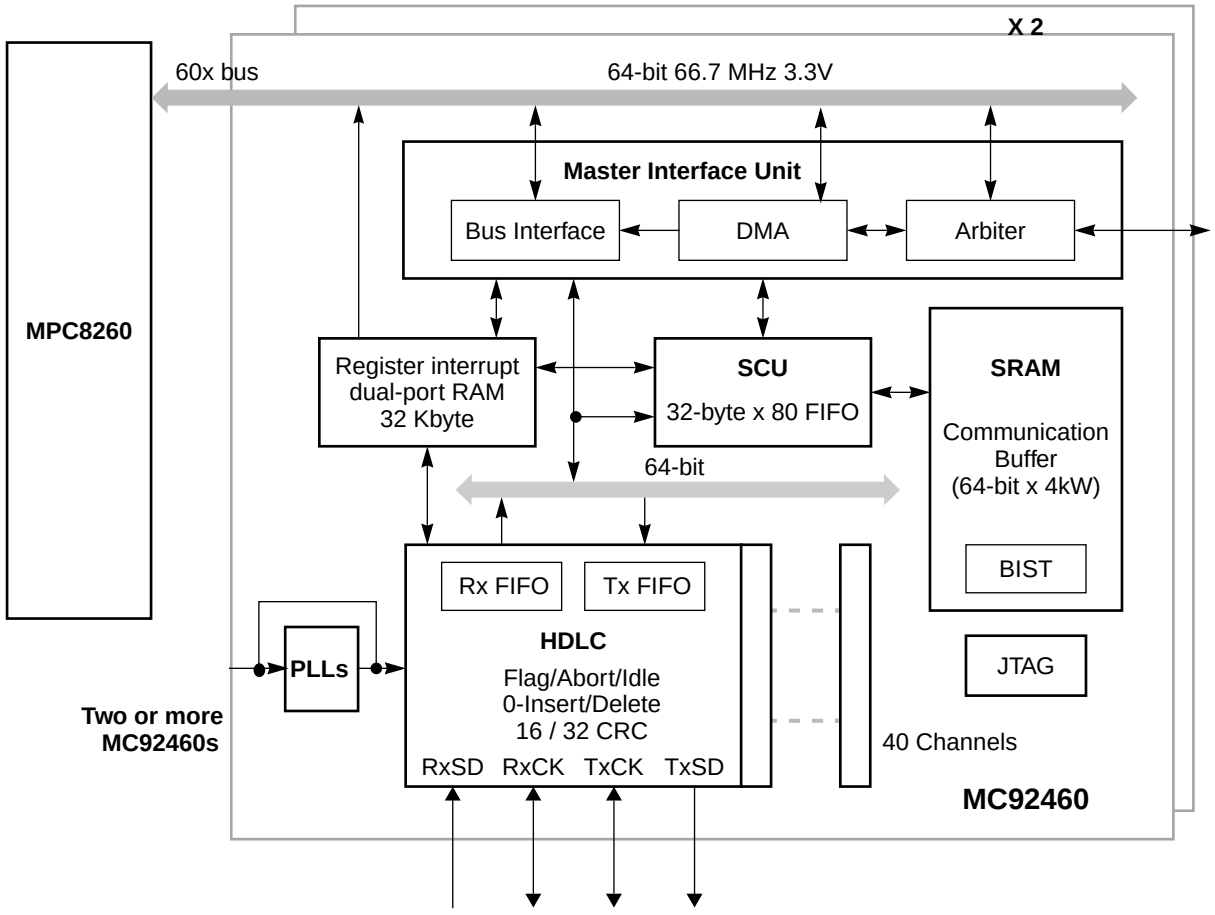


Figure 1-1. MC92460 Block Diagram

1.1 Features

The following is an overview of the MC92460 feature set:

- Channels
  - 40 full-duplex HDLC channels
  - Programmable channel assignment (any logical channel to any signal)
  - Each channel has a default of 64 buffer descriptors (Rx and Tx) but the number of buffer descriptors per channel is configurable
- Controllers
  - Maximum throughput of 1919 Mbps; individual controllers operate up to 66.7 Mbps
  - All communication controllers operate asynchronously
  - Programmable frame size (maximum 65,535 bytes)
  - Transparent memory access with internal memory controller
- 60x Bus
  - MC92460 directly connects with a 64-bit data and 32-bit address 60x bus
  - Supports 66.7 MHz 60x bus speed, with aggregate bandwidth of up to 1919 Mbps depending on the type of main memory used
  - Up to four MC92460's may be connected in parallel on the 60x bus
  - Bus supports multiple master design
- Communication Buffers
  - Data Buffer
    - 256 Kbits on-chip memory for data buffers
    - 256 Kbit communication buffer can store up to 819 bytes per frame.
    - 80 channel virtual DMA functionality executes between off-chip memory and the communication buffer
  - BD Buffer
    - 32 Kbyte on-chip dual-port RAM for buffer descriptors
    - A total of 4096 buffer descriptors (2048 TxBD and 2048 RxBD)
- JTAG Support
  - Supports the IEEE1149.1 JTAG controller standard
- Power and Clocks
  - Supports single-beat and burst accesses
  - On-chip PLL for baud rate generator (maximum of 66.7 MHz)
  - Separate power supplies for core internal logic (1.8V) and for I/O (3.3V)

- Package
  — 480 pin TPGA, 1.27 mm pitch

1.2 Memory and Register Map

The MC92460's internal memory resources are within a contiguous block of memory. The size of the internal space is 32 Kbytes. The location of this block within the global 4 Gbyte real memory space is mapped to an implementation specific special register called the internal memory map register (IMMR). Table 1-1 defines the internal memory map of the MC92460.

Table 1-1. Internal Memory Map

Internal Address		Abbreviation	Name	Size	Module
0x0_0000 - 0x0_7fff		HDLCB	Internal RAM for HDLC Tx & Rx Buffer	32 Kbytes	SCU
0x0_8000 - 0x4_ffff		Reserved			
0x5_0000 – 0x5_00ff	0x5_0000 – 0x50007	RxBD0-0	HDLC0-0 Receive Buffer Descriptor	8 bytes	HDLC
	0x50008 - 0x5000f	RxBD0-1	HDLC0-1 Receive Buffer Descriptor	8 bytes	
	0x50010 - 0x50017	RxBD0-2	HDLC0-2 Receive Buffer Descriptor	8 bytes	
	0x50018 - 0x5001f	RxBD0-3	HDLC0-3 Receive Buffer Descriptor	8 bytes	
	0x50020 - 0x50027	RxBD0-4	HDLC0-4 Receive Buffer Descriptor	8 bytes	
	0x50028 - 0x5002f	RxBD0-5	HDLC0-5 Receive Buffer Descriptor	8 bytes	
	0x50030 - 0x50037	RxBD0-6	HDLC0-6 Receive Buffer Descriptor	8 bytes	
	0x50038 - 0x5003f	RxBD0-7	HDLC0-7 Receive Buffer Descriptor	8 bytes	
	0x50040 - 0x5_00ff	RxBD0	HLDC0-8 to -31 Receive Buffer Descriptor	192 bytes	
0x5_0100 - 0x5_01ff		RxBD1	HDLC1 Receive Buffer Descriptor Table	256 bytes	
0x5_0200 - 0x5_02ff		RxBD2	HDLC2 Receive Buffer Descriptor Table	256 bytes	
0x5_0300 - 0x5_03ff		RxBD3	HDLC3 Receive Buffer Descriptor Table	256 bytes	
0x5_0400 - 0x5_04ff		RxBD4	HDLC4 Receive Buffer Descriptor Table	256 bytes	
0x5_0500 - 0x5_05ff		RxBD5	HDLC5 Receive Buffer Descriptor Table	256 bytes	
0x5_0600 - 0x5_06ff		RxBD6	HDLC6 Receive Buffer Descriptor Table	256 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
0x5_0700 - 0x5_07ff		RxBD7	HDLC7 Receive Buffer Descriptor Table	256 bytes	
0x5_0800 - 0x5_08ff		RxBD8	HDLC8 Receive Buffer Descriptor Table	256 bytes	
0x5_0900 - 0x5_09ff		RxBD9	HDLC9 Receive Buffer Descriptor Table	256 bytes	
0x5_0a00 - 0x5_0aff		RxBD10	HDLC10 Receive Buffer Descriptor Table	256 bytes	
0x5_0b00 - 0x5_0bff		RxBD11	HDLC11 Receive Buffer Descriptor Table	256 bytes	
0x5_0c00 - 0x5_0cff		RxBD12	HDLC12 Receive Buffer Descriptor Table	256 bytes	
0x5_0d00 - 0x5_0dff		RxBD13	HDLC13 Receive Buffer Descriptor Table	256 bytes	
0x5_0e00 - 0x5_0eff		RxBD14	HDLC14 Receive Buffer Descriptor Table	256 bytes	
0x5_0f00 - 0x5_0fff		RxBD15	HDLC15 Receive Buffer Descriptor Table	256 bytes	
0x5_1000 - 0x5_10ff		RxBD16	HDLC16 Receive Buffer Descriptor Table	256 bytes	
0x5_1100 - 0x5_11ff		RxBD17	HDLC17 Receive Buffer Descriptor Table	256 bytes	
0x5_1200 - 0x5_12ff		RxBD18	HDLC18 Receive Buffer Descriptor Table	256 bytes	
0x5_1300 - 0x5_13ff		RxBD19	HDLC19 Receive Buffer Descriptor Table	256 bytes	
0x5_1400 - 0x5_14ff		RxBD20	HDLC20 Receive Buffer Descriptor Table	256 bytes	
0x5_1500 - 0x5_15ff		RxBD21	HDLC21 Receive Buffer Descriptor Table	256 bytes	
0x5_1600 - 0x5_16ff		RxBD22	HDLC22 Receive Buffer Descriptor Table	256 bytes	
0x5_1700 - 0x5_17ff		RxBD23	HDLC23 Receive Buffer Descriptor Table	256 bytes	
0x5_1800 - 0x5_18ff		RxBD24	HDLC24 Receive Buffer Descriptor Table	256 bytes	
0x5_1900 - 0x5_19ff		RxBD25	HDLC25 Receive Buffer Descriptor Table	256 bytes	
0x5_1a00 - 0x5_1aff		RxBD26	HDLC26 Receive Buffer Descriptor Table	256 bytes	
0x5_1b00 - 0x5_1bff		RxBD27	HDLC27 Receive Buffer Descriptor Table	256 bytes	
0x5_1c00 - 0x5_1cff		RxBD28	HDLC28 Receive Buffer Descriptor Table	256 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
0x5_1d00 - 0x5_1dff		RxBD29	HDLC29 Receive Buffer Descriptor Table	256 bytes	
0x5_1e00 - 0x5_1eff		RxBD30	HDLC30 Receive Buffer Descriptor Table	256 bytes	
0x5_1f00 - 0x5_1fff		RxBD31	HDLC31 Receive Buffer Descriptor Table	256 bytes	
0x5_2000 - 0x5_20ff		RxBD32	HDLC32 Receive Buffer Descriptor Table	256 bytes	
0x5_2100 - 0x5_21ff		RxBD33	HDLC33 Receive Buffer Descriptor Table	256 bytes	
0x5_2200 - 0x5_22ff		RxBD34	HDLC34 Receive Buffer Descriptor Table	256 bytes	
0x5_2300 - 0x5_23ff		RxBD35	HDLC35 Receive Buffer Descriptor Table	256 bytes	
0x5_2400 - 0x5_24ff		RxBD36	HDLC36 Receive Buffer Descriptor Table	256 bytes	
0x5_2500 - 0x5_25ff		RxBD37	HDLC37 Receive Buffer Descriptor Table	256 bytes	
0x5_2600 - 0x5_26ff		RxBD38	HDLC38 Receive Buffer Descriptor Table	256 bytes	
0x5_2700 - 0x5_27ff		RxBD39	HDLC39 Receive Buffer Descriptor Table	256 bytes	
0x5_2800 - 0x5_9fff		Reserved			
0x5_a000 - 0x5_a0ff	0x5_a000 - 0x5_a007	TxBD0-0	HDLC0-0 Transmitter Buffer Descriptor	8 bytes	HDLC
	0x5_a008 - 0x5_a00f	TxBD0-1	HDLC0-1 Transmitter Buffer Descriptor	8 bytes	
	0x5_a010 - 0x5_a017	TxBD0-2	HDLC0-2 Transmitter Buffer Descriptor	8 bytes	
	0x5_a018 - 0x5_a01f	TxBD0-3	HDLC0-3 Transmitter Buffer Descriptor	8 bytes	
	0x5_a020 - 0x5_a027	TxBD0-4	HDLC0-4 Transmitter Buffer Descriptor	8 bytes	
	0x5_a028 - 0x5_a02f	TxBD0-5	HDLC0-5 Transmitter Buffer Descriptor	8 bytes	
	0x5_a030 - 0x5_a037	TxBD0-6	HDLC0-6 Transmitter Buffer Descriptor	8 bytes	
	0x5_a038 - 0x5_a03f	TxBD0-7	HDLC0-7 Transmitter Buffer Descriptor	8 bytes	
	0x5_a040 - 0x5_a0ff	TxBD0	HDLC0-8 – 31 Transmitter Buffer Descriptor	192 bytes	
0x5_a100 - 0x5_a1ff		TxBD1	HDLC1 Transmitter Buffer Descriptor Table	256 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
0x5_a200 - 0x5_a2ff		TxBD2	HDLC2 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a300 - 0x5_a3ff		TxBD3	HDLC3 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a400 - 0x5_a4ff		TxBD4	HDLC4 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a500 - 0x5_a5ff		TxBD5	HDLC5 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a600 - 0x5_a6ff		TxBD6	HDLC6 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a700 - 0x5_a7ff		TxBD7	HDLC7 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a800 - 0x5_a8ff		TxBD8	HDLC8 Transmitter Buffer Descriptor Table	256 bytes	
0x5_a900 - 0x5_a9ff		TxBD9	HDLC9 Transmitter Buffer Descriptor Table	256 bytes	
0x5_aa00 - 0x5_aaff		TxBD10	HDLC10 Transmitter Buffer Descriptor Table	256 bytes	
0x5_ab00 - 0x5_abff		TxBD11	HDLC11 Transmitter Buffer Descriptor Table	256 bytes	
0x5_ac00 - 0x5_acff		TxBD12	HDLC12 Transmitter Buffer Descriptor Table	256 bytes	
0x5_ad00 - 0x5_adff		TxBD13	HDLC13 Transmitter Buffer Descriptor Table	256 bytes	
0x5_ae00 - 0x5_aeff		TxBD14	HDLC14 Transmitter Buffer Descriptor Table	256 bytes	
0x5_af00 - 0x5_afff		TxBD15	HDLC15 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b000 - 0x5_b0ff		TxBD16	HDLC16 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b100 - 0x5_b1ff		TxBD17	HDLC17 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b200 - 0x5_b2ff		TxBD18	HDLC18 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b300 - 0x5_b3ff		TxBD19	HDLC19 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b400 - 0x5_b4ff		TxBD20	HDLC20 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b500 - 0x5_b5ff		TxBD21	HDLC21 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b600 - 0x5_b6ff		TxBD22	HDLC22 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b700 - 0x5_b7ff		TxBD23	HDLC23 Transmitter Buffer Descriptor Table	256 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address	Abbreviation	Name	Size	Module
0x5_b800 - 0x5_b8ff	TxBD24	HDLC24 Transmitter Buffer Descriptor Table	256 bytes	
0x5_b900 - 0x5_b9ff	TxBD25	HDLC25 Transmitter Buffer Descriptor Table	256 bytes	
0x5_ba00 - 0x5_baff	TxBD26	HDLC26 Transmitter Buffer Descriptor Table	256 bytes	
0x5_bb00 - 0x5_bbff	TxBD27	HDLC27 Transmitter Buffer Descriptor Table	256 bytes	
0x5_bc00 - 0x5_bcff	TxBD28	HDLC28 Transmitter Buffer Descriptor Table	256 bytes	
0x5_bd00 - 0x5_bdff	TxBD29	HDLC29 Transmitter Buffer Descriptor Table	256 bytes	
0x5_be00 - 0x5_beff	TxBD30	HDLC30 Transmitter Buffer Descriptor Table	256 bytes	
0x5_bf00 - 0x5_bfff	TxBD31	HDLC31 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c000 - 0x5_c0ff	TxBD32	HDLC32 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c100 - 0x5_c1ff	TxBD33	HDLC33 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c200 - 0x5_c2ff	TxBD33	HDLC34 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c300 - 0x5_c3ff	TxBD34	HDLC35 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c400 - 0x5_c4ff	TxBD36	HDLC36 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c500 - 0x5_c5ff	TxBD37	HDLC37 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c600 - 0x5_c6ff	TxBD38	HDLC38 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c700 - 0x5_c7ff	TxBD39	HDLC39 Transmitter Buffer Descriptor Table	256 bytes	
0x5_c800 - 0x6_3fff	Reserved			

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
Parameter RAM Area					
0x6_4000 - 0x6_407c	0x6_4000	MRBLR	HDLC0 Channel Maximum Rx Buffer Length	2 bytes	
	0x6_4008	Reserved		4 bytes	
	0x6_4010	Reserved		4 bytes	
	0x6_4018	C_MASK	HDLC0 CRC Mask	4 bytes	
	0x6_4020	C_PRES	HDLC0 CRC Preset	4 bytes	
	0x6_4028	MFLR	HDLC0 Max Frame Length	2 bytes	
	0x6_4030	HMASK	HDLC0 Mask Register	2 bytes	
	0x6_4038	HADDR1	HDLC0 Address1	2 bytes	
	0x6_403c	Reserved		2 bytes	
	0x6_4040	HADDR2	HDLC0 Address2	2 bytes	
	0x6_4048	HADDR3	HDLC0 Address3	2 bytes	
	0x6_4050	HADDR4	HDLC0 Address4	2 bytes	
	0x6_4058	MRA	HDLC0 Mode Register	4 bytes	
	0x6_4060	DSR	HDLC0 Data Sychronization	2 bytes	
	0x6_4064	ER	HDLC0 Event Register	2 bytes	
	0x6_4068	MR	HDLC0 Mask Register	2 bytes	
	0x6_406c	SR	HDLC0 Status Register	1 byte	
	0x6_4070	BRG	HDLC0 Baud Rate Generator	2 bytes	
	0x6_4074	RBASE	HDLC0 RxBD Table Base Address	4 bytes	
	0x6_4078	CMR	HDLC0 Command Register	2 bytes	
	0x6_407c	TBASE	HDLC0 TxBD Table Base Address	4 bytes	
0x6_4080		MRBLR	HDLC1 Channel Maximum Rx Buffer Length	128 bytes	
0x6_4100		MRBLR	HDLC2 Channel	128 bytes	
0x6_4180		MRBLR	HDLC3 Channel	128 bytes	
0x6_4200		MRBLR	HDLC4 Channel	128 bytes	
0x6_4280		MRBLR	HDLC5 Channel	128 bytes	
0x6_4300		MRBLR	HDLC6 Channel	128 bytes	
0x6_4380		MRBLR	HDLC7 Channel	128 bytes	
0x6_4400		MRBLR	HDLC8 Channel	128 bytes	
0x6_4480		MRBLR	HDLC9 Channel	128 bytes	
0x6_4500		MRBLR	HDLC10 Channel	128 bytes	
0x6_4580		MRBLR	HDLC11 Channel	128 bytes	
0x6_4600		MRBLR	HDLC12 Channel	128 bytes	
0x6_4680		MRBLR	HDLC13 Channel	128 bytes	
0x6_4700		MRBLR	HDLC14 Channel	128 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
0x6_4780		MRBLR	HDLC15 Channel	128 bytes	
0x6_4800		MRBLR	HDLC16 Channel	128 bytes	
0x6_4880		MRBLR	HDLC17 Channel	128 bytes	
0x6_4900		MRBLR	HDLC18 Channel	128 bytes	
0x6_4980		MRBLR	HDLC19 Channel	128 bytes	
0x6_4a00		MRBLR	HDLC20 Channel	128 bytes	
0x6_4a80		MRBLR	HDLC21 Channel	128 bytes	
0x6_4b00		MRBLR	HDLC22 Channel	128 bytes	
0x6_4b80		MRBLR	HDLC23 Channel	128 bytes	
0x6_4c00		MRBLR	HDLC24 Channel	128 bytes	
0x6_4c80		MRBLR	HDLC25 Channel	128 bytes	
0x6_4d00		MRBLR	HDLC26 Channel	128 bytes	
0x6_4d80		MRBLR	HDLC27 Channel	128 bytes	
0x6_4e00		MRBLR	HDLC28 Channel	128 bytes	
0x6_4e80		MRBLR	HDLC29 Channel	128 bytes	
0x6_4f00		MRBLR	HDLC30 Channel	128 bytes	
0x6_4f80		MRBLR	HDLC31 Channel	128 bytes	
0x6_5000		MRBLR	HDLC32 Channel	128 bytes	
0x6_5080		MRBLR	HDLC33 Channel	128 bytes	
0x6_5100		MRBLR	HDLC34 Channel	128 bytes	
0x6_5180		MRBLR	HDLC35 Channel	128 bytes	
0x6_5200		MRBLR	HDLC36 Channel	128 bytes	
0x6_5280		MRBLR	HDLC37 Channel	128 bytes	
0x6_5300		MRBLR	HDLC38 Channel	128 bytes	
0x6_5380		MRBLR	HDLC39 Channel	128 bytes	
0x6_5400		INTS	Interrupt Status Register	8 bytes	SCU
0x6_5408		Reserved		8 bytes	
0x6_5410		Reserved		4 bytes	
0x6_5414		Reserved		4 bytes	
0x6_5418		Reserved		4 bytes	
0x6_541c		Reserved		4 bytes	
0x6_5420		Reserved		4 bytes	
0x6_5424		Reserved		4 bytes	
0x6_5428		Reserved		4 bytes	
0x6_542c		Reserved		4 bytes	
0x6_5430		Reserved		4 bytes	

Table 1-1. Internal Memory Map (Continued)

Internal Address		Abbreviation	Name	Size	Module
0x6_5434		Reserved		4 bytes	
0x6_5438		MMCR	Max HDLC Channel	2 bytes	
0x6_5440		IMMR	Internal Memory Map Register	2 bytes	MI
0x6_5444		SMR	System Mode Register	2 bytes	
0x6_5448		PLL	PLL Mode Register	4 bytes	
0x6_544c		Reserved		4 bytes	
0x6_5450		Reserved		4 bytes	
0x6_5454		Reserved		4 bytes	
0x6_5458		Reserved		4 bytes	
0x6_545c		Reserved		4 bytes	
0x6_5460		Reserved		158 bytes	
0x6_5500		DMASA	Source Address of DMA	4 byte	
0x6_5504		DMATS	Transfer Size of DMA	4 byte	
0x6_5508		DMADA	Destination Address of DMA	4 bytes	
0x6_550c		DMAMR	Mode Register of DMA	4 bytes	
0x6_5510		DMASR	Status Register	4 bytes	
0x6_5514		DMAEA	Error Address Register	4 bytes	
0x6_5518		PRR	Process Revision Rregister	4 bytes	
0x6_551c		Reserved		4 bytes	
0x6_5520		ARBM0	Mode Register0	2 bytes	
0x6_5524		ARBM1	Mode Register1	2 bytes	
0x6_5526		Reserved		2 bytes	
0x6_5528		Reserved		320 bytes	

1.3 Internal Memory Map

The MC92460's internal memory map is defined with IMMR[ISB] and IMMR[CS]. The upper 9 bits (A[0:8]) are the ISB field that specify the internal space base, and the next 3 bits (A[9:11]) are the chip selects. The default base address 0xFD30_0000, for example, has an ISB of 0xFD*x* and a CS of 0x0.

Although IMMR can be read many times, it can be written only once after a reset operation. After a reset, a protection circuit is enabled allowing the IMMR to be written. After the first write access to the IMMR, the circuit is disabled. This register cannot be written by byte access.

1.3.1 Internal Memory Map Register (IMMR)

The internal memory map register (IMMR), shown in Table 1-2, identifies a specific device as well as holds the base address for the internal memory map. Software can deduce the availability and location of all on-chip system resources from the values in IMMR.

	0	8	9		11	12			15
Field	ISB		CS0	CS1	CS2				
Reset	At reset, bits 0–8 take the values of CSA[0:8]; the default setting is 1111_1101_0		X	X	X	–			
R/W	RW		R						

Figure 1-2. Internal Memory Map Register (IMMR)

Table 1-2. IMMR Field Description

Bits	Field	Description
0-8	ISB	Internal space base. Defines the base address of the internal memory space. The value of ISB is configured at reset to one of 32 addresses; it can then be changed to any value by the software. The default is 0xFD, which maps to address 0xFD*x*0_0000. As soon as ISB is written with a new base address, the IMMR base address is relocated according to the just written ISB. ISB can be configured to one of several possible addresses at reset to enable the configuration of multiple-MC92460 systems.
9-11	CS[0:2]	Chip select status. These bits indicate the state of the CS[0:2] signals at reset.
12-15	–	Reserved

1.3.2 Process Revision Register (PRR)

The process revision register (PRR) holds the code that identifies the revision number of the MC92560.

	0	15
Field	ID	
Reset	0000_0000_0000_0000	
R/W	R	

	16	31
Field	ID	
Reset	0000_0000_0000_0010	
R/W	R	

Figure 1-3. Process Revision Register (PRR)

Table 1-3. PRR Field Description

Bits	Field	Description
0-	ID	The revision number

1.4 IEEE1149.1 (JTAG) Interface

MC92460 supports the IEEE1149.1 standard for facilitating board test and chip debug. The IEEE1149.1 test interface is used for boundary-scan testing.

The JTAG user code is 0000_0010_1000_0000_1101_0000_0001_1101 (MSB to LSB).

1.5 Configurations of MC92460 with Compliant Devices

The following figures demonstrate how the MC92460 connects with 60x compliant devices.

1.5.1 MPC8260 with MC92460 Slaves

Figure 1-4 shows the configuration of an MPC8260 operating in 60x compatible bus mode with arbitration being done by the MPC8260. MC92460 #0 and MC92460 #1 are both set in slave mode and $\overline{BR}$, $\overline{BG}$, and $\overline{DBG}$ of the MC92460's slave #0 and slave #1 are connected in cascade. MC92460 slave #0's $\overline{BR}$, $\overline{BG}$, and $\overline{DBG}$, for example, are connected to MPC8260's $\overline{BR}$, $\overline{BG}$, and $\overline{DBG}$. And slave #1's $\overline{BR}$, $\overline{BG}$, and $\overline{DBGs}$ are connected to slave #0's $\overline{SBR}$, $\overline{SBG}$, and $\overline{SDBG}$. If additional slaves are added, their $\overline{BR}$, $\overline{BG}$, and $\overline{DBG}$ will be connected to the $\overline{SBR}$, $\overline{SBG}$, and $\overline{SDBG}$ of the slave before them. Note that this

configuration requires a 4-input AND gate off chip.

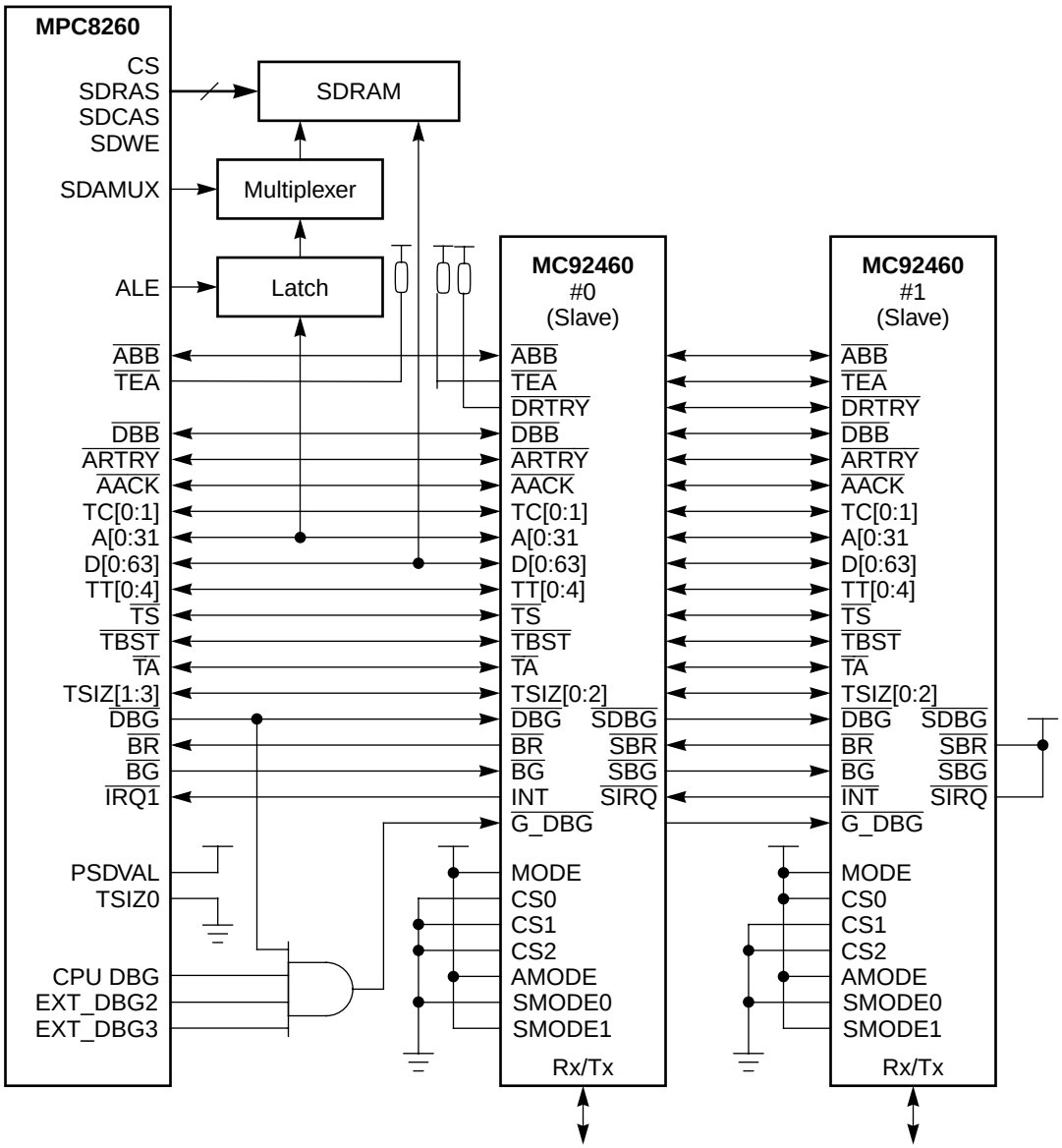


Figure 1-4. MPC8260 and MC92460 System Using MPC8260 Arbiter

1.5.2 MPC8260 with MC92460 Master and MC92460 Slave

Figure 1-5 shows the configuration with the MPC8260 operating in 60x compatible bus mode and external arbitration mode. The MC92460 can support two bus masters. The two MC92460 devices are connected to a 60x bus with the MC92460 #0 in master mode and doing the arbitration, and #1 set in slave mode. The SDRAM bank is controlled by an SDRAM controller on the MPC8260.

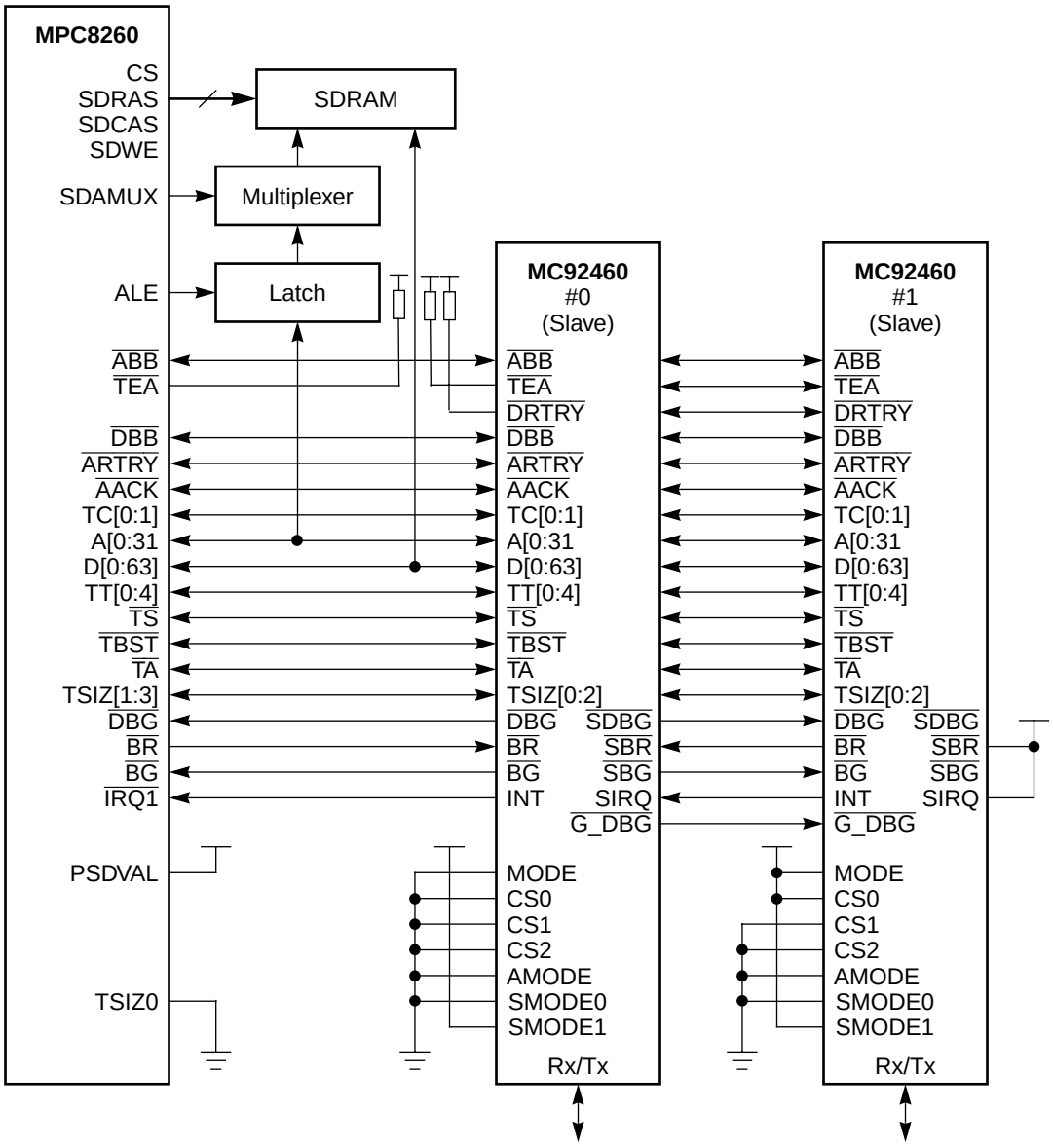


Figure 1-5. MPC8260 and MC92460 System Using MC92460#0 As Arbiter

1.5.3 MPC603e and MPC106 with MC92460s

Figure 1-6 shows the configuration of the MPC603e and the MPC106 with MC92460s. The MPC106 can handle four bus masters. In this configuration the MPC603e and the MC92460 #0 are in bus master mode, and the arbitration is done by the MPC106. MC92460 #1 is a slave connected to MC92460 #0 in cascade. The $\overline{\text{LBCLAIM}}$ signal is enabled by setting the PICRL1[CF_LBA_EN] bit of the MPC106.

Note that this configuration requires one 2-input AND gate and one 5-input AND gate off chip.

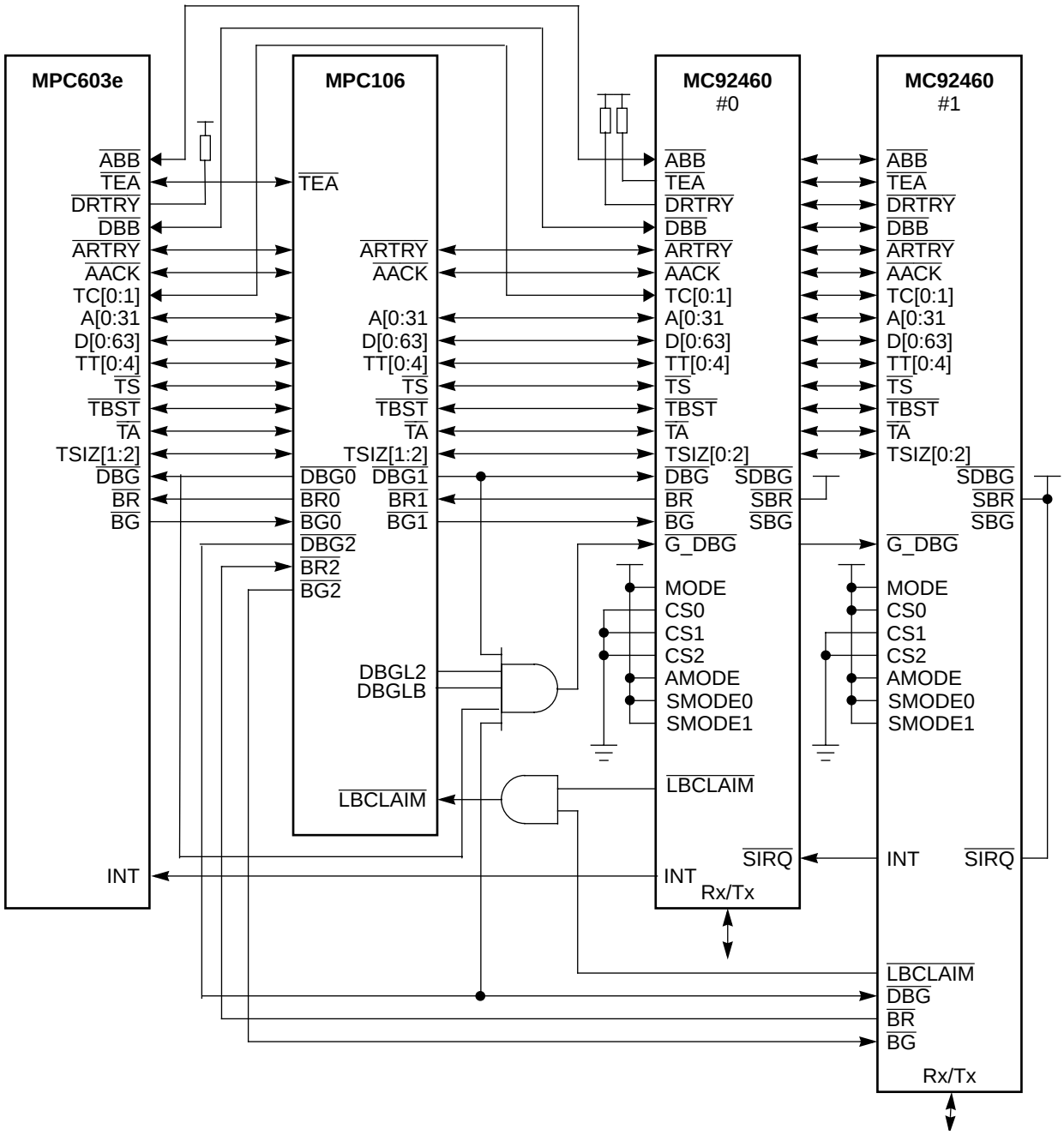


Figure 1-6. MPC603e, MPC106, and MC92460 System

Chapter 2
Signals

This chapter describes the MC92460 signals.

2.1 Signal Groups

Figure 2-1 shows the MC92460 external signals grouped by function.

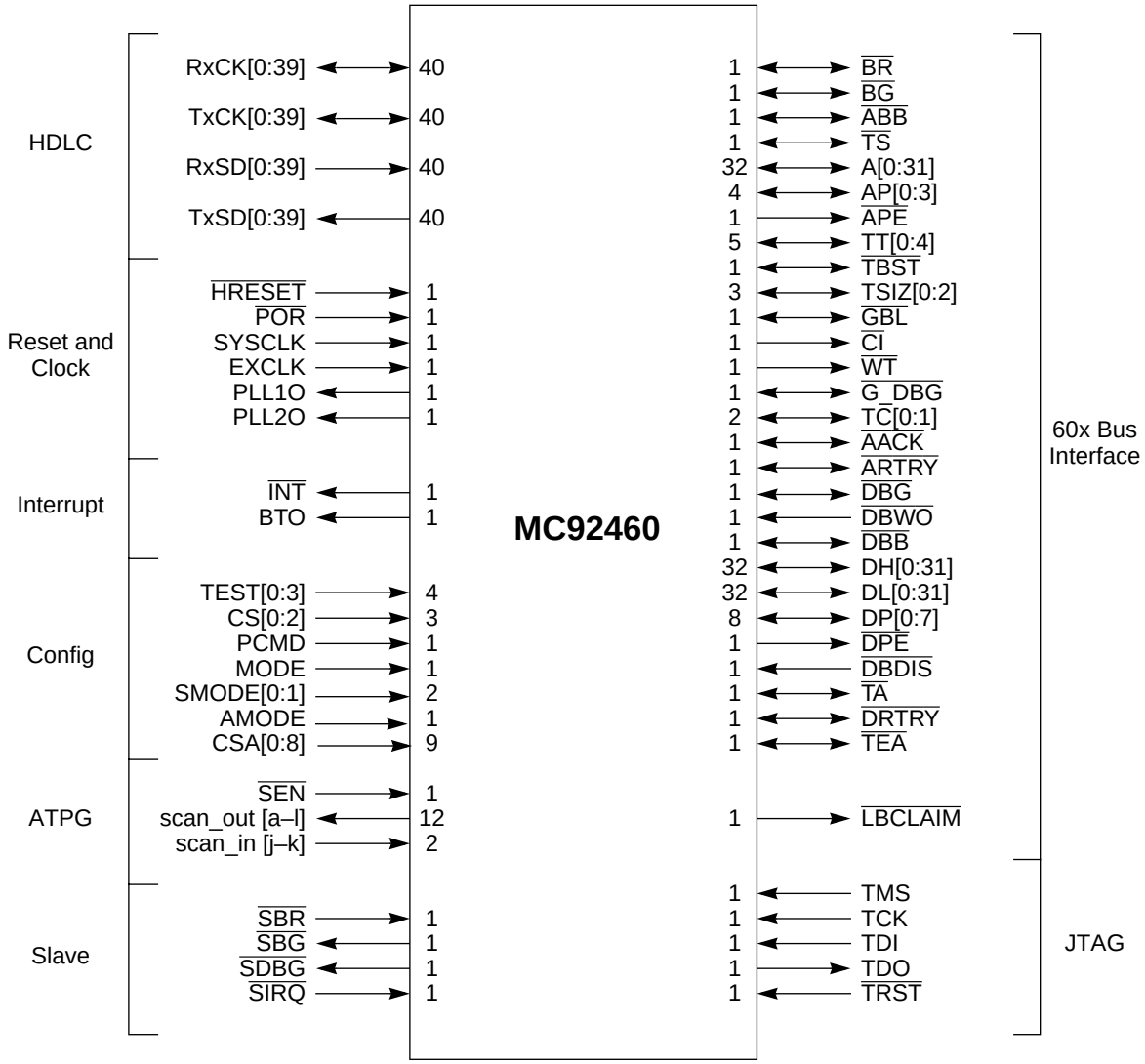


Figure 2-1. MC92460 External Signals

2.2 Signal Descriptions

The MC92460 system bus, shown in Table 2-1, consists of all the signals that interface with the external bus. Many of these signals perform different functions, depending on how the user assigns them.

Table 2-1. Signal Descriptions

Signal	Description	I/O	Type	mA	PU/PD
60x Bus Interface Signals					
$\overline{BR}$	60x bus request—This signal is an output when an external arbiter is used and an input when an internal main arbiter is used. As an output, the MC92460 asserts $\overline{BR}$ to request ownership of the 60x bus in slave mode. As an input, an external master should assert $\overline{BR}$ to request 60x bus ownership from the internal arbiter.	Input/ output	Three-state	10	
$\overline{BG}$	60x bus grant—This is an output signal when an internal main arbiter is used and an input when an external arbiter is used. As an output the MC92460 asserts $\overline{BG}$ to grant 60x bus ownership to an external bus master. As an input the external arbiter should assert $\overline{BG}$ to grant 60x bus ownership to the MC92460.	Output/ input	Three-state	10	
$\overline{ABB}$	60x address bus busy—As an output the MC92460 asserts this signal for the duration of the address bus tenure. Following an $\overline{AACK}$, which terminates the address bus tenure, the MC92460 negates $\overline{ABB}$ for a fraction of a bus cycle and then stops driving this signal. As an input, the MC92460 will not assume 60x bus ownership as long as it senses this signal is asserted by an external 60x bus master.	Input/ output	Three-state	10	
$\overline{TS}$	60x bus transfer start—Assertion of this signal indicates the beginning of a new address bus tenure. The MC92460 asserts $\overline{TS}$ when the DMA internal 60x bus master begins an address tenure. When the MC92460 senses $\overline{TS}$ being asserted by an external 60x bus master, it will respond to the address bus tenure as required.	Input/ output	Three-state	10	
A[0-31]	60x address bus—When the MC92460 is in external master bus mode, these signals function as the 60x address bus. The MC92460 drives the address of its internal 60x bus masters and respond to addresses generated by external 60x bus masters. When the MC92460 is in internal master bus mode, A[0-31] are used as address lines connected to memory devices and controlled by the memory controller.	Input/ output	Three-state	10	
AP[0-3]	Address parity —The 60x master that drives the address bus also drives the address parity signals. The value driven on the address parity signal should give odd parity (odd number of bits) on the group of signals that includes address parity 0 and A[0-7], AP1 A[8-15], AP2 A[16-23], AP3 A[24-31]	Input/ output	Three-state	10	

Table 2-1. Signal Descriptions (Continued)

Signal	Description	I/O	Type	mA	PU/PD
$\overline{\text{APE}}$	Address parity error—This output signal will be asserted when the MC92460 detects wrong parity driven on its address parity signals by an external master.	Output/ Input	Three-state	10	
TT[0-4]	60x bus transfer type—The 60x bus master drives TT[0-4] during the address tenure to specify the type of the transaction.	Input/ output	Three-state	10	
$\overline{\text{TBST}}$	60x bus transfer burst—The 60x bus master asserts $\overline{\text{TBST}}$ to indicate that the current transaction is a burst transaction (transfers 4 double words).	Input/ output	Three-state	10	
TSIZ[0-2]	60x transfer size—The 60x bus master drives these signals with a value indicating the amount of bytes transferred in the current transaction.	Input/ output	Three-state	10	
$\overline{\text{AACK}}$	60x address acknowledge—A 60x bus slave asserts $\overline{\text{AACK}}$ to indicate that it identified the address tenure. Assertion of this signal terminates the address tenure. Negation of $\overline{\text{AACK}}$ must occur one bus clock cycle after it is asserted.	Output/ input	Three-state	10	
$\overline{\text{ARTRY}}$	60x address retry—Assertion of $\overline{\text{ARTRY}}$ indicates that the bus transaction should be retried by the 60x bus master. The MC92460 never asserts this signal.	Input/ output	Three-state	10	
$\overline{\text{DBG}}$	60x data bus grant— In master mode the MC92460 asserts $\overline{\text{DBG}}$ to grant 60x data bus ownership to an external bus master. In slave mode, $\overline{\text{DBG}}$ is a data grant input signal. Note: see AMODE.	Output/ Input	Three-state	10	
$\overline{\text{DBB}}$	60x data bus busy—As an output the MC92460 asserts $\overline{\text{DBB}}$ for the duration of the data bus tenure. Following a $\overline{\text{TA}}$, which terminates the data bus tenure, the MC92460 negates $\overline{\text{DBB}}$ for a fraction of a bus cycle and then stops driving this signal. As an input, the MC92460 does not assume 60x data bus ownership as long as it senses $\overline{\text{DBB}}$ asserted by an external 60x bus master. Note: see AMODE.	Input/ output	Three-state	10	
DH[0-31] DL[0-31]	60x data bus—These are input/output signals. In write transactions, the 60x bus master drives the valid data on this bus. In read transactions, the 60x slave drives the valid data on this bus.	Input/ output	Three-state	10	
DP[0-7]	60x data parity—(Input/output)The 60x agent that drives the data bus drives also the data parity signals. The value driven on the data parity signals should give odd parity (odd number of bits) on the group of signals that includes data parity 0 and D[0–7], 1 and D[8-15], 2 and D[16-23], 3 and D[24-31], 4 and D[32-39], 5 and D[40-47], 6 and D[48-55], 7 and D[56-63].	Input/ output	Three-state	10	

Table 2-1. Signal Descriptions (Continued)

Signal	Description	I/O	Type	mA	PU/PD
$\overline{TA}$	Transfer acknowledge—Indicates that a 60x data beat is valid on the data bus. For 60x single beat transfers, assertion of this signal indicates the termination of the transfer. For 60x burst transfers, TA is asserted four times to indicate the transfer of four data beats with the last assertion indicating the termination of the burst transfer. Negation must occur after the one and half bus clock cycle after assertion of $\overline{TA}$.	Output/ input	Three-state	10	U100[1]
$\overline{TEA}$	Transfer error acknowledge—Assertion of this signal indicates a bus error. Any 60x masters within the MC92460 monitor the state of this signal. The MC92460's internal bus monitor may assert this signal in case it identifies a hung 60x bus transfer.	Input/ output	Three-state	10	
$\overline{GBL}$	Global—When an MC92460 master within the chip initiates a bus transaction, it drives this signal. Assertion of this signal indicates that the transfer is global and it should be snooped by caches in the system.	Output/ Input	Open Drain	10	
$\overline{CI}$	Cache inhibit—Used for L2 cache control. For each MC92460 60x transaction initiated in the core, the state of this signal indicates if this transaction should be cached. Assertion of the CI signal indicates that the transaction should not be cached. When the MC92460 is in master mode, it always asserts this signal.	Output	Open Drain	10	
$\overline{WT}$	Write-through—Used for L2 cache control. For each core-initiated MC92460 60x transaction, the state of this signal indicates if the transaction should be cached using write-through or copy-back mode. Assertion of WT indicates that the transaction should be cached using the write- through mode. When a MC92460 is in master mode, MC92460 always negates this signal.	Output	Open Drain	10	
$\overline{G\_DBG}$	Global $\overline{DBG}$ input/output If MODE is 1, $\overline{G\_DBG}$ becomes an input signal. One $\overline{DBG}$ is ANDed with another $\overline{DBG}$. If MODE is 0, this signal becomes an output signal. All $\overline{DBG}$s are ANDed signal outputs.	Input/ Output		5	U100[1]
$\overline{LBCLAIM}$	$\overline{LBCLAIM}$ output—Used with the MPC106 and MPC107, this signal connects to the MPC106 & MPC107 $\overline{LBCLAIM}$ input, and drives SMODE0 input high.	Input/ Output	Three-state	10	
TC[0-1]	Transfer Code—The transfer code output signals supply information that can be useful for debug purposes for each of the MC92460's initiated bus transactions.	Input/ Output	Three-state	10	

Table 2-1. Signal Descriptions (Continued)

Signal	Description	I/O	Type	mA	PU/PD
$\overline{\text{DBWO}}$	Data bus write only—The processor can run the data bus tenure for an outstanding write address even if a read address is pipelined before the write address. If write data is not available, the processor performs the first pending read transfer. Hiz output only..	Input/ Output	Three-state	10	
$\overline{\text{DPE}}$	Data parity error—The processor detects incorrect data bus parity on incoming read data.	Output/ Input	Three-state	10	
$\overline{\text{DBDIS}}$	Data bus disable—For a write transaction, the processor must release the data bus and DP[07] to high impedance in the next cycle. The data tenure remains active, $\overline{\text{DBB}}$ remains driven, and the transfer termination signals are still monitored by the processor. The $\overline{\text{DBDIS}}$ signal is ignored for reads.	Input/ Output	Three-state	10	
$\overline{\text{DRTRY}}$	Data retry—The master must invalidate the data from the previous read operation. $\overline{\text{DRTRY}}$ is ignored for write transactions and is not defined for direct-store transfers.	Output/ input	Three-state	10	
JTAG					
TMS	JTAG Test Mode Status	Input			U100[1]
TCK	JTAG Test Clock	Input			
TDI	JTAG Test Data Input	Input			U100[1]
TDO	JTAG Test Data Output	Output	Three-state	5	
$\overline{\text{TRST}}$	JTAG Test Reset	Input			U100[1]
HDLC Channels					
RxCK[0-39]	Receive Clock	Input/ Output	Three-state	5	
TxCK[0-39]	Transmit Clock	Input/ Output	Three-state	5	
RxSD[0-39]	Receive Serial Data Input	Input/ Output	Three-state	5	
TxSD[0-39]	Transmit Serial Data Output	Output/ Input	Three-state	5	
Interrupt					
$\overline{\text{INT}}$	Interrupt output for level interrupt	Output		10	
BTO	Bus time out; When $\overline{\text{TA}}$ does not assert within a 32 clock cycle, BTO goes high.	Input/ Output	Three-state	10	
Slave					
$\overline{\text{SBR}}$	Bus request from slave device	Input/ Output	Three-state	5	
$\overline{\text{SBG}}$	Bus grant to slave device	Output		5	
$\overline{\text{SDBG}}$	Data bus grant to slave device	Output		5	
$\overline{\text{SIRQ}}$	Slave interrupt request input	Input/ Output	Three-state	5	

Table 2-1. Signal Descriptions (Continued)

Signal	Description	I/O	Type	mA	PU/PD
Configuration					
TEST[0-3]	Function test select signal. This is always set to 1010.	Input			
CS[0-2]	Chip select signal	Input/ Output		5	
CSA[0–8]	Chip select signal	Input		5	D100[2]= CSA6 CSA8 U100[1]= CSA[0:5] CSA7
MODE	Select master or slave for the MC92460: Master mode = 0; Slave mode = 1	Input/ Output		5	
PCMD	CPU's PLL configure mode. Low input at bus: core = 1:1, High input at other bus clocking.	Input/ Output		5	
SMODE0	System Mode0—When the MC92460 is to be used with the MPC106, set SMODE0 high (= 1) When the MC92460 is to be used with the MPC8260, set SMODE0 low (= 0)	Input			U100[1]
SMODE1	System Mode1—Not used: SMODE1 is connected to VDD	Input			U100[1]
AMODE	Arbiter Mode- Select fast arbitration mode(AMODE = 1), or clocked arbitration mode(AMODE = 0). • When AMODE is 0 or 1, so long as MODE is 1, $\overline{DBG}$ is an input. • When AMODE is 0 or 1, MODE is 0, and $\overline{DBB}$ is negated, $\overline{DBG}$ will be asserted at the next cycle of $\overline{TS}$ assertion. • But when AMODE is 0, MODE is 0, and $\overline{DBB}$ is asserted, $\overline{DBG}$ is asserted as soon as $\overline{DBB}$ is negated • And when AMODE is 1, MODE is 0, and $\overline{DBB}$ is asserted, $\overline{DBG}$ is asserted one cycle after $\overline{DBB}$ is negated	Input			D100[2]
Reset and Clock					
$\overline{HRESET}$	Hard reset—When asserted, this line causes the MC92460 to enter a hard reset state.	Input	Schmitt		
$\overline{POR}$	Power on reset	Input			
SYSCLK	Clock in—This is the primary clock input.	Input			
EXCLK	External baud rate clock input	Input			
PLL1O	VCO1 output for test	Output	Analog		
PLL2O	VCO2 output for test	Output	Analog		
ATPG (Scan test)					
$\overline{SEN}$	Scan test enable • Low = scan test mode • High = normal mode	Input			D100[2]
scan_in[j-k]	Scan mode test input signals	Input			D100[2]

Table 2-1. Signal Descriptions (Continued)

Signal	Description	I/O	Type	mA	PU/PD
scan_out[a-l]	Scan mode test output signals	Output /Input		5	D100[2]
Power					
ACoreGnd1	GND for PLL1				
ACoreVdd1	VDD for PLL1				
ACoreGnd2	GND for PLL2				
ACoreVdd2	VDD for PLL2				

1 U100: internal pull-up 100K ohms

2 D100: internal pull-down 100K ohms

Chapter 3
System Control Unit

The system control unit (SCU) of the MC92460 consists of several functions that control internal memory, the high-level data link controllers (HDLCs), and the external system bus. Key features of the SCU include the following:

- System configuration and protection
- Arbitration of the internal memory (SRAM) access between the HDLCs and the external system bus
- Flexible, high-performance memory controller arbitration of the 60x bus data (coming through the bus interface) and the HDLC transmit and receive data to the buffers
- Storage of HDLC channel data in buffers by referencing 4096 buffer descriptors (2048 TxBDs and 2048 RxBDs) that reside in dual-port RAM. Refer to Section 4.4, “HDLC Buffer Descriptors (BDs) and Buffers.”

Figure 3-1 shows the SCU block diagram.

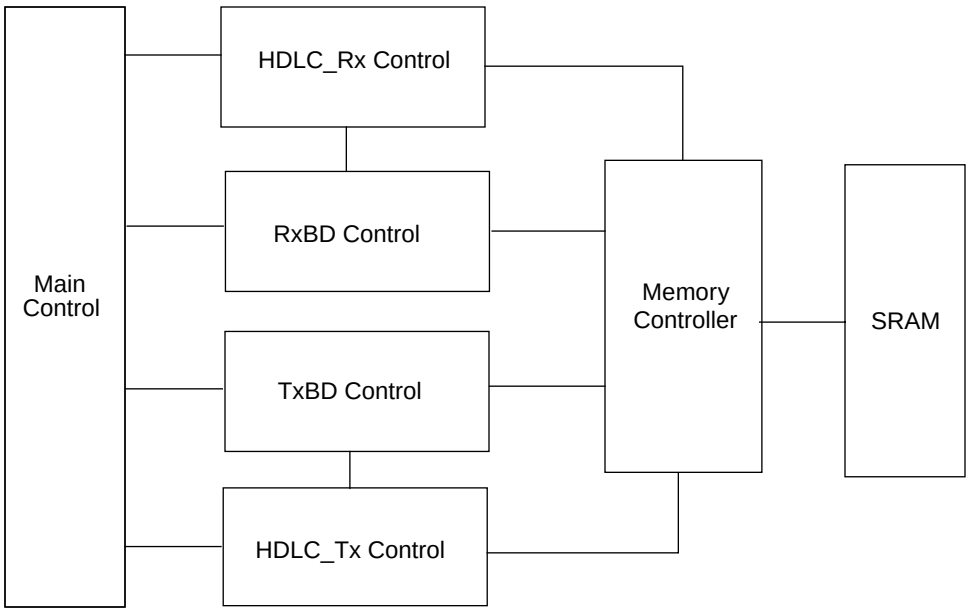


Figure 3-1. SCU Block Diagram

3.1 SCU Data Processing

Following are the data transmit and receive processes of the MC92460.

3.1.1 External Memory Receive

When data arrives, the SCU performs required processing on the data and moves resultant data to the buffer pointed to by the first BD; it continues until the buffer is full or an event, such as an error or end-of-frame detection, occurs. The buffer is then closed; subsequent data uses the next BD.

On the other hand, if the SCU tries to move data to a buffer that is not empty (E = 0), the buffer will report a busy error. The SCU will not move from this current BD until E is set to 1 (empty) by the core. The SCU waits until the core completes processing the BD. After a buffer descriptor is used (is full), the SCU clears E, and the BD is not reused until it has been processed by the core. However, in continuous mode (CM), E remains set to 1. When the SCU discovers a buffer descriptor's wrap (W) bit set (indicating it is the last BD in the circular BD table), it returns to the beginning of the table when it is time to move to the next buffer.

If BDs point to external memory, BDs access the 60x bus with burst beat access only. So a low-order buffer pointer must be oriented on an address that is a multiple of 8. Buffer descriptors transfer 32 bytes of data every time.

Figure 3-2 shows the sequence of data received with RxBD. In this case, the RxBD points to external memory.

1. Assuming the RxBD is empty, the core sets E = 1.
2. The SCU checks the enable bit of the RxBD to make sure the buffer descriptor is empty.
3. The core (application program) sets the enable bit.
4. Data is received by the HDLC .
5. Data is moved to the Rx FIFO.
6. A data transfer request from the arbiter is received by the SCU.
7. The DMA reads the Rx FIFO.
8. Data is transferred from the Rx FIFO to memory.

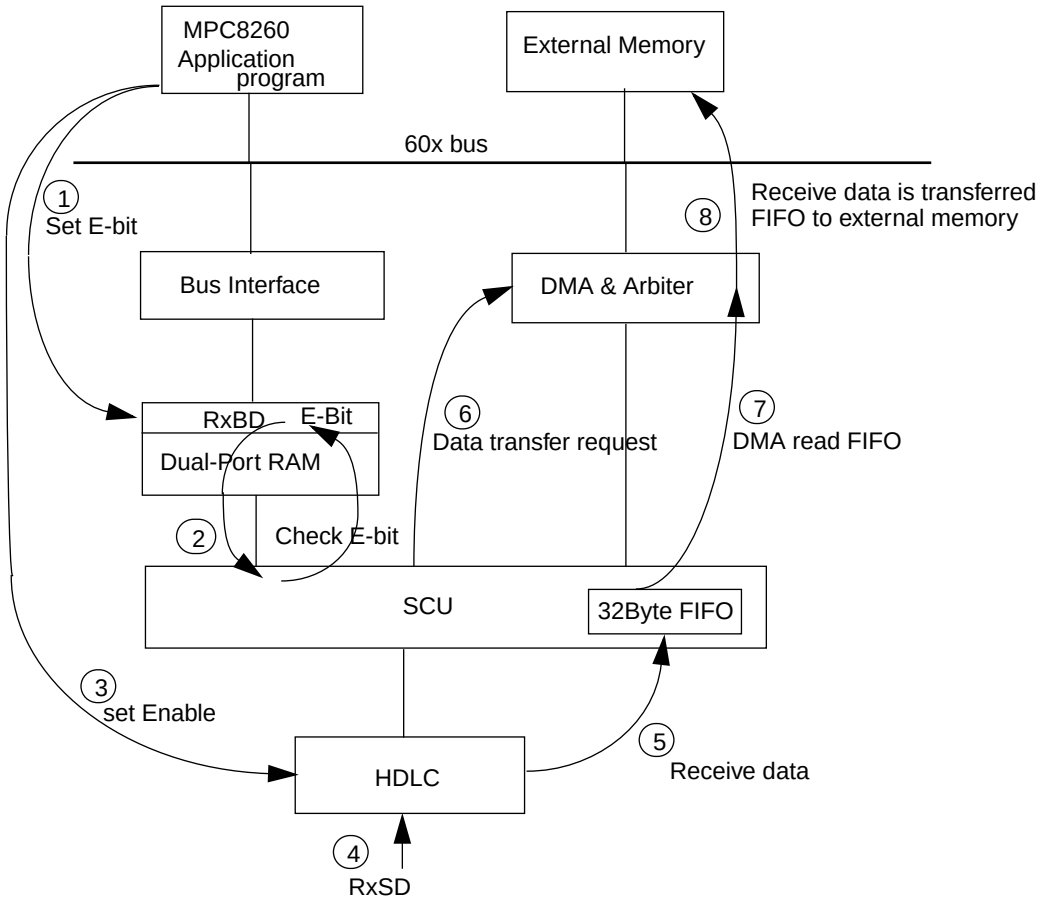


Figure 3-2. External Memory Receive Sequence with RxBD

3.1.2 External Memory Transmit

When the core sets the R bit to enable the transmit side of an HDLC, the SCU starts with the first BD in that TxBD table. Once the SCU detects that the R bit is set in the TxBD, it starts processing the buffer. The SCU detects that the BD is ready when it polls the R bit or when the user writes to the TODR. After data from the BD is put in the Tx FIFO, if necessary the SCU waits for the next descriptor's R bit to be set before proceeding. Thus, the SCU does no look-ahead descriptor processing and does not skip BDs that are not ready. When the SCU sees a buffer descriptor's W bit set, it returns to the start of the BD table after this last BD of the table is processed. The SCU clears R (not ready) after using a TxBD, which keeps it from being retransmitted before it is confirmed by the core. However, some protocols support a continuous mode (CM), for which R is not cleared (signifying that it is always ready).

Figure 3-3 shows the sequence of data transmitted with TxBD. TxBD points to external memory.

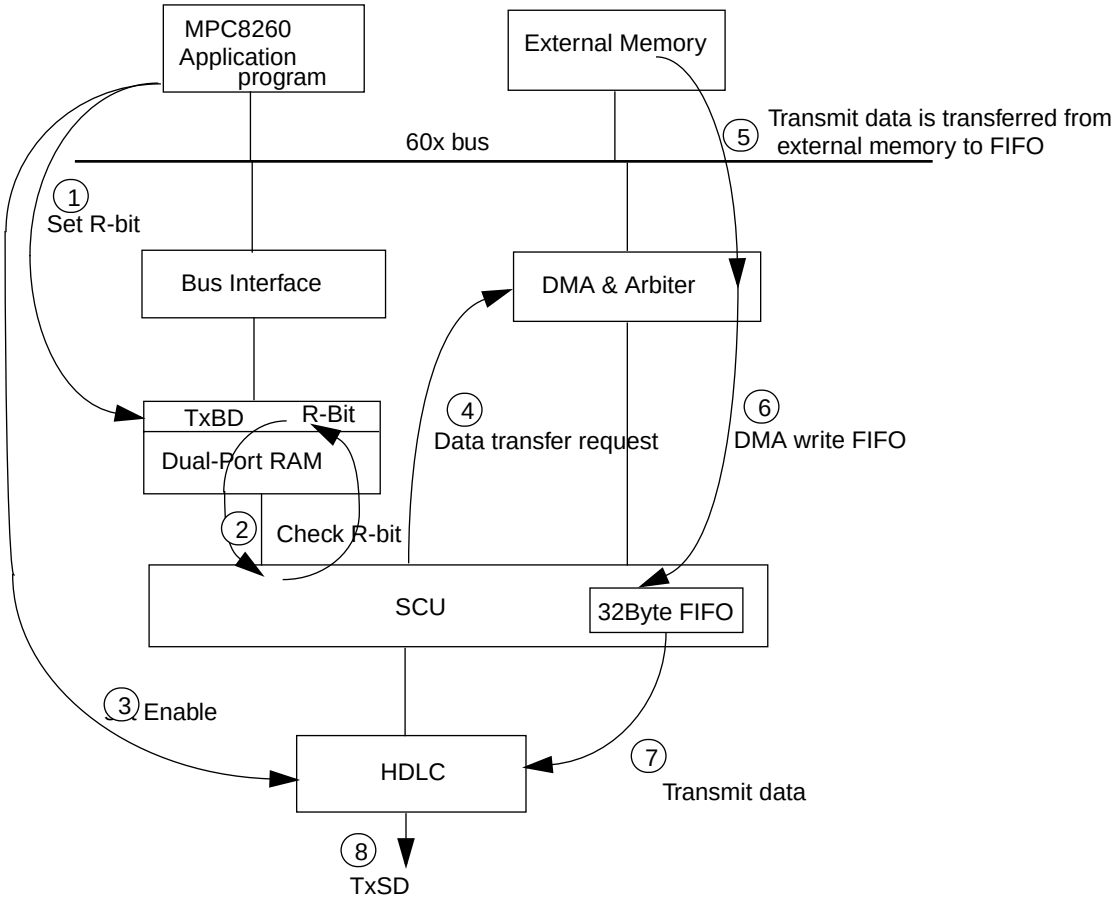


Figure 3-3. External Memory Transmit Sequence with TxBD

3.1.3 SRAM Transmission Using MC92460

Shown in Figure 3-4 is the data transmission sequence from external memory to internal SRAM of the MC92460 using a DMA write, and from there to the HDLC. The sequence of RxBDs is the same as the sequence of TxBDs.

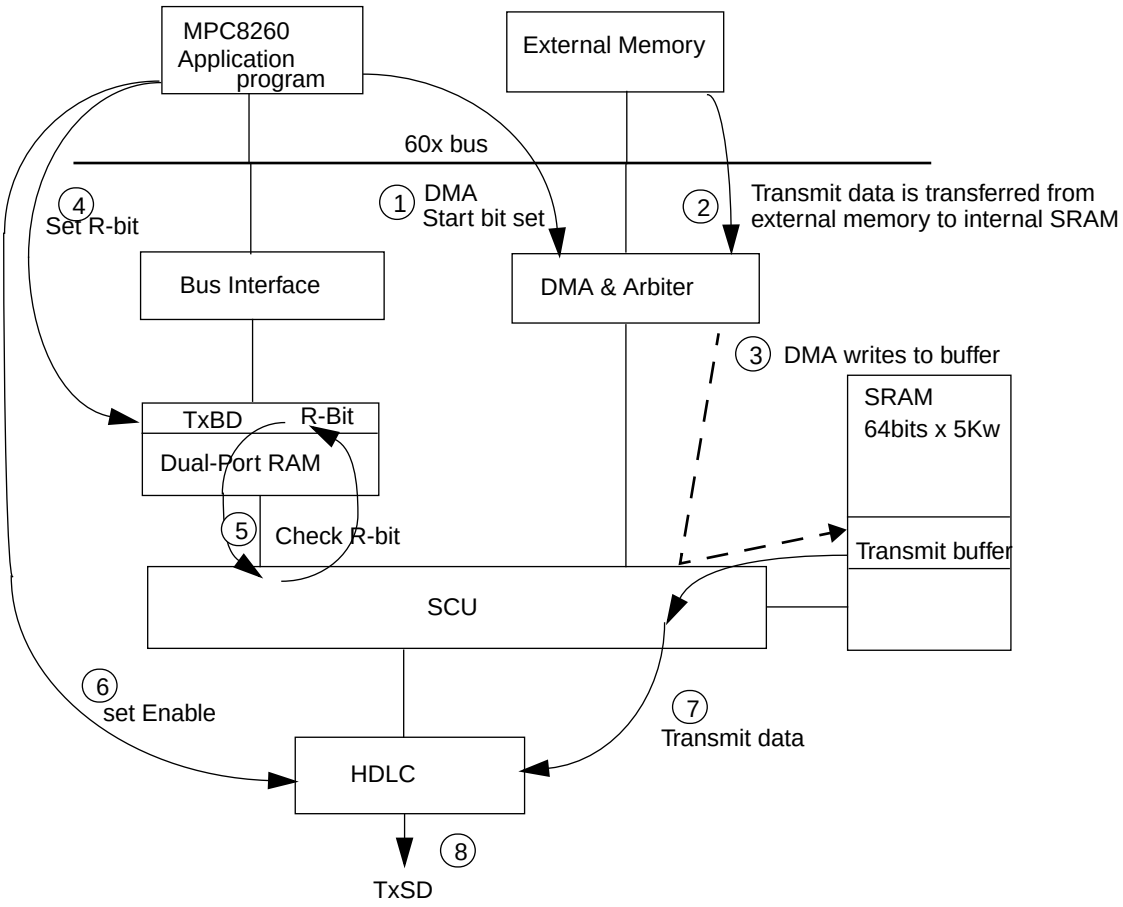


Figure 3-4. Internal SRAM Transmit Sequence Using MC92460 DMA

3.1.4 SRAM Transmission Using MPC8260

Shown in Figure 3-5 is the data transmission sequence from external memory to internal SRAM arbitrated by the MPC8260. The sequence of RxBDs is the same as the sequence of TxBDs.

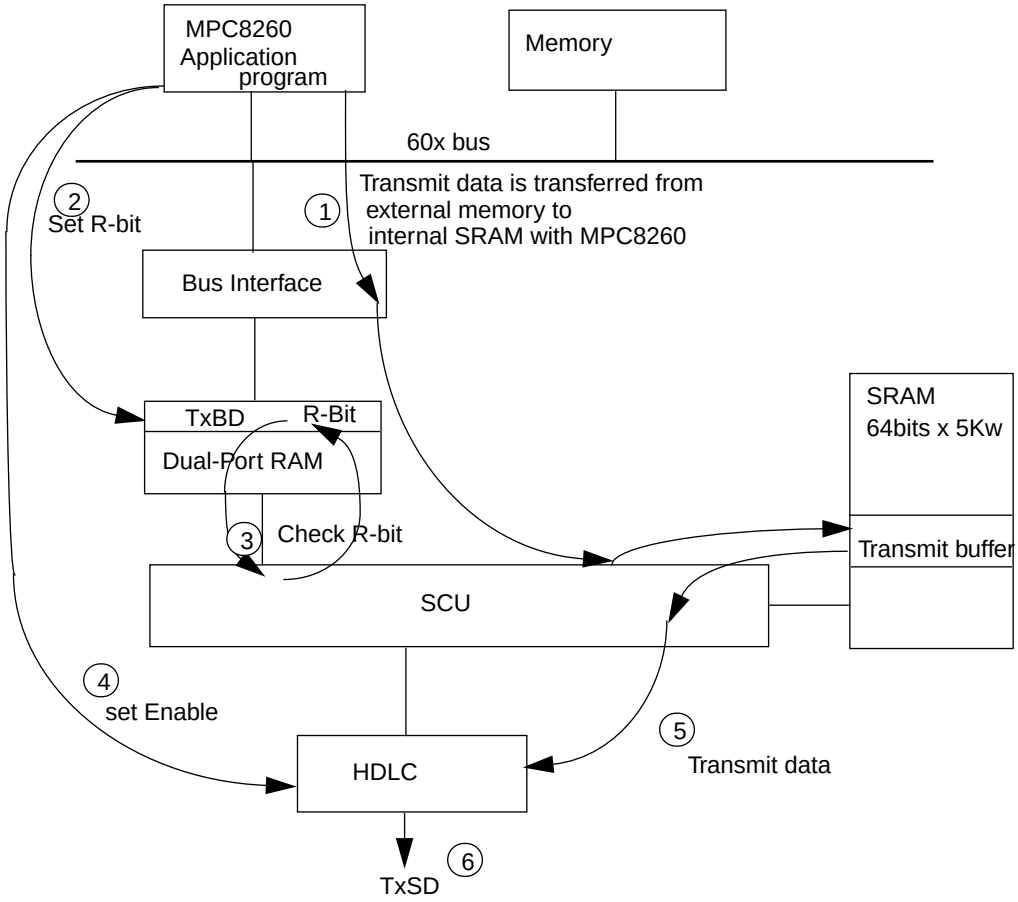


Figure 3-5. Internal SRAM Transmit Sequence Using MPC8260

3.2 Interrupt Status

Follow these steps to handle an HDLC interrupt:

1. When an interrupt occurs, read the event register (ER) of the HDLC to determine the interrupt sources and clear those ER bits (in most cases).
2. Process the TxBDs to reuse them if ER[TXB] or ER[TXE] = 1. If the transmit speed is fast or the interrupt delay is long, the HDLC may have sent more than one Tx BD. Thus, it is important to check more than one TxBD during interrupt handling. A common practice is to process all TxBDs in the handler until one is found with its R bit set.
3. Extract data from the RxBD if ER[RX], ER[RXB], or ER[RXF] is set. As with transmit buffers, if the receive speed is fast or the interrupt delay is long, the HDLC may have received more than one buffer and the handler should check more than one RxBD. A common practice is to process all RxBDs in the interrupt handler until one is found with its E bit set. This 40-bit, read-only INT1 and INT2 register allows monitoring of the real-time status.

NOTE:

The MC92460 supports level interrupts only.

3.2.1 Interrupt Status Register 1 (INTS1)

The INTS1 register is used to report events recognized by the 32 HDLC channels and to generate interrupts. INTS1 can be read at any time. Bits are cleared by writing a 1 to the ER of each HDLC; writing zeros does not affect bit values.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	HDLC 0	HDLC 1	HDLC 2	HDLC 3	HDLC 4	HDLC 5	HDLC 6	HDLC 7	HDLC 8	HDLC 9	HDLC 10	HDLC 11	HDLC 12	HDLC 13	HDLC 14	HDLC 15
Reset	0000_0000_0000_0000															
R/W	R															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	HDLC 16	HDLC 17	HDLC 18	HDLC 19	HDLC 20	HDLC 21	HDLC 22	HDLC 23	HDLC 24	HDLC 25	HDLC 26	HDLC 27	HDLC 28	HDLC 29	HDLC 30	HDLC 31
Reset	00000_0000_0000_0000															
R/W	R															

Figure 3-6. Interrupt Status Register (INTS1)

Table 3-1. Interrupt Status Register Field Description

Field	Description
HDLC[0-31]	HDLCx generate interrupt request

3.2.2 Interrupt Status Register 2 (INTS2)

The INTS2 register is used to report events recognized by the 8 HDLC channels, the DMASR, SIRQ, and to generate interrupts. INTS2 can be read at any time. Bits 0-7 of INTS2 are cleared by writing a one to the ER of each HDLC. Bit 15 is cleared by writing ones to DMASR, bit 16 is cleared by ones to the ER of all HDLCs, and bit 17 is cleared by writing ones to the ER of all HDLCs and to the DMASR of all slave MC92460s; writing zeros does not affect bit values.

	0	1	2	3	4	5	6	7	8	14	15
Field	HDLC32	HDLC33	HDLC34	HDLC35	HDLC36	HDLC37	HDLC38	HDLC39	—		DMA
Reset	0000_0000_0000_0000										
R/W	R										

	16	17	18	31
Field	INT	EXT	—	
Reset	0000_0000_0000_0000			
R/W	R			

Figure 3-7. Interrupt Status Register 2 (INTS2)

Table 3-2. Interrupt Status Register 2 Descriptions

Bits	Field	Description
0–7	HDLC [32-39]	HDLC*x* generate interrupt request
8–14	—	Reserved
15	DMA	DMA generate interrupt request
16	INT	HDLC0 to HDLC39 or DMA generate interrupt request
17	EXT	Slave devices drive $\overline{\text{SIRQ}}$ signal
18–31	—	Reserved

3.3 HDLC Channels

The SCU can support a maximum of 40 HDLC channels at the low baud rate. All HDLC channels can be programmed by the SCU.

3.3.1 Maximum HDLC Channel Register (MMCR)

The MMCR sets the maximum number of HDLC channels that can be programmed. The number can be up to 40.

The configuration in Table 3-8 should be chosen when 40 HDLC channels are needed. MMCR does not support dynamic reassigns, so before changing the MMCR, all HDLC channels should be disabled by setting MRA[ENT] and MRA[ENR] to zero.

	0 1	2 7	8 15
Field	—	MHC	—
Reset	00	10_1000	0000_0000
R/W	R	R/W	R

Figure 3-8. Max HDLC Channel Register (MMCR)

Table 3-3. Maximum HDLC Channel Register Description

Bits	Field	Description
0–1	—	Reserved. Writing a 1 to this register can cause boundedly undefined results.
2–7	MHC	Program Max HDLC channel. Example1: 0x28 SCU supports HDLC0 to HDLC39 Example2: 0x08 SCU supports HDLC0 to HDLC7
8–15	—	Reserved. Writing a 1 to this register can cause boundedly undefined results.

3.4 RxBD and TxBD Table Base Addresses

Each of the 40 channels has an RxBD and TxBD table in dual-port RAM that holds the buffer descriptors for that channel. The default number of BDs for each channel is 64, of which 32 are RxBDs and 32 are TxBDs.The base addresses of these RxBD and TxBD tables is defined by the RBASE and TBASE registers.

3.4.1 RxBD Table Base Address (RBASE)

The RBASE register defines the base address of each channel's RxBD table. RBASE should be 8 byte aligned.

	0 – 13	14 – 15
Field	—	Upper value
Reset	0000_0000_0000_00	Table 3-4
R/W	R	R/W

	16 – 28	29 – 31
Field	Lower value	—
Reset	Table 3-4	000
R/W	R/W	R

Figure 3-9. RBASE Register

The default base addresses of the RxBD table shown in Table 3-4 allocate 32 buffer descriptors for each of the 40 channels. Dual-port RAM, however, holds up to 2048 Rx buffer descriptors that can be distributed among the 40 RxBD tables. Thus it is possible to place all Rx buffer descriptors in one RxBD table and none in the other tables.

To reconfigure the number of buffer descriptors in each channel, change the RBASE addresses in parameter RAM.

Table 3-4. Default Base Addresses for RxBD Table

Name	Address	Name	Address	Name	Address	Name	Address
RBASE0	0x10000	RBASE1	0x10100	RBASE2	0x10200	RBASE3	0x10300
RBASE4	0x10400	RBASE5	0x10500	RBASE6	0x10600	RBASE7	0x10700
RBASE8	0x10800	RBASE9	0x10900	RBASE10	0x10a00	RBASE11	0x10b00
RBASE12	0x10c00	RBASE13	0x10d00	RBASE14	0x10e00	RBASE15	0x10f00
RBASE16	0x11000	RBASE17	0x11100	RBASE18	0x11200	RBASE19	0x11300
RBASE20	0x11400	RBASE21	0x11500	RBASE22	0x11600	RBASE23	0x11700
RBASE24	0x11800	RBASE25	0x11900	RBASE26	0x11a00	RBASE27	0x11b00
RBASE28	0x11c00	RBASE29	0x11d00	RBASE30	0x11e00	RBASE31	0x11f00
RBASE32	0x12000	RBASE33	0x12100	RBASE34	0x12200	RBASE35	0x12300
RBASE36	0x12400	RBASE37	0x12500	RBASE38	0x12600	RBASE39	0x12700

3.4.2 TxBD Table Base Address (TBASE)

The TBASE register defines the base address of each channel's TxBD table. TBASE should be 8 byte aligned.

	0	13	14	15
Field	—		Upper value	
Reset	0000_0000_0000_00		Table 3-5	
R/W	R		R/W	

	16	28	29	31
Field	Lower value		—	
Reset	Table 3-5		000	
R/W	R/W		R	

Figure 3-10. TBASE Register

The default base addresses of the TxBD table shown in Table 3-5 allocate 32 buffer descriptors for each of the 40 channels. Dual-port RAM, however, holds up to 2048 Tx buffer descriptors that can be distributed among the 40 TxBD tables. Thus it is possible to place all Tx buffer descriptors in one TxBD table and none in the other tables.

To reconfigure the number of buffer descriptors in each channel, change the TBASE addresses in parameter RAM.

Table 3-5. Default Base Addresses for TxBD Table

Name	Address	Name	Address	Name	Address	Name	Address
TBASE0	0x1a000	TBASE1	0x1a100	TBASE2	0x1a200	TBASE3	0x1a300
TBASE4	0x1a400	TBASE5	0x1a500	TBASE6	0x1a600	TBASE7	0x1a700
TBASE8	0x1a800	TBASE9	0x1a900	TBASE10	0x1aa00	TBASE11	0x1ab00
TBASE12	0x1ac00	TBASE13	0x1ad00	TBASE14	0x1ae000	TBASE15	0x1af00
TBASE16	0x1b000	TBASE17	0x1b100	TBASE18	0x1b200	TBASE19	0x1b300
TBASE20	0x1b400	TBASE21	0x1b500	TBASE22	0x1b600	TBASE23	0x1b700
TBASE24	0x1b800	TBASE25	0x1b900	TBASE26	0x1ba00	TBASE27	0x1bb00
TBASE28	0x1bc00	TBASE29	0x1bd00	TBASE30	0x1be00	TBASE31	0x1bf00
TBASE32	0x1c000	TBASE33	0x1c100	TBASE34	0x1c200	TBASE35	0x1c300
TBASE36	0x1c400	TBASE37	0x1c500	TBASE38	0x1c600	TBASE39	0x1c700

3.5 Communication Buffer (SRAM)

The MC92460 has a 64-bit X 5 Kword SRAM that can be used to hold the transmit and receive buffer data pointed to by TxBD and RxBD. The bus master (MPC8260, MPC603e,

and DMA devices) can be accessed as memory.

Chapter 4
High-level Data Link Controller (HDLC)

4.1 Introduction

High-level data link control (HDLC) is one of the most common protocols in the data link layer, layer 2 of the OSI model. Many other common layer 2 protocols, such as synchronous data link control (SDLC), link access protocol-balanced (LAPB), and link access protocol for the D channel (LAPD), are based on HDLC and its framing structure in particular. Figure 4-2 shows the HDLC frame structure. HDLC uses a zero insertion/deletion process (bit-stuffing) to ensure that a data bit pattern matching the delimiter flag does not occur in a field between flags. The HDLC frame is synchronous and relies on the physical layer for clocking and synchronization of the transmitter/receiver. An address field is needed to carry the frame's destination address because the layer 2 frame can be sent over point-to-point links, broadcast networks, packet-switched, or circuit-switched systems.

An address field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. SDLC and LAPB use an 8-bit address. LAPD divides its 16-bit address into different fields to specify various access points within one device. LAPD also defines a broadcast address. Some HDLC-type protocols permit addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends on the protocol using the frame. The length of the data in the data field depends on the frame protocol. Layer 3 frames are carried in this data field.

Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16 bits long but can be as long as 32 bits. In HDLC, the least significant bit of each octet is sent first; the most significant bit of the CRC is sent first. In a nonmultiplexed modem interface, HDLC outputs connect directly to external pins.

An MC92460 HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with respect to other HDLCs. Independent baud rate generators allow flexible clocking for each HDLC channel.

Figure 4-1 shows a block diagram of a HDLC.

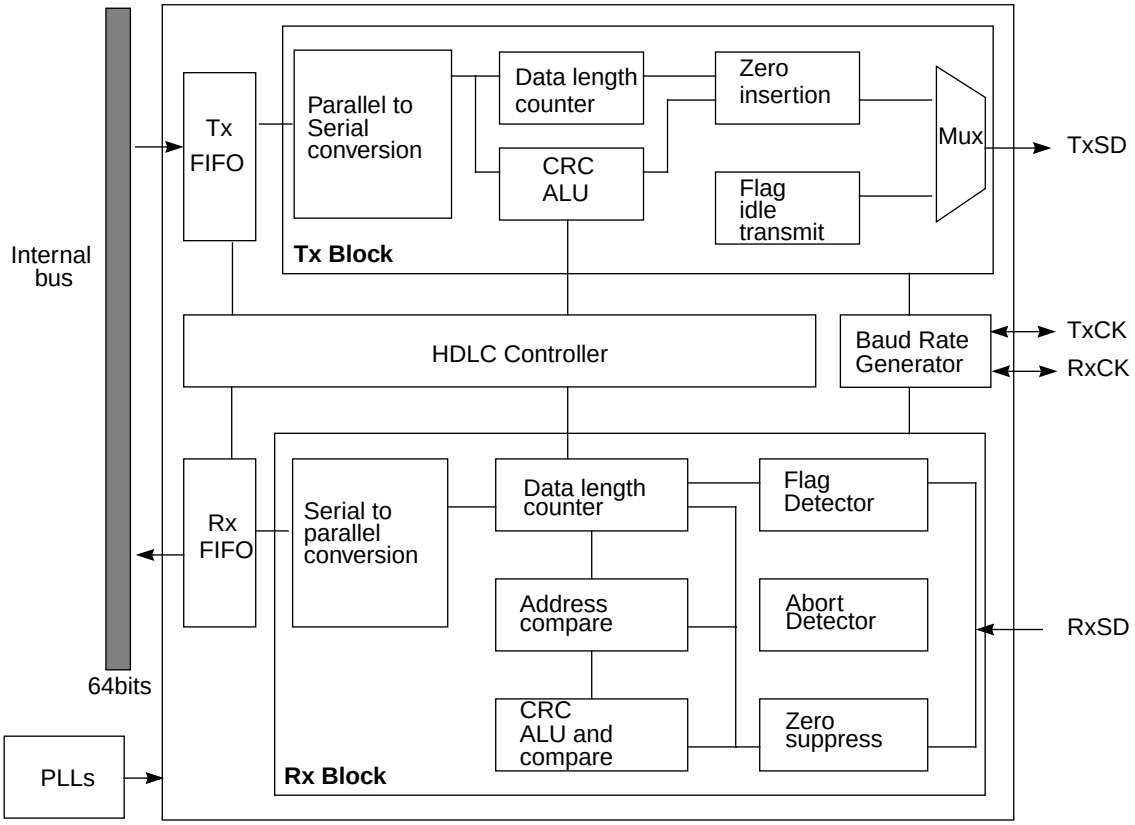


Figure 4-1. HDLC Controller Block Diagram

4.2 HDLC Features

The following are features of the HDLC controllers:

- Support for a maximum of 66.7 Mbps in each HDLC channel
- Flexible buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (Rx and Tx)
- Received-frames threshold to reduce interrupt overhead
- Flag/abort/idle generation and detection
- Zero insertion and deletion
- 16- or 32-bit CRC-CCITT generation and checking
- Detection of non-octet aligned frames
- Detection of frames that are too long
- Programmable flags (0-15) between successive frames
- Echo and local loopback mode for testing
- Independent baud rate generators providing clocks for the HDLC channels.

4.3 HDLC Channel Frame Transmission

HDLC protocol uses frames to send data. A frame is a packed bit stream of transmitted data. The HDLC frame has the following fields, explained in these sections:

- Section 4.3.1, “Flag Sequence Field”
- Section 4.3.2, “Address Field”
- Section 4.3.3, “Control Field”
- Section 4.3.4, “Information Field”
- Section 4.3.5, “Frame Check Sequence Field”

Figure 4-2 shows the HDLC frame structure.

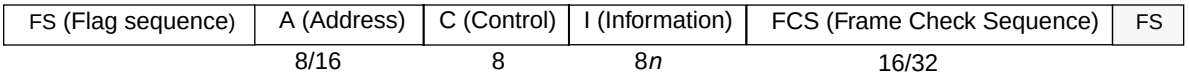


Figure 4-2. HDLC Frame Structure

4.3.1 Flag Sequence Field

All frames start and end with the flag sequence (FS) field that contains the sequence pattern 0x7E (0111_1110). All data stations that are attached to the data link continuously hunt for this sequence to distinguish the beginning and ending of frames. The flag shown in Figure 4-3 is used for frame synchronization.

Figure 4-3. Frame Sequence Flag

A single shared flag is used as both the opening and closing FS fields of a frame, as shown in Figure 4-4.

Figure 4-4. FS Used As Closing and Opening Flag

4.3.2 Address Field

In command frames, the address identifies the data station for which the command is intended. In response frames, the address identifies the data station from which the response originated.

Figure 4-5 shows the address field configuration.

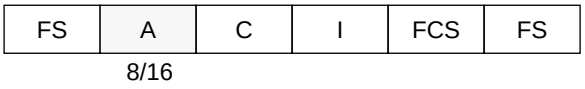


Figure 4-5. Address

The address field is 1 or 2 octets wide. Also the internal address register is a prepared16-bit length for a width of 2 octets. When the field needs only one octet, the HDLC controller recognizes the 8-bit field by the address mask register.

Table 4-1. Address Field

1 octet	2 octets	Description
0x00	0x0000	No-station address (TEST Address) This value must not be received by any receivers.
0x1A–0xFE	0x001A–0xFFFE	Address value. 254 addresses for 1 octet, 65534 addresses for 2 octets
0xFF	0xFFFF	Global address This value must be received by all receivers.

4.3.3 Control Field

The control field indicates the class of commands or responses to be carried out by the frame, and contains sequence numbers that specify any sequence that the command must follow. The control field conveys a command to perform a particular operation to the data station to which the command is addressed, or conveys a response to such a command from the addressed data station. The MC92460 should not recognize any data elements in the control field, as they are defined in the application.

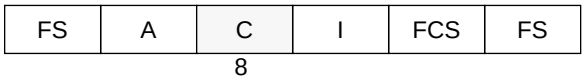


Figure 4-6. Control Field

Table 4-2 shows the three classes of HDLC control frames recognized by the HDLC controller used in the MC92460.

Table 4-2. Control Frame Classes

Control Frame Class	Description
I	Conveys not only information but also commands to operate transmission.
S	Conveys a command and response to supervise a data link.Applıcation programs define the S class.
U	Conveys a command and response for an operation mode setting. In some cases, the U class has no information field

4.3.4 Information Field

Information or data can be formatted in any sequence of bits and is stored in the information field that follows the control field, as seen in Figure 4-7.

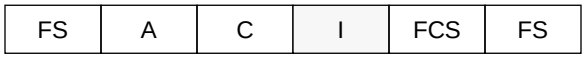


Figure 4-7. Information Field

In most cases, it will be linked to a convenient character structure (octets, for example) but if required, it may be an unspecified number of bits and unrelated to a character structure. For start/stop transmission, eight information bits exist between the start element and the stop element. If the information field is other than a multiple of 8 bits, the remaining bits are padded to complete the octet.

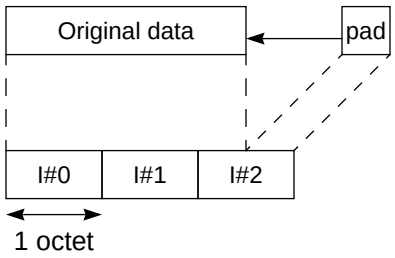


Figure 4-8. Padding to Fill Out an Octet

4.3.5 Frame Check Sequence Field

Two CRC-CCITT frame checking sequences are specified, a 16-bit frame checking sequence and a 32-bit sequence. The generator polynomials for each check sequence are described below in Figure 4-9.

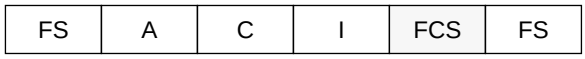


Figure 4-9. Frame Check

Table 4-3. CRC Generator Polynomials

Name	Generator Polynomial
CRC-CCITT16	X16+X12+X5+1
CRC-CCITT32	X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X1+1

4.4 HDLC Buffer Descriptors (BDs) and Buffers

The system control unit (SCU) stores the data associated with each HDLC channel in buffers (such as in SRAM), and each buffer is referenced by an 8-byte buffer descriptor (BD) that can reside anywhere in dual-port RAM. The MC92460 has 4096 BDs (2048 RxBDs and 2048 TxBDs).The total number of BDs is limited only by the size of the 32-Kbyte dual-port RAM. Each 64-bit BD has the following structure:

- The half word at offset + 0x0 contains status and control bits that control and report on the data transfer. These bits vary from protocol to protocol. The HDLC updates the status bits after the buffer is sent or received.
- The half word at offset + 0x2 (data length) holds the number of bytes sent or received.
  - For an RxBD, this is the number of bytes the controller writes into the buffer. The HDLC writes the length after received data is placed into the associated buffer and the buffer closed.

    In frame-based protocols such as is used by the MC92460, this field contains the total frame length, including CRC bytes. Also, if a received frame's length, including CRC, is an exact multiple of MRBLR, the last BD holds no actual data but does contain the total frame length.
  - For a TxBD, this is the number of bytes the controller should send from its buffer. Normally, this value should be greater than zero. The HDLC never modifies this field.
- The word at offset + 0x4 (buffer pointer) points to the beginning of the buffer in memory (internal or external). Both Rx and Tx buffer pointers can be even or odd in external memory. Because Rx and Tx buffers are 32 byte-aligned, the pointers at offset + 0x4 must be a multiple of 8 (8 byte-aligned).

Writes to buffer descriptors should be 32-/64-bit single beat or burst writes. Reads are described in Section 4.4.2, "Receive Buffer Descriptor (RxBD)" and Section 4.4.3, "Transmit Buffer Descriptor (TxBD)."

Shown in Figure 4-10, the format of Tx and Rx BDs is the same in each HDLC.

Bit	0															15
Offset + 0	Status and Command															
Offset + 2	Data Length															
Offset + 4	High-Order Buffer Pointer															
Offset + 6	Low-Order Buffer Pointer															

Figure 4-10. Buffer Descriptors

4.4.1 SRAM Buffers

The MC92460's Rx and Tx buffers are stored in a 64-bit x 4K word SRAM. Both Rx and Tx buffers are 32-byte aligned.

For frame-oriented protocols, a message can reside in as many buffers as necessary. Each buffer has a maximum length of 65,535 bytes. The SCU does not assume that all buffers of a single frame are currently linked to the BD table. The SCU does assume, however, that the unlinked buffers are provided by the core in time to be sent or received; otherwise, an error condition reports an underrun when sending and a busy when receiving.

Figure 4-11 shows the SCU BD table and buffer structure.

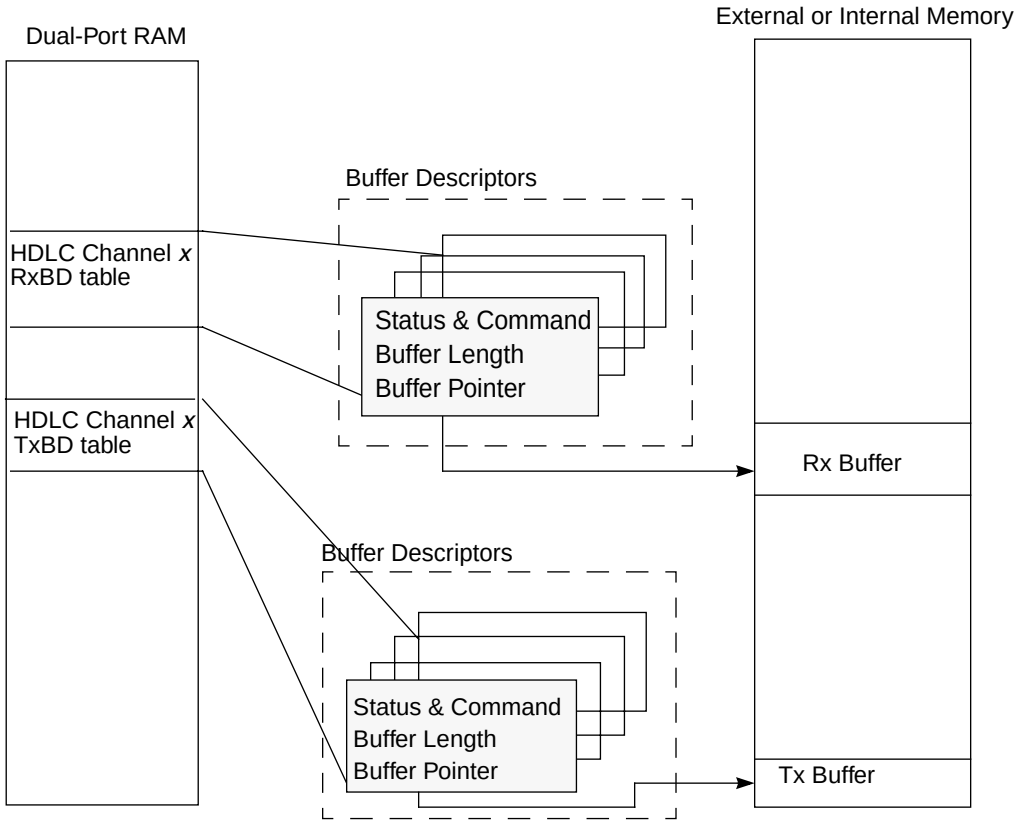


Figure 4-11. BD Table and Buffer Memory Structure

In all protocols, BDs can point to buffers in the internal RAM. However, because internal RAM is used for descriptors, buffers are usually put in external RAM, especially if they are large.

4.4.2 Receive Buffer Descriptor (RxBD)

Figure 4-12 shows the receive buffer descriptor (RxBD) format.

BIT	0	1	2	3	4	5	6	7		9	10	11	12	13	14	15
Offset + 0	E	X	W	I	L	F	CM	—			LG	NO	AB	CR	OV	BE
Offset + 2	Data length															
Offset + 4	Rx buffer pointer															
Offset + 6																

Figure 4-12. Receive Buffer Descriptor (RxBD)

Note that offsets + 0 through + 2 are accessed by half-words, whereas offsets +4 and +6 are accessed by words and half-words. The data length is not allowed to be zero.

Table 4-4 shows the function of each receive buffer descriptor bit.

Table 4-4. Receive Buffer Descriptor Bit Functions

Bit	Name	Description
0	E	Empty. 0 The buffer is full or reception stopped because of an error. The core can read or write to any fields of this RxBD. The SCU does not use this BD while E = 0. 1 The buffer is not full. The SCU controls the BD and buffer. The core should not update the BD.
1	X	External buffer. 0 This RxBD access the internal SRAM 1 This RxBD access the external memory
2	W	Wrap (last BD in the RxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the SCU receives incoming data using the BD pointed to by RBASE. The number of BDs in this table are programmable and determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 ER[RXB] is not set after this buffer is used; ER[RXF] is unaffected. 1 ER[RXB] or ER[RXF] is set when the HDLC uses this buffer.
4	L	Last buffer in frame. 0 Not the last buffer in frame. 1 Last buffer in frame. Indicates reception of a closing flag or an error, in which case one or more of the OV, AB, and LG bits are set. The HDLC writes the number of frame octets to the data length field.
5	F	First in frame. 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	CM	Continuous mode. Note that RxBD[E] is cleared if an error occurs during reception, regardless of CM. 0 Normal operation. 1 RxBD[E] is not cleared by the HDLC after this BD is closed, allowing the associated buffer to be overwritten next time the HDLC accesses it
7-9	—	Reserved

Table 4-4. Receive Buffer Descriptor Bit Functions (Continued)

Bit	Name	Description
10	LG	Rx frame length violation. Set when a frame larger than the maximum defined for this channel is recognized. Only the maximum-allowed number of bytes (MFLR) is written to the buffer. This event is not reported until the buffer is closed, ER[RXF] is set, and the closing flag is received. The total number of bytes received between flags is still written to the data length field.
11	NO	Rx non-octet aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	AB	Rx abort sequence. Set when at least seven consecutive ones are received during frame reception.
13	CR	Rx CRC error. Set when a frame contains a CRC error. CRC bytes received are always written to the Rx buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	BE	Bus transfer error. Set when a bus transfer error occurs.

4.4.3 Transmit Buffer Descriptor (TxBD)

Figure 4-13 shows the transmit buffer descriptor (TxBD) format.

Bits	0	1	2	3	4	5	6	7						13	14	15
Offset + 0	R	X	W	I	L	TC	CM	Reserved							UN	BE
Offset + 2	Data Length															
Offset + 4	Tx Buffer Pointer															
Offset + 6																

Figure 4-13. Transmit Buffer Descriptor

Offsets + 0 and + 2 are accessed by half-word access, and offsets + 4 and +6 are accessed through word access. The data length is not allowed to be zero.

Table 4-5 shows the transmit buffer descriptor (Tx BD) bit functions.

Table 4-5. Transmit Buffer Descriptor Bit Functions

Bit	Name	Description
0	R	Ready. 0 The buffer is not ready for transmission. Both the buffer and the BD can be updated. The SCU clears R after the buffer is sent or an error is encountered. 1 The buffer has not been sent or is being sent and the BD cannot be updated.
1	X	External buffer. 0 This TxBD access the internal SRAM 1 This TxBD access the external memory

Table 4-5. Transmit Buffer Descriptor Bit Functions (Continued)

Bit	Name	Description
2	W	Wrap (last BD in TxBD table). 0 Not the last BD in the table. 1 Last BD in the BD table. After this buffer is used, the SCU sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined by TxBD[W] and the space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is processed. 1 ER[TXB] or ER[TXE] is set when this buffer is processed, causing interrupts if not masked.
4	L	Last. 0 Not the last buffer in the frame 1 Last buffer in the frame
5	TC	Tx CRC valid only when TxBD[L]=1. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	CM	Continuous mode. 0 Normal operation. 1 The SCU does not clear TxBD[R] after this BD is closed allowing the buffer to be resent the next time the SCU accesses this BD. However, TxBD[R] is cleared if an error occurs during transmission, regardless of CM.
7-13	—	Reserved
14	UN	Underrun. Set after the HDLC sends a buffer and a transmitter underrun occurred.
15	BE	Bus transfer error. Set When a bus transfer error occurs.

4.4.4 Receiving Using RxBDs

Figure 4-14 shows an example of how RxBDs are used in receiving.

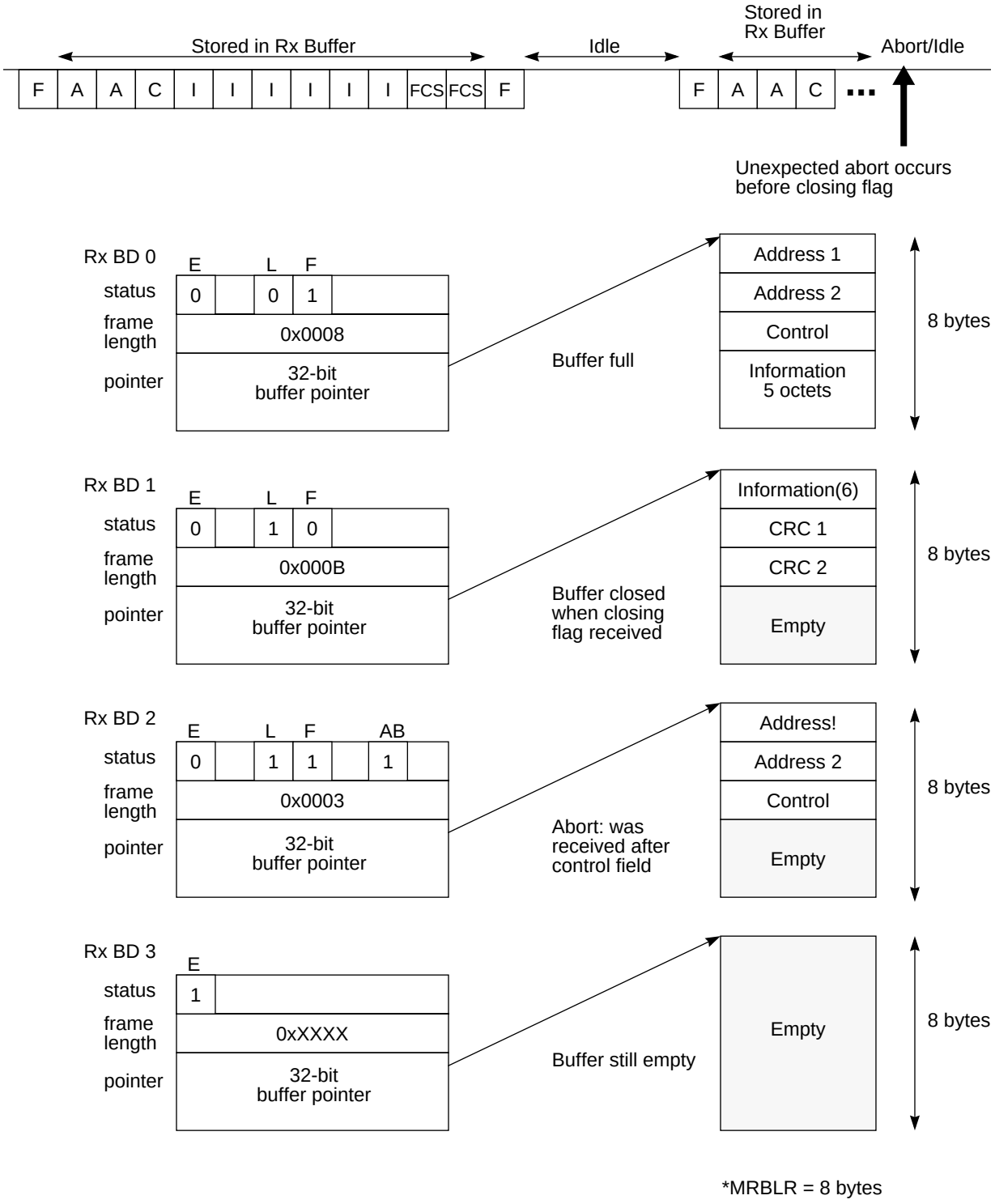


Figure 4-14. HDLC Receiving Using RxBDs

4.4.5 Transmitting Using TxBDs

Figure 4-15 shows an example of how TxBDs are used in transmitting.

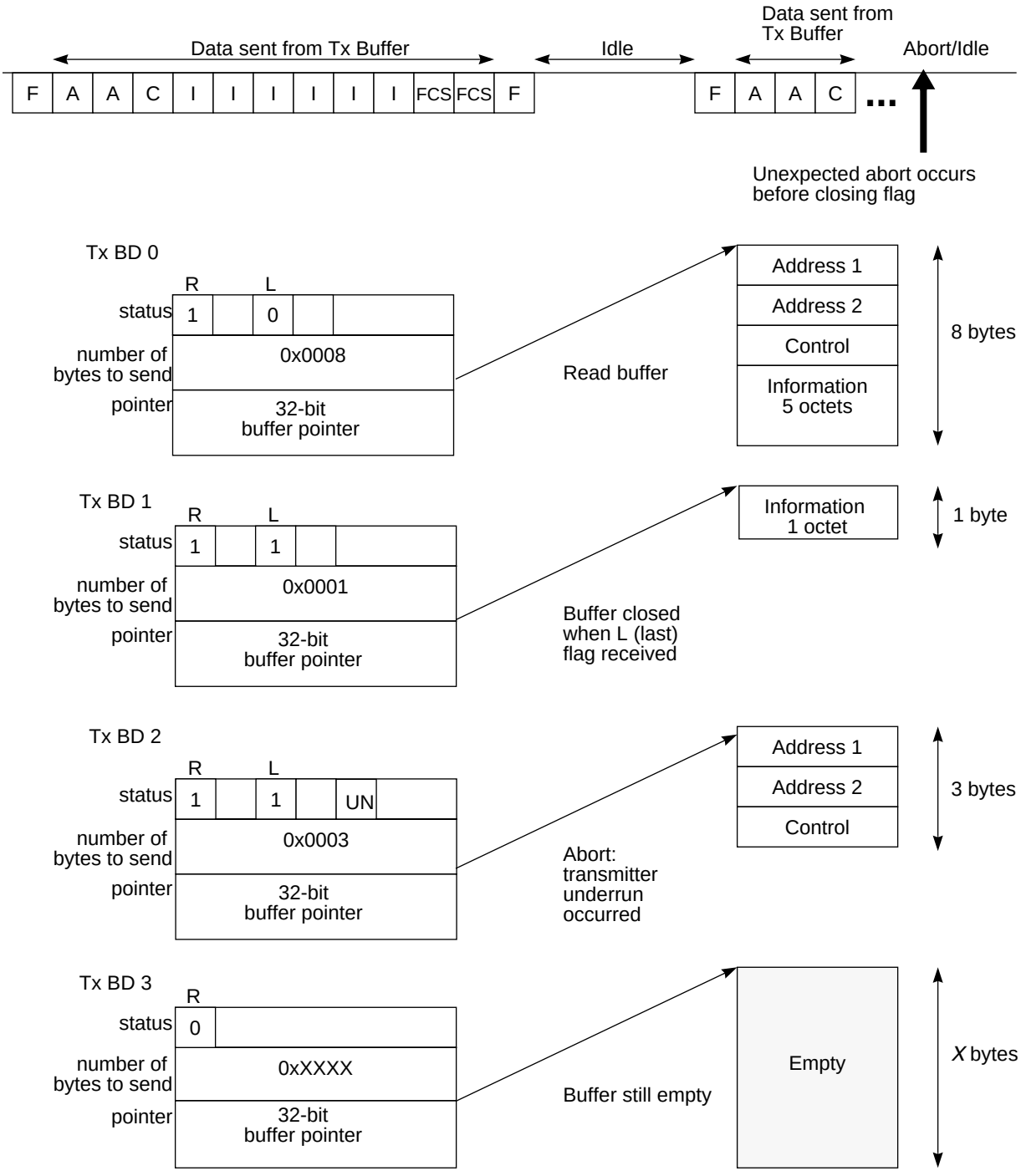


Figure 4-15. HDLC Transmitting Using TxBDs

4.4.6 HDLC Frame Transmission

The HDLC transmitter is designed to work with little or no core intervention. Once enabled by the core, a transmitter starts sending flags or idles as programmed in the HDLC mode register. The HDLC polls the first BD in the TxBD table. When there is a frame to transmit, the HDLC fetches the data from memory and starts sending the frame after sending the minimum number of flags specified between frames. When the end of the current buffer is reached and TxBD[L] (the last buffer in the frame) is set, the CRC and the closing flag are appended. The lsb of each octet and the msb of the CRC are sent first.

After a closing flag is sent, the HDLC updates the frame status bits of the BD and clears TxBD[R] (buffer ready). At the end of the current buffer, if TxBD[L] is not set (multiple buffers per frame), only TxBD[R] is cleared. Before the HDLC proceeds to the next TxBD in the table, an interrupt can be issued if TxBD[I] is set. This interrupt programmability allows the core to intervene after each buffer, after a specific buffer, or after each frame.

4.4.7 HDLC Frame Reception

The HDLC receiver is designed to work with little or no core intervention to perform address recognition, CRC checking, and maximum frame length checking. Received frames can be used to implement any HDLC-based protocol. Once enabled by the core, the receiver waits for an opening flag character. When it detects the first byte of the frame, the HDLC compares the frame address with four user-programmable, 16-bit address registers and an address mask. The HDLC compares the received address field with the user-defined values after masking with the address mask. To detect broadcast (all ones) address frames, one address register must be written with all ones.

If an address match is detected, the HDLC fetches the next BD and starts transferring the incoming frame to the buffer if it is empty. When the buffer is full, the HDLC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. If the incoming frame is larger than the current buffer, the HDLC continues receiving using the next BD in the table. During reception, the HDLC checks for frames that are too long (using the max frame length register, MFLR). When the frame ends, the CRC field is checked against the recalculated value and written to the buffer.

The data length of the last RxBD in the HDLC frame contains the entire frame length. This also enables software to identify the frames in which maximum frame length violations occur. The HDLC sets RxBD[L] (last buffer in frame), writes the frame status bits, and clears RxBD[E]. It then generates a maskable event (ER[RXF]; see Table 4-17) to indicate a frame was received. The HDLC then waits for a new frame.

Back-to-back frames can be received with only one shared flag between frames. The received frames threshold parameter (RFTH) can be used to postpone interrupts until a specified number of frames is received. This function can be combined with a timer to implement a time-out if fewer than the specified number of threshold frames is received.

HDLC mode, or any other synchronous mode, must receive a minimum of eight clocks after the last bit arrives to account for Rx FIFO delay.

4.5 HDLC-Specific Parameter RAM

The SCU maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the HDLC channels. Each channel has 128 bytes of parameter RAM space whose starting address is an offset of the HDLC base offset address.

The parameter RAM includes features such as a maximum receive buffer length, CRC preset and mask, protocol conditions, and so forth, that are common to all HDLC protocols. Table 4-6 shows these common parameters.

Some parameter RAM values must be initialized before the HDLC can be enabled. Other values are initialized or written by the HDLC. Once initialized, most parameter RAM values do not need to be accessed because most activity centers around the descriptors rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled.

NOTE:

Boldfaced entries in the parameter RAM table must be initialized by the user.

Table 4-6. Parameter RAM Map

Offset[1]	RAM Area	Width	Description
0x00	**MRBLR**	Half-word	Maximum receive buffer length. Defines the maximum number of bytes the MPU writes to a receive buffer before it goes to the next buffer. The MPU can write fewer bytes than MRBLR if a condition such as an error or end-of-frame occurs. It never writes more bytes than the MRBLR value. Therefore, user-supplied buffers should be no smaller than MRBLR. MRBLR should be greater than 8 for all modes. It should be a multiple of 8 for HDLC modes, and in totally transparent mode unless the Rx FIFO is 8-bits wide. Note that although MRBLR is not intended to be changed while the HDLC is operating, it can be changed dynamically. Changing MRBLR has no immediate effect. To guarantee the exact Rx BD on which the change occurs, change MRBLR only while the receiver is disabled. Transmit buffer length is programmed in TxBD [Data Length] and is not affected by MRBLR.
0x02 -0x17	—		Reserved
0x18	**C_MASK**	Word	CRC mask. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8; for 32-bit CRC-CCITT, initialize with 0xDEBB_20E3.
0x20	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF; for 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF.

Table 4-6. Parameter RAM Map (Continued)

Offset[1]	RAM Area	Width	Description
0x28	**MFLR**	Half-word	Max frame length register. The HDLC compares the incoming HDLC frame's length with the user-defined limit in MFLR. If the limit is exceeded, the rest of the frame is discarded and RxBD[LG] is set in the last BD of that frame. At the end of the frame the HDLC reports frame status and frame length in the last RxBD. The MFLR is defined as all in-frame bytes between the opening and closing flags.
0x30	**HMASK**	Half-word	Mask register (HMASK) and four address registers (HADDR*n*) for address recognition. The HDLC reads the frame address from the HDLC receiver, compares it with the HADDRs, and masks the result with HMASK. Setting an HMASK bit enables the corresponding comparison bit, clearing a bit masks it. When a match occurs, the frame address and data are written to the buffers. When no match occurs and a frame is error-free, the nonmatching address received counter is incremented. The eight low-order bits of HADDRn should contain the first address byte after the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDRn should contain 0xAA68 and HMASK should contain 0xFFFF. For 8-bit addresses, clear the eight high-order HMASK bits.
0x38	**HADDR1**	Half-word	
0x40	**HADDR2**	Half-word	
0x48	**HADDR3**	Half-word	
0x50	**HADDR4**	Half-word	
0x58	**MRA**	Word	Mode register
0x60	DSR	Half-word	Data sync register
0x64	ER	Half-word	Event register
0x68	MR	Half-word	Mask register
0x6c	SR	Byte	Status register
0x70	**BRG**	Half-word	Baud rate generator configuration register
0x74	RBASE	Word	RBASE Table register
0x78	CMR	Half-word	HDLC command register
0x7c	TBASE	Word	TBASE table register

1 Offset from HDLC Base (IMMR + 0x64000 + HDLC base offset address)

4.5.1 HDLC Base Offset Address

This section describes the address offsets for the 40 HDLC channels. The HDLC base offset addresses are offsets of IMMR + 0x64000.

The MC92460 supports HDLC mode only.

Table 4-7. HDLC Base Offset Addresses

Page	Offset Address[1]	Peripheral	Size (bytes)
0	0x000	HDLC0	128
1	0x080	HDLC1	128
2	0x100	HDLC2	128
3	0x180	HDLC3	128
4	0x200	HDLC4	128

Page	Offset Address[1]	Peripheral	Size (bytes)
20	0xa00	HDLC20	128
21	0xa80	HDLC21	128
22	0xb00	HDLC22	128
23	0xb80	HDLC23	128
24	0xc00	HDLC24	128

Table 4-7. HDLC Base Offset Addresses (Continued)

Page	Offset Address[1]	Peripheral	Size (bytes)
5	0x280	HDLC5	128
6	0x300	HDLC6	128
7	0x380	HDLC7	128
8	0x400	HDLC8	128
9	0x480	HDLC9	128
10	0x500	HDLC10	128
11	0x580	HDLC11	128
12	0x600	HDLC12	128
13	0x680	HDLC13	128
14	0x700	HDLC14	128
15	0x780	HDLC15	128
16	0x800	HDLC16	128
17	0x880	HDLC17	128
18	0x900	HDLC18	128
19	0x980	HDLC19	128

Page	Offset Address[1]	Peripheral	Size (bytes)
25	0xc80	HDLC25	128
26	0xd00	HDLC26	128
27	0xd80	HDLC27	128
28	0xe00	HDLC28	128
29	0xe80	HDLC29	128
30	0xf00	HDLC30	128
31	0xf80	HDLC31	128
32	0x1000	HDLC32	128
33	0x1080	HDLC33	128
34	0x1100	HDLC34	128
35	0x1180	HDLC35	128
36	0x1200	HDLC36	128
37	0x1280	HDLC37	128
38	0x1300	HDLC38	128
39	0x1380	HDLC39	128

1 Offset from IMMR + 0x64000

4.6 Error Handling by the HDLC Controller

The HDLC controller reports frame reception and transmission errors using BDs and the event register (ER). The transmission error is shown in Table 4-8.

Table 4-8. Transmit Errors

Error	Description
Transmitter Underrun	The channel stops transmitting, closes the buffer, sets TxBD[UN], and generates a TXE interrupt if not masked.

Receiving errors are shown in Table 4-9.

Table 4-9. Receive Errors

Error	Description
Overrun	Each HDLC maintains an internal FIFO for receiving data. When an Rx FIFO overrun occurs, the previous byte is overwritten by the next byte. The previous data byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates an RXF interrupt if not masked. The receiver then enters hunt mode. Even if an overrun occurs during a frame whose address is not recognized, an RxBD with data length 2 is opened to report the overrun and the interrupt is generated.

Table 4-9. Receive Errors (Continued)

Error	Description
Abort Sequence	Occurs when seven or more consecutive ones are received. When this occurs while receiving a frame, the channel closes the buffer, sets RxBD[AB] and generates a maskable RXF interrupt. The CRC and non connected error status conditions are not checked on aborted frames. The receiver then enters hunt mode.
CRC	The channel writes the received CRC to the buffer, closes the buffer, sets RxBD[CR], generates a maskable RXF interrupt. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.
Nonoctet-Aligned Frame	The channel writes the received data to the buffer, closes the buffer, sets RxBD[NO], and generates a maskable RXF interrupt. CRC error status should be disregarded on nonoctet frames. An immediate back-to-back frame is still received. The nonoctet data may be derived from the last word in the buffer as follows: msb — lsb — | 1 | 0 0 Valid Data | Nonvalid Data

4.6.1 HDLC Mode Register (MRA)

Each HDLC channel has the protocol-specific mode register (MRA), shown in Table 4-10. No bits (except ENT and ENR) can be modified while the receiver and transmitter are enabled.

Table 4-10. Mode Register (MRA)

	0–3	4	5	6	7	8	9	10–12	13	14	15
Field	NOF	CRC		—	FLG	—	DRT	—	DIAG0	DIAG1	DIAG2
Reset	0000_0000_0000_0000										
R/W	R/W			R	R/W	R	R/W	R	R/W		

	16	17	18	19–21	22	23	24–31
Field	ENT	—	ENR	—	IRCK	ITCK	RFTH
Reset	0000_0000						0000_1000
R/W	R/W	R	R/W	R	R/W		

Table 4-11 shows the descriptions of each mode register field.

Table 4-11. Mode Register Description

Bit	Field	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames. If NOF = 0b0000, no flags are inserted between frames and the closing flag of one frame is followed by the opening flag of the next frame in the case of back-to-back frames. NOF can be modified on the fly.
4–5	CRC	00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1. This field defines the protocol format. 01 Reserved. 11 Reserved. 10 32-bit CCITT-CRC (Ethernet and HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8+ X7 + X5 + X4 + X2 + X1 +1
6	—	Reserved.
7	FLG	Send flag or IDLE during frame 0 Send IDLE 1 Send flag
8	—	Reserved.
9	DRT	Disable receiver while transmitting. 0 Normal operation. 1 As the HDLC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$. This helps if the HDLC channel is on a multidrop line and the HDLC does not need to receive its own transmission.
10–12	—	Reserved.
13–15	DIAG[0–2]	Diagnosis mode 000 Normal operation 001 Loop back mode TxSD high 010 Auto Echo mode TxSD through 011 Reserved 100 Normal 101 Loop back mode TxSD data out 110 Auto Echo mode re-synchronize TxSD with RxCK 111 Reserved
16	ENT	Enable transmitter 0 Disable 1 Enable
17	—	Reserved.
18	ENR	Enable receiver 0 Disable 1 Enable
19–21	—	Reserved.
22	IRCK	Invert RxCK 0 Not invert RxCK 1 Invert RxCK
23	ITCK	Invert TxCK 0 Not invert TxCK 1 Invert TxCK
24–31	RFTH	Receive frame threshold parameter Number of receive frames that generate RXF interrupt

4.6.2 HDLC Data Synchronization Register (DSR)

Each HDLC channel has a data synchronization register (DSR) that specifies the pattern used for frame synchronization. The pattern that each SYN1 and SYN2 flag specifies is 0x7E and, because this is the default sequence, it does not need to be written.

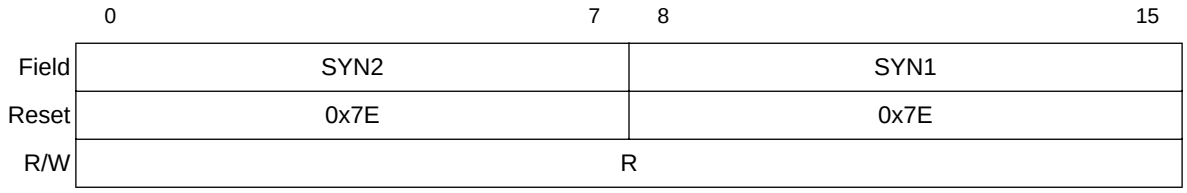


Figure 4-16. Data Synchronization Register (DSR)

Table 4-12. Data Synchronization Register Description

Bit	Field	Description
0-7	SYN2	Defaults and fixes to 0x7E for HDLC flag sequence pattern
8-15	SYN1	Defaults and fixes to 0x7E for HDLC flag sequence pattern

4.6.3 HDLC Event Register/Mask Register (ER)/MR)

The HDLC event register (ER) is used to report events recognized by the HDLC channel and to generate interrupts. When an event is recognized, the HDLC controller sets the corresponding ER bit. Interrupts generated through ER can be masked in the HDLC mask register (MR) which has the same bit format as the ER. Setting an MR bit enables the corresponding interrupt; clearing a bit masks it. ER bits are cleared by writing ones; writing zeros has no effect. All unmasked bits must be cleared before the HDLC controller clears the internal interrupt request. Table 4-17 shows the ER/MR register for HDLC operation.

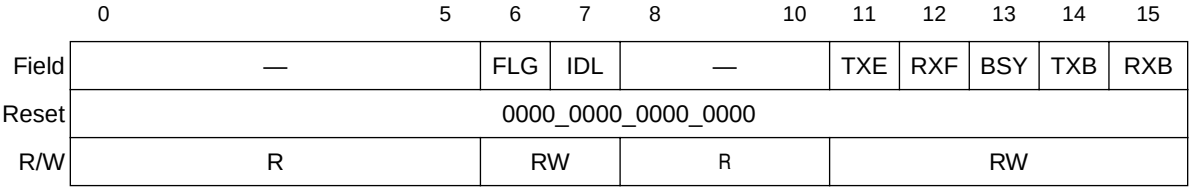


Figure 4-17. Event/Mask Register (ER/MR)

Table 4-13. ER/MR Register Description

Bit	Field	Description
0–5	—	Reserved.
6	FLG	Flag status. Set when the HDLC channel stops or starts receiving HDLC flags. Real-time status can be read in SR[FG].
7	IDL	Idle sequence status changed. Set when HDLC line status changes. Real-time status of the line can be read in SR[ID].
8–10	—	Reserved.

Table 4-13. ER/MR Register Description (Continued)

Bit	Field	Description
11	TXE	Tx error. Indicates an error has occurred on the transmitter channel.
12	RXF	Rx frame. Set when the number of receive frames specified in RFTH are received on the HDLC channel. It is set no sooner than two clocks after the last bit of the closing flag is received. This event is not maskable via the RxBD[I] bit.
13	BSY	Busy condition. Indicates a frame arrived but was discarded due to a lack of buffers.
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. TXB is set when a buffer is sent on the HDLC channel. For the last buffer in the frame, TXB is not set before the last bit of the closing flag begins its transmission.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when the HDLC channel receives a buffer(MRBLR) that is not the last in a frame.

4.6.4 HDLC Status Register (SR)

The HDLC status register (SR), shown in Table 4-18, permits monitoring of real-time status conditions on RxD.

	0 3	4	5	6	7
Field	—	BSYSR	FG	—	ID
Reset	0000_0000				
R/W	R	RW		R	RW

Figure 4-18. HDLC Status Register (SR)

Table 4-14. SR Field Descriptions

Bit	Field	Description
0–3	—	Reserved.
4	BSYSR	Busy condition. This bit is set when a frame is received.
5	FG	Flags. The line is checked after the data has been decoded. 0 HDLC flags are not being received. The most recently received 8 bits are examined every bit time to see if a flag is present. 1 HDLC flags are being received. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once it is set, FG remains set at least eight bit times and the next eight received bits are examined. If another flag occurs, FG stays set for at least another eight bits. If not, it is cleared and the search begins again.
6	—	Reserved.
7	ID	Idle status. 0 The line is busy. 1 Set when RXD is a logic 1 (idle) for 15 or more consecutive bit times. It is cleared after a single logic 0 is received.

4.7 HDLC Commands

The transmit and receive commands are issued to the command register (CMR).

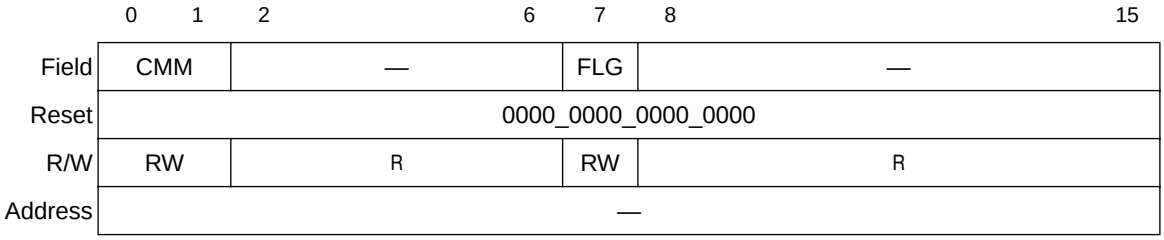


Figure 4-19. HDLC Command Register (CMR)

Table 4-15. HDLC Command Register Field Descriptions

Bit	Field	Description
0–1	CMM	CMM = 00 (Stop Transmit). After a hardware or software reset and a channel is enabled, the transmitter starts polling the first BD and begins sending data if TxBD[R] is set. If the HDLC receives the STOP TRANSMIT command while not transmitting, the transmitter stops polling the BDs. If the HDLC receives the command during transmission, transmission is aborted after a maximum of 64 additional bits, the Tx FIFO is flushed, and the current BD pointer is not advanced (no new BD is accessed). The transmitter then sends an abort sequence (0x7F) and stops polling the BDs. When not transmitting, the channel sends flags or idles as programmed in the MRA. CMM = 01 (Restart Transmit) Enables frames to be sent on the transmit channel. The HDLC controller expects this command after a STOP TRANSMIT is issued. The transmitter resumes from the current BD. CMM = 10 (Enter Hunt Mode). After a hardware or software reset, once an HDLC is enabled, the receiver is automatically enabled and uses the first BD in the RxBD table. While the HDLC is looking for the beginning of a frame, that HDLC is in hunt mode. The ENTER HUNT MODE command is used to force the HDLC receiver to stop receiving the current frame and enter hunt mode, in which the HDLC continually scans the input data stream for a g sequence. After receiving the command, the buffer is closed and the CRC is reset. Further frame reception uses the next BD CMM = 11 Received
2–6	—	Reserved.
7	FLG	Set by the MPU and cleared by the HDLC 0 The HDLC is ready to receive a new command 1 The HDLC is currently processing. The HDLA clear this bit at the end of a command.
8–15	—	Reserved.

4.8 Baud Rate Generator

The MC92460 contains an independent baud rate generator (BRG) for each of the 40 HDLCs. Inputs to the BRGs come from the PLLs. The clocks produced by the BRGs are sent to each HDLC. In addition, the output of a BRG can be routed to TxCK*n* and RxCK*n* to be used externally. The following is a list of the main features of the MC92460 baud rate generators:

- 40 independent and identical baud rate generators
- On-the-fly changes allowed
- Input is from the PLLs
- A 16x divider option allows slow baud rates at high system frequencies
- BRGO output can be routed to a pin (TxCK*n* and RxCK*n*)

Figure 4-20 shows a baud rate generator.

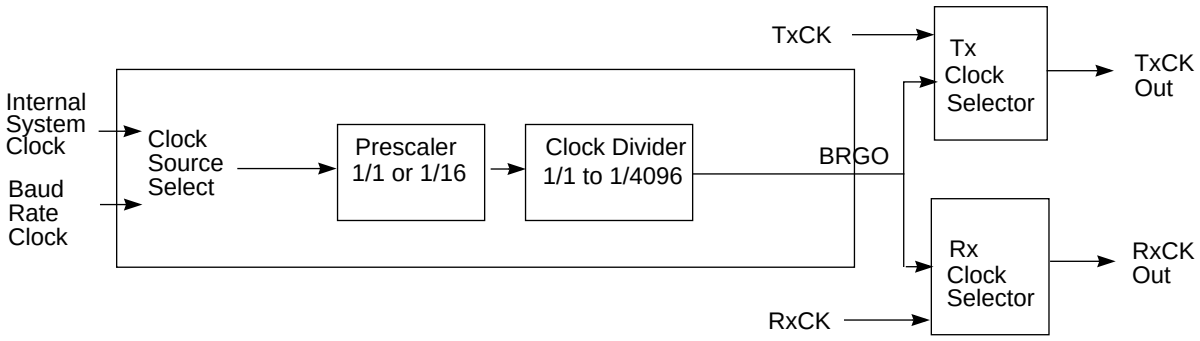


Figure 4-20. Baud Rate Generator Block Diagram

Whether the internal system clock or the baud rate clock is used as the BRG clock source depends on BRGC*x* [EXTC]). The internal system clock is generated in PLL1 and the baud rate clock is generated by PLL2 as source clocks especially for the BRGs, see Chapter 7, “Phase-Locked Loop.” These source options allow flexible baud rate frequency generation, independent of the system frequency. The clock signals are not synchronized internally before being used by the BRG.

The BRG provides a divide-by-16 prescaler option (BRGC*x* [DIV16]) and a 12-bit clock divider (BRGC*x*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on the fly; however, two changes should not occur within two clock source periods. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the falling edge of the source clock. If the divide factor is an odd value, the transitions alternate between the falling and rising edges of the source clock.

The baud rate that is generated by PLL2 may be passed through an HDLC baud rate generator with no change from its entry rate if zero values are selected for the prescaler and the clock divider. Table 4-17 shows the various baud rates that can be generated from a 66 MHz baud rate clock input given specific DIV16 and CD values. If the baud rate clock input

is other than 66 MHz, however, other baud rates than those shown in the table will be generated from the BRGs.

4.8.1 BRG Configuration Registers (BRGC*x*)

The baud rate generator configuration registers (BRGC*x*) are shown in Table 4-21. A reset disables the BRG and drives the baud rate generator output (BRGO) clock high. The BRGC can be written at any time with no need to disable the HDLCs or external devices that are connected to the BRGO. Configuration changes occur at the end of the next BRG clock cycle (no spikes occur on the BRGO output clock). The BRGC can be changed on the fly; however, two changes should not occur within two clock source periods.

	0	1	2	3	4 ... 15
Field	EXTC	TCS	RCS	DIV16	CD
Reset	0	1	1	0	0000_0000_0000
R/W	RW				

Figure 4-21. Baud Rate Generator Configuration Register (BRGC*x*)

Table 4-16. BRGC*x* Register Description

Bit	Field	Description
0	EXTC	External clock source. Selects the BRG input clock. 0 The BRG input clock comes from the internal system clock (may be 66MHz) 1 The BRG input clock comes from the baud rate clock
1	TCS	Select transmitter clock source 0 Internal baud rate generator output 1 TxCK*n*
2	RCS	Select receiver clock source 0 Internal baud rate generator output 1 RxCK*n*
3	DIV16	Prescaler: selects a divide-by-1 or divide-by-16 prescaler before reaching the clock divider. 0 Divide by 1 1 Divide by 16
4–15	CD	Clock divider. CD presets an internal 12-bit counter that is decremented at the DIV16 output rate. When the counter reaches zero, it is reloaded with CD. If CD = 0xFFF the minimum clock rate for BGRO (divide by 4096)is generated If CD = 0x000 the maximum rate (divide by 1) is generated.

4.8.2 Baud Rate Examples

This formula is used to calculate a baud rate for each channel:

Baud rate = main clock (internal system clock or baud rate clock) / (1 or 16 according to BRGC*x*[DIV16]) ∗ (BRGC*x*[CD] + 1)

Table 4-17 shows the baud rate generated from a 66-MHz main clock.

Table 4-17. Baud Rate

DIV16 (Prescaler)	CD (Clock Divider)	Baud Rate (Actual Frequency)
0	0	66 MHz
0	1	33 MHz
0	2	22 MHz
0	3	16.5 MHz
0	4095	16113.28 Hz
1	0	4.125 MHz
1	1	2.0625 MHz
1	2	1.375 MHz
1	3	1.03125 MHz
1	4095	1007.08 Hz

Chapter 5
Master Interface Unit

The master interface unit (MIU) supports the 60x bus. This module is connected through the 60x-compatible bus mode of the MPC8260. Key features of the MIU include the following:

- Direct connection with 64-bit data and 32-bit address 60x bus
- Virtual DMA supports an 80 channel HDLC master (40 Tx, 40 Rx)
- Virtual DMA functionality executing off-chip memory to internal memory
- Supports single-beat and burst access
- Either the MC92460 master or the MC92460 slave can be the 60x bus master
- Supports 66 MHz 60x bus speed

Figure 5-1 shows the MIU block diagram.

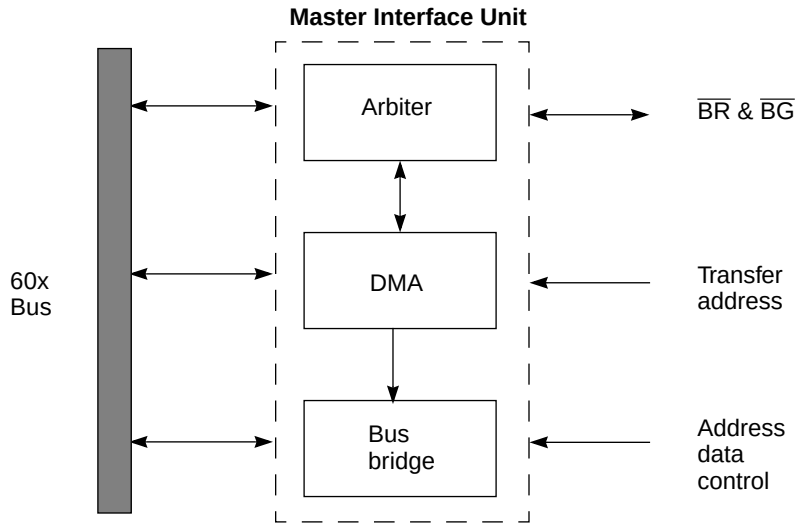


Figure 5-1. Master Interface Unit Block Diagram

5.1 System Mode

The system mode is implemented by the system mode register (SMR), shown in Figure 5-2. This register defines the 60x bus mode and the internal RAM wait cycle.

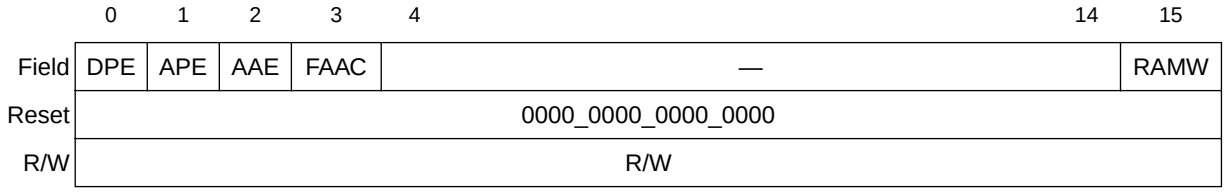


Figure 5-2. System Mode Register (SMR)

Table 5-1. SMR Register Descriptions

Bits	Field	Description
0	DPE	Data parity error signal control 0 Data parity error signal output disable 1 Data parity error signal output enable
1	APE	Address parity error signal control 0 Address parity error signal output disable 1 Address parity error signal output enable
2	AAE	Address ACK enable at address only access 0 $\overline{\text{AACK}}$ is asserted at address only access on Master mode 1 $\overline{\text{AACK}}$ is NOT asserted at address only access on Master mode
3	FAAC	$\overline{\text{AACK}}$ select 0 MPC8260 pipeline-1 mode 1 Fast AACK mode; support 60x bus fastest AACK
4–14	—	Reserved
15	RAMW	Internal RAM wait cycle select 0 0x0 to 0x63fff is two wait access 1 0x0 to 0x63fff is one wait access

5.2 Direct Memory Access (DMA)

The modules that are required to perform DMA transfers arbitrate for the system master bus (external 60x bus). These DMA modules are referred to as master modules. If such a module also has registers accessible to the processor core, then the module also will connect to the system slave bus. The detailed specification of the master bus is dependent on the type of processor core being used.

The DMA has two modes, external and direct. These are selected by DMAMR[EXT]. Ordinarily the external mode should be selected, because this mode generates and enables 80 DMA channels of HDLC. When the MC92460 has no need to perform HDLC data transfers, the DMA module can be used as a generic DMA module by selecting direct mode.

A user can transfer data among all the memories on the 60x bus, among all of the MC92460's memories, or between the memories on the 60x bus and the MC92460. Direct mode supports only data that is aligned on 8 byte boundaries and of a minimum 8 byte data size. Thus, values written to the lower 3 bits of DMASA, DMADA, and DMATS are ignored and have no meaning.

5.2.1 Source Address (DMASA)

The DMA source address register (DMASA) holds the starting address of internal memory source data. The DMA reads the data from the internal memory source address and then transfers the data to the destination address.

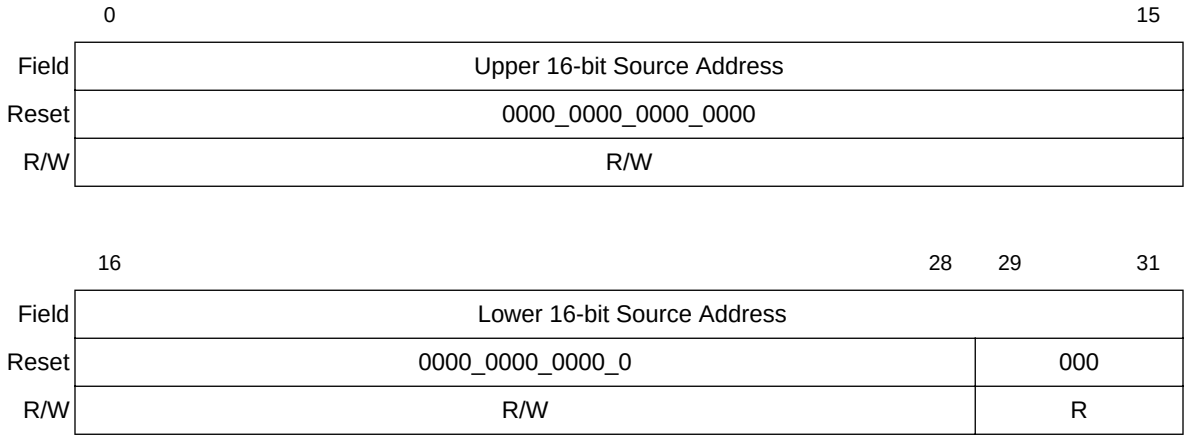


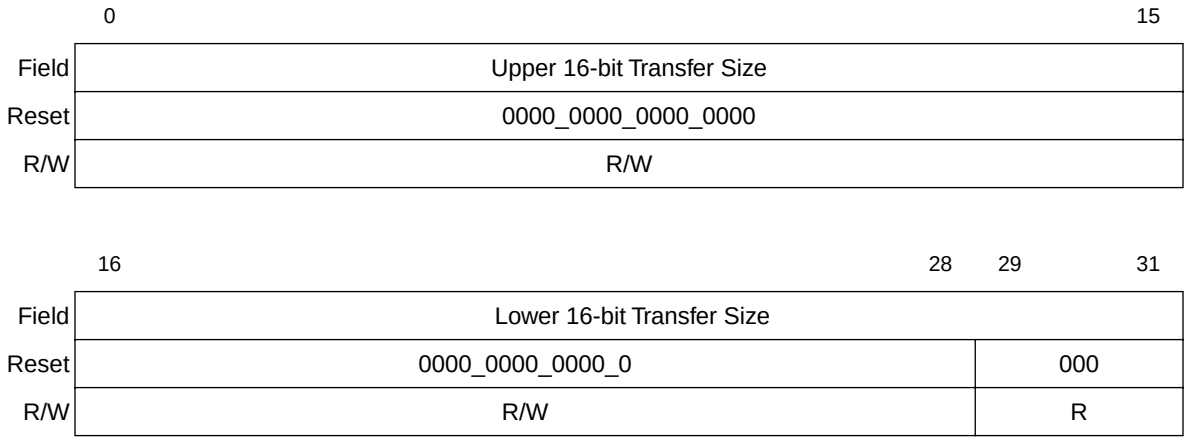
Figure 5-3. DMA Source Address Register (DMASA)

Table 5-2. DMASA Register Description

Field	Description
Source Address	DMA transfer source address. This value is used in DIRECT MODE only

5.2.2 Transfer Size (DMATS)

The DMA transfer size register (DMATS) determines the maximum allowed byte size of reads and writes to the source and destination registers.



	0	15
Field	Upper 16-bit Transfer Size	
Reset	0000_0000_0000_0000	
R/W	R/W	

	16	28	29	31
Field	Lower 16-bit Transfer Size			
Reset	0000_0000_0000_0		000	
R/W	R/W		R	

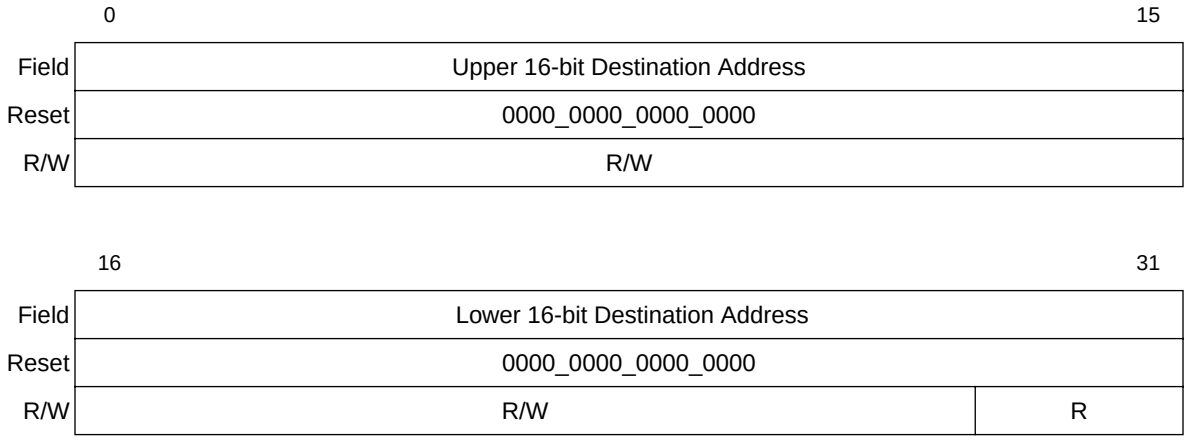
Figure 5-4. DMA Transfer Size Register (DMATS)

Table 5-3. DMATS Register Description

Field	Description
Transfer Size	DMA transfer byte count. This value is used in DIRECT MODE only. A data length of zero is not allowed.

5.2.3 Destination Address (DMADA)

The DMA destination address register (DMADA) indicates the start of the MC92460's internal memory destination address. The DMA reads the data from the internal memory source address and then transfers the data to the destination address.



	0	15
Field	Upper 16-bit Destination Address	
Reset	0000_0000_0000_0000	
R/W	R/W	

	16	31
Field	Lower 16-bit Destination Address	
Reset	0000_0000_0000_0000	
R/W	R/W	R

Figure 5-5. Destination Address Register (DMADA)

Table 5-4. DMADA Register Description

Field	Description
Destination Address	DMA transfer destination address. This value is used in DIRECT MODE only.

5.2.4 Mode Register (DMAMR)

The DMA mode register (DMAMR) defines the MC92460's DMA parameters; refer to Table 5-5. Parameters except STRT should be modified only while the EN bit is 0 (the DMA is disabled).

	0	1	2	3	4	5 7	8 10	11 13	14	15
Field	EN	EXT	EOTIE	ERIE	SNP	—	TSIZ	—	DAL	SAL
Reset	0100_0000_0000_0000									
R/W	R/W					R	R/W	R	R/W	

	16	17 31
Field	STRT	—
Reset	0000_0000_0000_0000	
R/W	R/W	R

Figure 5-6. DMA Mode Register (DMAMR)

Table 5-5. DMAMR Register Descriptions

Bits	Field	Description
0	EN	DMA enable/disable 0 Disable 1 Enable
1	EXT	External Mode 1 External mode. DMA uses the value that is provided by the transfer requesting device/module(I/O) through the interface signals. 0 Direct mode. DMA uses the source/destination registers, in which the transfer information is described, for the data transfer.
2	EOTIE	End-of-transfer interrupt enable/disable 0 No interrupt will be generated at the end-of-transfer. 1 If the current DMA transfer is finished an interrupt will be generated.
3	ERIE	Transfer error interrupt enable 0 If there is an error during a DMA transfer, an interrupt will be generated. 1 No interrupt will be generated when a transfer error occurs.
4	SNP	Transfer is snoopable. 0 Access is non-global. 1 Access is global.
5-7	—	Reserved.
8-10	TSIZ	Source transfer size. The value in this field is valid. These values must be set. 000 8-byte for the burst read/write 001 1-byte 010 2-byte 011 3-byte NOTE: MC92460 supports 000 (burst read/write) only
11-13	—	Reserved.

Table 5-5. DMAMR Register Descriptions (Continued)

Bits	Field	Description
14	DAL	Destination address location. 0 Destination address is located in the MC92460 internal SRAM memory space (0x0~0x7fff) 1 Destination address is located in the external memory space on 60x bus.
15	SAL	Source address location. 0 Source address is located in the MC92460 internal SRAM memory space (0x0~0x7fff) 1 Source address is located in the external memory space on 60x bus.
16	STRT	DMA Transfer start 1 When BUSY is negated and write 1 allows to start the DMA transfer. If BUSY is asserted and write 1, it will not affect to the current DMA process. 0 Always 0 when this bit is read
17–31	—	Reserved.

5.2.5 Status Register (DMASR)

The DMA status register reports the status of DMA transfer events.

	0	1	2	3	4	5	15
Field	BUSY	EOTI	ERI	DMATT		—	
Reset	0000_0000_0000_0000						
R/W	R	R/W	R/W	R		—	

Figure 5-7. DMA Status Register (DMASR)

Table 5-6. DMASR Field Descriptions

Bits	Field	Description
0	BUSY	DMA transfer busy 0 - No DMA transfer. 1 - DMA transfer is in progress.
1	EOTI	DMA end-of-transfer interrupt status 0 - No EOT interrupt occurs. 1 - EOT interrupt occurs. Writing 1 clears this bit.
2	ERI	DMA Transfer error interrupt status 0 - No Transfer Error interrupt occurs. 1 - Transfer Error interrupt occurs. If write 1, this bit is cleared.
3–4	DMATT	DMA transfer type status. This status field indicates the data transfer direction of the current DMA transfer. 00 - Memory to Memory transfer 01 - Memory to I/O transfer 10 - I/O to Memory transfer 11 - Reserved
5–15	—	Reserved

5.2.6 DMA Error Address Register (DMAEA)

The DMA error address register holds the system address accessed during a transfer error on the host bus.

	0	15
Field	EA High	
Reset	0000_0000_0000_0000	
R/W	R	

	16	31
Field	EA Low	
Reset	0000_0000_0000_0000	
R/W	R	

Figure 5-8. DMA Error Address Register (DMAEA)

Table 5-7. DMAEA Register Field Description

Bits	Field	Description
0–15	EA	DMA transfer error address. When transfer error occurs, the address value on the host bus will be saved in this field.

5.3 Arbiter

The system bus arbiter of the master interface unit (MIU) is responsible for allocating both the internal master bus and the 60x external bus whenever the MPU core (Power PC architecture core, MPC603e, or MPC8260) relinquishes bus control. This arbiter manages the internal master bus and the 60x external bus for the DMA module

The system bus arbiter supports one internal master device (the MIU) and one external device, another MC92460 for example, which can be either a master or a slave. Refer to Figure 5-9. The external master supported does not include the MPU core.

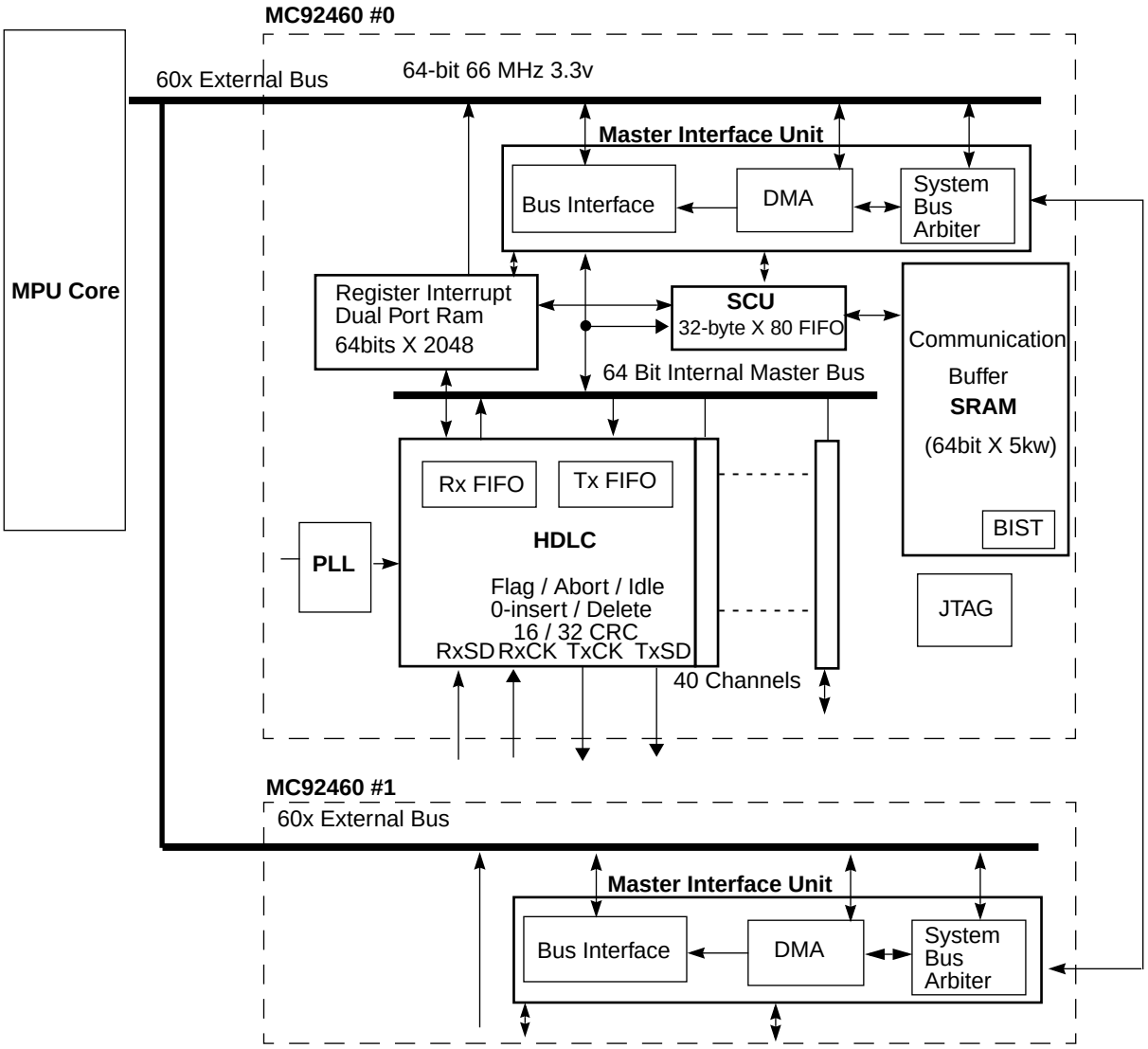


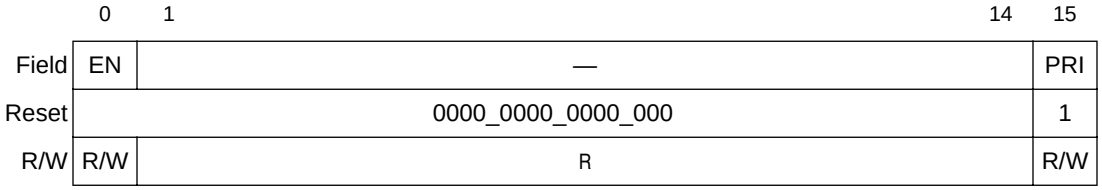
Figure 5-9. Arbitration Between Two MC92460s

The arbitration scheme enables DMA master modules to access other internal modules including the SCU (system integration module), chip selects, and so forth. Because most external master devices will not have all the required master bus signals available as external signals, they cannot arbitrate for the internal master bus directly; they can only arbitrate for the external bus.

When an external alternate master is granted control of the external bus, the external device drives its signals and the system bus arbiter drives the internal master bus signals. The arbiter controls priority automatically. The MPU is the highest priority after a system reset. But if the arbiter is enabled, the MPU will be the lowest priority.

5.3.1 Mode Register 0 (ARBM0)

The mode register 0 (ARBM0) defines the system bus arbiter parameters for a channel that connects to an MC92460 master.



	0	1	14	15
Field	EN	—		PRI
Reset	0000_0000_0000_000			1
R/W	R/W	R		R/W

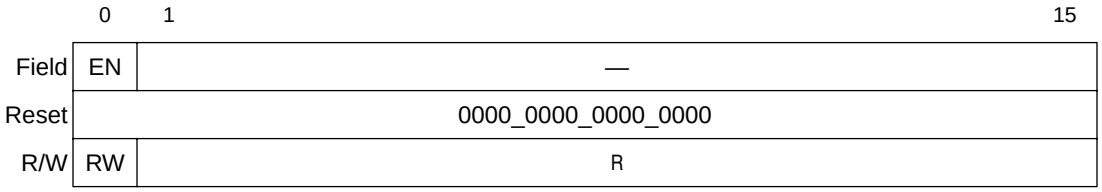
Figure 5-10. Mode Register 0 (ARBM0)

Table 5-8. ARBM0 Register Field Descriptions

Bits	Field	Description
0	EN	Arbiter for internal master Enable/Disable 0 Disable. 1 Enable.
1–14	—	Reserved.
15	PRI	Priority 0 Slave MC92460 is higher priority than Master MC92460 1 Master MC92460 is higher priority than Slave MC92460

5.3.2 Mode Register 1 (ARBM1)

The mode register 1 (ARBM1) defines the system bus arbiter parameters for a channel that connects to an MC92460 slave.



	0	1	15
Field	EN	—	
Reset	0000_0000_0000_0000		
R/W	RW	R	

Figure 5-11. Mode Register 1 (ARMB1)

Table 5-9. ARBM1 Register Field Description

Bit	Field	Description
0	EN	Arbiter for internal master Enable/Disable 0 Disable. 1 Enable.
1–15	—	Reserved.

5.4 Notes

- In an MPC8260 SDRAM system with an attached MC92460, if the arbiter is enabled (by pulling MODE low), the clocked mode of the arbiter cannot be used. Pulling AMODE high to enable clocking of the enabled arbiter is not supported.
- Assigning L2 cache space to the MC92460 address area is not supported.
- The AMODE of an MC92460 slave should be set to the same value as the AMODE of an MC92460 master.

Chapter 6
IEEE 1149.1 Test Access Port

The MC92460 provides a test access port (TAP) that is compatible with the IEEE 1149.1 standard test access port and boundary scan architecture. This implementation of IEEE 1149.1 allows boundary scan compatibility with other JTAG components in a circuit-board. A boundary scan description language (BSDL) file for this device is provided in another text file located on Motorola's MC92460 product web site.

The TAP includes five dedicated signal pins, a 16-state TAP controller, a bypass register, an instruction register, and a device identification (ID) register. The ID register contains the specific JTAG ID code. A boundary scan register links all device I/O signal pins into a shift-register chain around the periphery of the device. This path is provided with serial input and output signal pins and is controlled by appropriate clock and control signals.

The JTAG test logic is independent of the device system logic. The JTAG implementation on the MC92460 provides for the following capabilities:

- Boundary scan operations
- BYPASS mode, which bypasses the MC92460 boundary scan chain
- Sampling system data from I/O signals (for inputs) and core system data (for outputs) and shifting out the result through the boundary scan register
- Test of component interconnect with EXTEST mode. This instruction updates I/O and system logic with data that is shifted in serially. Output pins are driven with the updated values, and input pins have the updated values driven into the core logic.
- CLAMP mode, which drives as in EXTEST mode, but puts boundary scan chain in BYPASS mode
- IDCODE mode allows the ID code to be shifted from the ID register.
- HIGHZ mode tristates all system outputs and bi-directionals for component isolation.

6.1 Functional Blocks

The MC92460 implementation of the IEEE 1149.1 standard includes a TAP controller, an instruction register, a bypass register, an ID code register, and the I/O boundary scan register. The associated signal pins for the JTAG logic are:

- TCK—Test clock input for the JTAG test logic

- TMS—Test mode select input used to sequence the TAP controller's state machine, sampled on the rising edge of TCK
- TDI—Test data input sampled on the rising edge of TCK
- TDO—Test data output active during the Shift-DR and Shift-IR controller states
- $\overline{\text{TRST}}$—Asynchronous test reset signal that initializes the TAP controller and other JTAG logic, active low

A block diagram is shown in Figure 6-1.

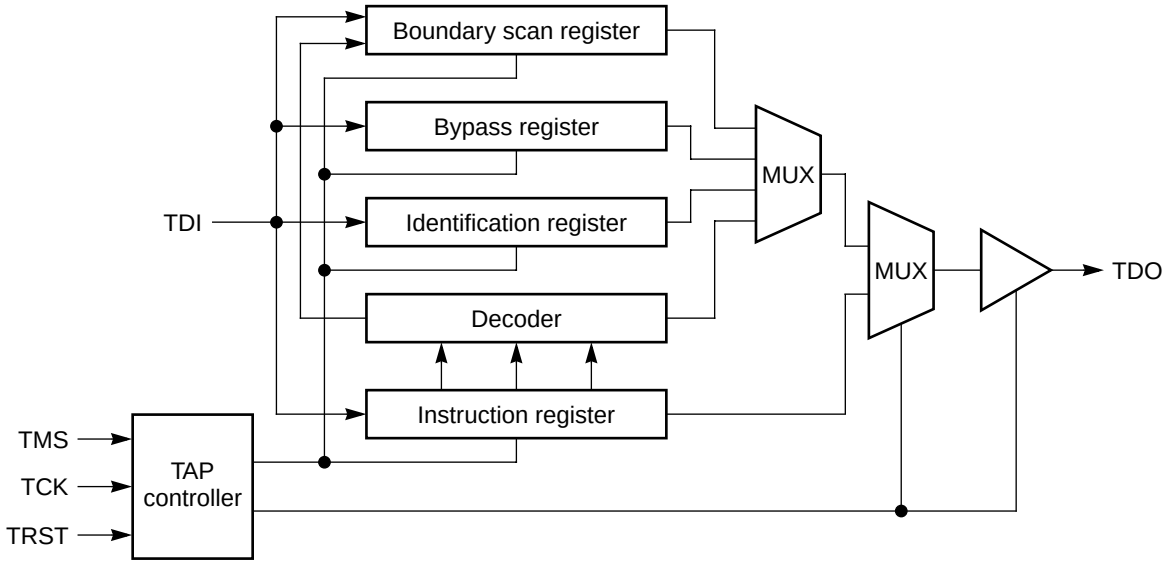


Figure 6-1. JTAG Logic Block Diagram

6.1.1 TAP Controller

The TAP controller is a synchronous, 16-state machine that controls and manages the mode of operation for the test circuitry. The controller interprets the sequence of logical values on the TMS signal. The TAP controller states are explained in the IEEE 1149.1 document. The state diagram for the controller is shown in Figure 6-2.

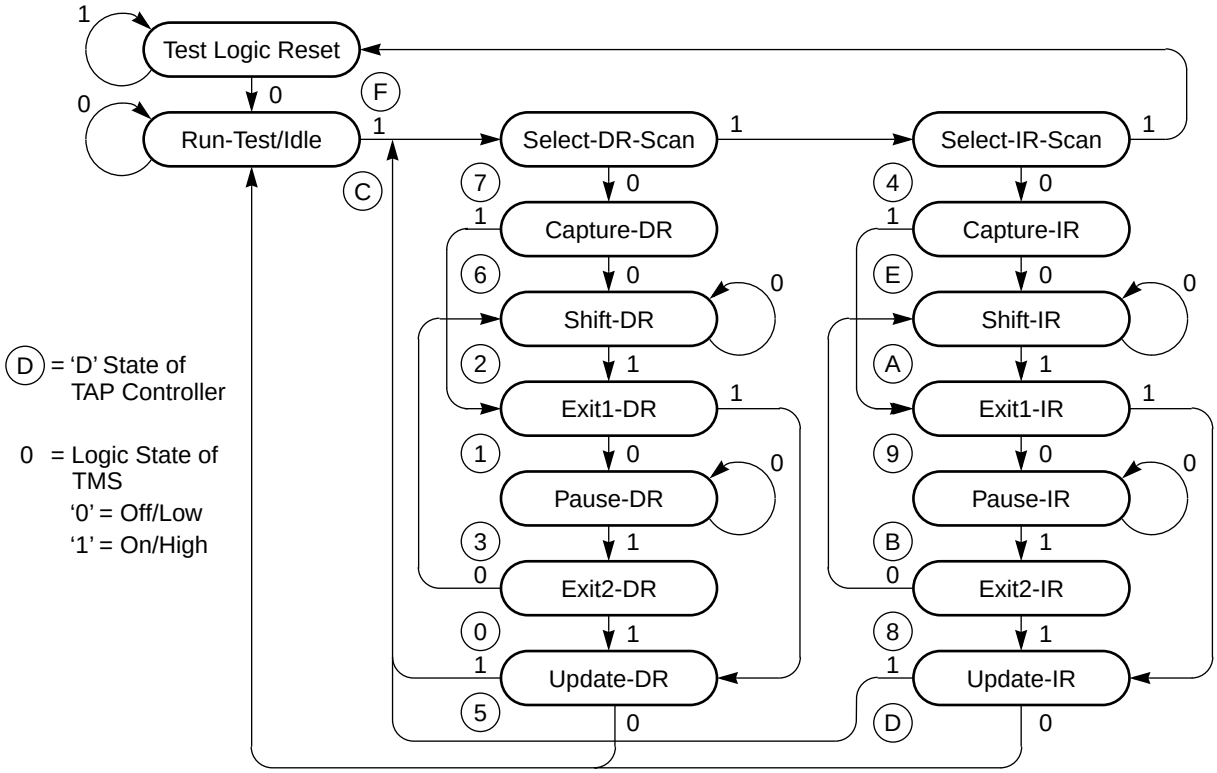


Figure 6-2. TAP Controller State Diagram

6.1.2 Instruction Register

The 4-bit instruction register holds various instructions that define which test registers are to be used and the serial test data register path between TDI and TDO signals. The instructions are described in Section 6.2 JTAG Instruction Support.

6.1.3 Device Identification Register

The device identification register is a 32-bit register that holds the manufacturer's identity code, sequence number (equivalent to a part number), and version code. The bit assignment for the ID code is shown in Table 6-1. The ID code data is shifted out with the least significant bit (0) shifted out first.

Table 6-1. Device Identification Register ID Codes

	ID CODE (32bits)
BIN	(MSB) 0000 0010 1000 0000 1101 0000 0001 1101(LSB)
HEX	0280D01D

6.1.4 Bypass Register

The 1-bit bypass register is connected between TDI and TDO when either the BYPASS instruction or the CLAMP instruction is active. This allows for bypassing the boundary scan register while other components on a board are being tested. When the bypass register is selected by the current instruction, the data that is shifted out on the first clock edge is the previously latched data.

6.1.5 Boundary Scan Register

The boundary scan register allows testing of circuitry external to the MC92460. It also allows the signals flowing through the system pins to sampled and examined without interfering with the operation of the MC92460. The boundary scan register includes all the functional pins of the MC92460 and additional stages that control the direction and driving state of some of the pins.

6.2 JTAG Instruction Support

The following instructions are supported by the MC92460:

- SAMPLE—The SAMPLE instruction captures the input or output data from the I/O pads. The data is captured on the rising edge of TCK when the TAP controller is in the capture-DR state, and can be observed when shifted out through the boundary scan register. This instruction does not update the core logic or the I/O pads. This makes it useful for preloading the boundary scan register with known data when entering the EXTEST instruction.
- BYPASS—The BYPASS instruction selects the bypass register for serial access between TDI and TDO. This instruction does not update the core logic or the I/O pads.
- EXTEST—The EXTEST instruction selects the boundary scan register. When the EXTEST instruction is active, the output pins and the internal core input signals are updated with test data that has been previously shifted into the boundary scan register. Also, the state of all input ports of the chip core are defined by the data shifted into the boundary scan register, and are updated on the falling edge of TCK in the update-DR controller state. Refer to the IEEE 1149.1 document for more details on the function and use of EXTEST.
- IDCODE—The MC92460 includes a device identification register. The IDCODE instruction selects only the device identification register to be connected for serial access between TDI and TDO in the shift-DR controller state. When the IDCODE instruction is selected, the vendor identification code is loaded into the ID register in the capture-DR state.
- CLAMP—The CLAMP instruction selects the bypass register for serial access between TDI and TDO. The CLAMP instruction drives the core data and the chip

I/O signals in the same manner as EXTEST, while the bypass register is selected as the serial TDI/TDO path.

- HIGHZ—The HIGHZ instruction selects the bypass register for serial access between TDI and TDO. This instruction also puts all system logic output pins (including bi-directional pins) in their inactive drive state. This instruction is provided to help isolate the component during circuit board testing.

Table 6-2 shows the coding of the instructions supported by the MC92460. Bit 0 is the first bit shifted in from TDI and the first bit shifted out through TDO. The shaded rows of the table indicate the code point that is defined in the BSDL file for each instruction. The other values are provided for completeness, but are not expected to be used.

Table 6-2. Instruction Decoding

Code				Instruction
Bit3	Bit2	Bit1	Bit0	
0	0	0	0	EXTEST
0	0	0	1	IDCODE
0	0	1	0	SAMPLE
0	0	1	1	RESERVED
0	1	0	0	RESERVED
0	1	0	1	RESERVED
0	1	1	0	RESERVED
0	1	1	1	PRIVATE
1	0	0	0	PRIVATE
1	0	0	1	HIGHZ
1	0	1	0	PRIVATE
1	0	1	1	PRIVATE
1	1	0	0	CLAMP
1	1	0	1	PRIVATE
1	1	1	0	PRIVATE
1	1	1	1	BYPASS

6.3 Boundary Scan Register Path

The MC92460 boundary scan register path is defined in another text file located on the Motorola product web site in Boundary Scan Description Language (BSDL) format.

Chapter 7
Phase-Locked Loop

The MC92460 has two phase-locked loops, PLL1 and PLL2. Phase-locked loop 1 (PLL1) generates all internal clocks for the MC92460. Using the SYSCLK external reference, PLL1 as seen in Figure 7-1 generates a master clock as a reference for a bus clock and an internal system clock that can be used by the baud rate generator in each HDLC controller.

Phase-locked loop 2 (PLL2) shown in Figure 7-2 receives the EXCLK to generate a baud rate clock that also can be used by the baud rate generator in each HDLC controller. See Chapter 4, “High-level Data Link Controller (HDLC)” for an explanation of the MC92460 baud rate generators.

7.1 PLL1

Phase-locked loop 1 divides SYSCLK to provide the system and bus clock for the MC92460. VCO1 test output is provided by PLL1O.

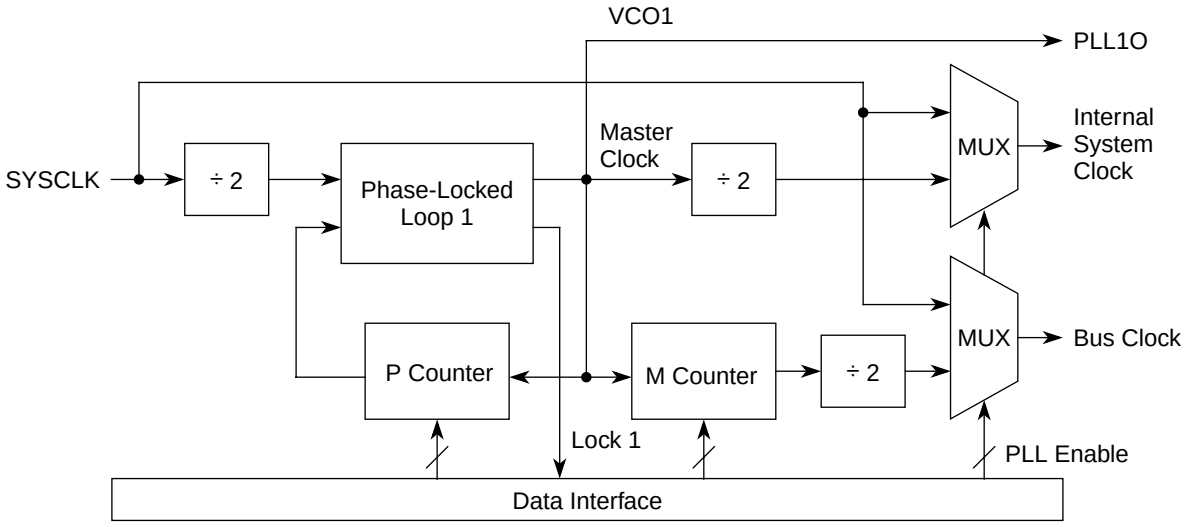


Figure 7-1. PLL1 Block Diagram

7.2 PLL2

The phase-locked loop 2 (PLL2) EXCLK is divided by the values loaded into the P and M counters from the PLL mode register to generate various frequencies for the baud rate clock. VCO2 test output is provided by PLL2O.

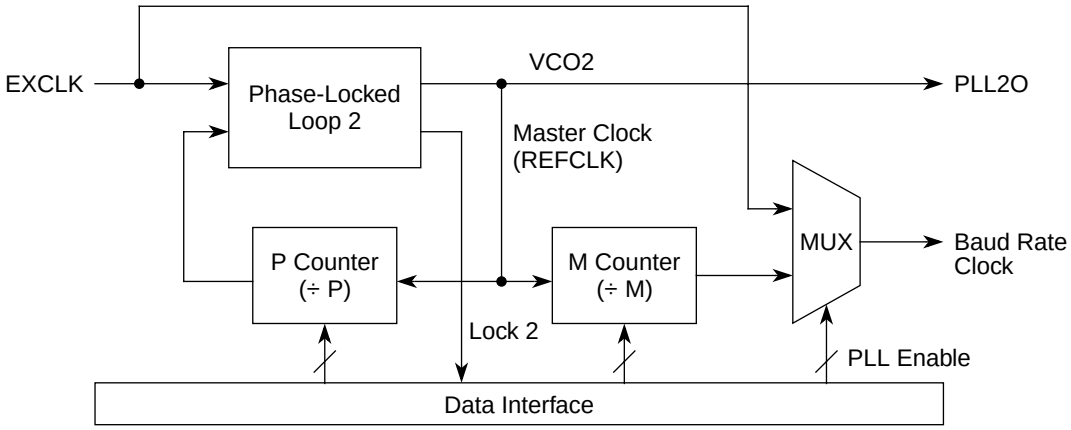


Figure 7-2. PLL2 Block Diagram

7.3 Programmer's Model of PLL 1 and PLL2

PLL1 has an M and a P counter with hardwired values that manipulate SYSCLK to generate an internal system clock and a bus clock. The PLL2 has an M and a P counter, which are programmable and divide EXCLK for setting the baud rate:

- P Counter: Divides VCO output frequency and feeds it back to the PLL phase comparator for comparison with the reference clocks SYSCLK or EXCLK.
- M Counter: This is a PLL post-divider that divides VCO.
- Baud rate = REFCLK X P value ÷ M value.

7.3.1 PLL Mode Register (PLL)

The PLL mode register controls PLL operation for the MC92460. By setting PLLEN, the PLLs are enabled which generate a master clock used for an internal system clock, a bus clock, and a baud rate clock. PLL2 uses the values written into PLL[P] and PLL[M] to generate the desired baud rate clock. PLL1 uses P and M counter values that are hardwired into the PLL1 to generate a 66 MHz internal clock and bus clock, given a 66 MHz SYSCLK. PLL1 does not use writable PLL[P] or PLL[M].

When the PLLs are turned off, EXCLK goes directly to baud rate clock out, and SYSCLK is used for the internal system clock and the bus clock. The lock bits are PLL status bits.

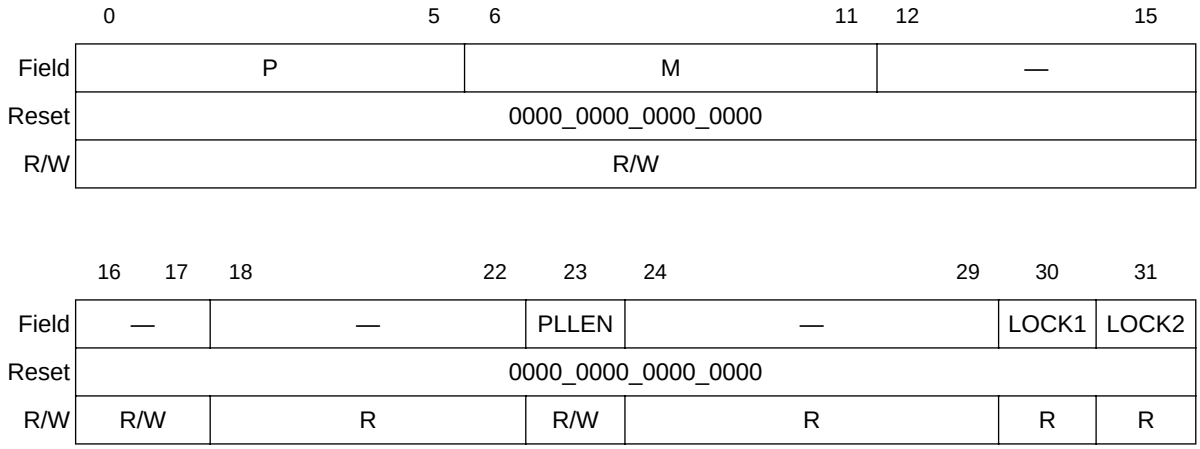


Figure 7-3. PLL Mode Register (PLL)

Table 7-1. PLL Mode Register Field Descriptions

Bit	Field	Description
0–5	P	P-counters value. Only used in PLL2. Refer to Table 7-2.
6–11	M	M-counters value. Only used in PLL2. Refer to Table 7-2.
12-17	—	Reserved
18-22	—	Reserved.
23	PLLEN	The PLLs can be enabled with this bit. When the PLLs are on, the internal master clock is used. When the PLLs are turned off, the external SYSCLK and EXCLK are used. 0: PLLs Off 1: PLLs On
24–29	—	Reserved.
30	Lock1	SYSCLK PLL1 lock/unlock status (read only) 0: PLL1 unlocked 1: PLL1 locked
31	Lock2	Baud rate PLL2 lock/unlock status (read only) 0: PLL2 unlocked 1: PLL2 locked

7.4 Operation of PLL1

PLL1 is enabled by setting PLLEN, and generates an internal system clock and a bus clock at the same frequency as SYSCLK.

7.5 Operation of PLL2

This section describes the operation and preferred sequence for controlling phase-locked loop 2. To generate a specific baud rate clock from PLL2, follow these steps:

- Set PLLEN, thereby turning on PLL operation
- Write PLL[P] and PLL[M] register values into the PLL mode register. Refer to the P and M register values of Table 7-2 to see the values needed.
- These register values are bit reversed and loaded into the respective P and M counters. Thus, if the hex value 0x0a, for example, is written into the register's six-bit P field with the binary value 0b001010, this binary value is bit reversed and loaded into the P Counter with 0b010100, which has the decimal value of 20.
- Wait for the Lock1 and Lock2 bits to be set to 1 (showing PLL1 and PLL2 are locked).

Table 7-2 shows the baud rate clock frequencies with a 14.72 MHz EXCLK. The formula used to calculate the baud rate clock is the following:

Baud rate clock = EXCLK * P / M

Table 7-2. Baud Rate Clock Output with 14.72 MHz EXCLK

Baud Rate (MHz)	P		M		PLLEN Value
	Register Value[1]	Counter Value[2]	Register Value[1]	Counter Value[2]	
EXCLK	X	X	X	X	0
7.350	0x0a	20	0x05	40	1
14.720	0x0a	20	0x0a	20	1
22.080	0x06	24	0x02	16	1
29.440	0x06	24	0x0c	12	1
36.800	0x0a	20	0x04	8	1
44.160	0x06	24	0x04	8	1
51.520	0x0e	28	0x04	8	1
58.880	0x01	32	0x04	8	1
66.240	0x09	36	0x04	8	1
73.600	0x0a	20	0x08	4	1
80.960	0x1a	22	0x08	4	1
88.320	0x06	24	0x08	4	1
95.680	0x16	26	0x08	4	1
103.040	0x0e	28	0x08	4	1
110.400	0x1e	30	0x08	4	1
117.760	0x01	32	0x08	4	1
125.120	0x11	34	0x08	4	1
132.480	0x09	36	0x08	4	1
139.840	0x19	38	0x08	4	1
147.200	0x0a	20	0x10	2	1
154.560	0x2a	21	0x10	2	1

Table 7-2. Baud Rate Clock Output with 14.72 MHz EXCLK (Continued)

161.920	0x1a	22	0x10	2	1
169.280	0x3a	23	0x10	2	1
176.640	0x06	24	0x10	2	1
184.000	0x26	25	0x10	2	1
191.360	0x16	26	0x10	2	1
198.720	0x36	27	0x10	2	1
206.080	0x0e	28	0x10	2	1
213.440	0x2e	29	0x10	2	1
220.800	0x1e	30	0x10	2	1
228.160	0x3e	31	0x10	2	1
235.520	0x01	32	0x10	2	1
242.880	0x21	33	0x10	2	1
250.240	0x11	34	0x10	2	1
257.600	0x31	35	0x10	2	1
264.960	0x09	36	0x10	2	1
272.320	0x29	37	0x10	2	1
279.680	0x19	38	0x10	2	1
287.040	0x39	39	0x10	2	1
294.400	0x05	40	0x10	2	1
294.400	0x0a	20	0x10	1	1

1 This is the hex value of the register field. Convert this hex value into a binary value and enter into the register.

2 The register's binary value is bit reversed and loaded into the counter. This table show the decimal of the counter binary number.

Chapter 8
Reset

The MC92460 has several inputs to the reset logic:

- Power-on reset ($\overline{\text{POR}}$)
- External hard reset ($\overline{\text{HRESET}}$)
- JTAG reset ($\overline{\text{TRST}}$)

All of these reset sources are fed into the reset controller and, depending on the source of the reset, different actions are taken.

8.1 Reset Causes

Table 8-1 describes reset causes.

Table 8-1. Reset Causes

Name	Description
$\overline{\text{POR}}$	Input pin. Asserting this pin initiates the power-on reset sequence that resets both PLLs.
$\overline{\text{HRESET}}$	Input pin. Asserting this pin initiates the hard reset sequence that resets all of the chip except the PLL and JTAG blocks.
$\overline{\text{TRST}}$	When JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated.

8.1.1 Reset Actions

The reset block has reset control logic that determines the cause of reset, synchronizes it if necessary, and resets the appropriate logic modules. The system control unit, HDLC controller, master interface, register and I/O pins are initialized only on hard reset.

Table 8-2 identifies reset actions for each reset source.

Table 8-2. Reset Action for Each Reset Source

Name	Resets Logic	Resets PLL	Resets JTAG
$\overline{\text{POR}}$	No	Yes	No
$\overline{\text{HRESET}}$	Yes	No	No
$\overline{\text{TRST}}$	No	No	Yes

8.1.2 Power-On Reset Sequence

Assertion of the $\overline{\text{POR}}$, $\overline{\text{TRST}}$, and $\overline{\text{HRESET}}$ external signals initiates the power-on reset sequence (reset flow). $\overline{\text{POR}}$ and $\overline{\text{TRST}}$ should be asserted externally for at least two input clock cycles after external power to the chip reaches at least 2/3 Vcc (the CORE and I/O supply voltages). PLL1 will be locked after the $\overline{\text{POR}}$ is negated and then will count 4096 SYSCLK cycles. When the PLL1 is locked, the clock block starts distributing clock signals in the chip.

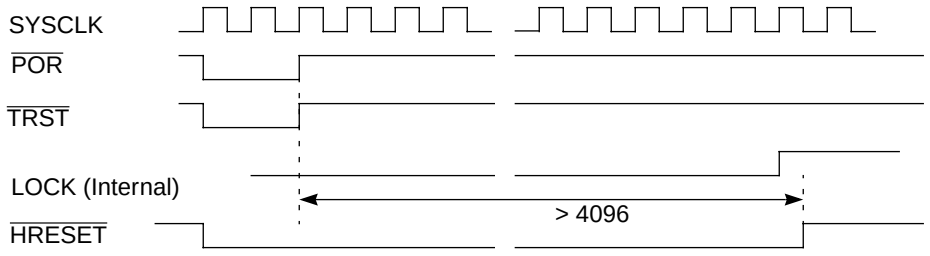


Figure 8-1. Reset Flow

8.1.3 $\overline{\text{HRESET}}$ Sequence

The $\overline{\text{HRESET}}$ sequence may be initiated externally by asserting $\overline{\text{HRESET}}$ for at least 4096 input clock cycles after $\overline{\text{POR}}$ is negated. Refer to Figure 8-1.

8.1.4 $\overline{\text{TRST}}$

The test reset signal ($\overline{\text{TRST}}$) is an asynchronous reset with an internal pull-up resister that provides initialization of the TAP controller and other logic required by the IEEE1149.1 standard.

Appendix A
Transaction and Bandwidth Calculations
for the MC92460

This appendix specifies the number of access cycles the MC92460 requires to process transactions, and provides information on improving 60x bus performance of HDLC data transfers through using the MC92460's external DMA mode.

A.1 Memory Cycles

Table A-1 shows the average number of clock cycles for DMA transactions. This cycle count is for the whole period of a transaction, from preparation, bus grant, bus transfer, through closing the transaction. The number of clock cycles for a transaction depends on various factors including memory specifications, the last address or data tenure, the number of wait cycles defined by a memory controller or a bus arbiter, whether a bottleneck exists on the bus, and the pipeline-status at the transaction start.

NOTE

Table A-1 shows average clock cycles and is not related to Figure A-1, which is an example of pipelined 4-beat burst accesses.

Table A-1. Average Clock Cycles by External DMA

Type of Main Memory	Single Beat		4-Beat Burst		Two 4-Beat Bursts	
	READ	WRITE	READ	WRITE	READ	WRITE
SDRAM (cas latency=2)	7 + 1	5 + 1	13 + 1	11 + 1	19 + 1	15 + 1
SyncBurstSRAM (pipelined)	5 + 1	4 + 1	8 + 1	7 + 1	16 + 1	14 + 1

In this Table, (+1) is a minimum idle cycle and is inserted before each transaction.

Figure A-1 shows wave forms during a DMA access of two continuous 4-beat bursts. The first read access to *radr1* (memory address) is counted 9+1 for a 4-beat burst, and the second 4-beat burst access *radr2* is counted 16+1.

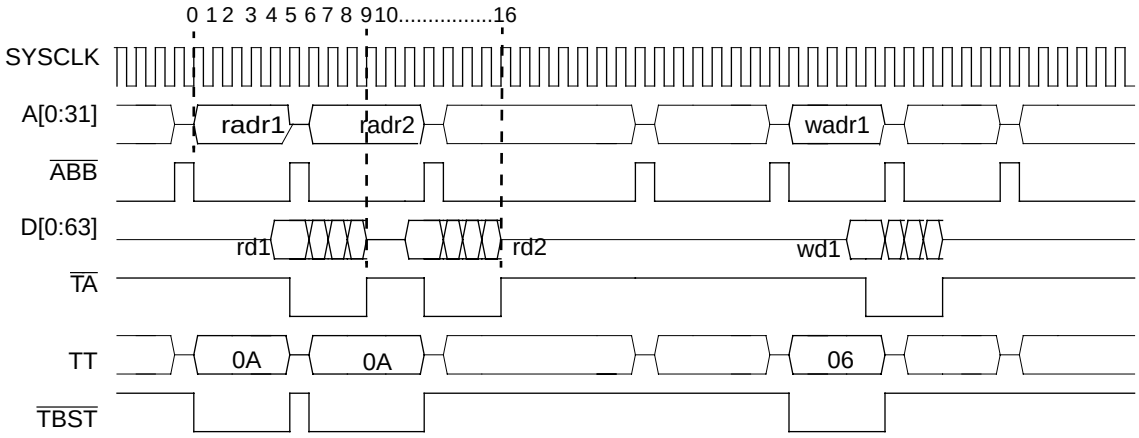


Figure A-1. DMA Accesses: Two 4-Beat Burst Reads and One Write

A.2 60x Bus Performance Consideration

The MC92460's buffer descriptor transmits data to and receives data from the DMA module in a maximum 32 byte data size. The MC92460's DMA module divides or combines the data to adhere to a 32-byte alignment.

Table A-2 shows the transactions generated by a DMA read. In the table, "single" refers to a single beat and "burst" means a 4-beat burst.

Table A-3 shows single-beat transactions generated by a DMA write. In the table, 7+8+8+8+1 means a single beat of 7 bytes, three beats of 8 bytes, and a single beat of 1 byte.

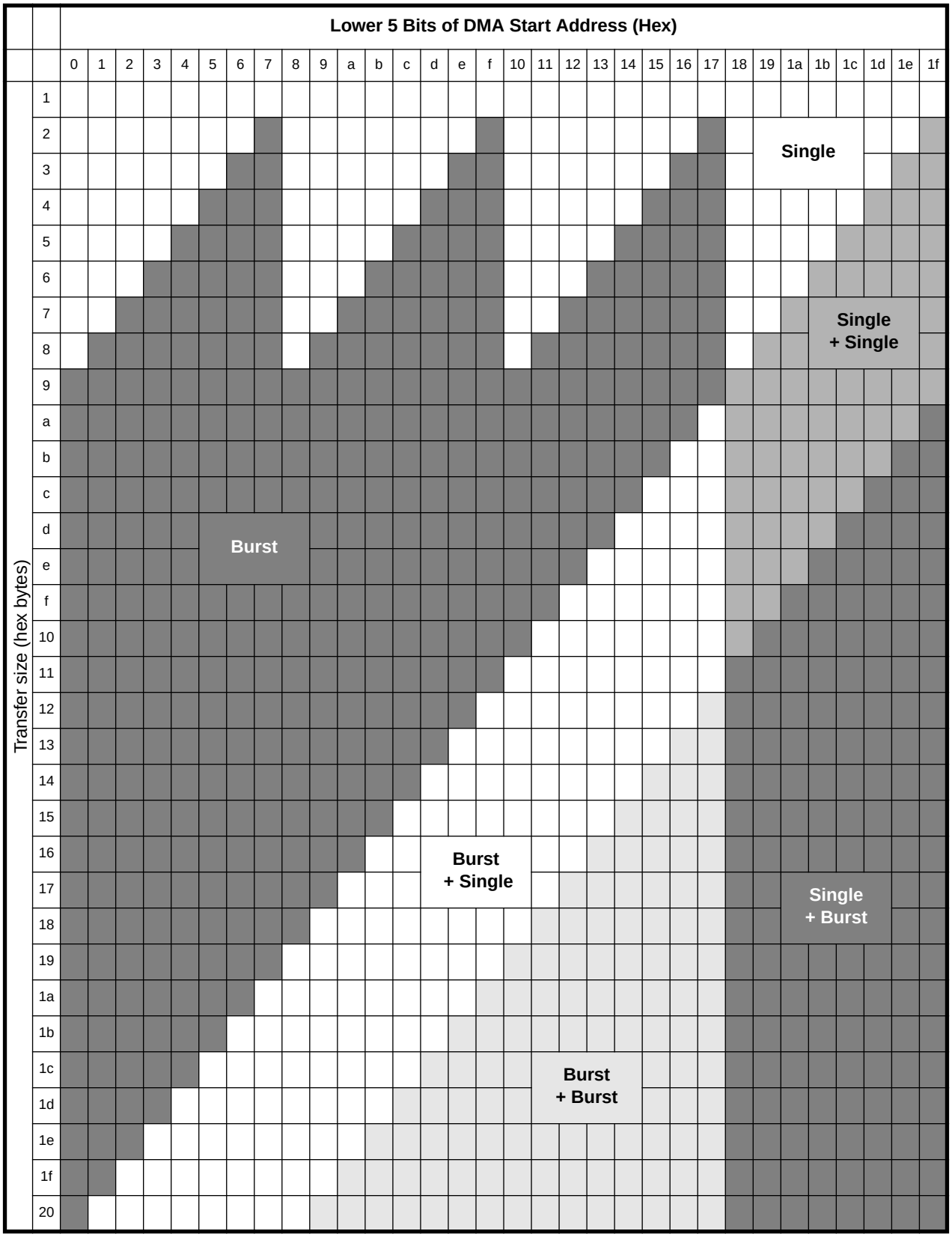
From these tables the data transfer speed can be calculated when the data is aligned along a burst boundary. If the data's destination or source address is burst aligned, it will take 1 burst access of 4 beats for a 32-byte read data event and two continuous burst accesses of 4 beats for a 32-byte write data event.

However, in the case of non-burst-aligned write data, it can take 5 write cycles for a 32-byte data transfer. In addition, write cases such as an irregular start address or small data size cause low bus performance (see the shading cells in Table A-3). In most cases cycle latency will be lowest by aligning data on burst boundaries.

These considerations apply only to MC92460's external DMA mode. The direct DMA mode of the MC92460 accepts data only when the data is 8-byte aligned and in minimum 8-byte data units.

In Table A-2, the rows are the data transfer size per set of external DMA read sequences, and the columns are the lower start address of source memory by the same sequence. This table shows the transactions that will be generated by the DMA in all cases.

Table A-2. External DMA Transactions–Read Sequence



In Table A-3, the rows are labeled data transfer size by one set of external DMA write sequence, and the columns are labeled lower start address of destination memory by the sequence. This table shows single-beat transactions that will be generated by the DMA write in all cases.

Table A-3. External DMA Transactions–Write Sequence

	Lower 3 Bits of Start Address							
Size	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	1	1	1	1	1	1	1
2	2	2	2	1-1	2	2	2	1-1
3	3	3	2-1	1-2	3	3	2-1	1-2
4	4	3-1	2-2	1-3	4	3-1	2-2	1-3
5	5	3-2	2-3	5	4-1	3-2	2-3	1-4
6	6	3-3	6	5-1	4-2	3-3	2-4	1-5
7	7	7	6-1	5-2	4-3	3-4	2-5	1-6
8	8	7-1	6-2	5-3	4-4	3-5	2-6	1-7
9	8-1	7-2	6-3	5-4	4-5	3-6	2-7	1-8
a	8-2	7-3	6-4	5-5	4-6	3-7	2-8	1-8-1
b	8-3	7-4	6-5	5-6	4-7	3-8	2-8-1	1-8-2
c	8-4	7-5	6-6	5-7	4-8	3-8-1	2-8-2	1-8-3
d	8-5	7-6	6-7	5-8	4-8-1	3-8-2	2-8-3	1-8-4
e	8-6	7-7	6-8	5-8-1	4-8-2	3-8-3	2-8-4	1-8-5
f	8-7	7-8	6-8-1	5-8-2	4-8-3	3-8-4	2-8-5	1-8-6
10	8-8	7-8-1	6-8-2	5-8-3	4-8-4	3-8-5	2-8-6	1-8-7
11	8-8-1	7-8-2	6-8-3	5-8-4	4-8-5	3-8-6	2-8-7	1-8-8
12	8-8-2	7-8-3	6-8-4	5-8-5	4-8-6	3-8-7	2-8-8	1-8-8-1
13	8-8-3	7-8-4	6-8-5	5-8-6	4-8-7	3-8-8	2-8-8-1	1-8-8-2
14	8-8-4	7-8-5	6-8-6	5-8-7	4-8-8	3-8-8-1	2-8-8-2	1-8-8-3
15	8-8-5	7-8-6	6-8-7	5-8-8	4-8-8-1	3-8-8-2	2-8-8-3	1-8-8-4
16	8-8-6	7-8-7	6-8-8	5-8-8-1	4-8-8-2	3-8-8-3	2-8-8-4	1-8-8-5
17	8-8-7	7-8-8	6-8-8-1	5-8-8-2	4-8-8-3	3-8-8-4	2-8-8-5	1-8-8-6
18	8-8-8	7-8-8-1	6-8-8-2	5-8-8-3	4-8-8-4	3-8-8-5	2-8-8-6	1-8-8-7
19	8-8-8-1	7-8-8-2	6-8-8-3	5-8-8-4	4-8-8-5	3-8-8-6	2-8-8-7	1-8-8-8
1a	8-8-8-2	7-8-8-3	6-8-8-4	5-8-8-5	4-8-8-6	3-8-8-7	2-8-8-8	1-8-8-8-1
1b	8-8-8-3	7-8-8-4	6-8-8-5	5-8-8-6	4-8-8-7	3-8-8-8	2-8-8-8-1	1-8-8-8-2
1c	8-8-8-4	7-8-8-5	6-8-8-6	5-8-8-7	4-8-8-8	3-8-8-8-1	2-8-8-8-2	1-8-8-8-3

Table A-3. External DMA Transactions–Write Sequence (Continued)

	Lower 3 Bits of Start Address							
1d	8-8-8-5	7-8-8-6	6-8-8-7	5-8-8-8	4-8-8-8-1	3-8-8-8-2	2-8-8-8-3	1-8-8-8-4
1e	8-8-8-6	7-8-8-7	6-8-8-8	5-8-8-8-1	4-8-8-8-2	3-8-8-8-3	2-8-8-8-4	1-8-8-8-5
1f	8-8-8-7	7-8-8-8	6-8-8-8-1	5-8-8-8-2	4-8-8-8-3	3-8-8-8-4	2-8-8-8-5	1-8-8-8-6
20	8-8-8-8 *1	7-8-8-8-1	6-8-8-8-2	5-8-8-8-3	4-8-8-8-4	3-8-8-8-5	2-8-8-8-6	1-8-8-8-7

Note: If the lower 4 bits of the starting address is 2'b_0000, this transaction is a 4-beat burst transaction.

A.3 Total Theoretical Bus Bandwidth

The MC92460's total bus bandwidth, or throughput (T_{dt}), is calculated by multiplying the speed of the bus by its width by the number of beats per bus cycle. The formula for the current version of the MC92460 is as follows:

$$T_{dt}\ (\text{Mbps}) = 66.7\ \text{MHz} * 64\ \text{bits} * 4\ \text{beats} / N\ \text{bus clocks})$$

In most cases, the theoretical maximum data transfer speed is defined by the number of cycles required to access memory.

If, for example, one burst write cycle takes 7 clocks, and one burst read cycle takes 8clocks, the average memory access cycle time *N* will be (8+1 +7+1)/2 = 8.5 clocks. In this case, the theoretical maximum data transfer rate (T_{dt}) is as shown below:

$$T_{dt}\ (\text{Mbps}) = 64\ \text{bits x 4 beat x 66.7 (MHz)} / (N\text{=8.5 cycles}) = 2000\ \text{Mbps}*$$

*This is the theoretical data transfer value and ignores any memory service time (such as refresh time)

Table A-4. Theoretical Maximum Data Transfer Rate

Type of Main Memory	Typical Cycle Count for 4-Beat Burst		*N* (Average Number of Cycles)	Tdt (Mbps)
	READ	WRITE		
SDRAM (cas latency=2)	13 + 1	11 + 1	13	1300
SyncBurst SRAM (pipelined)	8 + 1	7 + 1	8.5	1987

The cycle count (clock count during address issue and $\overline{TA}$ negation) is + 1.

The above value is calculated with ideal bus transactions, thus in the ideal environment, read (or write) burst transactions are performed continuously, and only the MC92460 will be granted the 60x bus.

A.4 Typical Bandwidth

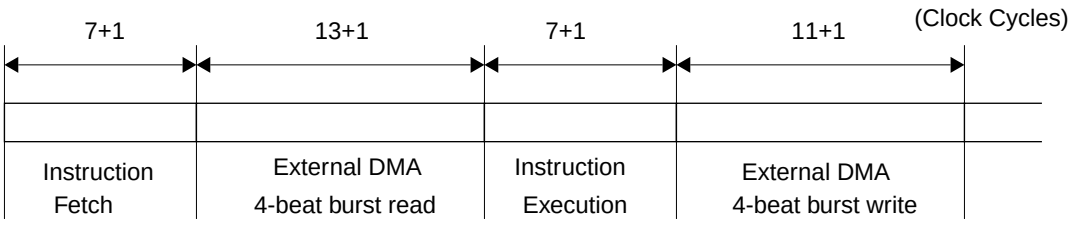
In real applications, it is impossible for the MC92460 to ensure the T_{dt} value, because there are so many cases in which the bus grant might be issued to another master. Additionally, the total data transfer speed of the HDLCs is different from the speed on the 60x bus over a short time period. The following, therefore, should be used to calculate the realistic data transfer speed.

R_{bg} = Estimated occupation rate of 60x bus by the MC92460's external DMA transactions (depends on the arbiter, firmware, HDLC channel usage)

Thus, the T_d may be as shown below in the application.

$$T_d = R_{bg} \times T_{dt}$$

Figure A-2 shows a example of 60x bus behavior with a PQII. In this example, instructions are located in a SDRAM, the instruction cache is disabled, and the instruction has access to an HDLC register. The approximate data transfer speed in the system can be estimated as T_d = 0.62 x 1300 = 800 Mbps. In most cases, overhead cycles from each bus-master change can be ignored because of bus pipelining.



Cm1 = Average clock cycles of PQII occupation = 7+1 = 8
N = Average clock cycles of MC92460 occupation = (13+1 +11 +1)/2 = 13
Rbg= N / (Cm1 + N) = 0.62

*The value is without any memory service time (for example refresh time)
*This figure is modeled to calculate the Rbg value, the real bus behavior is pipelined

Figure A-2. 60x Bus Behavior with PQII System

Note that single beat transactions may cause low bus performance. See Section A.2 .

Although the throughput of one HDLC channel can be 133 Mbps, the maximum throughput of all channels is not a simple extrapolation of 40 X 133. Throughput approaches the maximum 1919 Mbps linearly as illustrated in Figure A-3. The primary bottleneck on throughput is that the maximum data tranfer speed for each of the transmission paths (Rx and Tx) is 960 Mbps.

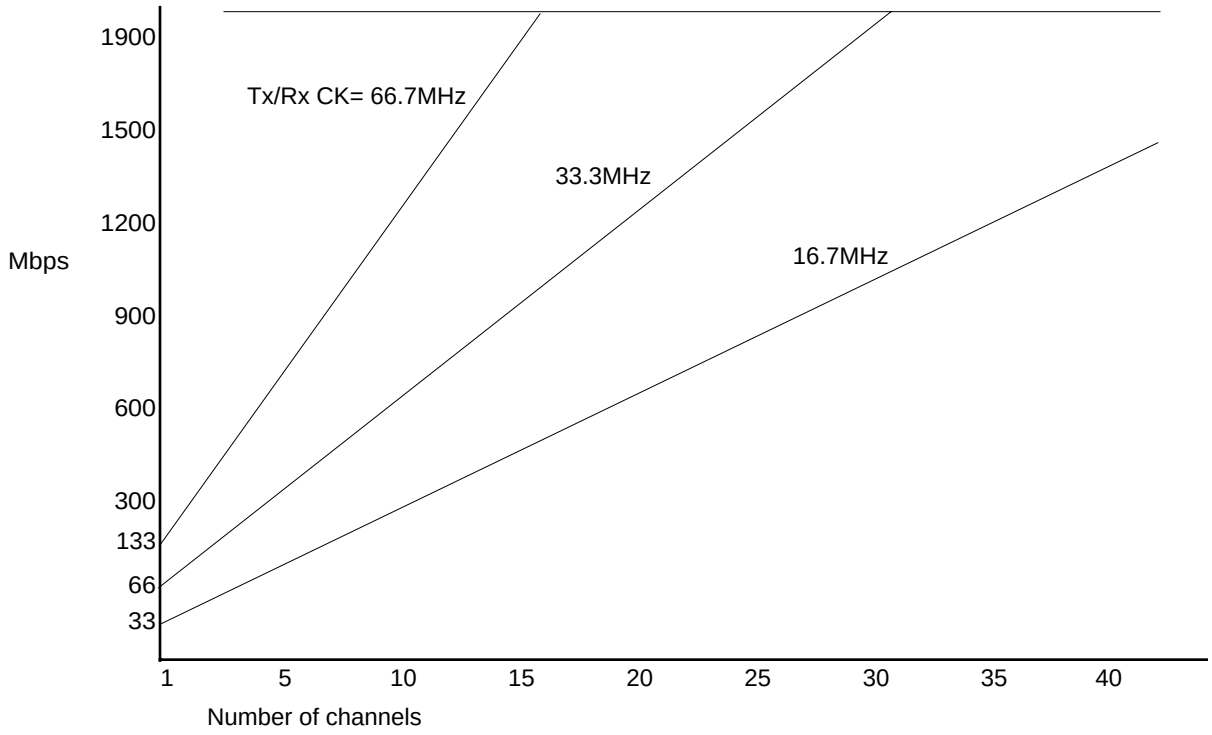


Figure A-3. Throughput Increase per Number of Channels Used

Finally, it is important to note that when the MC92460's own communication buffers are used, the maximum data transfer speed may be faster. An example of this is a configuration using the MC92460's communication buffer as the transmit buffer and accessing external memory through a DMA transaction for the receive buffer.

A.5 Maximum Throughput Measurement

Table A-5 shows the MC92460's performance data measured from a real application.

All the HDLCs are assigned loop-back mode to measure the strict maximum performance of the device. All 80 channels (40 Rx and 40 Tx) are enabled, and each channel uses a default BD. All TxBDs indicate the same address as their Tx buffer memory, each RxBD indicates the independent (unique) memory address as their Rx buffer memory. All TxBDs and RxBDs are configured in "continuous mode," thus each channel continues to transmit/receive frames during the test. Because some BDs are closed by overrun or underrun status, the maximum data throughput can be measured by counting the number of these error BDs.

Table A-5. Maximum Throughput

Tx Buffer	Rx Buffer	Tx/Rx CLK Frequency	Number of channels		Transferred Data Size TT (Mbps)	Actual Throughput AT (Mbps)
			Tx	Rx		
SPRAM	SPRAM	33.3MHz	29ch	29ch	1933	1919
SDRAM	SDRAM	16.7MHz	40ch	39ch	1317	1297
SPRAM	none	33.3MHz	29ch	-	967	960
none	SPRAM	33.3MHz	-	29ch	967	960
SDRAM	SPRAM	33.3MHz	22ch	22ch	1466	1443
SDRAM	none	22.2MHz	33ch	-	733	722
none	SDRAM	22.2MHz	-	35ch	778	766
Shows the source buffer location for the transmit	Shows the destination buffer location for the reception.		Shows the maximum number of channels which succeed to transfer data without overrun/underrun.		Shows the data throughput.	Shows the actual data throughput excluding flags.

SPRAM : communication buffer 0x0~0x7FFF.
AT = TT x (PD / AD)
PD : Pure data size excluding start/end flag.
AD : Full data size which is transferred.
The size of a frame is 128 or 288 bytes.

Index

T