Hybrid Memory Cube – HMC Gen2

MT43A4G40200 – 2GB 4H DRAM stack

HMC Memory Features

- V_{DDM} = 1.2V ±0.06V
- V_{CCP} = 2.5V ±0.125V
- 2GB configuration
  - 128 memory banks
- Configured as 16 independent memory vaults
- Closed-page memory architecture
- Built-in memory controller for each vault
  - Automatic refresh control over all temperatures
- Internal ECC data correction
- Advanced RAS features including data scrubbing
- Post-assembly repair capability
- In-field repair for ultimate reliability

HMC Interface Features

- V_{DD} = 0.9V
- V_{TT}, V_{TR}, V_{DDPLL} = 1.2V
- V_{DDK} = 1.5 V
- 10 Gb/s, 12.5 Gb/s, or 15 Gb/s SerDes I/O interface
- Up to four 16-lane, full-duplex serialized links
  - Half-width link (8-lane) configuration also supported
  - Up to 160 GB/s bandwidth
- Packet-based data/command interface
- Supports 16, 32, 48, 64, 80, 96, 112, and 128 byte data payloads per request
- Cyclic redundancy check (CRC) error detection for packets with automatic retry
- Power management supported per link
- Through-silicon via (TSV) technology
- Built-in self-test (BIST)
- JTAG interface (IEEE 1149.1-2001, 1149.6)
- I^2C Interface (UM-10204 I^2C bus specification)

Options	Marking
• Configuration	
– 2GB cube (4Gb x 4H DRAM stack)	4G4
• BGA package (Pb-free)	
– 4-link (31mm x 31mm) 2GB	NFA
• Operating temperature	
– DRAM: (0°C ≤ T_J ≤ 105°C)	None
– Logic: (0°C ≤ T_J ≤ 110°C)	None
• PHY	
– HMC Gen2	-S15
• Logic revision	02
• DRAM revision	:A

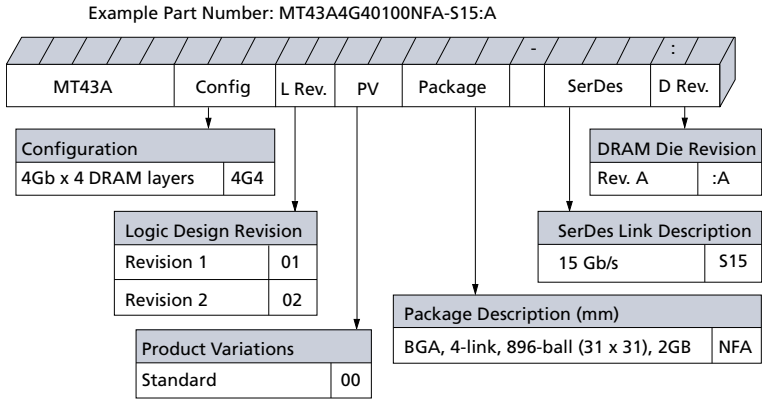
Description

Hybrid Memory Cube (HMC) is a single package containing four DRAM die and one logic die, all stacked together using through-silicon via (TSV) technology.

Within each cube, memory is organized vertically—portions of each memory die are combined with the corresponding portions of the other memory die in the stack. Each grouping of memory partitions is combined with a corresponding section of the logic die, forming what is referred to as a vault.

Specifications within this data sheet are compatible with the HMC Consortium Specification.

Figure 1: HMC Part Numbers



Contents

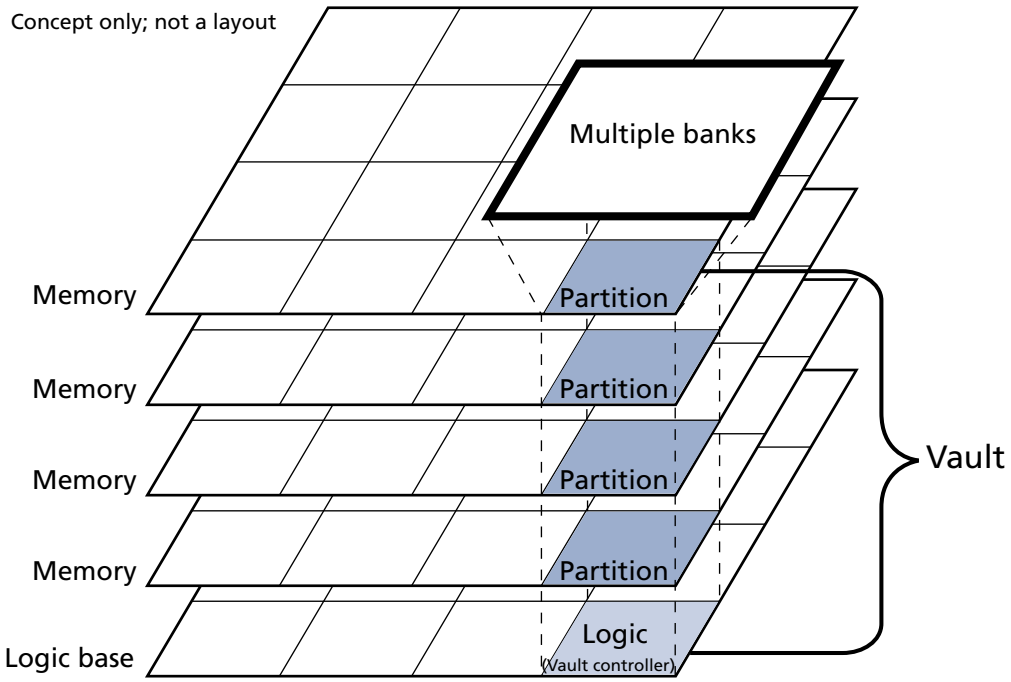
List of Figures

List of Tables

HMC Architecture

HMC consists of a single package containing multiple memory die and one logic die, stacked together using through-silicon via (TSV) technology (see the example below). Within HMC, memory is organized into vaults. Each vault is functionally and operationally independent.

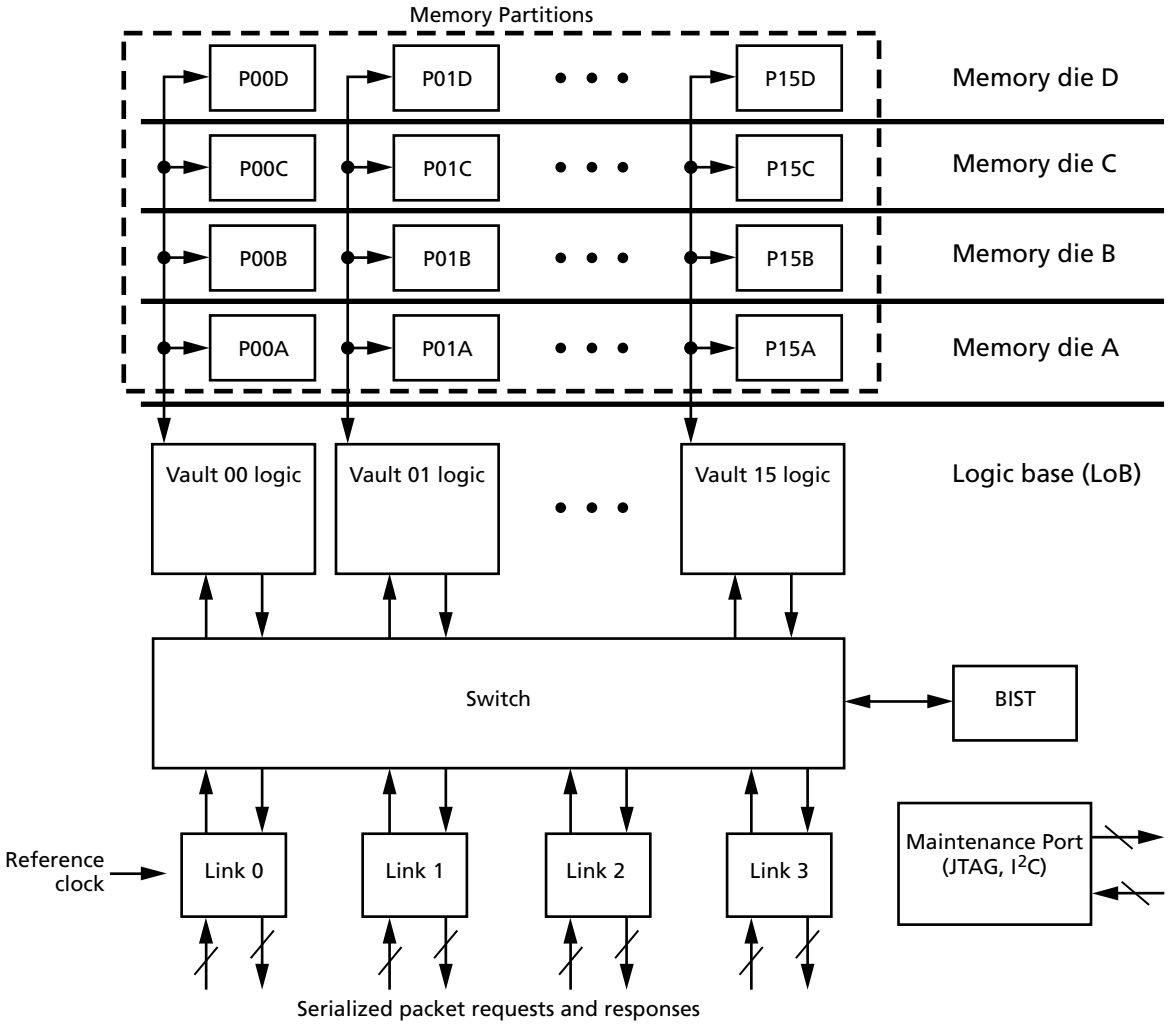
Figure 2: Example HMC Organization



Each vault has a memory controller (called a vault controller) in the logic base that manages all memory reference operations within that vault. Each vault controller determines its own timing requirements. Refresh operations are controlled by the vault controller, eliminating this function from the host memory controller.

Each vault controller has a queue used to buffer references for that vault's memory. The vault controller may execute references within that queue based on need rather than order of arrival. Therefore, responses from vault operations back to the external serial I/O links may be out of order. However, requests from a single external serial link to the same vault/bank address are executed in order. Requests from different external serial links to the same vault/bank address are not guaranteed to be executed in a specific order and must be managed by the host controller.

Figure 3: HMC Block Diagram Example Implementation (4-link HMC configuration)



Logic Base Architecture

The logic base manages multiple functions for HMC (see the example implementation in the figure above):

- All HMC I/O, implemented as multiple serialized, full-duplex links
- Memory control for each vault; data routing and buffering between I/O links and vaults
- Consolidated functions removed from the memory die to the controller
- Mode and configuration registers
- BIST for the memory and logic layer
- JTAG test access port compliant to JTAG IEEE 1149.1-2001, 1149.6
- I^2C test access port compliant to UM-10204 I^2C bus specification
- Spare resources enabling field recovery from some internal hard faults

The collective, internally available bandwidth from all of the vaults is made accessible to the I/O links. A crossbar switch is an example implementation of how this could be achieved. The external I/O links consist of multiple serial links, each with a default of 16 input lanes and 16 output lanes for full duplex operation. A half-width configuration is also supported, comprised of 8 input lanes and 8 output lanes per link. Each lane direction in each link has additional power-down signals for power management. The following configurations are those supported within this specification.

Table 1: HMC Configuration

	Configurations
Number of links in package	4
Link speed (Gb/s)	10, 12.5, 15
Memory density	2GB
Number of vaults	16
Memory banks	128 banks
Maximum aggregate link bandwidth[1]	240 GB/s
Maximum DRAM data bandwidth	160 GB/s (1.28 Tb/s)
Maximum vault data bandwidth	10 GB/s (80 Gb/s)

Note: 1. Link bandwidth includes data as well as packet header and tail. Full-width (16 input and 16 output lanes) configuration is assumed.

All in-band communication across a link is packetized. Packets specify single, complete operations. For example, a READ reference of 64 data bytes. There is no specific timing associated with memory requests, and responses can generally be returned in a different order than requests were made because there are multiple vaults, each executing independently of the others. The vaults generally reorder their internal requests to optimize bandwidths and to reduce average latencies.

Pin Descriptions

Table 2: Pin Descriptions

"P" pins are the true side of differential signals; "N" pins denote the complement side

Signal Pin Symbol	Type	Description
Link Interface[1]		
LxRXP[n:0]	Input	Receiving lanes[2]
LxRXN[n:0]		
LxTXP[n:0]	Output	Transmitting lanes[2]
LxTXN[n:0]		
LxRXPS	Input	Power-reduction input
LxTXPS	Output	Power-reduction output
FERR_N	Output	Fatal error indicator with an open-drain output that drives LOW if fatal error occurs in HMC. It should be tied HIGH with an external pull-up resistor on the PCB to the V_{DDK} rail. It can optionally be tied together (wire-OR'd) with those of other HMC devices to be monitored as a single fatal error flag, if desired. The output pin model for FERR_N is included in the HMC Gen2 IBIS model. Pin leakage current for the open-drain state is not modeled in the IBIS model, but is specified in the AC and DC Operating Conditions section of this data sheet. This leakage current should be considered when sizing the external pull-up resistor value, particularly if there are multiple HMCs with their FERR_N pins wire-OR'd together.
Clocks and Reset		
REFCLKP	Input	Reference clock for all links
REFCLKN		
REFCLKSEL	Input	Determines on-die termination for REFLCLKP/N. Set DC HIGH ($\geq V_{DDK}$ - 0.5V) if REFCLKP/N are AC-coupled. Set DC LOW ($\leq V_{SS}$ + 0.5V) if REFCLKP/N are DC-coupled.
P_RST_N	Input	System reset. Active LOW CMOS input referenced to V_{SS}. The P_RST_N is a rail-to-rail signal with DC HIGH ≥ (V_{DDK} - 0.5V) and DC LOW ≤ (V_{SS} + 0.5V).
JTAG Interface[3]		
TMS	Input	JTAG test mode select
TCK	Input	JTAG test mode clock
TDI	Input	JTAG test data-in
TDO	Output	JTAG test data-out
TRST_N	Input	JTAG test reset (active LOW)
I^2C Interface[3]		
SCL	Input	I^2C clock
SDA	Bidirectional	I^2C data
Bootstrapping Pins		

Table 2: Pin Descriptions (Continued)

"P" pins are the true side of differential signals; "N" pins denote the complement side

Signal Pin Symbol	Type	Description
CUB[2:0]	Input	User-assigned HMC identification to enable the host to map the unique location of HMC in a system. The I^2C slave address is equivalent to the state of the CUB[2:0] bits. This value will also be used as the default value for the CUB field of the Request packet header. DC HIGH ≥ (V_{DDK} - 0.5V) and DC LOW ≤ (V_{SS} + 0.5V).
REFCLK_BOOT[1:0]	Input	Bootstrapping pins that enable autonomous PLL configuration. Tie pins DC HIGH or DC LOW to match reference clock frequency being used. DC HIGH ≥ (V_{DDK} - 0.5V) and DC LOW ≤ (V_{SS} + 0.5V)
		REFCLK_BOOT[1:0] / **Reference Clock Frequency** 00 / 125 MHz 01 / 156.25 MHz 10 / 166.67 MHz 11 / Reserved
Analog Pins		
EXTRESTP	–	Pins used to connect upper (top) precision 200Ω resistor used for termination impedance auto-calibration
EXTRESTN	–	
EXTRESBP	–	Pins used to connect lower (bottom) precision 200Ω resistor used for termination impedance auto-calibration
EXTRESBN	–	
Reserved Pins		
DNU	–	Do not use; must not be connected on the board
TST_V_{SS}	Input	Must be connected to ground
Supply Pins		
V_{DD}	Supply	Logic supply: 0.9V
$V_{DDPLLA[3:0]}$, $V_{DDPLLB[3:0]}$, V_{DDPLLR}	Supply	Filtered PLL supplies: 1.2V ±0.06V. PLLAs are used for 10 Gb/s and 12.5 Gb/s operation and PLLBs are used for 15 Gb/s operation, though all V_{DDPLLx} pins must be connected to 1.2V supply.
V_{TT}, V_{TR}	Supply	Link transmit termination and link receive termination supplies: 1.2V
V_{DDM}	Supply	Memory supply: V_{DDM} = 1.2V
V_{CCP}	Supply	DRAM wordline boost supply 2.5V
V_{DDK}	Supply	NVM, I^2C, JTAG, and power management pin supply: 1.5V
V_{SS}	Supply	Ground

Notes:
1. x represents specific link number and will be 0 to 3 depending upon configuration.
2. In full-width mode, n = 15. In half-width mode, n = 7 and lanes 8–15 are considered DNU.
3. Additional usage details related to the JTAG and I^2C interfaces can be found in the HMC Gen2 Users Guide on Micron's web site.

Link Data Transmission

Commands and data are transmitted in both directions across the link using a packet-based protocol where the packets consist of 128-bit flow units called "FLITs." These FLITs are serialized, transmitted across the physical lanes of the link, then re-assembled at the receiving end of the link. Three conceptual layers handle packet transfers:

- The physical layer handles serialization, transmission, and deserialization.
- The link layer provides the low-level handling of the packets at each end of the link.
- The transaction layer provides the definition of the packets, the fields within the packets, and the packet verification and retry functions of the link.

Two logical blocks exist within the link layer and transaction layer:

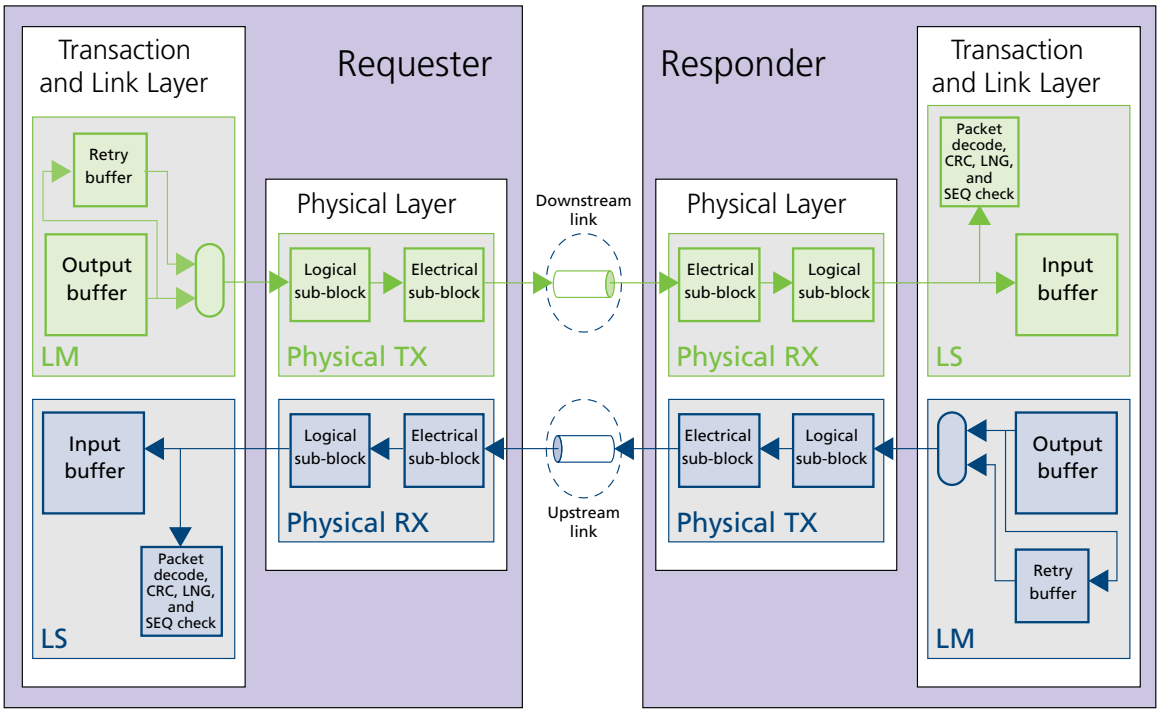
- The link master, is the logical source of the link where the packets are generated and the transmission of the FLITs is initiated.
- The link slave, is the logical destination of the link where the FLITS of the packets are received, parsed, evaluated, and then forwarded internally.

The nomenclature below is used throughout the specification to distinguish the direction of transmission between devices on opposite ends of a link. These terms are applicable to both host-to-cube and cube-to-cube configurations.

Requester: Represents either a host processor or an HMC link configured as a pass-through link. A requester transmits packets downstream to the responder.

Responder: Represents an HMC link configured as a host link. A responder transmits packets upstream to the requester.

Figure 4: Link Data Transmission Implementation Example



Logical Sub-Block of Physical Layer

The transfer of information across the links consists of 128-bit FLITs. Each FLIT takes the following path through the serial link:

1. 128-bit FLITs are generated by the link master and sent in parallel to the transmitting logical sub-block in the physical layer.
2. The transmitting logical sub-block serializes each FLIT and drives it across the link interface in a bit-serial form on each of the lanes.
3. The receiving logical sub-block deserializes each lane and recreates the 128-bit parallel FLIT. Receive deserializer FLIT alignment is achieved during link initialization.
4. The 128-bit parallel FLIT is sent to the link layer link slave section.

Link Serialization

Link serialization occurs with the least-significant portion of the FLIT traversing across the lanes of the link first. During one unit interval (UI) a single bit is transferred across each lane of the link. For the full-width configuration, 16 bits are transferred simultaneously during the UI, meaning it takes 8 UIs to transfer the entire 128-bit FLIT. For the half-width configuration, 8 bits are transferred simultaneously, taking 16 UIs to transfer a single FLIT. The following table shows the relationship of the FLIT bit positions to the lanes during each UI for both full-width and half-width configurations.

Table 3: Unit Interval FLIT Bit Positions for Full-Width Configuration

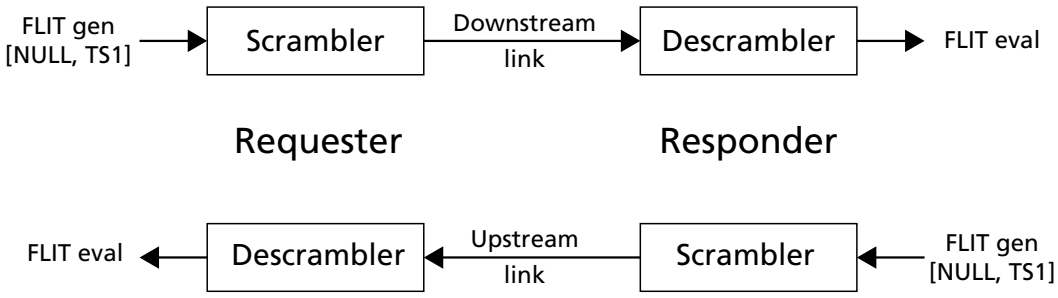
	Lane															
UI	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
7	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112

Table 4: Unit Interval FLIT Bit Positions for Half-Width Configuration

	Lane							
UI	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
…	…	…	…	…	…	…	…	…
15	127	126	125	124	123	122	121	120

Scrambling and Descrambling

Figure 5: Scrambler and Descrambler Paths from Requester to Responder



Use the following polynomial for the scrambler and descrambler logic: $1 + x^{-14} + x^{-15}$. To scramble the data, XOR the LSB of the linear feedback shift register (LFSR) with the data, as shown in Figure 6 (page 15). Seeding values vary per lane, as shown in Table 5 (page 16). Designers should include a mode register bit to bypass scrambling for debug purposes.

Figure 6: Scrambler Logic

```
// Title  :  scr_x15_serial.v
//
// This module implements a serial test circuit for a scrambler based on the
// polynomial 1+ x^(-14) + x^(-15).
//
// Such Scrambler is typically shown as a 15 bit Linear Feedback Shift Register
// (LFSR) with bits shifting from register 1 on the left to register 15 on the
// right, with register 14 and 15 combining to shift into register 1.
//
// The HMC Serializer outputs data[0] first from  parallel tx data[n:0],
// so if data[n:0] is to be bitwise scrambled with LFSR[n:0], we need the LFSR
// to shift from n -> 0, the opposite direction from the typical illustration.
// This implementation shifts data from LFSR[14] on the left to LFSR[0] on the
// right, with LFSR[1] and [0] combining to shift into LFSR[14].  This way
// LFSR[14:0] can bitwise scramble data[14:0] and be compatible with serializ-
// ation that shifts out on the data[0] side.
//
// Put otherwise: Polynomial 1+ x^(-14) + x^(-15) is equiv to  x^15 + x^1 + x^0

`timescale 100ps/100ps

module scr_x15_serial
(   input         CLK           ,
    input         I_ASYNC_RESETN,
    input  [14:0] I_SCRAM_SEED  , // unique per lane
    input         I_DATA        , // input  serial data
    output        O_DATA          // output serial data
);

reg  [14:0] LFSR; // LINEAR FEEDBACK SHIFT REGISTER

// SEQUENTIAL PROCESS
always @ (posedge CLK or negedge I_ASYNC_RESETN)
begin
    if (~I_ASYNC_RESETN)  LFSR[14:0] <= I_SCRAM_SEED;
    else                  LFSR[14:0] <= { (LFSR[1] ^ LFSR[0]) , LFSR[14:1] };
end                                     // serial shift right with left input

// SCRAMBLE
assign O_DATA = I_DATA ^ LFSR[0];

endmodule
```

Table 5: Scrambler Logic Seed Values

Lane	Seed Value	Lane	Seed Value
0	15'h4D56	8	15'h3EB3
1	15'h47FF	9	15'h2769
2	15'h75B8	10	15'h4580
3	15'h1E18	11	15'h5665
4	15'h2E10	12	15'h6318
5	15'h3EB2	13	15'h6014
6	15'h4302	14	15'h077B
7	15'h1380	15	15'h261F

Note: 1. Scrambler logic seed values for lanes 0–7 are the same with full-width and half-width configurations.

Table 6 contains a series of NULL packets to illustrate how the scrambler operates on each lane. The Lane 0 LFSR starts with seed value 0xCD56. This is the 15-bit seed value from Table 5 (15'h4D56) with the next (16th) bit of the PRBS included to enable scrambling of 16 data payload bits per clock cycle; the scrambler PRBS is calculated at 16 bits per clock cycle. The scrambler LFSR[15:0] is bitwise exclusive-OR'ed with data[15:0] to generate scrambled data. When NULLs, or all zeroes payload data per lane, are scrambled, the continual exclusive-or with (payload) zero causes the scrambled data to be identical to the scrambler LFSR value each clock cycle. As a result, the list of the scrambler LFSR sequence is also the list of scrambled data when the payload data is all NULLs.

Table 6: Scrambler Example

	Lane x LFSR[15:0]													
UI	0	1	2	3	4	5	6	7	8	9	10	11	12	13
16	CD56	47FF	75B8	1E18	2E10	BEB2	C302	1380	3EB3	A769	4580	D665	6318	601
32	D5FE	3200	A7B2	888A	1C8C	B0F5	9141	0D20	D0F5	FA6E	33A0	1EAB	A94A	680
48	5F80	1580	BA35	2667	C965	3447	6CF0	05D8	1C47	43AC	1538	48FF	3EF7	2E0
64	3820	0FA0	3397	9AAA	16EB	9732	2D44	839A	8932	F13D	8FD2	3640	90C6	5C8
80	1218	0438	952E	2BFF	4ECF	AE55	5DF3	212B	A6D5	84D1	A41D	16B0	EC52	F96
96	8D8A	8312	6FDC	1F00	34D4	3CBF	F985	58DF	3ADF	635C	BB09	0EF4	8D3D	42E
112	25A7	A14D	EC19	0840	575F	1170	02A3	3A58	13D8	E979	F346	44C7	A5D1	B1C
128	9BBA	B8F5	CD0A	0630	3E78	0CE4	C1F9	93BA	8D1A	CEE2	C572	B352	7B9C	749
144	2B33	3247	15C7	0294	90A2	454B	D082	2D33	25CB	94C9	93E5	B57D	E329	676
160	DF55	95B2	8F92	41EF	AC79	73F7	9C61	DDD5	5B97	EF56	2D0B	B7E1	C95E	2A6
176	187F	AFB5	A42D	308C	FD22	A506	6928	199F	BB2E	CC7E	5DC7	7608	56F8	DFA
192	0A20	3C37	BB1D	D465	81D9	FBC2	AEDE	0AA8	735C	5520	B992	A686	BEC2	983
208	0798	9116	B349	1F2B	E09A	8311	7CD8	87FE	E579	3FD8	B2AD	FAE2	B0D1	AA1
224	822A	ECCE	F576	485F	086B	614C	A15A	6200	CBE2	901A	B5FD	83C9	745C	7F8
240	219F	4D54	C7E6	3638	462F	E8F5	38FB	2980	9709	2C0B	B781	E116	E739	E02
256	18A8	75FF	D20A	9692	329C	0E47	5243	1EA0	EE46	5D07	7620	C8CE	CA52	081

Table 7: TS1 Definition

TS1 Bit Position															
(Sent Last) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	(Sent First) 0
1	1	1	1	0	0	0	0	L	L	L	L	S	S	S	S
0xF				0x0				Dependant on lane[1]				Sequence[2]			

Notes:
1. Lane field = 0xC for lane 15, or lane 7 in half-width configuration, 0x3 for lane 0, and 0x5 for all other lanes.
2. Sequence[3:0] is a 4-bit counter that increments from 0 to 15 with each successive TS1 sent on a lane. The sequence value will roll back to 0 after 15 is reached.
3. The half-width link configuration uses lanes 0–7.
4. The 2-byte TS1 character establishes the framing that identifies byte pairs that belong to the same FLIT after training completes in half-width mode. The transition from sending TS1s to sending NULL FLITs can occur on any byte boundary within the entire 32-byte TS1 sequence, including mid-TS1 character, so this transition boundary to NULL bytes should NOT be used to infer the FLIT boundary in half-width configuration.

Table 8: TS1 Training Sequence Example

TS1 Ordering	Lane 0	Lane 1–14	Lane 15
0	16'hF030	16'hF050	16'hF0C0
1	16'hF031	16'hF051	16'hF0C1
2	16'hF032	16'hF052	16'hF0C2
3	16'hF033	16'hF053	16'hF0C3
...	...	...	...
14	16'hF03E	16'hF05E	16'hF0CE
15	16'hF03F	16'hF05F	16'hF0CF

Lane Run Length Limitation

For each HMC RX lane, the scrambled data pattern must meet a run length limit of 85 UI. This is the maximum number of consecutive identical digits allowed. A run of 86 or more consecutive ones (or zeroes) must not be generated by the transmitter on any of the downstream lanes at any time. This restriction is in place so that the clock recovery circuit at each HMC RX lane meets its minimum required transition density to assure correct data alignment. This restriction does not require a particular implementation of the transmitter logic, as long as the run length limit is met at HMC RX lane inputs.

Although a given host may not have a run length limit on the scrambled upstream data, HMC is designed to meet run length limitations of 85 UI on each lane at all times. This is implemented through HMC's ability to monitor the scrambled data at each TX lane. If the scrambled TX data on any lane exceeds 85 consecutive digits, the TX scramble logic forces at least one transition. Forcing a transition corrupts the upstream response and results in a link retry (see Link Retry). The probability of such an event is approximately 2^{-80} for random payload data. HMC's run length limitation feature may be disabled within the mode register.

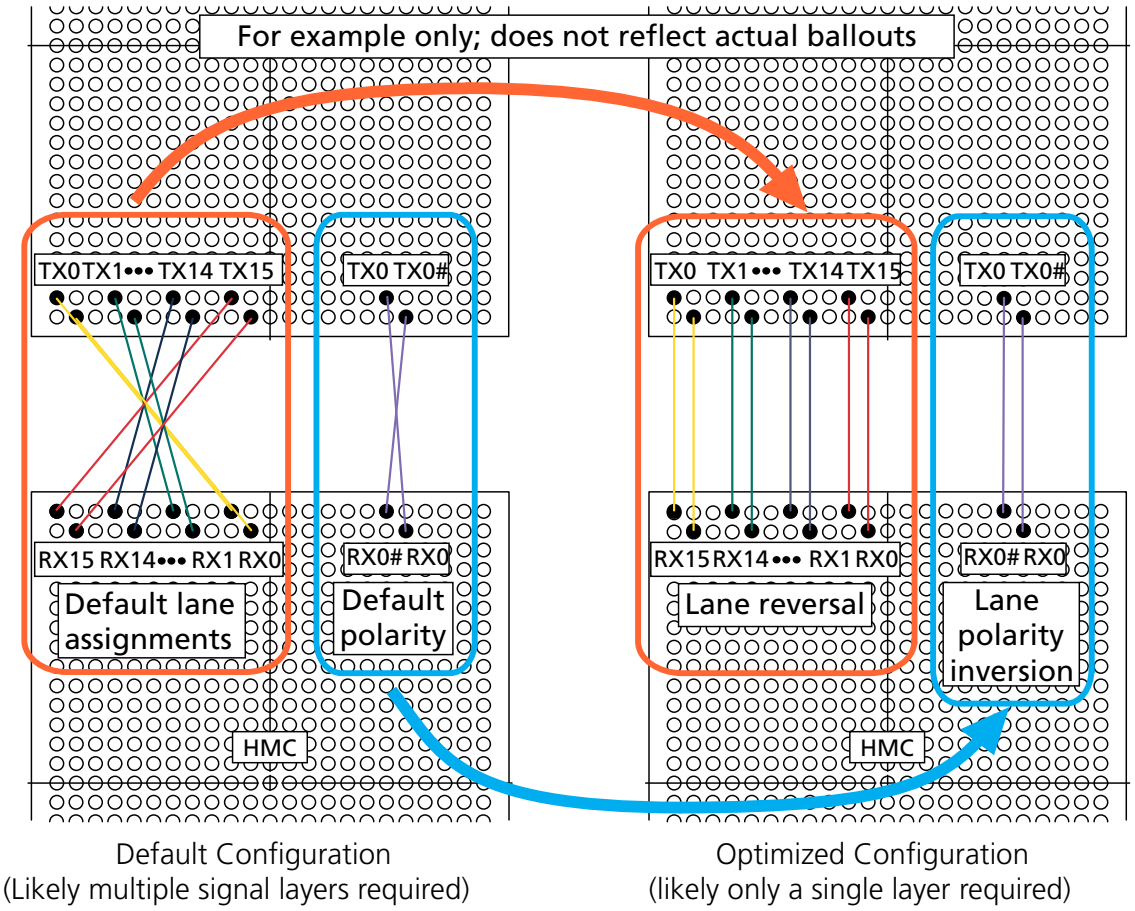
Lane Reversal

To accommodate different external link routing topologies, the RX logical sub-block has the capability to reverse the lane bit number assignment. The RX logical sub-block detects lane reversal at HMC when it receives the TS1 training sequence during initialization. Logic in each receiving logical sub-block automatically compensates for lane reversal and reassigns external Lane 0 to connect to Lane 15 internally, Lane 1 to connect to Lane 14 internally, and so on. Refer to Figure 7 (page 20). If a link is configured as a half-link, the logical sub-block reassigns external Lane 0 to connect to Lane 7 internally, Lane 1 to connect to Lane 6 internally, and so on. When lane ordering is detected and set by HMC's logical sub-block, it remains unchanged thereafter. The lane reversal mode does not have to be the same for both directions of the link. HMC may have one or more links with lane reversal enabled at their respective receivers. The host is responsible for any implementation of lane reversal from the upstream lanes.

Lane Polarity Inversion

In order to accommodate different external Link routing topologies, the RX logical sub-block has the capability to logically invert the received data on a lane-by-lane basis. Refer to Figure 7 (page 20). The polarity is determined during link initialization and remains unchanged thereafter. The training sequence is used to infer polarity at HMC's receiver. The requester is responsible for any implementation of lane polarity inversion from upstream lanes.

Figure 7: Lane Reversal, Lane Polarity Inversion Example



Default Configuration
(Likely multiple signal layers required)

Optimized Configuration
(likely only a single layer required)

Chaining

Multiple HMCs may be chained together to increase the total memory capacity available to a host. A network of up to four HMCs are supported. Each HMC in the network is identified through the value in its CUB field, located within the request packet header. The host processor must load routing configuration information into each HMC. This routing information enables each HMC to use the CUB field to route request packets to their destination.

Each HMC link in the cube network is configured as either a host link or a pass-through link, depending upon its position within the topology. See Figure 8. Figure 9, Figure 10 and Figure 11 for illustration. A host link uses its link slave to receive request packets and its link master to transmit response packets. After receiving a request packet, the host link will either propagate the packet to its own internal vault destination (if the value in the CUB field matches its programmed cube ID) or forward it towards its destination in another HMC via a link configured as a pass-through link. In the case of a malformed request packet whereby the CUB field of the packet does not match any device in the chain (but CRC is correct), a write request may cause silent data corruption within the local cube or a read response may return invalid data from an unintended address.

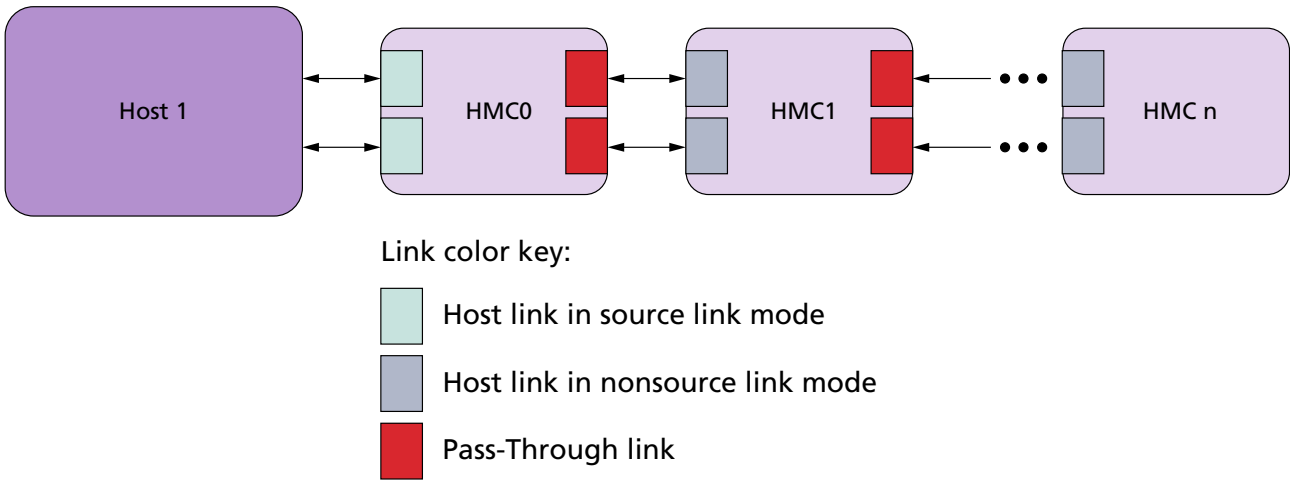
A pass-through link uses its link slave to receive response packets and its link master to transmit the request packet towards its destination cube.

A link can only be configured as one of three types of defined links, a host link in source link mode, a host link in nonsource link mode or a pass-through link. Each link type has a defined direction of operation as described above. The connection between two cubes will always be a pass-through link connected to a host link in nonsource linked mode. The defined direction of operation of these two link types will only allow requests and responses to flow in one direction. A single link connecting two HMCs does NOT support requests and responses flowing in both directions. The supported multihost topology shown in Figure 10 uses two links with the pass-through link and the host link in nonsource linked mode configured in opposite directions. This allows requests and responses to properly flow between the hosts and the cubes in the chain.

The HMC link connected directly to the host processor must be configured as a host link in source mode. The link slave of the host link in source mode has the responsibility to generate and insert a unique value into the source link identifier (SLID) field within the tail of each request packet. The unique SLID value is used to identify the source link for response routing. The SLID value does not serve any function within the request packet other than to traverse the cube network to its destination vault where it is then inserted into the header of the corresponding response packet. The routing information loaded by the host enables each HMC to use the SLID value to route response packets to their destination. On the opposite side of a pass-through link is a host link that is NOT in source mode. This host link operates with the same characteristics as the host link in source mode except that it does not generate and insert a new value into the SLID field within a request packet. All link slaves in pass-through mode use the SLID value generated by the host link in source mode for response routing purposes only. The SLID fields within the request packet tail and the response packet header are considered "Don’t Care" by the host processor.

Figure 8, Figure 9, Figure 10, and Figure 11 show the supported multicube topologies.

Figure 8: Serial Chain Topology



For this topology, n ≤7.

Figure 9: Star Topology

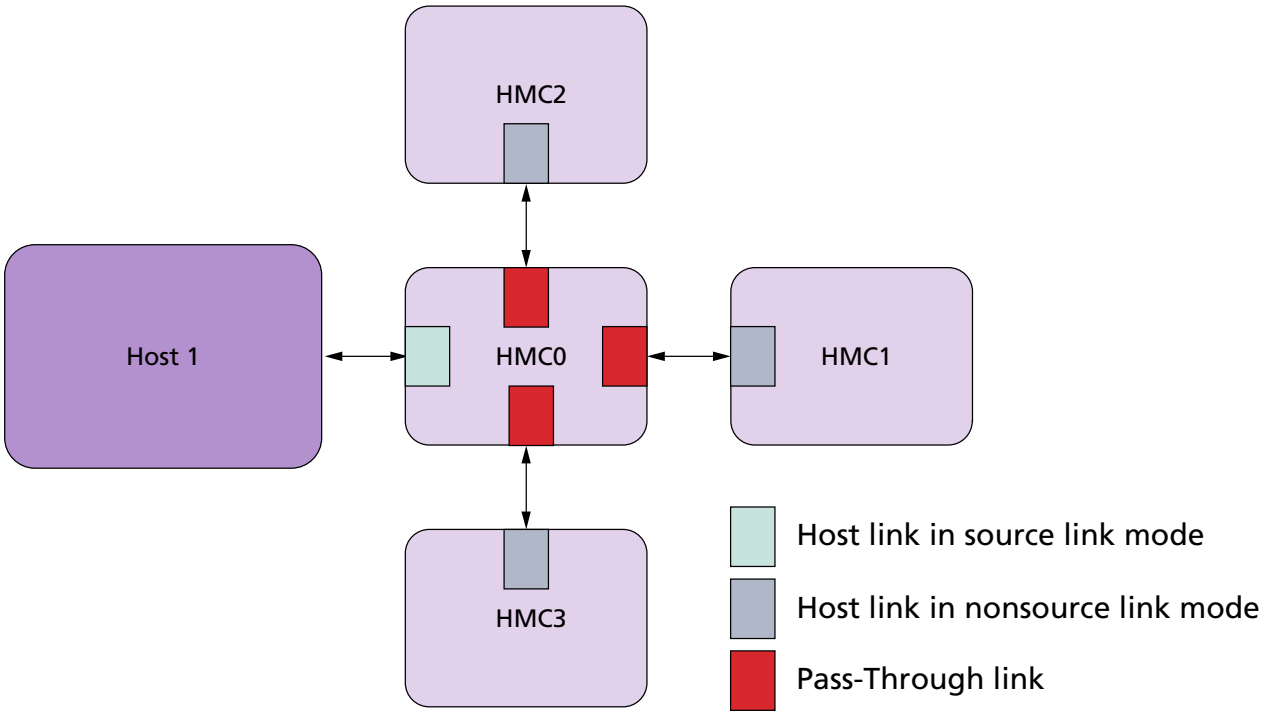
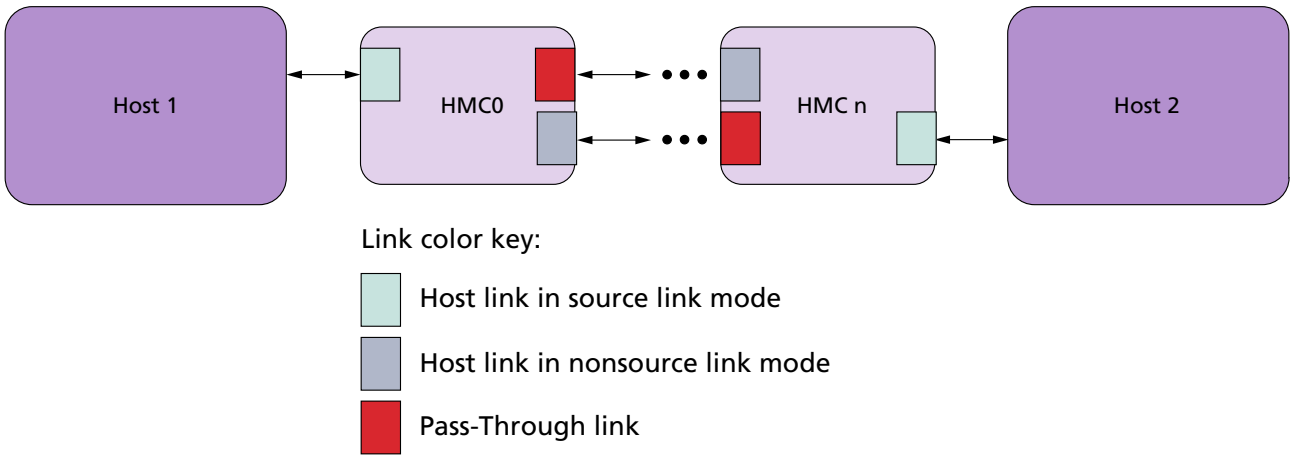
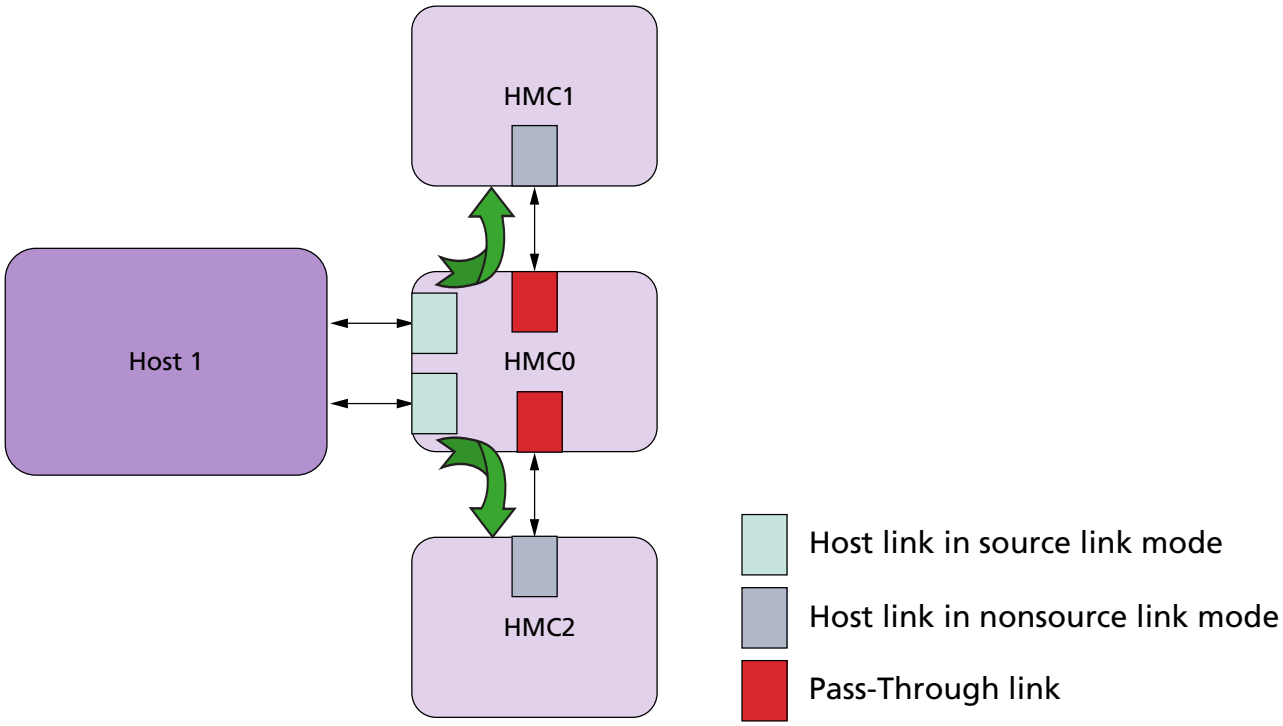


Figure 10: Multihost Serial Chain Topology



For this topology, n ≤7.

Figure 11: Modified Star Topology



Note: 1. The green arrows highlight the fact that each link does not have access to all three cubes. One link has access to HMC0 and HMC1 and the other link has access to HMC0 and HMC2.

Power-On and Initialization

HMC must be powered on and initialized in a predefined manner. All timing parameters specified in the following sequence can be found in the Initialization Timing Parameters table. HMC does not include or require time-out mechanisms during any step within the initialization routine. Registers indicating the status of HMC during initialization can be accessed via the I^2C or JTAG bus. All links on HMC may be initialized in parallel. The steps in this section referring to the initialization of a single link are applicable to all links of HMC being initialized at the same time.

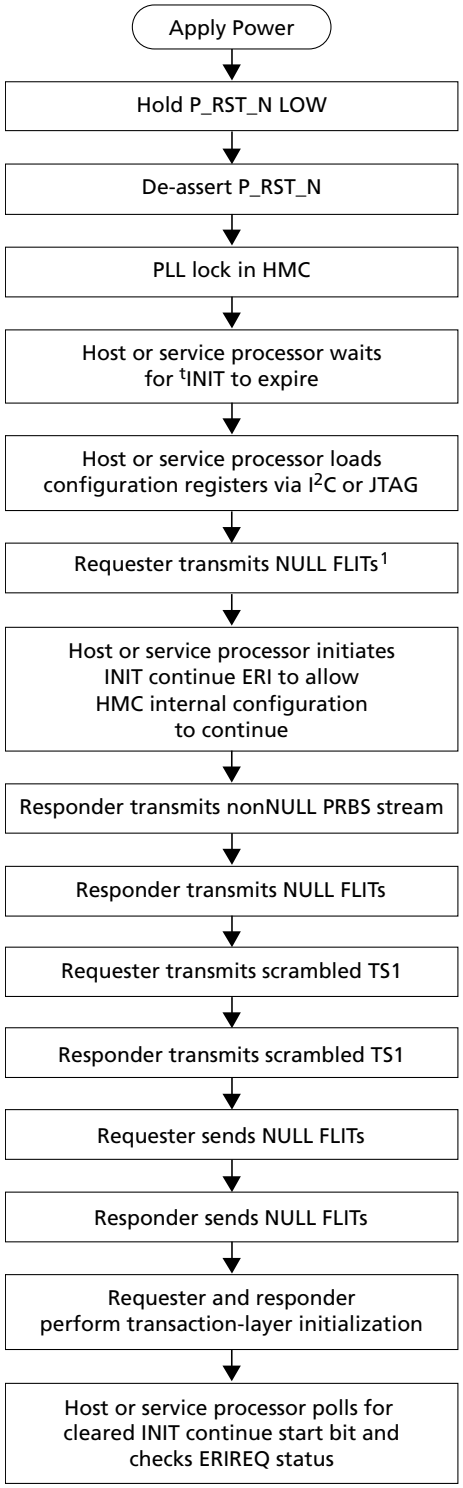
The following sequence is required for power-up:

1. Apply power (all supplies). V_{CCP} must ramp to full rail before the V_{DDM} ramp begins. All supplies must ramp to their respective minimum DC levels within ^{t}DD, but supply slew rates must not exceed V_{DVDT}.
2. Ensure that P_RST_N is held LOW (≤V_{SS} + 0.5V) during power ramp to ensure that outputs remain disabled (High-Z) and on-die termination (ODT) is off. Link transmit signals (LxTXP[n:0] and LxTXN[n:0]) will remain High-Z until Step 5 below. All other HMC inputs can be undefined during the power ramp with the exception of TRST_N, which must be held LOW. TRST_N will track the subsequent transition of P_RST_N if using the JTAG port.
3. After the voltage supplies and reference clocks (REFCLKP and REFCLKN) are within their specified operating tolerance, P_RST_N must be held LOW for at least ^{t}RST prior to beginning the initialization process. The LxRXPS signal for each link must be set for at least ^{t}IS prior to the de-assertion of the P_RST_N signal. If the link is active, the corresponding LxRXPS signal must be set HIGH (≥V_{DDK} – 0.5V). If a link is unused or disabled, the corresponding LxRXPS signal must be set LOW (≤V_{SS} + 0.5V).
4. After P_RST_N de-asserts HIGH (≥V_{DDK} - 0.5V), HMC initialization begins. P_RST_N must meet the minimum slew rate of $V_{RSTDVDT}$ = 0.1 V/ns as it transitions from LOW to HIGH. The following initialization steps are applicable for active links only.
5. Global HMC PLL lock occurs within ^{t}INIT. After ^{t}INIT has passed, the I^2C or JTAG bus is used to load the selected configuration registers. When complete, the INIT continue ERI must be run in order to allow internal configuration to continue. The requester must transmit a scrambled PRBS stream (may be NULL or non-NULL) coincident with or before running the INIT continue ERI. While there is no requirement on when the requester begins transmission of NULL FLITs, any delay in doing so after the INIT continue ERI begins execution will delay the training process and may cause the status bits of INIT continue to return a link status that indicates an error.
6. After the INIT continue ERI begins, HMC continues with internal initialization, including link PLL lock, receiver DFE training, and CDR clock recovery (which depends upon scrambled data supplied by the requester). The responder enters its start state and the transmit signals (LxTXP[n:0] and LxTXN[n:0]) begin to transmit a non-NULL PRBS stream to the requester. This stream is intended to allow the requester to achieve DFE training and CDR clock recovery, and also prohibit the requester from acquiring descrambler synchronization, which is not desired until Step 8.
7. To initialize the responder's descramblers, the requester enters the idle state and issues continuous scrambled NULL FLITs to the responder (see NULL Command for NULL description). The responder descrambler synchronization should occur within ^{t}RESP1 of the PLL locking. Upon descrambler synchronization, the res-

ponder will also enter the IDLE state and begin to transmit scrambled NULL FLITs.

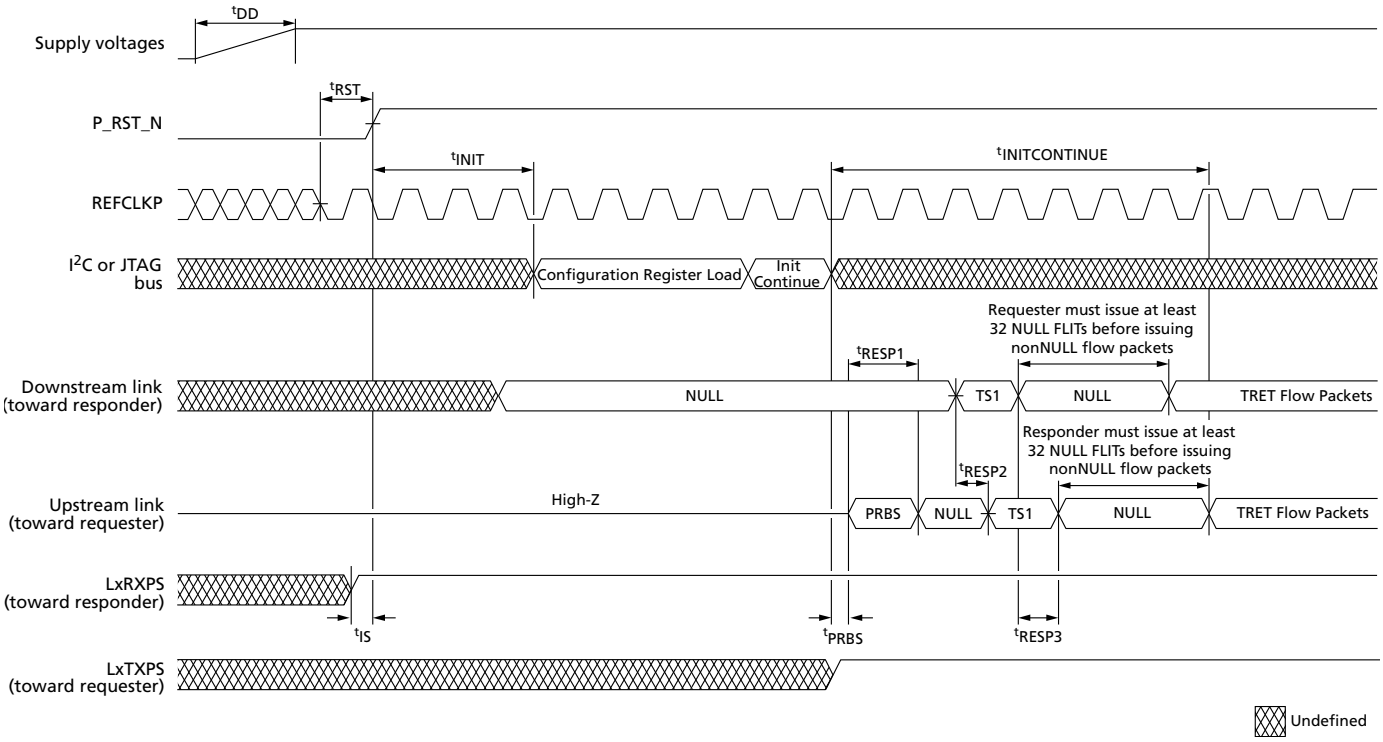
8. The responder issues continuous scrambled NULL FLITs, and the requester waits for these at its descrambler outputs. The purpose of this step is to achieve synchronization of the requester's descramblers.
9. After the requester has synchronized its descramblers, it enters the TS1 state and issues the scrambled TS1 training sequence continuously to the responder to achieve FLIT synchrony (see the TS1 Definition table for TS1 sequence). Responder link lock should occur within ^{t}RESP2.
10. After responder link lock occurs, it will enter the TS1 state and issue a series of scrambled TS1 training sequence back to the requester. The requester waits for one or more valid TS1 training sequence(s) at the local descrambler output. The requester must then align per-lane receiver timing to achieve host FLIT synchronization.
11. The requester stops sending TS1 training sequences to the responder after it has achieved link lock, whereby it then starts sending scrambled NULL FLITs. The requester must send a minimum of 32 NULL FLITs before sending the first non-NULL flow packet. Upon detecting the first NULL FLIT, the responder must enter the active state within a period of 32 FLITs, so that it can accept non-NULL flow packets after the requester has transmitted at least 32 NULL FLITs.
12. Upon detection of the first NULL FLIT after the TS1 sequence, the responder stops sending TS1 sequences and starts sending scrambled NULL FLITs back to the requester. The responder will transmit a minimum of 32 NULL FLITs before sending the first non-NULL flow packet. Upon detecting the first NULL FLIT, the requester must enter the active state within a period of 32 FLITs, so that it can accept non-NULL flow packets after the responder has transmitted at least 32 NULL FLITs. CRC errors on both ends of the link may be disregarded prior to becoming active. After becoming active, IRTRY packets must be held off until after sending 32 NULL FLITs. If the requester is running in response open loop mode, it should not set the RSOPENLOOP bit in the link configuration register until it is ready to accept an unsolicited error response packet that may occur before the first request is transmitted.
13. The link-layer initialization is complete. Both sides of the link will perform transaction layer initialization. The first TRET sent from HMC during the transaction layer initialization will have FRP = 1 and RRP = 0. Host commanded power-state transitions are supported at this point. The requester should not transmit TRET packets until it is able to receive transaction layer responses, including error response packets.

Figure 12: HMC Initialization Flowchart



Note: 1. The requester may begin transmission of NULL FLITs at any point prior to this step.

Figure 13: Initialization Timing



Notes:
1. Data on links is scrambled.
2. tINITCONTINUE represents the range of time that the INIT continue ERI function may take to complete. All aspects of link training on all active links must be completed before the INIT continue ERI completes, and therefore the minimum time should be used as the upper bound for link training time between the host and HMC. If a violation occurs, a link critical error status will be returned upon completion of INIT continue ERI.
3. Link receiver phase acquisition times vary based on the time needed to acquire decision feedback equalization (DFE) synchronization. Better signal integrity on the lanes will reduce synchronization time.
4. The host may send TRET flow packets prior to receiving TRET flow packets from HMC. However, the host must send a minimum of 32 NULL FLITs after completion of tRESP3. Alternatively, if the host receives TRET flow packets from HMC following tRESP3, it may send TRET flow packets at that time.
5. If the requester is running in response open loop mode, it should not set the RSOPENLOOP bit in the link configuration register until it is ready to accept an unsolicited error response packet that may occur before the first request is transmitted.

Power State Management

Each link can independently be set into a lower power state through the use of the power state management pins, LxRXPS and LxTXPS. Each of the links can be set into a low-power state, sleep mode, as well as a minimum power state called down mode.

Power is reduced when sleep mode is entered from normal operation (active mode) through the disablement of the link's high-speed SerDes circuitry. After a link has been initialized and is in active mode, the requester can transition its power state management pin, LxTXPS, from HIGH to LOW to put the opposite side of the link (responder)

into sleep mode. As it begins to enter sleep mode, the responder will transition its LxTXPS from HIGH to LOW within the ^{t}PST specified timing. Values for ^{t}PST and all other power state management related timing can be found in the Link Power Management Parameters table. Transition of the responder's LxTXPS will initiate the requester's RX and TX lanes to enter into sleep mode as well. Prior to entering sleep or down mode on a link, all link traffic must be quiesced such that all requests have received a response or ^{t}QUIESCE has passed since the last request. Any request or response packets still in flight may be lost. After packet transmission resumes following sleep or down mode exit, the sequence number (SEQ filed) must continue where it left off prior to entering sleep or down mode. This is true in both directions on the link.

Down mode allows a link to go to an even lower power state than sleep mode by disabling both the high-speed SerDes circuitry as well as the link's PLLs. A link enters down mode if its corresponding LxRXPS signal is LOW when the P_RST_N signal transitions from LOW to HIGH either during initialization or a reset event. HMC can be configured so that all of its links will enter down mode after the last link in active mode transitions to sleep mode. In this setting, any link already in sleep mode when the last link exits active mode will be transitioned to down mode to further reduce power consumption. Transitioning any links from sleep mode to down mode will take up to 150µs (represented by ^{t}SD specification).

When all links exit active mode and transition to down mode, HMC enters a down mode state. Although not accessible from the links, data stored within the memory is still maintained. Upon entering down mode state, HMC must stay in this state for a minimum of 1ms (^{t}DOWN). This is measured from the time HMC's last link transitions its LxRXPS LOW (entering self refresh) to its first LxRXPS to transition HIGH (entering active mode).

All links will independently respond to power state control; however, in the event of simultaneous power state transitions, HMC will stagger transitions between active mode and sleep modes in order to avoid supply voltage shifts. The amount of stagger time incurred is defined by the ^{t}SS specification. Multiply ^{t}SS by n - 1 to determine when the final link will transition states, where n equals the number of simultaneous link transitions. For example, if the requester sets its L0TXPS, L1TXPS, L2TXPS, and L3TXPS to 0 at the same time, it will take 1.5µs for HMC to transition all four links into sleep mode (^{t}SS × 3). This is also true for links exiting sleep mode and entering active mode. As an example, if L0RXPS and L1RXPS transition from LOW to HIGH at the same time, the second links will not start the transition to active mode until up to 500ns later (^{t}SS × 1).

The transition of a link to active mode from either sleep mode or down mode requires a re-initialization sequence as illustrated in Figure 14 (page 29).

Bringing a link from down mode into active mode requires additional time to complete the HSS PLL self-calibration as specified by ^{t}PSC (see Note 2 of Figure 14 (page 29)).

The transition of a pass-through link's LxTXPS signal is dependent on the power state transition of the host links within the same HMC:

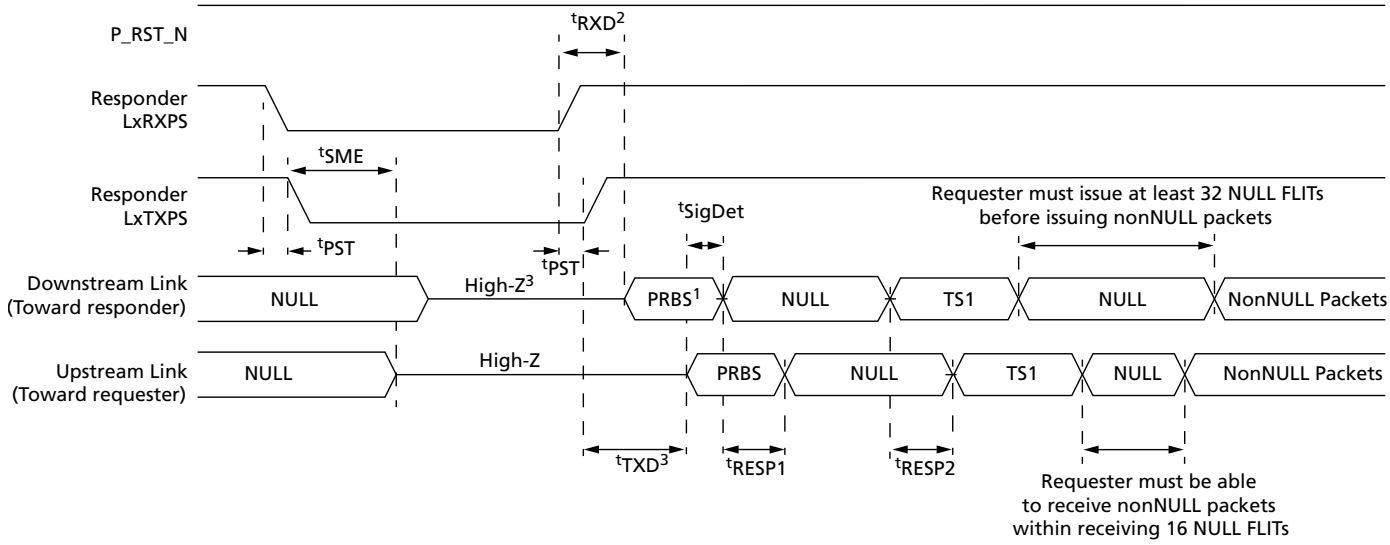
- A pass-through link's LxTXPS will transition LOW after the last host link in the same cube enter sleep mode.
- A pass-through link's LxTXPS will transition HIGH as soon as the LxTXPS of any host link in the same cube transitions HIGH.

Table 9: Link Power States and Conditions

Mode	Method of Entry	HMC TX Lane Status	HMC RX Lane Status	Link PLL Status
Active	Standard initialization	Enabled	Enabled	Enabled
Sleep	LxRXPS asserted LOW while in active mode	High-Z	Disabled	Enabled
Down	1) LxRXPS asserted LOW when P_RST_N transitions from LOW to HIGH; or 2) After the last link in active mode enters sleep mode (all other links already in sleep or down mode)	High-Z	Disabled	Disabled

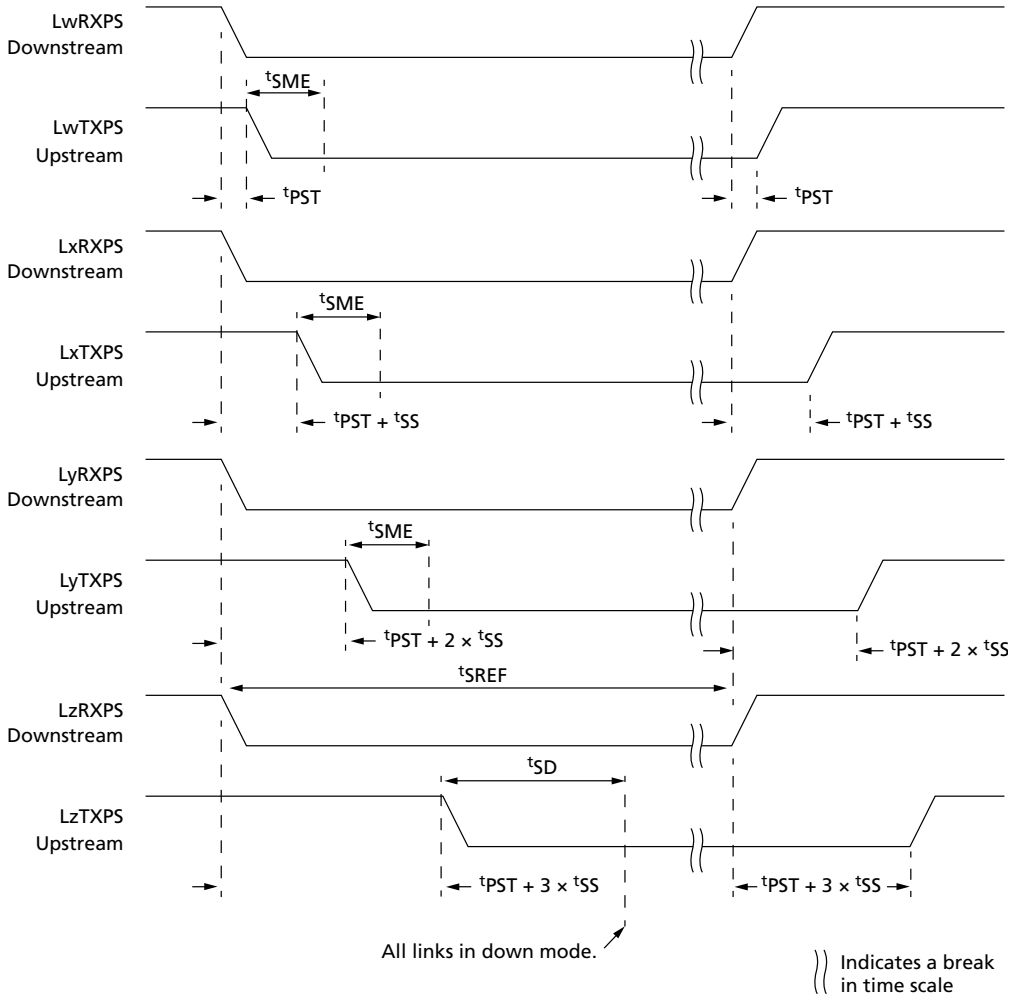
Notes:
1. Data from all vaults available through any link in active mode, even if other links are in sleep or down mode.
2. Down mode state is entered when all links exit active mode whereby data in HMC memory is retained but unavailable.
3. Down mode can be inhibited by use of the inhibit link down field of the link configuration register. See the HMC Gen2 Register Addendum for further details.

Figure 14: Sleep Mode Entry and Exit (Single Link Only)



Notes:
1. PRBS transmitted by HMC pass-through link; NULLs or PRBS may be transmitted from host.
2. tPSC must be added to this delay when exiting down mode.
3. The host must continue to issue NULL FLITs on the downstream link until the upstream link transmitters transition to High-Z. At that time, the host may transition to High-Z or continue to issue NULL FLITs.
4. For any link transitioning to sleep mode, all link traffic must be quiesced such that all requests have received a response or tQUIESCE has passed since the last request.

Figure 15: Simultaneous Transition of Four Host Links to Sleep Mode, Entry into Down Mode and Return to Active Mode (Single HMC, Four-Link Example)



Cold Reset and Power-Off Considerations

A cold reset is defined as asserting the P_RST_N LOW, effectively clearing out HMC configuration registers and the DRAM contents. HMC includes nonvolatile memory (NVM), which is used for storing the device firmware, DRAM repair information, and can include fatal error logs. Related to these last two things, it is critical that HMC never be powered off or cold reset while it is in the process of logging a repair or fatal error to the NVM. If this combination of events happen, it is possible that the NVM could get corrupted. To ensure that HMC is safe from NVM corruption during a power-off or cold reset event, the Shutdown ERI must be executed prior to powering off or cold resetting the device. After the Shutdown ERI command has successfully completed, no further operations will be supported and power may be removed, or the device cold reset. Note that the device NVM can only be written to during device operation if the NVM write disable register is set to 0 (enabling NVM write events). Refer to the HMC Register Addendum for more details about the Shutdown ERI command.

Link Layer

The link layer handles communication across the link not associated with the transaction layer. This consists of the transmission of NULL FLITs and flow packets.

All FLITs of a packet must be transmitted sequentially, without interruption, across the link. NULL FLITs are generated at the link layer when no other packets are being transmitted. A NULL FLIT is an all-zeros pattern that is scrambled prior to transmission. Any number of packets can be streamed back-to-back across the link, or they can be separated by NULL FLITs, depending upon system traffic and transaction layer flow control. There is no minimum or maximum requirement associated with the number of NULL FLITs transmitted between packets. NULL FLITs are not subject to flow control. The first nonzero FLIT following a NULL FLIT is considered to be the first FLIT of a packet. Data FLITs within a packet may be all zeros, but these lie between a header FLIT and a tail FLIT.

Flow packets are generated by the link master to pass flow control and retry control commands to the opposite side of the link. Flow packets are sent when no other link traffic is occurring or when a link retry sequence is to be initiated. Flow packets are single-FLIT packets with no data payload and are not subject to flow control. Flow packets utilize the same header and tail format as request packets, but are not considered requests, and do not have corresponding response commands. See Flow Commands for specifics on the commands transmitted with flow packets.

Transaction Layer

As noted previously, all information transmitted across the links is included in packets that consist of one or more FLITs. A packet always includes an 8-byte header at the beginning of the packet and an 8-byte tail at the end of the packet. The header includes the command field that identifies the packet type, other control fields, and when required, the addressing information. The tail includes flow and link-retry control fields along with the CRC field. There are three different packet types. The first two, request and response packets, are used at the transaction layer. The third type, the flow packet, is only used at the link layer, however it is included below so that all packet types can be described within a single section.

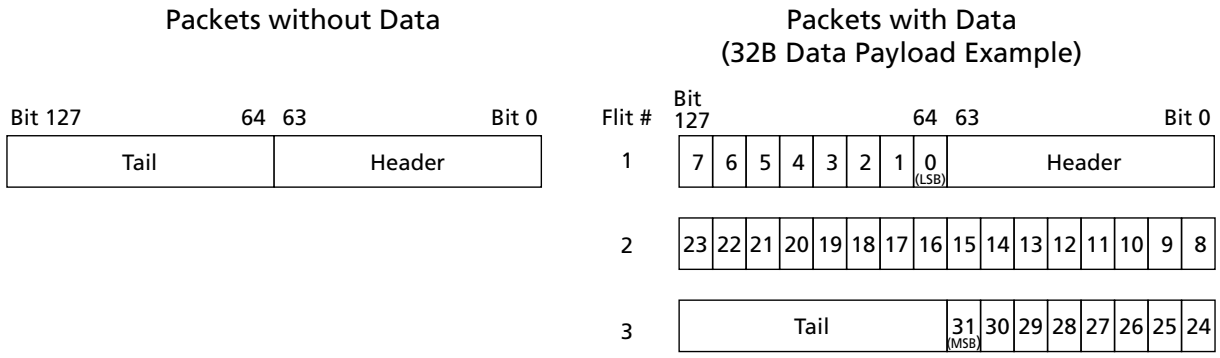
1. Request packets are issued by the requester (host or HMC configured as a pass-through link). The request packet header includes address information to perform a READ or WRITE operation. Write request packets include data FLITs.
2. Response packets are issued by the responder (HMC configured as a host link). Response headers do not include address information. READ responses include data FLITs. Write response packets do not include data FLITs.
3. Flow packets are not part of the transaction layer protocol and are not subject to flow control. They are only used at the link to pass flow control and retry control information back to the link master when normal transaction layer packets are not flowing in the return direction. Flow packets are generated at the link master and are decoded and then dropped at the link slave; they are not forwarded and do not pass through HMC.

Each packet type has its own defined header and tail formats, described in the Request Packets, Response Packets, and Flow Packets sections.

Packets that consist of multiple FLITs are transmitted across the link with the least significant FLIT number first. The first FLIT includes the header and the least significant

bits (LSBs) of the data. Subsequent data FLITs progress with more significant data following. The figure below illustrates the layout for packets with and without data.

Figure 16: Packet Layouts



Note: 1. Each numbered data field represents a byte with bit positions [7(MSB): 0(LSB)].

A CRC field is included in the tail of every packet that covers the entire packet, including the header, all data, and the nonCRC tail bits. The link retry logic retransmits packets across the link if link errors are detected, as described in the Link Retry section. Upon successful transmission of packets across the link, the packets are transmitted through the logic base all the way to the destination vault controller. CRC provides command and data integrity checking to the destination vault controller. For the remaining path between the vault controller and the DRAM arrays, parity bits are used to maintain the integrity of the command and address values, and Error Correction and Control (ECC) is used to maintain the integrity of the data. The section titled Packet Integrity, Parity, ECC and Scrubbing describes this in greater detail. Thus, data is protected along the entire path to and from the memory device. If the vault controller detects a correctable data error on a read cycle, error correction is executed before the read response is returned to the host. If an uncorrectable data error is detected during a read request, the read response packet will include: Data Invalid (DINV) flag asserted, the Error Status (ERRSTAT) field set to indicate the multiple uncorrectable error (MUE), and the corrupted data in the packet body.

The CRC algorithm used on HMC is the Koopman CRC-32K. This algorithm was chosen for HMC because of its balance of coverage and ease of implementation. The polynomial for this algorithm is:

$$x32 + x30 + x29 + x28 + x26 + x20 + x19 + x17 + x16 + x15 + x11 + x10 + x7 + x6 + x4 + x2 + x + 1$$

The CRC calculation operates on the LSB of the packet first. The packet CRC calculation must insert 0s in place of the 32 bits representing the CRC field before generating or checking the CRC. For example, when generating CRC for a packet, bits [63: 32] of the Tail presented to the CRC generator should be all zeros. The output of the CRC generator will have a 32-bit CRC value that will then be inserted in bits [63:32] of the Tail before forwarding that FLIT of the packet. When checking CRC for a packet, the CRC field should be removed from bits [63:32] of the Tail and replaced with 32 bits of zeros, then presented to the CRC checker. The output of the CRC checker will have a 32-bit CRC val-

ue that can be compared with the CRC value that was removed from the tail. If the two compare, the CRC check indicates no bit failures within the packet.

The two tables that follow represent a single FLIT packet and a multi-FLIT packet, respectively. These examples illustrate how CRC is generated.

Table 10: Single FLIT Example: TRET Packet

UI	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Hex Value
0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0x0882
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0x0100
5	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0xf801
6	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0xa78a
7	0	1	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0x5e91

Table 11: Multi-FLIT Example: 2ADD8 Request Packet

UI	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Hex Value
0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0x1112
1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0x801D
2	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	1	0x282B
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
4	1	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0xE664
5	0	0	1	1	0	1	0	0	1	0	1	0	1	1	1	0	0x34AE
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
8	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0xBF91
9	0	0	1	1	0	0	1	1	1	0	0	0	1	1	0	0	0x338C
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000
12	0	1	0	0	1	0	1	1	1	1	0	0	0	0	1	0	0x4BC2
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0x0002
14	1	1	1	1	1	0	0	0	1	1	0	1	0	1	0	0	0xF8D4
15	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0x6418

Overall packet lengths are from 1 to 9 FLITs. Write request packets and read response packets will contain from 16B to 128B of data between the header and the tail.

All requests are operationally closed as follows:

- Request packets are uniquely identified with a tag field embedded in the packet.
- All read request packets have a corresponding read response packet returned to the requesting link that includes the corresponding request tag and requested data.

- Normal write request packets are acknowledged with an explicit write response packet that includes the write request's tag.
  - This positive acknowledgment provides an indication to the requesting processor that the request has been executed without error.
- Because HMC does not use the TAG field of posted write requests, the host can populate this field with any value.
- If unrecoverable errors occur in the request path, they are captured in the error status that is included in the response packet.
- If unrecoverable errors occur in the request path for posted write requests, an error response packet is returned to the requester indicating an error.

Flow packets are not considered a request, they do not have a valid tag, and they do not have a corresponding response packet.

The flow of request packets and response packets are managed with tokens, described in the Packet Flow Control section. Flow packets have no flow control and can be sent at any time by the link master.

Memory Addressing

A request packet header includes an address field of 34 bits for internal memory addressing within HMC. This includes vault, bank, and DRAM address bits. The configuration currently defined provides a total of 2GB of addressable memory locations within one HMC. This requires the lower 31 address bits of the 34-bit field; the upper 3 bits are for future expansion and are ignored by HMC.

The user can select a specific address mapping scheme so that access of HMC's vaults and banks is optimized for the characteristics of the request address stream. If the request address stream is either random or generally sequential from the low-order address bits, the host can choose to use one of the default address map modes offered in HMC (see the Default Address Map Mode Table). The default address mapping is based on the maximum block size chosen in the address map mode register. It maps the vault address toward the less significant address bits, followed by the bank address just above the vault address. This address mapping algorithm is referred to as "low interleave," and forces sequential addressing to be spread across different vaults and then across different banks within a vault, thus avoiding bank conflicts. A request stream that has a truly random address pattern is not sensitive to this specific method of address mapping.

Memory Addressing Granularity

HMC supports READ and WRITE data block accesses with 16-byte granularity from 16B to the value of the maximum block size setting (32B, 64B, or 128B). The maximum block size is set with the Address Configuration register. See Data Access Performance Considerations for information on how to select maximum block size such that it optimizes performance in a given system. The maximum block size specified in the address map mode register determines the number of bits within the byte address field, of which the four LSBs are ignored due to the 16-byte granularity (an exception is the BIT WRITE command). The 4 LSBs of the byte address may be used for future density expansion if 16B block aligned access is preserved. Details for address expansion are outside the scope of this specification version. The remaining upper bits of the byte address indicate the starting 16-byte boundary when accessing a portion or all of the block. The vault controller uses these upper bits of the byte address as part of its DRAM column addressing. If the starting 16-byte address in conjunction with the data access length of the request causes the access to go past the end of the maximum block size boundary,

the DRAM column addressing will wrap within the maximum block size and continue. For example, if the maximum block size is 64 bytes and a 48-byte READ command is received starting at byte 32, the DRAM access will occur starting at byte 32 up to byte 63, then continue at byte 0 up to byte 15.

Memory Address-to-Link Mapping

Each link is associated with four local vaults that are referred to as a quadrant. Access from a link to a local quadrant may have lower latency than to a link outside the quadrant. Bits within the vault address designate both the specific quadrant and the vaults within that quadrant.

Table 12: Addressing Definitions

Address	Description	Comments
Byte address	Bytes within the maximum supported block size	The four LSBs of the byte address are ignored for READ and WRITE command requests (with the exception of BIT WRITE command; See BIT WRITE Command for details)
Vault address	Addresses vaults within HMC	Lower two bits of the vault address specifies 1 of 4 vaults within the logic chip quadrant
		Upper two bits of the vault address specifies 1 of 4 quadrants
Bank address	Addresses banks within a vault	Addresses 1 of 8 banks in the vault
DRAM address	Addresses DRAM rows and column within a bank	The vault controller breaks the DRAM address into row and column addresses, addressing 1Mb blocks of 16 bytes each

Default Address Map Mode Table

Table 13: Default Address Map Mode Table

Request Address Bit	2GB		
	32-Byte Max Block Size	64-Byte Max Block Size	128-Byte Max Block Size
33	Ignored	Ignored	Ignored
32	Ignored	Ignored	Ignored
31	Ignored	Ignored	Ignored
30	DRAM[19]	DRAM[19]	DRAM[19]
29	DRAM[18]	DRAM[18]	DRAM[18]
28	DRAM[17]	DRAM[17]	DRAM[17]
27	DRAM[16]	DRAM[16]	DRAM[16]
26	DRAM[15]	DRAM[15]	DRAM[15]
25	DRAM[14]	DRAM[14]	DRAM[14]
24	DRAM[13]	DRAM[13]	DRAM[13]
23	DRAM[12]	DRAM[12]	DRAM[12]
22	DRAM[11]	DRAM[11]	DRAM[11]
21	DRAM[10]	DRAM[10]	DRAM[10]
20	DRAM[9]	DRAM[9]	DRAM[9]
19	DRAM[8]	DRAM[8]	DRAM[8]
18	DRAM[7]	DRAM[7]	DRAM[7]
17	DRAM[6]	DRAM[6]	DRAM[6]
16	DRAM[5]	DRAM[5]	DRAM[5]
15	DRAM[4]	DRAM[4]	DRAM[4]
14	DRAM[3]	DRAM[3]	DRAM[3]
13	DRAM[2]	DRAM[2]	Bank[2]
12	DRAM[1]	Bank[2]	Bank[1]
11	Bank[2]	Bank[1]	Bank[0]
10	Bank[1]	Bank[0]	Vault[3]
9	Bank[0]	Vault[3]	Vault[2]
8	Vault[3]	Vault[2]	Vault[1]
7	Vault[2]	Vault[1]	Vault[0]
6	Vault[1]	Vault[0]	Byte[6], DRAM[2]
5	Vault[0]	Byte[5], DRAM[1]	Byte[5], DRAM[1]
4	Byte[4], DRAM[0]	Byte[4], DRAM[0]	Byte[4], DRAM[0]
3	Byte[3] = Ignored	Byte[3] = Ignored	Byte[3] = Ignored
2	Byte[2] = Ignored	Byte[2] = Ignored	Byte[2] = Ignored
1	Byte[1] = Ignored	Byte[1] = Ignored	Byte[1] = Ignored
0	Byte[0] = Ignored	Byte[0] = Ignored	Byte[0] = Ignored

Address Mapping Mode Register

The address mapping mode register provides user control of mapping portions of the ADRS field within the header of request packets to the vault and bank addresses. When using default address mapping, the address mapping mode field should be set based on specific system requirements. See Data Access Performance Considerations for details.

In addition to the default address mapping algorithms, user-defined fields are provided for the user to specify a unique mapping algorithm. There is a user-defined field for the vault address and the bank address. Each field holds a 5-bit encoded value pointing to the corresponding bit position in the ADRS field of the request packet header. This represents the LSB that will be used for the vault or bank address, with the more significant bits of the subaddress coming from the bit positions immediately above it in the request header ADRS field. When operating with the user-defined modes, it is the user's responsibility to set the user-defined encoded fields so that the vault and bank addresses do not overlap each other or overlap the byte address within the request header ADRS field. The byte address is always fixed at the LSBs of the request header ADRS field. The user must also be aware that HMC size determines the number of bits in the bank address. The remaining bits of the request ADRS field not specified as the vault and bank addresses become the DRAM address.

The address map mode register is writable via the MODE WRITE command or via the maintenance interface (I^2C or JTAG bus). Because the MAX block size is defaulted to 64 bytes, the host boot block code must either be compatible (issue requests of 64 bytes or smaller) or issue a MODE WRITE command to change the MAX block size.

DRAM Addressing

Each HMC bank contains 16,777,216 memory bytes (16MB). A WRITE or READ command to a bank accesses 32 bytes of data for each column fetch. The bank configuration is identical in all specified HMC configurations.

Packet Length

As noted previously, packets are always multiples of 16-byte FLITs. Within each packet header there is a length (LNG) field that indicates the total number of FLITs in the packet. A packet that contains no data FLITs would have LNG = 1; this represents the single FLIT holding the 8-byte header and 8-byte tail. When generating response packets, the necessary data size is determined from the CMD field in the corresponding request packet. Specific packet lengths are provided in the following tables: Transaction Layer – Request Commands, Transaction Layer – Response Commands, and Flow Commands.

A duplicate length (DLN) field is included within each packet header to provide additional packet integrity. The value of DLN must equal that of LNG within a packet. The DLN field is compared to the LNG field prior to the CRC being checked. Note that LNG enables the packet parser to determine the location of the CRC field (where a value of 1 points to the current FLIT). A miscompare of DLN and LNG is treated like a CRC check and will trigger the link retry mechanism. If LNG does not equal DLN when the packet is generated at the host, link retries will be unsuccessful and a fatal error will result.

Packet Flow Control

Flow control occurs in both directions on each link. The flow of transaction layer packets is managed with token counts. Each token represents the storage to hold one FLIT (16 bytes). The link slave input buffer temporarily stores transaction layer packets as

they are received from the link. (Flow packets are not stored in the input buffer, and therefore, are not subject to flow control.) The minimum size of this buffer must be equal to or greater than the number of FLITs in a single packet of the largest supported length, but in general can hold several of the largest packets to enable a constant flow across the link. The available space in this input buffer is represented in a token count register at the link master. This gives the link master knowledge of the available buffering at the other end of the link and the ability to evaluate whether there is enough buffer space to receive every FLIT of the next packet to be transmitted.

The token count register is loaded during the initialization sequence with the maximum number of tokens representing the available buffer space at the link slave when it is empty. As the link master sends each transaction layer packet across the link, it must decrement the token count register by the number of FLITs in the transmitted packet. As the packet is received and stored in the input buffer, its FLITs use up the space represented by the decremented value of token count register at the link master. As each packet is read out of the input buffer, it is forwarded to a destination, and its FLIT location is freed up. This requires a mechanism to return packet tokens to the link transmitter, using the opposite link direction. The input buffer control logic sends a count (representing the number of FLITs that were read out of the input buffer when the packet was forwarded) to the local (adjacent) link master. This count is returned in the return token count (RTC) field of the next possible packet traveling in the opposite direction on the link. At the link slave, the RTC field is extracted from the incoming packet and sent back to the original link transmitter where its value is added to the current value of the token count register. If no transaction layer packet traffic is occurring in the return direction, the link master must create a flow packet known as a token return (TRET) to return the token. Not returning tokens can lead to performance degradation or stalling. TRET is described in TOKEN RETURN (TRET) Command.

Appropriate sizing of the link slave input buffer requires weighing the trade-offs between latency, throughput, silicon real estate, and routability. If there is a temporary pause in the packet forwarding priority at the output of the input buffer (internal switch conflicts, or destination vault flow control), a packet might not be emptied from the input buffer. This would cause a pause in the return of the tokens. As long as this is infrequent and the input buffer is sized to accommodate it, the packet flow will not be disrupted unless the link is running at 100% busy. If the input buffer is empty, a specific implementation may choose to bypass the input buffer and forward a packet immediately to its destination. In this case, the tokens for the packet would be immediately returned to the link master.

As noted previously, the link master must not transmit a packet if there is insufficient space in the input buffer at the other end of the link. The LNG field within the header of each outgoing packet must be compared to the token count register to determine whether the packet should be transmitted or the packet stream should be paused.

Token Return Loop Time

As mentioned in the Packet Flow Control section, the link input buffer in the link slave provides buffering of received FLITs while tokens are being returned to the transmitter. While tokens are being returned additional FLITs are being transmitted and filling the link input buffer at a rate up to the maximum link bandwidth. An adequately sized link input buffer will have room to buffer enough FLITs to accommodate the time it takes for the token to return to the transmitting link master and allow it to transmit more FLITs. This is referred to as the "token return loop time."

The total delay for the token return loop time starts from the point in the link master where it is determined that there are enough tokens to transmit the next packet to the link; the first transmitted FLIT includes the following delays:

- Logical and physical block serialization time at the transmitting end of the link
- Physical and logical block deserialization time at the receiving end of the link
- The time for the link slave to forward the received FLIT and generate a token for the FLIT
  (Note that the link input buffer is empty and the FLIT does not get queued behind a previously transmitted FLIT. Any additional queuing in the link input buffer does not get added to the token return loop time.)
- Transfer of the token from the link slave to the local link master
- Time to wait in the link master from just missing embedding the token into the RTC field of the previous packet and waiting for the next tail of the longest packet
- Logical and physical serialization time for the packet with the embedded token in the transmitting link master
- Physical and logical deserialization time for the packet with the embedded token in the receiving link slave
- Time taken for this longest packet's tail to be received in the link slave, to the point of CRC check and token extraction from the RTC field
- Transfer of the extracted token from the link slave to the local link master (the original transmitter in the first delay above)
- Time taken to increment the token counter
- Time taken to determine that the token counter has enough tokens to allow the next packet to be transmitted (which is the same point where the loop started)

In order to maintain link performance, the maximum allowable delay of this token return loop time must be less than the amount of time it takes to fill the link input buffer. If the actual token return loop time is larger than the time it takes to fill the link input buffer, the requester will become starved for tokens, causing the packet stream to be throttled and resulting in a drop of bandwidth across the link.

There are two token return loops, one beginning in the host's link master, which corresponds to HMC's link input buffer size, and the other beginning in HMC's link master, which corresponds to the host's link input buffer size.

HMC link input buffer size (in FLITs) is controlled by the starting number of tokens specified in the Link Input Buffer Max Token Count field in the Input Buffer Token Count register. This determines the usable depth of the input buffer and therefore determines the maximum amount of time available for token return loop time. The link input buffer in HMC has a maximum usable size of 228 FLITs. However, there must be sufficient input buffer depth reserved for one poisoned packet (poisoned packets are described in the Packet Integrity section) in the case of a link retry. If this space is not reserved, there is the potential for a link input buffer overrun if a link retry does occur. For example, if the maximum block (packet) size available to the host is 128B (9 FLITs), the maximum usable tokens are 219 (228 - 9) FLIT locations. This is the space to be used and tracked with tokens by the requester. This maximum allowable size may be reduced by initializing the Link Input Buffer Max Token Count field (within the Input Buffer Token Count register) to a value smaller than 219 during HMC initialization. There are latency and bandwidth trade offs that can be made with this setting by increasing or decreasing the Input Buffer Token Count, but the optimal setting is dependent upon many factors such as: memory utilization, link rate, number of links used, and host token and

pointer return latencies. The optimal setting can be derived by simulating specific use cases using HMC performance model in the appropriate system simulation environment.

HMC token return delay represents the elapsed time from when the first bit of a FLIT is received, the token for that FLIT is generated (when the link input buffer is empty), the token is embedded into a single-FLIT packet, and the first bit is transmitted out of HMC.

Table 14: Token Return Delay for HMC Link Input Buffer

Link			219-FLIT HMC Input Buffer Full Period (ns)[1]	HMC Token Return Delay (ns)[2]	Host Allowable Delay (ns)
Bit Rate	Link Width	ps/FLIT			
15 Gb/s	Full	533	116.7	19.2	97.5
15 Gb/s	Half	1066	233.4	20.8	212.8
12.5 Gb/s	Full	640	140.2	22.4	117.8
12.5 Gb/s	Half	1280	280.4	24	256.3
10 Gb/s	Full	800	175.2	23.2	152
10 Gb/s	Half	1600	350.4	24.8	325.6

Notes:
1. If the link input buffer size is reduced with a value smaller than 219 in the Link Input Buffer Max Token Count field of the Input Buffer Token Count register, the buffer full period can be recalculated as: Buffer full period in ns = (ps/FLIT × Link Input Buffer Max Token Count)/1000
2. The following additional delays may be added to the HMC token return delay values shown in the table if multi-FLIT packets are in flight on the return link:
   2–4 FLIT packet: 1.6ns (3.2ns for half-width link)
   5–7 FLIT packet: 3.2ns (6.4ns for half-width link)
   8–9 FLIT packet: 4.8ns (9.6ns for half-width link)

HMC token update delay is the elapsed time from when the first bit of a single-FLIT packet is received until the host token from the RTC field of that FLIT is extracted, the token counter in the link master is updated, the next FLIT to be transmitted to the SerDes TX is allowed, and the SerDes TX time to the first bit of the FLIT is transmitted.

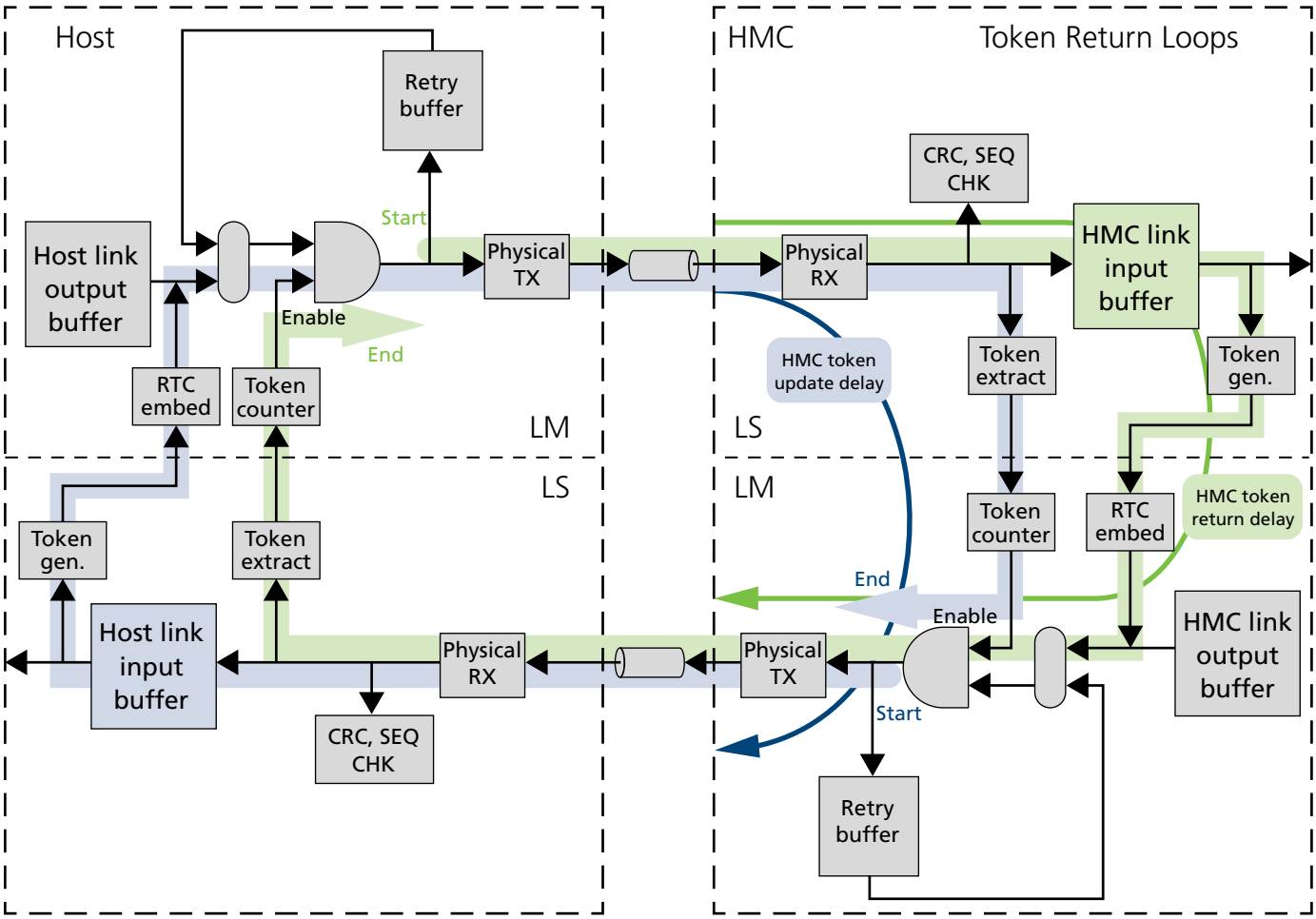
Table 15: HMC Token Update Delay for Host Link Input Buffer

Link			HMC Token Update Delay (ns)[1]
Bit Rate	Link Width	ps/FLIT	
15 Gb/s	Full	533	19.2
15 Gb/s	Half	1066	20.8
12.5 Gb/s	Full	640	22.4
12.5 Gb/s	Half	1280	24
10 Gb/s	Full	800	23.2
10 Gb/s	Half	1600	24.8

Note:
1. The following additional delays may be added to the HMC token update delay values shown in the table if multi-FLIT packets are in flight on the return link:
   2–4 FLIT packet: 1.6ns (3.2ns for half-width link)

5–7 FLIT packet: 3.2ns (6.4ns for half-width link)
8–9 FLIT packet: 4.8ns (9.6ns for half-width link)

Figure 17: Token Return Loops



Tagging

Operational closure for requests is accomplished using the tag fields in request and response packets. The tag value remains associated with the request until a response is received indicating that the request has been executed. Tags in READ requests are returned with the respective read data in the read response packet header. Write response tags are returned within a write response packet. In the case of posted write requests, because no response is returned and HMC does not use the TAG field of posted write requests, the host can populate this field with any value. When multiple host links are connected to HMC, each response packet will be returned on the same link as its associated request. This keeps the tag range independent for each host link.

Tag fields in packets are nine bits long, which is enough space for 512 tags. All tags are available for use. Tag assignment and reassignment are managed by the host. There is no required algorithm to assign tag values to requests. HMC does not use the tag for in-

ternal control or identification, only for copying the tag from the request packet to the response packet.

Tags are assigned by control logic at the host link master and must not be used in another request packet until a response tag with the same tag number is returned to that host link. Other host links will use the same tag range of 512 tags, but they are uniquely identified by their association with each different host link. Thus, the total maximum number of unique tags is 512 times the number of host links. For example, if four links are connected from HMC to the host, there are 2048 usable tags for requests to HMC. As an implementation example, a host control logic might decide to provide a time-out capability for every transaction to determine whether a request or response packet has been lost or corrupted, preventing the tag from being returned. The timer in this implementation example would consider link retry periods that occur to account for response packets that are delayed, but still returned. Variability in response time due to transaction mix would also need to be considered when determining an appropriate timeout value.

Packet Integrity, Parity, ECC, and Scrubbing

There are several methods used to check for errors and protect the integrity of the data both in flight and while at rest within HMC. Integrity is maintained throughout by use of:

- CRC for the packets on the links between host and HMC
- CRC between the link slave/link master and the vault controller
- Parity for the command and address between the vault controller and the DRAM array
- ECC for the data between the vault controller and the DRAM array
- Scrubbing for data at rest in the DRAM array

These are all interrelated; for example if there is a parity error detected on the command or address signals, the vault controller can intentionally poison the CRC in the response packet. See the Parity section for more details.

Packet Integrity

As previously described in the Transaction Layer section, the integrity of the packet contents is maintained with the CRC field in the tail of every packet. Because the entire packet (including header and tail) is transmitted from the source link all of the way to the destination vault, CRC is used to detect failures that occur not only on transmission across the link, but along the entire path. CRC may be regenerated along the path if flow control fields within the header or tail change. CRC regeneration is done in an overlapped fashion with respect to a CRC check to ensure that no single point of failure will go undetected. There may be cases when the first FLITs of a packet are forwarded before the tail is received and the CRC is checked. This occurs in HMC's link slave after a request packet crosses a link and is done to avoid adding latency in the packet path. If the packet is found to have a CRC error as it passes through the link slave, the packet is intentionally poisoned by the link slave so that its destination (in this case a vault controller) will recognize it as corrupted and will not use it. The link slave poisons the packet by inverting a recalculated value of the CRC and inserting that into the tail in place of the previously calculated CRC. This resulting modified packet is known as a poisoned packet.

Parity

Parity is used to maintain the integrity of the command and address bits between the vault controller and the DRAM. After deconstructing a packet that was received from the link slave, the vault controller will calculate and transmit parity bits associated with the command and address lines to the DRAM. The DRAM will check these parity bits, and signal to the vault controller if there is a parity error. For example, if the DRAM access is a read, a read response packet may be generated for transmission back to the link master when a parity error is detected on the command or address bits. As a result of the parity error, the vault controller will poison the read response packet (see Poisoned Packets), and transmit it back to the host which should drop the packet. If enabled, the vault controller will retry the DRAM request in parallel to this, and upon a successful DRAM access, a valid read response packet will be generated and transmitted to the requester. If the retried access is not successful, the command execution is halted until a reset is performed and a Vault Critical Error ERRSTAT will be included in an error response packet.

If a packet travels across a link after it is poisoned, a link master will still be able to embed flow control fields in the packet by recalculating the CRC with the previously embedded flow control values, then inverting the recalculated CRC so that the poisoned state will be maintained. Because the flow control fields are valid in a poisoned packet, it is stored in the retry buffer. In this case, the link slave on the other end of the link will recognize the poisoned CRC and will still extract the flow control fields. Whenever a packet is poisoned, the CMD, ADRS, TAG, and ERRSTAT fields are not valid, but the flow control fields (FRP, RRP, RTC) are valid.

ECC and Scrubbing

Data is protected in memory using ECC. ECC will provide correction of single-bit errors (SBE) as well as multi-bit correctable errors (MCE) which is categorized as all data bits that are transmitted over a single failed TSV between the memory and the vault controller. Memory locations are initialized with correct ECC data patterns before the host begins using the locations. This eliminates a write-before-read requirement to avoid SBEs and multiple unrecoverable errors (MUE).

Scrubbing is used inside the cube to mitigate soft failures in memory. Two categories of scrubbing are provided in each vault controller:

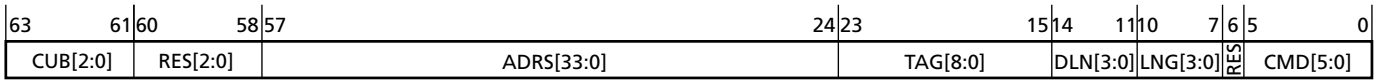
Demand scrubbing: Performed during read requests. If the vault controller detects a correctable SBE or MCE the data is corrected and returned to the host in the response packet. When this takes place, a scrubbing cycle is invoked that writes the corrected data back into the memory location that required correction. If an MUE is found such that the data is not correctable, the vault controller will assert the DINV field and setting the MUE ERRSTAT in the tail of the read response packet.

Patrol scrubbing: Performed by HMC refresh logic in the vault controller. If a correctable SBE or MCE is detected, a scrubbing cycle is invoked that writes the correct data back into the memory location immediately. If an MUE is found during patrol scrubbing, an unsolicited error packet will be sent with the MUE ERRSTAT in the tail.

Request Packets

Request packets carry request commands from the requester (host or HMC link configured as pass-through link) to the responder (HMC link configured as host link). All request packets are subject to normal packet flow control.

Figure 18: Request Packet Header Layout



Request packet headers for request commands and flow commands contain the fields shown in the table below. Specific commands may require some of the fields to be 0. These are provided in the Valid Field and Command Summary Table.

Table 16: Request Packet Header Fields

Name	Field Label	Bit Count	Bit Range	Function
Cube ID	CUB	3	[63:61]	CUB field is used to match requests with the target HMC. Each HMC's cube ID register defines the target device's ID for matching with the CUB field in the request header and defaults to the value set by the CUB pins of each HMC device.
Reserved	RES	3	[60:58]	Reserved: These bits are reserved for future address or cube ID expansion. The responder will ignore bits in this field from the requester except for including them in the CRC calculation. HMC can use portions of this field range internally.
Address	ADRS	34	[57:24]	Request address. For some commands, control fields are included within this range.
Tag	TAG	9	[23:15]	Tag number uniquely identifying this request.
Duplicate length	DLN	4	[14:11]	Duplicate of packet length field.
Packet length	LNG	4	[10:7]	Length of packet in FLITs (1 FLIT is 128 bits). Includes header, any data payload, and tail.
Reserved	RES	1	[6]	Reserved: The responder will ignore this bit from the requester except for including it in the CRC calculation. HMC can use this field internally.
Command	CMD	6	[5:0]	Packet command. (See the Transaction Layer – Request Commands table for the list of request commands.)

Figure 19: Request Packet Tail Layout

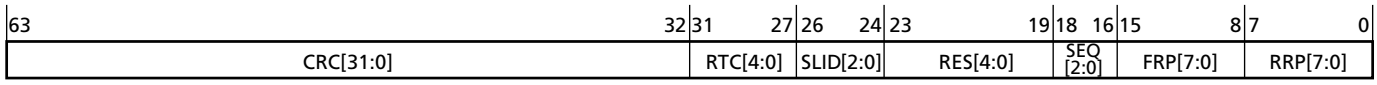


Table 17: Request Packet Tail Fields

Name	Field Label	Bit Count	Bit Range	Function
Cyclic redundancy check	CRC	32	[63:32]	The error-detecting code field that covers the entire packet.

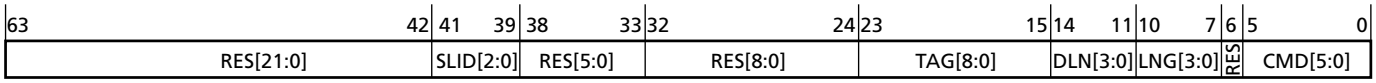
Table 17: Request Packet Tail Fields (Continued)

Name	Field Label	Bit Count	Bit Range	Function
Return token count	RTC	5	[31:27]	Return token count for transaction-layer flow control. In the request packet tail, the RTC contains the tokens that represent available space in the requester's input buffer.
Source Link ID	SLID	3	[26:24]	Used to identify the source link for response routing. The physical link number is inserted into this field by HMC. The host can ignore these bits with the exception of including them in the CRC calculation.
Reserved	RES	5	[23:19]	Reserved: The responder will ignore bits in this field from the requester except for including them in the CRC calculation. HMC can use portions of this field range internally.
Sequence number	SEQ	3	[18:16]	Incrementing value for each packet transmitted, except for PRET and IRTRY packets (see Packet Sequence Number Generation).
Forward retry pointer	FRP	8	[15:8]	Retry pointer representing this packet's position in the retry buffer (see Retry Pointer Description).
Return retry pointer	RRP	8	[7:0]	Retry pointer being returned for other side of link (see Retry Pointer Description).

Response Packets

Response packets carry response commands from the responder (HMC link configured as host link) to the requester (host or HMC link configured as pass-through link). All response packets are subject to normal packet flow control.

Figure 20: Response Packet Header Layout



Response packet headers for response commands contain the fields shown in the table below. Specific commands may require some of the fields to be 0. These are provided in the Valid Field and Command Summary Table.

Table 18: Response Packet Header Fields

Name	Field Label	Bit Count	Bit Range	Function
Reserved	RES	22	[63:42]	Reserved: The host will ignore bits in this field from HMC except for including them in the CRC calculation.

Table 18: Response Packet Header Fields (Continued)

Name	Field Label	Bit Count	Bit Range	Function
Source link ID	SLID	3	[41:39]	Used to identify the source link for response routing. A unique value is generated and inserted into this field by HMC and is considered a "Don't Care" by the host processor. The value in the SLID field is copied into the response header from the corresponding request packet for response routing purposes.
Reserved	RES	15	[38:24]	Reserved: The host will ignore bits in this field from HMC except for including them in the CRC calculation.
Tag	TAG	9	[23:15]	Tag number uniquely associating this response to a request.
Duplicate length	DLN	4	[14:11]	Duplicate of packet length field
Packet length	LNG	4	[10:7]	Length of packet in 128-bit FLITs
Reserved	RES	1	[6]	Reserved: The host will ignore this bit from HMC except for including it in the CRC calculation.
Command	CMD	6	[5:0]	Packet command (see the Transaction Layer – Response Commands table for response commands)

Figure 21: Response Packet Tail Layout

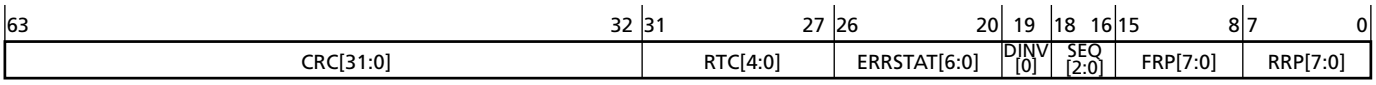


Table 19: Response Packet Tail Fields

Name	Field Label	Bit Count	Bit Range	Function
Cyclic redundancy check	CRC	32	[63:32]	Error-detecting code field that covers the entire packet.
Return token counts	RTC	5	[31:27]	Returns token count for transaction-layer flow control. In the response packet tail, the RTC contains the tokens that represent available space in HMC's input buffer.
Error status	ERRSTAT	7	[26:20]	Error status bits (see the ERRSTAT[6:0] Bit Definitions table)
Data Invalid	DINV	1	[19]	Indicates validity of packet payload. Data in packet is valid if DINV = 0 and invalid if DINV = 1.
Sequence number	SEQ	3	[18:16]	Incrementing value for each packet transmitted. See Packet Sequence Number Generation).
Forward retry pointer	FRP	8	[15:8]	Retry pointer representing this packet's position in the retry buffer (see Retry Pointer Description).
Return retry pointer	RRP	8	[7:0]	Retry pointer being returned for the other side of link (see Retry Pointer Description).

The ERRSTAT field included in the response tail is valid for all read, write, and error responses. All bits in the ERRSTAT field being zero indicates a normal response with no

errors or flags. There are 6 different error and flag categories defined by ERRSTAT[6:4] bits. The unique error or flag descriptions within each category indicate a specific error or item that requires attention. Additional status on errors or flags can be obtained by accessing internal status registers using the MODE READ command or sideband (I^2C or JTAG). The table that follows shows the error categories and descriptions along with the method of response (read/write response packet or error response packet).

Table 20: ERRSTAT[6:0] Bit Definitions

ERRSTAT Bit							Description	Response Packet Type	Notes
6	5	4	3	2	1	0			
0	0	0	0	0	0	0	No errors/conditions		
Warnings									
0	0	0	0	0	0	1	Reliability temperature threshold: The specified maximum device junction temperature has been reached. Any additional temperature increase will be harmful to long term device reliability.	Error response packet (TAG = CUB number)	
0	0	0	0	0	1	0	Host token overflow warning: The host sent more than 1023 tokens to HMC on a link.		
0	0	0	0	1	0	1	Repair warning: There are one or more regions of memory that have no available repair resources left.		
0	0	0	0	1	1	0	Repair alarm: A request has encountered a hard SBE or MUE, and there are no repair resources available.		
DRAM Errors									
0	0	1	0	0	0	0	SBE or MCE occurred (this status is normally disabled): Data is corrected before being returned to the requester. The SBE is scrubbed and tested to determine if it is a hard error, in which case it can be dynamically repaired. The MCE is tested to determine if it is a hard error, in which case it will be reported/logged for repair.	Read response packet (TAG = tag of corresponding request)	1
0	0	1	1	1	1	1	MUE has occurred: This indicates that a multiple uncorrectable error has occurred in the DRAM for read data. The DINV (data invalid) flag will also be active. Error packet will be unsolicited if MUE was encountered during patrol scrub.	Read response packet (TAG = tag of corresponding request) or Error response packet (TAG = CUB number) if MUE is discovered during patrol scrubbing	
Link Errors									

Table 20: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type	Notes
6	5	4	3	2	1	0			
0	1	0	0	0	0	0	Link 0 retry in progress: Link slave 0 has detected a link error on an incoming packet and is initiating a link retry.	Error response packet (TAG = CUB number)	
0	1	0	0	0	0	1	Link 1 retry in progress: Link slave 1 has detected a link error on an incoming packet and is initiating a link retry.	Error response packet (TAG = CUB number)	
0	1	0	0	0	1	0	Link 2 retry in progress: Link slave 2 has detected a link error on an incoming packet and is initiating a link retry.		
0	1	0	0	0	1	1	Link 3 retry in progress: Link slave 3 has detected a link error on an incoming packet and is initiating a link retry.		
Protocol Errors									
0	1	1	0	0	0	0	Invalid command: An unsupported command existed in a request packet.	Write response packet (TAG = tag of corresponding request)	
0	1	1	0	0	0	1	Invalid length: An invalid length was specified for the given command in a request packet.		2
Vault Critical Errors									
1	1	0	0	0	0	0	Vault 0 critical error: Command/Address parity error has not cleared after the command/address retry count threshold has been met. Vault 0 command execution is halted until a reset is performed.	Error response packet (TAG = CUB number)	
1	1	0	0	0	0	1	Vault 1 critical error		
1	1	0	0	0	1	0	Vault 2 critical error		
1	1	0	0	0	1	1	Vault 3 critical error		
1	1	0	0	1	0	0	Vault 4 critical error		
1	1	0	0	1	0	1	Vault 5 critical error		
1	1	0	0	1	1	0	Vault 6 critical error		
1	1	0	0	1	1	1	Vault 7 critical error		
1	1	0	1	0	0	0	Vault 8 critical error		
1	1	0	1	0	0	1	Vault 9 critical error		
1	1	0	1	0	1	0	Vault 10 critical error		
1	1	0	1	0	1	1	Vault 11 critical error		
1	1	0	1	1	0	0	Vault 12 critical error		
1	1	0	1	1	0	1	Vault 13 critical error		
1	1	0	1	1	1	0	Vault 14 critical error		
1	1	0	1	1	1	1	Vault 15 critical error		

Table 20: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type	Notes
6	5	4	3	2	1	0			
Fatal Errors									
1	1	1	0	0	0	0	Link 0 retry failed and was unsuccessful after the retry limit was reached.	Error response packet (TAG = CUB number)	
1	1	1	0	0	0	1	Link 1 retry failed and was unsuccessful after the retry limit was reached.		
1	1	1	0	0	1	0	Link 2 retry failed and was unsuccessful after the retry limit was reached.		
1	1	1	0	0	1	1	Link 3 retry failed and was unsuccessful after the retry limit was reached.		
1	1	1	1	0	0	0	Input buffer overrun link 0: The host has issued too many FLITs and has overrun the link input buffer, or has issued a MODE CMD before the previous MODE CMD has returned a response packet.		3
1	1	1	1	0	0	1	Input buffer overrun link 1: The host has issued too many FLITs and has overrun the link input buffer, or has issued a MODE CMD before the previous MODE CMD has returned a response packet.		3
1	1	1	1	0	1	0	Input buffer overrun link 2: the host has issued too many FLITs and has overrun the link input buffer, or has issued a MODE CMD before the previous MODE CMD has returned a response packet.		3
1	1	1	1	0	1	1	Input buffer overrun link 3: The host has issued too many FLITs and has overrun the link input buffer, or has issued a MODE CMD before the previous MODE CMD has returned a response packet.		3
1	1	1	1	1	0	0	Reserved		
1	1	1	1	1	0	1	Critical temperature threshold: HMC's junction temperature has reached a critically high limit and must be shut down immediately.		
1	1	1	1	1	1	0	Packet length fatal error: Packet length larger than MAX supported size (LNG = DLN > 9 FLITs)		

Table 20: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type	Notes
6	5	4	3	2	1	0			
1	1	1	1	1	1	1	Internal fatal error: This group includes but is not limited to CRC error at vault controller, internal state machine errors, and internal buffer underruns/ overruns.	Error response packet (TAG = CUB number)	

Notes:
1. Requires that the Enable SBE report field of the vault control register is set to enable.
2. (LNG = DLN) ≤9
   If (LNG = DLN) >9, a fatal error occurs.
3. The input buffer overrun fatal error only happens when HMC's input buffer actually overruns, not when the host sends more FLITs than it has tokens for.

Flow Packets

Flow packets are generated at the link master to convey information for link flow control and link retry control to the link slave on the other end of the link. They are not part of the transaction layer. The link slave extracts the control information from the flow packets and then removes the flow packets from the packet stream without forwarding them to the link input buffer. For this reason flow packets are not subject to normal packet flow control. The link master can generate and transmit flow packets without requiring tokens, and it will not decrement the token count when it transmits flow packets.

Flow packets use the request packet header and tail layouts described in Request Packets. The flow packet commands are described in Flow Commands.

Some header fields and tail fields within the flow packets are not used and should be set to 0, as specified in the Valid Field and Command Summary Table.

Poisoned Packets

A poisoned packet is uniquely identified by the fact that its CRC is inverted from the correct CRC value. Packet CRC verification logic of both the requester and the responder should include checking for an inverted value of good CRC. A poisoned packet starts out as a good packet (as identified by a good CRC) but for various reasons can become poisoned. The most common cause of a poisoned packet is a link error, as described in Link Retry. Other examples of how a packet becomes poisoned are described in Packet Integrity. There are both valid and invalid fields within the header and tail of poisoned packets as is described in the Valid Field and Command Summary Table. Receiving a poisoned packet does not trigger a link retry.

Poisoned Packet Rules

- A packet of any valid length can be poisoned.
- NULL FLITs will not be poisoned.
- If a link slave (LS) in the requester or responder receives a previously poisoned packet (the packet was poisoned before it was transmitted on the link) of any valid length, the LS must extract the FRP, RRP, and RTC fields as in any other received packet. After extraction, the LS has the option of dropping the packet or forwarding the packet if

the LS is using cut-through routing, based on the packet routing algorithm that the link slave normally executes (evaluating the CUB and ADRS field of requests or evaluating the SLID field of responses). CUB and ADRS fields are not valid in poisoned packets, and could lead to routing the poison packet to a unintended destination. This unintended routing must not cause a routing error to be reported.

- Previously poisoned packets travelling across the link use tokens like any other valid packet. The link slave should return the correct number of tokens when a previously poisoned packet is pulled out of the link input buffer. If the link slave drops a previously poisoned packet, it must still return the correct number of tokens back to the other side of the link.
- The routing fabric and all packet destinations must be able to tolerate stray poisoned packets of any valid length. At any point in the routing fabric the packet could be dropped, but if cut-through routing applies (packet routing begins before the tail arrives indicating that the packet is poisoned), the packet may continue to propagate. All normal credit and token handling applies to the propagating poisoned packet. Credits or tokens for any previously poisoned packet that has been dropped must also be returned.
- If the previously poisoned packet is routed to the link master of an outgoing link, the LM treats it like any other valid packet, where the LM embeds FRP, RRP, and RTC values into the tail of the poisoned packet. As the LM recalculates the CRC, which now includes these embedded values, it must result in another poisoned CRC state (being an inverted value of the newly generated CRC value). The poisoned packet will also be temporarily stored in the link retry buffer when it is transmitted on the link.
- A packet destination (including a host receiving packets from a cube) must be able to tolerate receiving a poisoned packet of any valid length and silently dropping it. Note that a poisoned packet could have corrupted CUB, ADRS, ERRSTAT, and/or TAG fields, along with corrupted data FLITs if present.

Request Commands

All request commands are issued by the requester (host or HMC link configured as pass-through link) to the responder (HMC link configured as host link), using request packets described in the Request Packets section. Every request must have a unique tag included in the request packet header. A corresponding response packet (one that includes the same tag) will be issued by the responder.

All request packets are saved in the link retry buffer as they cross a link. This means that these packets are retried if an error occurs on their transfer.

Note that there are vendor-specific command patterns listed at the bottom of the table. These are reserved for internal Micron use only. If they are entered in request packets, a response packet could be returned with Invalid Command/Length errors, or potentially no response at all.

Table 21: Transaction Layer – Request Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs	Response Returned
		5	4	3	2	1	0		
WRITE Requests									
16-byte WRITE request	WR16	0	0	1	0	0	0	2	WR_RS

Table 21: Transaction Layer – Request Commands (Continued)

Command Description	Symbol	CMD Bit						Packet Length in FLITs	Response Returned
		5	4	3	2	1	0		
32-byte WRITE request	WR32	0	0	1	0	0	1	3	WR_RS
48-byte WRITE request	WR48	0	0	1	0	1	0	4	WR_RS
64-byte WRITE request	WR64	0	0	1	0	1	1	5	WR_RS
80-byte WRITE request	WR80	0	0	1	1	0	0	6	WR_RS
96-byte WRITE request	WR96	0	0	1	1	0	1	7	WR_RS
112-byte WRITE request	WR112	0	0	1	1	1	0	8	WR_RS
128-byte WRITE request	WR128	0	0	1	1	1	1	9	WR_RS
MODE WRITE, BIT WRITE and ATOMIC Requests									
MODE WRITE request	MD_WR	0	1	0	0	0	0	2	MD_WR_RS
BIT WRITE request	BWR	0	1	0	0	0	1	2	WR_RS
Dual 8-byte add immediate	2ADD8	0	1	0	0	1	0	2	WR_RS
Single 16-byte add immediate	ADD16	0	1	0	0	1	1	2	WR_RS
POSTED WRITE Requests									
16-byte POSTED WRITE request	P_WR16	0	1	1	0	0	0	2	None
32-byte POSTED WRITE request	P_WR32	0	1	1	0	0	1	3	None
48-byte POSTED WRITE request	P_WR48	0	1	1	0	1	0	4	None
64-byte POSTED WRITE request	P_WR64	0	1	1	0	1	1	5	None
80-byte POSTED WRITE request	P_WR80	0	1	1	1	0	0	6	None
96-byte POSTED WRITE request	P_WR96	0	1	1	1	0	1	7	None
112-byte POSTED WRITE request	P_WR112	0	1	1	1	1	0	8	None
128-byte POSTED WRITE request	P_WR128	0	1	1	1	1	1	9	None
POSTED BIT WRITE and POSTED ATOMIC Requests									
Posted BIT WRITE request	P_BWR	1	0	0	0	0	1	2	None
Posted dual 8-byte add immediate	P_2ADD8	1	0	0	0	1	0	2	None
Posted single 16-byte add immediate	P_ADD16	1	0	0	0	1	1	2	None
Read Requests									
16-byte READ request	RD16	1	1	0	0	0	0	1	RD_RS
32-byte READ request	RD32	1	1	0	0	0	1	1	RD_RS
48-byte READ request	RD48	1	1	0	0	1	0	1	RD_RS
64-byte READ request	RD64	1	1	0	0	1	1	1	RD_RS
80-byte READ request	RD80	1	1	0	1	0	0	1	RD_RS
96-byte READ request	RD96	1	1	0	1	0	1	1	RD_RS
112-byte READ request	RD112	1	1	0	1	1	0	1	RD_RS
128-byte READ request	RD128	1	1	0	1	1	1	1	RD_RS
Mode Read Requests									
MODE READ request	MDRD	1	0	1	0	0	0	1	MD_RD_RS

Table 21: Transaction Layer – Request Commands (Continued)

Command Description	Symbol	CMD Bit						Packet Length in FLITs	Response Returned
		5	4	3	2	1	0		
Vendor Specific									
Reserved	–	0	0	0	x	x	x	–	–
Reserved	–	1	1	1	x	x	x	–	–
Reserved	–	0	1	0	1	x	x	–	–
Reserved	–	1	0	1	1	x	x	–	–

READ and WRITE Request Commands

READ and WRITE request commands are issued by the requester to access the DRAM memory. They are block commands that access a contiguous number of memory-addressable bytes with 16-byte granularity, meaning they can begin and end on any 16-byte boundary up to a maximum length of 128 bytes. Due to the 16-byte granularity of READ and WRITE request commands, the four LSBs of the request address are ignored. As shown in the Request Commands table, there are unique commands for each supported memory access length.

READ and WRITE operations will not cross a row (page) boundary within the DRAM array. The maximum block size selected in the address map mode register determines the boundary at which the addressing will wrap if the block access goes beyond the end of the maximum block size. All read requests are acknowledged with a read response packet that includes the tag of the request and the data payload. All nonposted write requests are acknowledged with an explicit write response packet that includes the tag of the write request. If a READ or WRITE request command specifies a data payload length that is longer than the configured maximum block size, the addressing will continue to wrap at the max block boundary until the data length is satisfied.

POSTED WRITE Request Commands

POSTED WRITE request commands operate similarly to standard write requests, except that there is no response returned to the requester. Because HMC does not use the TAG field of posted write requests, the host can populate this field with any value. If an error occurs during the execution of the write request, an Error Response packet will be generated indicating the occurrence of the error to the host. In this case, the write request's tag will not be included in the Error Response packet.

ATOMIC Request Commands

Dual 8-Byte Add Immediate

This atomic request performs two parallel ADD IMMEDIATE operations. Each add operation uses an 8-byte (64-bit) two's complement signed integer memory operand from DRAM and a 4-byte (32-bit) two's complement signed integer immediate operand from the atomic request data payload. The 32-bit operand is sign-extended to 64 bits before the add operation is performed. The results are written back to DRAM, overwriting the original memory operands.

It is up to the host performing the atomic operation to verify that the data value written has not overflowed. An overflow has occurred if the carry in to the most significant bit of

the result is not equal to the carry out of the most significant bit of the result; that is, an overflow has occurred if the sum of two positive numbers yields a negative result or if the sum of two negative numbers yields a positive result.

The following table provides three examples of overflow analysis, using 4-bit arithmetic for clarity. A 4-bit, two's complement signed value can represent numbers in the range of –8 to +7.

Table 22: Overflow Analysis Examples

Description	Example 1		Example 2		Example 3	
	Binary	Decimal	Binary	Decimal	Binary	Decimal
Memory operand	0111	7	1000	–8	1000	–8
Immediate operand (2-bit)	01	1	10	–2	01	1
Immediate operand (sign-extended to 4 bits)	0001	1	1110	–2	0001	1
Addition (unlimited precision)	01000	8	10110	–10	1001	–7
Addition (4-bit two's complement precision)	1000	–8	0110	6	1001	–7
Analysis of result		Overflow		Overflow		Correct

Table 23: Operands from DRAM

Byte	15 (MSB)	14	13	12	11	10	9	8 (LSB)	7 (MSB)	6	5	4	3	2	1	0 (LSB)[1]
Field	Memory operand 2								Memory operand 1							

Notes:
1. Corresponds to the start of the 16-byte aligned address provided in the request address field.
2. Each byte field contains bit positions [7(MSB):0(LSB)].

Table 24: Immediate Operands Included in Atomic Request Data Payload

Byte	15	14	13	12	11 (MSB)	10	9	8 (LSB)	7	6	5	4	3 (MSB)	2	1	0 (LSB)
Field	4 bytes of zeros				Imm operand 2				4 bytes of zeros				Imm operand 1			

Note:
1. Each byte field contains bit positions [7(MSB):0(LSB)].

Single 16-Byte Add Immediate

This request performs a single ADD IMMEDIATE operation. It uses a 16-byte (128-bit) two's complement signed integer memory operand from DRAM and an 8-byte (64-bit) two's complement signed integer immediate operand from the atomic request data payload. The 64-bit operand is sign-extended to 128 bits before the add operation is performed. The result is written back to the DRAM, overwriting the original memory operand.

It is up to the host performing the atomic operation to verify that the data value written has not overflowed. An overflow has occurred if the carry in to the most significant bit of the result is not equal to the carry out of the most significant bit of the result; that is, an

overflow has occurred if the sum of two positive numbers yields a negative result or if the sum of two negative numbers yields a positive result.

Table 25: Operands from DRAM

Byte	15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)[1]
Field	Memory operand 1															

Notes:
1. Corresponds to the start of the 16-byte aligned address provided in the request address field.
2. Each byte field contains bit positions [7(MSB):0(LSB)].

Table 26: Immediate Operand Included in Atomic Request Data Payload

Byte	15	14	13	12	11	10	9	8	7 (MSB)	6	5	4	3	2	1	0 (LSB)
Field	8 bytes of zeros								Immediate operand 1							

Note:
1. Each byte field contains bit positions [7(MSB):0(LSB)].

MODE READ and WRITE Request Commands

MODE READ and WRITE request commands access the internal hardware mode and status registers within the logic layer of HMC. Each mode request accesses up to 32 contiguous bits of the register.

The address field (ADRS) within the mode request packet is redefined to provide a register address, a start field, and a size field as shown in the table below. MODE READ and WRITE request commands must include address values that reference the registers defined in the Register Addendum. A MODE READ request command that contains an address that references a nonexistent register results in a MODE READ response that includes all zeros within the data payload. A MODE WRITE request that references a nonexistent register will not be performed. There will be no error indication.

Table 27: Mode Request Addressing

Field	Bits	Number of Bits	Description
Unused	33:32	2	Unused bits
Start	31:27	5	**Start field:** Encoded to provide a value from 0 to 31 that points to the starting bit position within the 32 bits of data, 0 being the LSB position and 31 being the most significant bit (MSB) position.
Size	26:22	5	**Size field:** Indicates the number of contiguous bits to read or write (from 1 to 32), and is encoded as follows: 0x0 = 32 bits, 0x1–0x1F = 1–31 bits. The minimum size is 1 bit.
Register address	21:0	22	**Register address field:** References a specific mode or status register. All registers are 32 bits or less in size.

Notes:
1. A full 32-bit access would be 0 in both the start and size fields.
2. See HMC Register Addendum for more details on MODE READ and WRITE Request Commands and applicable registers.

The valid data bytes within the data payload in the mode write request packet and the mode read response packet are right justified at the four least significant bytes as shown in the following table.

Table 28: Valid Data Bytes

Byte	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Unused												Data			

If less than 32 bits are being accessed, as determined by the start and size fields within the address, the valid data bits are right justified to the LSBs within the 4-byte data field.

HMC can only process one mode request at a time. After a mode request is issued, the host(s) must wait for the corresponding response before issuing another mode request on any link. Because of this requirement, HMC does not support posted MODE WRITE requests.

BIT WRITE Command

The BIT WRITE request is a 2-FLIT request packet that provides a 0- to 8-byte write capability as indicated by write data mask bits included in the data payload. The vault controller performs a READ-UPDATE-WRITE operation to the DRAM to merge the write data bytes with the existing bytes in the memory. The three LSBs of the byte address field within the address in the request packet header must be 000 when addressing an 8-byte block. The specific bits to be written within the 8-byte block are identified by the data mask bits in bytes[15:8] of the 16-byte data payload within the request packet as shown in the following table.

Table 29: Request Packet Bytes to be Written

Byte	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Data mask								Write data							

Notes: 1. Data mask: Each bit inhibits writing the corresponding write data bit when 1.
2. Write data: Up to eight bytes of byte-aligned write data.

The bit(s) to be written must be positioned in the correct bit position within the write data field, as identified with the cleared bit(s) in the data mask field. For example, if the data mask = 0x00FFFFFFFFFF00FF, then bytes 7 and 1 of the write data field will be written into the same bytes in the memory. If the data mask = 0xFFFFFFFFFFFFFFFF, the READ-UPDATE-WRITE operation will be performed with the original value of the data being rewritten into the memory.

Response Commands

All response commands are issued by the responder (HMC link configured as host link) to the requester (host or HMC link configured as pass-through link), using response packets described in Response Packets. Every response other than the error response includes a valid tag that matches the tag in its corresponding request. All response packets are saved in the link retry buffer as they cross a link, which means they are retried if an error occurs on the link.

Table 30: Transaction Layer – Response Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs
		5	4	3	2	1	0	
READ response	RD_RS	1	1	1	0	0	0	1 + data FLITs
WRITE response	WR_RS	1	1	1	0	0	1	1
MODE READ response	MD_RD_RS	1	1	1	0	1	0	2
MODE WRITE response	MD_WR_RS	1	1	1	0	1	1	1
ERROR response	ERROR	1	1	1	1	1	0	1
Reserved	–	1	1	1	1	1	1	–

READ and MODE READ Response Commands

A READ or MODE READ response command is issued by the responder to return requested data to the requester. A read response packet always includes data payload along with the header and tail, and it includes the same tag as its corresponding read request. If an error occurred such that the data is not valid, as indicated in the DINV field in the response tail, the data payload FLITs are still present in the response packet but they hold invalid data. For this reason, the response packet lengths are always consistent with the corresponding request.

A MODE READ command must include address values that point to the registers defined in the Request Commands section. A MODE READ request command that contains an address that points to a non-existent register will get a MODE READ response back that includes all 0s within the data payload, and the DINV bit will not be set.

WRITE and MODE WRITE Response Commands

A WRITE or MODE WRITE response command provides an explicit write response packet that returns the tag for the original write request. If errors occurred during the transfer or execution of the WRITE request command, status will be included in the ERRSTAT field of the write response packet. A lack of error status bits indicates that the write was successful.

A MODE WRITE command must include address values that point to the registers defined in the Request Commands section. A MODE WRITE request command that contains an address that points to a non-existent register will get a MODE WRITE response back; however, the request will not be performed.

ERROR Response Command

HMC uses the READ, MODE READ, WRITE, and MODE WRITE response commands to report errors or flags that are associated with a specific request and tag. These error reports are included in the ERRSTAT field of the response packet.

The ERROR response command is used by HMC to report a critical event when the request type and/or the original request tag are not known. The three LSBs of the ERROR response command packet tag field contain the values the CUB[2:0] signals are tied to. TAG[2:0] = CUB[2:0] and TAG[8:3] = 0. This gives the host visibility to which HMC the Error Response packet originated from in a topology with multiple HMCs. The ERROR response command is unsolicited and can be configured to transmit ERROR response commands on any or all links except for those in chained topologies, where the error

response command must only be routed to a single source link. The following situations cause HMC to transmit an ERROR response command packet:

- The responder receives a request packet from the requester that is successfully transferred across a link but is internally corrupted before it reaches its destination. In this case the original command and tag are lost and the responder cannot generate a proper response packet or the necessary tag to close the transaction.
- HMC has an internal error not associated with a transaction.
- HMC reports a condition or status to the host.

Flow Commands

Flow commands facilitate retry and flow control on the link and are not part of the transaction layer. They are generated at the link master to send information to the other end of the link when no other link traffic is occurring or when a link retry sequence is to be initiated. Flow commands are not forwarded from that point. Flow commands use the request packet header and tail layouts described in the Request Packets section, but are not considered requests, and do not have corresponding response commands. Flow commands are not subject to token flow control. The link master is allowed to send a flow command even if there are no tokens available.

Table 31: Flow Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs
		5	4	3	2	1	0	
Null	NULL	0	0	0	0	0	0	1
Retry pointer return	PRET	0	0	0	0	0	1	1
Token return	TRET	0	0	0	0	1	0	1
Init retry	IRTRY	0	0	0	0	1	1	1

NULL Command

A NULL command field of all zeros, along with all zeros in the rest of the FLIT, define a NULL FLIT. NULL FLITs are driven on the link when there are no packets to be transmitted. (Null FLITs are also used during link initialization.) Note that a FLIT of all zeros will have a correct CRC. A poisoned version of the NULL FLIT is not supported.

RETRY POINTER RETURN (PRET) Command

This command is issued by the link master to return retry pointers when there is no other link traffic flowing at the time the pointer is to be returned. It is a single-FLIT packet with no data payload. PRET packets are not saved in the retry buffer, thus their SEQ and FRP fields are ignored by HMC. Tokens should not be returned in these packets, the RTC field is ignored by HMC.

TOKEN RETURN (TRET) Command

This command is issued by the link master to return tokens when there is no other link traffic flowing at the time the token is to be returned. It is a single-FLIT packet with no data payload. Token return packets are saved in the retry buffer, thus their SEQ and FRP fields are valid.

INIT RETRY (IRTRY) Command

A programmable stream of INIT RETRY (IRTRY) command packets is issued by the link master when it is in the Retry_Init state. This is a single-FLIT packet with no data payload. IRTRY command packets are not saved in the retry buffer, and their SEQ field is ignored by HMC. The FRP field is used to indicate the following: FRP[0] = Start Retry Flag, FRP[1] = Clear Error Abort Flag. Tokens must not be returned in these packets, because the RTC field is ignored by HMC (see the LinkRetry_Init State section).

Valid Field Command Summary

Table 32: Valid Field and Command Summary Table

	Request		Response			Flow				Previously Poisoned[2, 8]
Symbol	READ	WRITE	READ	WRITE	ERROR	NULL	PRET	TRET	IRTRY	All
ADRS	V	V	N/A	N/A	N/A	0	0	0	0	NV
CMD	V	V	V	V	V	0	V	V	V	NV
CRC	V	V	V	V	V	0	V	V	V	V
CUB	V	V	N/A	N/A	N/A	0	0	0	0	NV
DINV	N/A	N/A	V	0	0	N/A	N/A	N/A	N/A	NV
DLN	V	V	V	V	V	0	V	V	V	V[3]
ERRSTAT	N/A	N/A	V	V	V	0	N/A	N/A	N/A	NV
FRP	V	V	V	V	V	0	0	V	V[4]	V
LNG	V	V	V	V	V	0	V	V	V	V[3]
RRP	V	V	V	V	V	0	V	V	V	V
RTC	V[7]	V[7]	V	V	V	0	0	V	0	V
SEQ	V	V	V	V	V	0	0	V	0	V
TAG	V	V[5]	V	V	V[6]	0	0	0	0	NV

Notes:
1. V: Valid field
   NV: Not valid, do not use field
   N/A: Not applicable, field doesn't exist
   0: Field must be 0.
2. A poisoned packet is defined by an inverted CRC.
3. DLN and LNG are valid as long as they match each other.
4. FRP[0] = Start Retry Flag
   FRP[1] = Clear Error Abort Flag.
5. The TAG field is not used within POSTED WRITE command requests; the host can use any value for TAG, including zero.
6. TAG[2:0] = CUB[2:0] and TAG[8:3] = 0.
7. If using response open loop mode, the value in the RTC field is "Don't Care."
8. Previously poisoned packets are defined as packets that are poisoned before being transmitted across the link.

Transaction Layer Initialization

During link initialization, after the responder stops sending TS1 patterns on the link, the link layer is considered active. Both ends of the link are in the active state, sending

NULL FLITs, as described in Step 11 in the Power-On and Initialization section. At this point, HMC will continue internal initialization.

Transaction layer initialization continues with the following steps:

1. After internal initialization is complete, the responder will send one or more TRET packets that will transfer the number of its link input buffer tokens to the requester. Each TRET packet can transmit a count of 31 tokens, therefore there will be multiple TRETs required to transfer the entire number of tokens representing the link input buffer's available space.
2. After the requester receives at least one TRET packet from the responder, the requester should transmit a series of TRET packets back to the responder carrying the tokens representing the available space in the requester's link input buffer. HMC can support up to 1023 tokens from the host on each link.
3. After the TRET packets are transmitted from the requester to the responder, the requester can now start sending transaction packets. There is no required time frame when the transaction packets can start.

Within HMC, all links that are configured as pass-through links must complete their transaction layer initialization before the links configured as host links begin to send TRET packets upstream. This ensures that all links within a multicube topology are initialized after the link configured as a host link in source mode has finished its transaction layer initialization.

Configuration and Status Registers

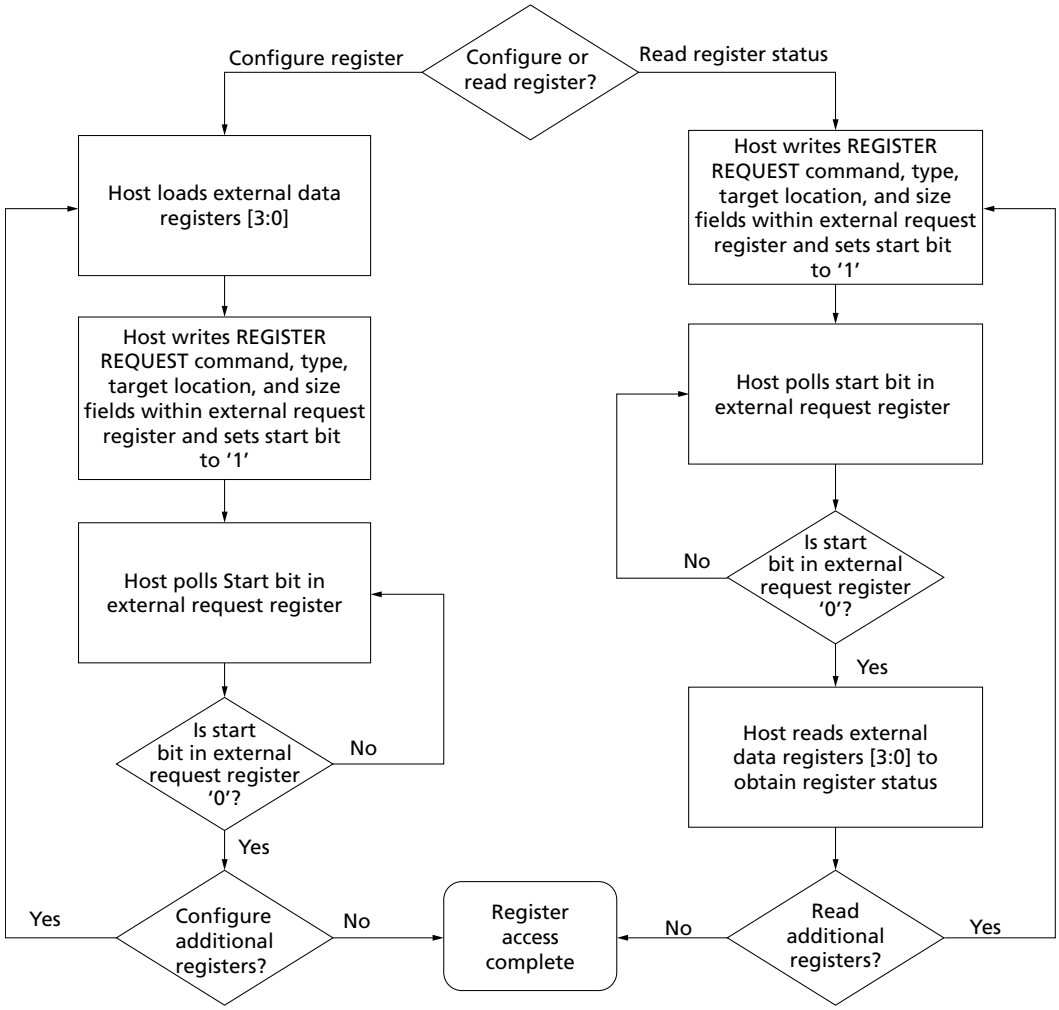
There are two methods supported for accessing HMC's configuration and status registers:

Direct access: This type of access uses the in-band MODE WRITE and MODE READ request commands or the sideband (JTAG/I^2C) to directly read or write the configuration and status registers. See the HMC Register Addendum for specific direct access configuration and status registers.

Indirect access through the external data interface (ERI): This type of access utilizes four external data registers and an external request register to indirectly access HMC's configuration and status registers. The external data and request registers are accessible through MODE WRITE and MODE READ request commands or sideband (JTAG/I^2C). Specific configuration and status registers accessible through the ERI are not contained within this specification, only the framework to enable this type of register access. See the HMC Register Addendum for specific indirect access configuration and status registers. See the following figure for the steps required to access the configuration and status registers through the ERI.

For specific register functionality, including descriptions, addresses and ERI sequences, see the HMC Register Addendum.

Figure 22: Configuration and Status Register Access Through ERI



Link Retry

Each link transmits data at a very high speed and requires a mechanism for correcting failures to increase system reliability and availability. Link retry is a fault-tolerant feature that recovers the link when link errors occur. Link packet integrity is insured with a CRC code that resides in the tail of every packet. It covers all bits in the packet, including the header, any data payload, and the tail. CRC is generated at the link master before the packet bits are serialized and transmitted, and is checked at the link slave after the packet bits are received and deserialized. A mismatch indicates that some of the data within the packet has changed since the original CRC code was generated at the transmitting end of the link. An additional check is provided using an incrementing sequence number included in every packet. If the CRC check is successful upon receipt of the packet, the link slave will verify that the sequence number has a +1 value compared to the last successful packet received.

The link master saves a copy of each packet transmitted across the link. Each packet is held until an acknowledge occurs indicating that the packets have been received without errors. This acknowledgment comes from a retry pointer (included in every trans-

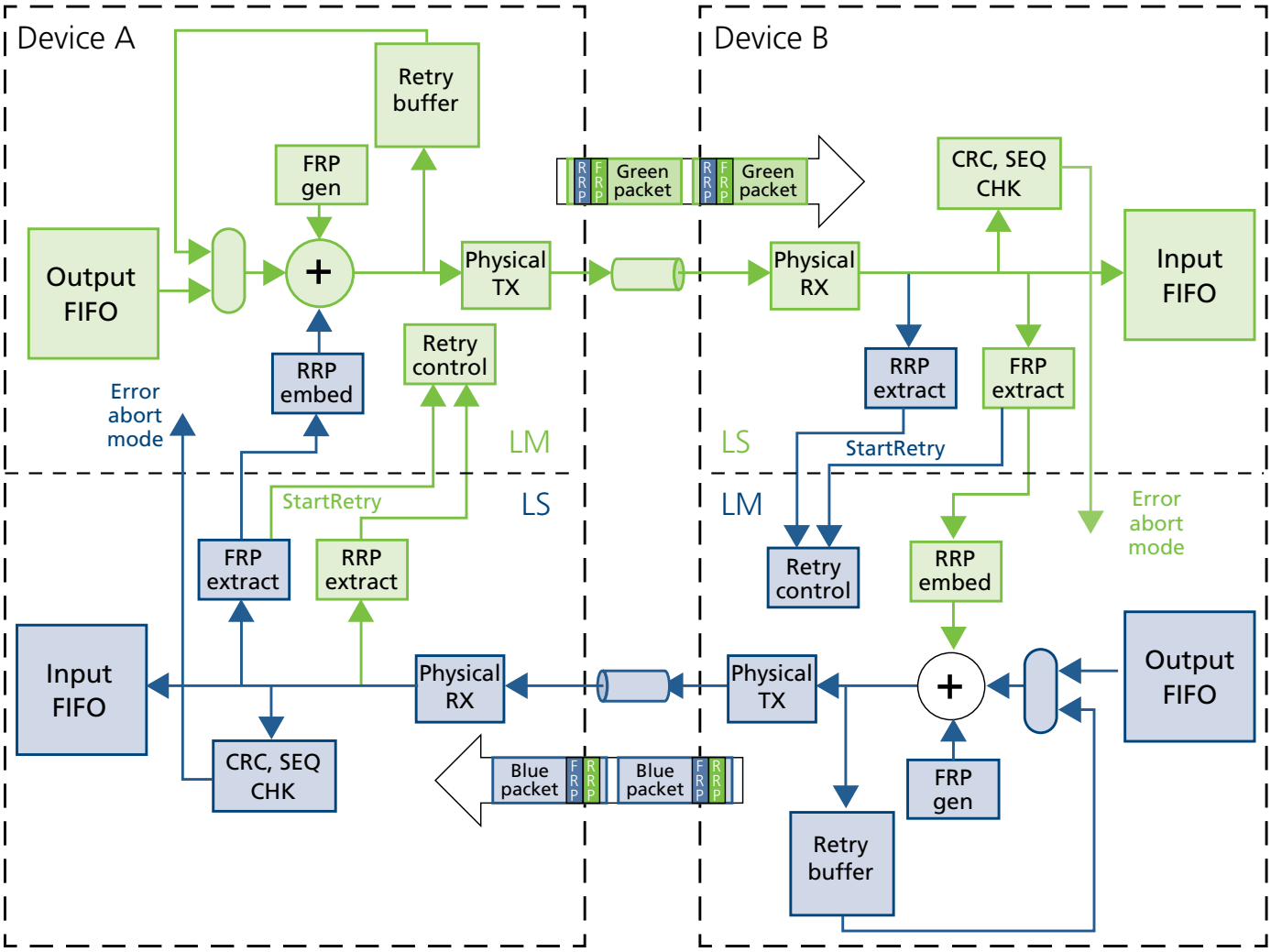
mitted packet); it is returned to the link master embedded in packets traveling in the opposite direction on that link.

If the link slave detects a packet error, it enters error abort mode. This results in the failed packet being poisoned or dropped and all subsequent packets being dropped. Error abort mode triggers the local link master to send a StartRetry flag back to the transmitting end using the other side of the link. The poisoned and dropped packets are then retransmitted by the link master during the retry sequence.

The following figure illustrates an example implementation of the link retry blocks on both ends of a link. The forward retry pointer (FRP) and return retry pointer (RRP) extract and embed blocks are color coded to show the path the retry pointer takes for each link direction.

The link is symmetrical and in the following discussions the requester can be either device A or device B. Similarly, the responder can be either device A or device B.

Figure 23: Implementation Example of Link Retry Block Diagram



Retry Pointer Description

The retry pointer represents the packet's position in the retry buffer. The retry pointer transmitted with the packet points to the retry buffer location + 1 (to which the tail is being written). This is the starting address of the next packet to be transmitted.

There are two retry pointer versions:

- The FRP as it travels in the forward direction on the link
- The RRP, as the same retry pointer travels back on the other side of the link, completing the round trip and eventually getting back to the retry buffer control logic in the link master

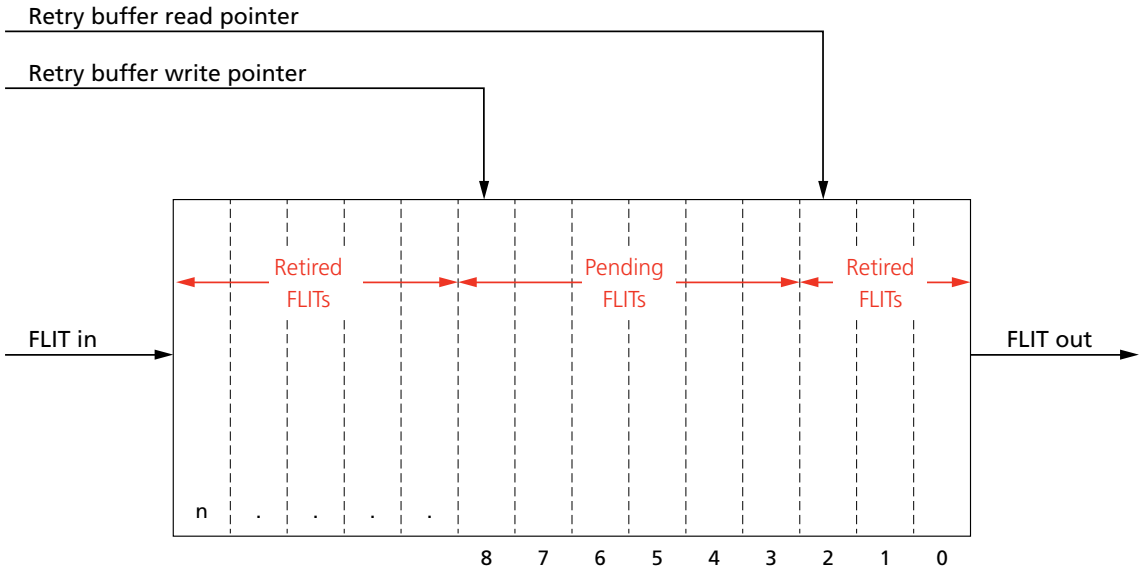
The FRP is sent with every packet that is part of the normal packet flow (other than IRTRY packets and PRET packets described later). The retry buffer is addressed using FLIT addresses, and the retry pointer represents a FLIT position in the retry buffer. Even though a packet may be multiple FLITs, the retry pointer will always point to the header FLIT position of a packet.

There are two pointer fields in the tail of a packet:

- FRP field: This is the forward retry pointer as the packet is traveling in the forward direction on the link. The FRP field is ignored in PRET packets and is redefined to contain two retry control flags in IRTRY packets.
- RRP field: This is the return retry pointer embedded in any packet traveling in the return direction of the link. If the return direction has no normal packet traffic, when an FRP is extracted in the local link slave and sent to the link master, a PRET packet is then created in the link master and the FRP is loaded into the RRP field. The RRP is not associated with the packet in which it is embedded, other than to “ride along” to get back to the other end of the link. The link is symmetrical in both directions, so RRPs are embedded in packet traffic moving in both directions. The RRP field of the packet is always valid, including IRTRY packets and PRET packets.

Link Master Retry Functions

Figure 24: Implementation Example of a Retry Buffer



In this implementation example, a retry buffer is used to save packets for potential re-transmission if a link error occurs. The retry buffer relies on a write pointer and read pointer to track the status of the FLITs stored within it. The write pointer is advanced with valid outgoing FLITs as determined by the packet stream traveling out of the link master. As FLITs are written into the retry buffer, they are considered to be pending FLITs, because they may need to be re-transmitted if an error occurs. They remain pending FLITs until the read pointer is advanced past them, indicating that one or more returned retry pointers were received, representing a successful packet transmission.

The read pointer in this example is used to read data from the retry buffer during a retry sequence. If no errors occur, retry pointers are returned (RRPs) and used to advance the read pointer. As RRPs are received, the read pointer will rotate around the retry buffer address space, following the write pointer, and turn pending FLITs into retired FLITs.

Retired FLITs are unneeded residue FLITs that will eventually be overwritten by new pending FLITs as the write pointer advances and wraps around the retry buffer address space.

During a retry sequence, the read pointer is incremented until it equals the write pointer, indicating that all pending FLITs have been pulled out of the retry buffer and retransmitted across the link. All RRPs received during this period are saved in a read pointer hold register until all pending FLITs have been read out of the retry buffer. At the end of the LinkRetry_FLIT state (see LinkRetry_FLIT State), as the retry control switches back to a flow of new packets, the latest value of the read pointer hold register is copied to the read pointer. This action may retire many of the pending FLITs. After the copy is made, any newly arrived RRPs are used to advance the read pointer directly, thus retiring more FLITs.

A threshold is implemented in this example to detect a near-full condition of the retry buffer. The near-full condition is detected by comparing the value of the write pointer to the read pointer. This threshold leaves enough room remaining in the retry buffer for the current packet in transmission to be completely saved in the retry buffer, without overflow, when the near-full detection occurs. At this point, the link master will stop the packet flow to the link.

Forward Retry Pointer Generation

The link master generates the retry pointer and includes it in the FRP field in the tail of the packet being transmitted. The retry pointer is created by using the value where the tail FLIT of the packet is to be saved for potential retransmission, incrementing by +1, and loading it into the FRP field within the tail of the packet. This value points to the location where the header of the next packet to be transmitted will be saved. The first packet written into HMC's retry buffer (a TRET packet during transaction layer initialization) will have an FRP value of 1. Until a link master receives a forward retry pointer from the local link slave, a value of zero will be embedded into the RRP field of outgoing packets. The FRP value must be included in the CRC generation, which is also loaded into the tail of the packet being transmitted. The FRP field is valid for all packets that are saved for potential retransmission. All packets that are transmitted other than IRTRY, NULL, and PRET packets are saved until the success of their transmission has been acknowledged. If the original packet stream on the link included NULL FLITs between some of the packets, those NULL FLITs are not saved for potential retransmission. However, during a retry sequence, as the retried packet sequence is transmitted, NULL FLITs may be transmitted between packets.

Packet Sequence Number Generation

The link master generates a sequence number for every outgoing packet that is part of the normal packet flow and loads it into the SEQ field within the tail of the packet. The sequence number should be incremented by + 1 for every normal packet. The sequence number should increment by +1 for every packet that is saved for retransmission. It will not be generated for PRET or IRTRY packets, as these are not saved for retransmission. Any packets replayed during a retry sequence already have their sequence number embedded and are re-sent without updating the SEQ field. After the last retried packet is transmitted, the sequence number generator should increment for the next new packet that follows the retried packet stream. This accommodates a sequence checking across a retry sequence, verifying that all packets are sequential before, during, and after the retry sequence. The sequence number register in the link slave should be set to zero upon a cold or warm reset. The first packet sent from the link master after a cold or warm reset should contain a sequence number of 1.

Forward Retry Pointer Reception and Embedding

One link master function is to return retry pointers that represent packets traveling in the opposite direction on the other side of the link. The link master captures the FRP from the link slave and then embeds it in the RRP field in the tail of the current or next packet being transmitted. At that point it becomes the return retry pointer. Transmitted packets may consist of multiple FLITs that span multiple core cycles, so one or more new FRPs could be received from the local link slave before the previous one has been embedded. In this case, the latest FRP overwrites the previous one. The link master does not have to embed every retry pointer received from the link slave; only the latest one, which supersedes any previous pointer.

If there is no forward traffic being transmitted by the link master when a retry pointer arrives from the link slave, the link master generates a single-FLIT retry pointer return (PRET) packet where it embeds the latest value of the retry pointer into the RRP field. The PRET packet generated for this purpose is not saved in the retry buffer.

If there are no FRPs being sent from the link slave to the link master, the last valid FRP value received from the link slave will be embedded in subsequent packets sent from the link master. However, in this case, no explicit PRET packets need to be generated if the last valid FRP value has been embedded in at least one packet.

The link master must always embed the current value of the retry pointer collection register into the RRP field of every nonNULL packet transmitted, including:

- Normal transactional packets from the link output buffer
- IRTRY packets transmitted during the StartRetry stream
- IRTRY packets transmitted during the LinkRetry sequence
- Pending packets transmitted during the LinkRetry_FLIT state of the LinkRetry sequence. In this case, the old pointer values of the RRP field (previously transmitted) are overwritten by the latest pointer. The CRC must be recalculated for each packet due to the RRP change.

Return Retry Pointer Reception

The link master not only receives the FRPs for packets traveling in the opposite direction from the local link slave, as described in the previous section, but also receives RRPs from the local link slave. Receipt of an RRP completes the loop that the retry pointer travels and acts as an acknowledgment of the successful transmission of the associated packet.

Link Master Sequences

The link master provides one of two different sequences, depending on where the link error occurs, that interrupts the normal packet flow being transmitted from the link master. When only one link error occurs (the most likely case), the local link master adjacent to the link slave that detected the error will execute the StartRetry sequence. The link master transmitting in the direction where the link error occurred will execute the LinkRetry sequence.

StartRetry Sequence

When a link slave detects an error, it sets “Error Abort Mode,” which is monitored by the local link master. When this signal activates, it triggers the local link master to initiate its StartRetry sequence, which interrupts the normal transaction packet flow and transmits a programmable number of single-FLIT IRTRY packets. A packet currently being transmitted must be allowed to complete before starting the IRTRY stream. Within each of these IRTRY packets, the StartRetry flag is set with FRP[0] = 1. This stream of IRTRY packets must be contiguous, meaning each IRTRY FLIT must be followed by the next IRTRY FLIT without the insertion of any other types of FLITs/packets, including NULL FLITs, for the duration of the stream. The transmission of the IRTRY stream follows these rules:

- The link master should always embed the latest FRP it received into the RRP field of each IRTRY packet as it does for any outgoing packet.
- None of the IRTRY packets should be loaded into the Retry Buffer.

The execution of the StartRetry sequence also starts a Retry Timer. This is used to monitor the success of the link retry. If the link retry is unsuccessful and Error Abort Mode does not clear, the retry timer will time-out which will initiate the execution of another StartRetry sequence. This will continue until the number of retry attempts equals the retry attempt limit specified in the link retry control register.

StartRetry Sequence States

StartRetry_Begin State

This state enables the completion of an in-flight packet transmission, after which the normal packet flow stops. The in-flight packet must also be saved for retransmission.

StartRetry_Begin State Control
- Enter: When the link master detects the rising edge of the Error Abort Mode signal from the local link slave, OR when the retry timer expires and Error Abort Mode is set.
- Actions:
  1. Allow any packet currently being transmitted to complete.
  2. Stop normal packet flow and enter StartRetry_Init state. This will force the packet stream to be queued-up in the link output FIFO.
- Exit: When the in-flight packet transmission is complete.

StartRetry_Init State

In this state, the link master transmits a programmable number of single-FLIT IRTRY packets across the link. These packets are not saved for retransmission, thus their forward retry pointer and sequence number aren't used. Within each IRTRY packet, Bit 0 of the FRP field is redefined to hold the StartRetry flag, which signals the other end of the link to initiate its LinkRetry sequence. The link master also embeds the last FRP value it received in the RRP field of every IRTRY packet. This stream of IRTRY packets accomplishes the following:

- It signals to the link slave on the far end of the link that this end detected a link error.
- It provides the latest value of the retry pointer, embedded in the RRP field, to the other end of the link.

StartRetry_Init State Control

- Set: When StartRetry_Begin state clears
- Action: Start transmission of contiguous stream of IRTRY packets that include the StartRetry flag (FRP[0]=1). The number of packets is determined by the IRTRY packet transmit number field in the Link Retry Control Register. NULL FLITs must not be inserted between IRTRY packets.
- Clear: After the programmable number of IRTRY packets has been transmitted.

When the link master exits the StartRetry_Init state, it reverts back to its normal packet stream that has been accumulating in its link output buffer.

Retry Timer

The Retry Timer monitors the success of the current retry attempt by timing the assertion of the Error Abort mode signal from the local link slave.

The vast majority of link retries will be successful on the first retry attempt, in which case the Error Abort mode will clear out within the normal retry loop time, and the Retry Timer will not affect the delay of the overall retry operation. Given that the retry loop time is less than or equal to the retry buffer full period, the retry timer should be set to at least 3 times the retry buffer full period, as shown in the right most column of the

Retry Pointer Loop Time table. This is to eliminate the possibility of issuing a second StartRetry before the first retry has a chance of completing during a multiple error case where both sides of the link may be executing a link retry sequence, before the first Init_Retry/StartRetry stream is allowed to be transmitted.

Retry Timer Algorithm

If Error_Abort_Mode = 0

- Retry_Timer = 0
- Retry_attempts = 0
- Retry_Timer_expired = 0

Elseif Retry_Timer = Retry Timeout Period AND retry_attempts < retry_attempt_limit

- Retry_Timer_expired = 1
- Retry_attempts = Retry_attempts + 1
- Retry_Timer = 0

Elseif retry_attempts = retry_attempt_limit

- Set link_retry_failed

Else

- Retry_Timer = Retry_Timer + 1
- Retry_Timer_expired = 0

EndIF

LinkRetry Sequence

If the link master receives a StartRetry pulse from the local link slave, it is an indication that the link slave on the other end of the link detected an error, and the link master should initiate its LinkRetry sequence. This sequence will suspend the flow of packets from the link output buffer and send the packets currently being saved for retransmission.

LinkRetry Sequence States

The LinkRetry sequence includes three states: LinkRetry_Begin, LinkRetry_Init, and LinkRetry_FLIT. The LinkRetry_FLIT state may or may not be entered depending on if there are any packets to be retransmitted.

LinkRetry_Begin State

This state enables the completion of an in-flight packet transmission, after which the normal packet flow stops. The in-flight packet must also be saved for retransmission.

LinkRetry_Begin State Control
- Enter: When the link master receives a StartRetry pulse from the local link slave.
- Actions:
  1. Allow the packet currently in transmission to complete.
  2. Stop normal packet flow and enter next retry state.
- Exit: When the in-flight packet transmission is complete.

LinkRetry_Init State

In this state, the link master transmits a programmable number of single-FLIT IRTRY packets across the link. Within each of these IRTRY packets, the ClearErrorAbort flag

will be set (FRP[1]=1). These packets are not saved for retransmission, thus their Forward Retry Pointer and SEQ number are not used. The link master also embeds the last valid FRP it received from its local link slave in the RRP field of every IRTRY packet. The stream of IRTRY packets enables the following:

- The ClearErrorAbort flag enables the link slave section on the far end of the link to detect the link master is in the retry sequence.
- It provides a period of time for the link slave to detect a number of good IRTRY packets that establish confidence in the link.

LinkRetry_FLIT State

In this state, the link master transmits the packets currently being saved for retransmission. These packets have already been transmitted but have been aborted at the receive end of the link. If an error occurs in a packet that this link master had previously transmitted, this state will be entered. If an error occurs only during the transmission of a stream of continuous NULL FLITs and there is not any FLITs currently saved for retransmission, this state will not be entered.

If the link master is retransmitting packets in the LinkRetry_FLIT state and it detects the assertion of the Error Abort mode signal, it can choose to either:

1. Finish the LinkRetry_FLIT state and allow all packets to be retransmitted before sending a stream of IRTRY packets with the StartRetry flag set.
2. Pause the LinkRetry_FLIT state and send a stream of IRTRY packets with the StartRetry flag set, and then revert back to the completion of the LinkRetry_FLIT state.

HMC implementation completes the LinkRetry_FLIT state to empty the retry buffer, and then sends the stream of IRTRY packets.

LinkRetry_FLIT State Control

- Enter: When LinkRetry_Init state clears AND there are packets currently saved for retransmission.
- Actions: Retransmit all saved packets across the link. All packets retransmitted must have the latest valid FRP value received from the local link slave embedded into their RRP field.
- Exit: When all packets saved for retransmission have been sent to the link.

LinkRetry Sequence Exit

The link master exits the retry sequence from either the LinkRetry_Init state or the LinkRetry_FLIT state depending on whether there were packets saved for retransmission when the LinkRetry_Init state was exited. The normal packet stream may now resume.

Link Slave Retry Functions

The link slave section parses incoming packets, performs the CRC and sequence check for each packet, and if there are no errors, forwards the packets to the internal destination. The link slave also extracts the FRP and RRP fields from each good packet and forwards those to the local (adjacent) link master section. When a link error occurs, the link slave goes into error abort mode, which stops the forwarding of the incoming packets. It also stops the forwarding of FRP and RRP pointers to the local link master. The assertion of error abort mode causes the local link master to transmit a stream of IRTRY packets to signal to the other end to activate its link retry sequence. When a link retry is in progress, the link slave detects and counts two different types of IRTRY packets, in-

cluding those with the StartRetry flag set (FRP[0]=1) and those with the ClearErrorAbort flag set (FRP[1]=1).

Packet CRC/Sequence Check

As incoming packets flow through the link slave, CRC is calculated from the header to the tail (inserting 0s into the CRC field) of every packet. As the tail passes through, the calculated CRC is compared to the incoming CRC included in the tail of the packet. A mismatch indicates that bits in the packet have changed from when the CRC was originally generated at the link master. If the calculated CRC compares correctly, an additional packet check is provided by extracting the packet sequence number out of the tail and comparing it with the saved value of the sequence number from the last successful packet. The sequence number should always be incremented by + 1 for a correct transmission of packets across the link.

FRP and RRP Extraction

If the incoming packet passes the CRC and sequence checks, and the link slave is not in error abort mode, the link slave extracts the values of the FRP and RRP fields and forwards them to the local link master section.

Error Abort Mode

If an LNG, CRC, or sequence error occurs on an incoming packet, the link slave executes the following actions:

- The link slave sets the Error Abort mode.
- The link slave drops or poisons the packet that had the CRC or sequence error. Poisoning involves inverting the calculated CRC for the packet, so it will be dropped by a downstream receiver of the packet. This action is required because in the case of packets that may span multiple core cycles, the beginning of the packet may have been forwarded before the CRC check was performed at the tail, to minimize latency.
- During error abort mode, the link slave will drop (not forward) all subsequent packets following the packet that caused the error.
- During error abort mode the link slave normally does not extract values from the RRP, FRP, or RTC fields. There is one exception to this, which is described in the IRTRY Packet Operation section.
- Error Abort mode is cleared by monitoring the IRTRY packets as described in the IRTRY Packet Operation section.

IRTRY Packet Operation

When the link slave receives an IRTRY packet, it performs a LNG check and CRC check similar to checking any other incoming packet. The SEQ check is inhibited for IRTRY packets because they are not retried if they have a failure. A stream of multiple IRTRY packets up to a programmable threshold must be received before an action is performed so that a data payload that looks like an IRTRY packet arriving after a link error occurred doesn't falsely trigger a retry action.

Each successfully received IRTRY packet will cause the link slave to increment one of two counters as described below. This occurs even if Error Abort mode is set. No IRTRY packets are forwarded beyond the link slave. If an IRTRY packet has a LNG error or CRC error, the link slave will set error abort mode if it is not already set.

The link slave decodes two types of IRTRY packets:

1. StartRetry: IRTRY with FRP[0] = 1, indicating the StartRetry flag. In this case the link slave will increment the StartRetry counter for each StartRetry flag that it receives in a continuous stream of IRTRY packets. After the counter reaches the threshold specified in the Init_Retry Packet Receiver Number field of the link retry control register, the link slave will generate a StartRetry pulse that is sent to the local link master. Additional StartRetry flags may arrive after the threshold is met if the transmitting link master has its Init_Retry Packet Transmit Number programmed to be greater than the threshold. These excess StartRetry flags should not affect the operation and should not cause a second StartRetry pulse. If a FLIT or packet other than an IRTRY is received, the link slave will clear the StartRetry counter, which should arm the counter to begin counting again if additional IRTRY packets with the StartRetry flag set arrive later. If the link slave detects a link error and sets Error Abort mode, it will also clear the StartRetry counter. When receiving IRTRY packets with the StartRetry flag, there are two cases of RRP extraction:
    a. If error abort mode is not set, the link slave extracts an RRP out of every good packet that it receives, including all IRTRY packets with the StartRetry flag set. These RRPs are sent to the local link master.
    b. When error abort mode is set, normally the link slave does not extract any retry pointers or tokens from incoming packets. There is one exception: If Error Abort mode is set and the link slave is receiving IRTRY packets with the StartRetry flag, this is a special case where a link error occurred on both sides of the link. The link slave will count the StartRetry flags, and when reaching the threshold it will extract the RRP from the IRTRY packet that caused the threshold to be reached. The link slave will send this RRP along with the StartRetry pulse to the local link master.
2. ClearErrorAbort : IRTRY with FRP[1] = 1, indicating the ClearErrorAbort flag. In this case the link slave will increment the ClearErrorAbort counter for each ClearErrorAbort flag that it receives in a continuous stream of IRTRY packets. After the counter reaches the threshold specified in the Init_Retry Packet Receiver Number field of the link retry control register, the link slave will clear error abort mode if it is currently set. It will also extract the RRP out of the IRTRY packet that reached the receiver number threshold and send the RRP to the local link master. Additional IRTRY packets with the ClearErrorAbort flag set may arrive after the threshold is met if the transmitting link master has its Init_Retry Packet Transmit Number programmed to be greater than the threshold. Specific logical implementation may require more IRTRY packets with the ClearErrorAbort flag set to be transmitted than the link slave is expecting. These excess ClearErrorAbort flags will not affect the operation as long as Error Abort mode clears before a retried packet is received. The link slave should follow normal procedure and extract all RRPs after error abort mode clears and send them to the local link master. If a FLIT or packet other than an IRTRY is received, the link slave will clear the ClearErrorAbort counter, which should arm the counter to begin counting again if additional IRTRY packets with the ClearErrorAbort flag set arrive later. If the link slave detects a link error on any IRTRY packet with ClearErrorAbort flag set when error abort mode is clear, it should set error abort mode and also clear the ClearErrorAbort counter.

Resumption of Normal Packet Stream after the Retry Sequence

At some point after the retry sequence, a normal packet (either a retried one or a new one) will be received by the link slave. The link slave operates normally on that packet, verifying the LNG, CRC, and sequence number fields and forwarding it on to its internal

destination. Because the link slave captured a valid sequence number from the last good packet received before the retry sequence (no sequence numbers were extracted from the IRTRY packets), it will compare that sequence number with the first one received after the retry sequence. The sequence number should be incremented by 1 if the retry sequence was successful and the normal packet stream resumed correctly.

Retry Pointer Loop Time

Table 33: Retry Pointer Loop Time Implementation Example

Link		Retry Buffer Full Period (ns)	HMC Delay (ns)	Host Allowable Delay (ns)	Recommended Retry Timer Value (ns)	Recommended Timeout Period Encode Value
Bit Rate	ps/FLIT					
10 Gb/s	800	153.6	26.5	127.1	> 450	4
12.5 Gb/s	640	163.8	25.9	138.0	> 500	4
15 Gb/s	533	136.4	22.3	114.2	> 400	4

HMC protocol includes retry pointer space to support a retry buffer that can hold up to 256 FLITs. In order to maintain link performance, the maximum allowable retry pointer loop time must be less than the retry buffer full period shown in the table. If the maximum retry pointer loop time was allowed to be greater than the time it takes to fill the retry buffer, the retry buffer full condition would throttle the packet stream. The result would be NULL FLITs sent between transaction packets, degrading the performance of the link. To avoid link throttling, the host's portion of the retry pointer loop delay must be less than the "Host Allowable Delay" shown in the table.

The following implementation example lists the measurements used to identify the total retry pointer loop time:

- The time it takes for a packet (and its FRP) to travel from a link master to a link slave, including:
  - Serialization time at the transmitting end of the link
  - Deserialization time at the receiving end of the link
- The time for the longest packet's tail to be received in the link slave, to the point of CRC check and FRP extraction
- The transfer of the extracted FRP from the link slave to the local link master
- Time to wait in the link master for just missing embedding the retry pointer into the RRP field of the previous packet and waiting for the next tail of the longest packet
- The time it takes for a packet with the embedded retry pointer to travel back to the source end of the link, including:
  - Serialization time at the transmitting end of the link
  - Deserialization time at the receiving end of the link
- Time for the longest packet's tail to be received in the link slave, to the point of CRC check and RRP extraction
- The transfer of the RRP from the link slave to the local link master (the original transmitter in the first step)

Link Flow Control During Retry

When the normal packet stream is transmitted by the link master, it has already determined that the link slave has enough buffer space to accept all FLITs of the transmitted packets, as controlled by the return token count protocol. If a link error occurs and the packet stream is aborted during error abort mode in the link slave, the reserved space in the receive buffers still exists, and therefore, when those packets are retransmitted across the link, there is no need for any additional flow control, except for the following clarification. At the receiver, the link slave may poison the packet that has errored, but it has already propagated it forward into the receive buffer. Because this takes up some FLIT locations, the return token count control logic must not return the tokens for a poisoned packet. That packet will be retransmitted, and when it is finally loaded into the receive buffer with good CRC and propagated further, the tokens will be returned. All subsequent packets (after the errored one) dropped in the link slave will not be forwarded into the receive buffer. As a result, the buffer space will be available for those packets as they are retransmitted.

A consequence of forwarding the poisoned packet and not returning tokens until the packet is retransmitted successfully is as follows: The poisoned packet could have been the last packet that caused the full condition at the receive buffer, and then the receive buffer could be temporarily stalled due to downstream flow control. If this were the case, the failed packet would be the only packet to be transmitted. When the packet was retransmitted, there would be no room for that packet in the receive buffer. An easy solution to this is to adjust the maximum tokens available for that receive buffer to 9 FLITs less than the actual buffer size. This accommodates an extra copy (the poisoned one) of the largest packet that could be retransmitted during a link retry.

Example Implementation Link Error and Retry Sequence

The following example refers to Figure 23 (page 62).

The green link master, upper left, transmits packets across the link to the green link slave in the upper right. The blue link master, lower right, transmits packets across the link to the blue link slave in the lower left.

1. Successful green packets, flowing from the green link master to the green link slave across the top of the diagram, have green FRPs and blue RRPs embedded in them. Successful blue packets, flowing from the blue link master to the blue link slave across the bottom of the diagram, have blue FRPs and green RRPs embedded in them.
2. An error occurs in the green packet stream across the top. The green link slave detects CRC error and sets error abort mode. The errored packet is poisoned and all following packets are dropped.
3. Flow of FRPs and RRPs stops from the green link slave to the blue link master.
4. The blue link master receives the error abort mode signal from the green link slave, suspends its normal packet stream from its output FIFO, and inserts a stream of IRTRY packets each with the StartRetry flag set (FRP[0] = 1). The blue link master embeds the latest FRP it received from the green link slave (via its retry pointer collection register) into the RRP field of each IRTRY packet. The value embedded into the RRP field is the same as the FRP for the last good packet received at the green link slave before the link error occurred. After sending the stream of IRTRY packets, the blue link master resumes its normal packet stream from its output FIFO.

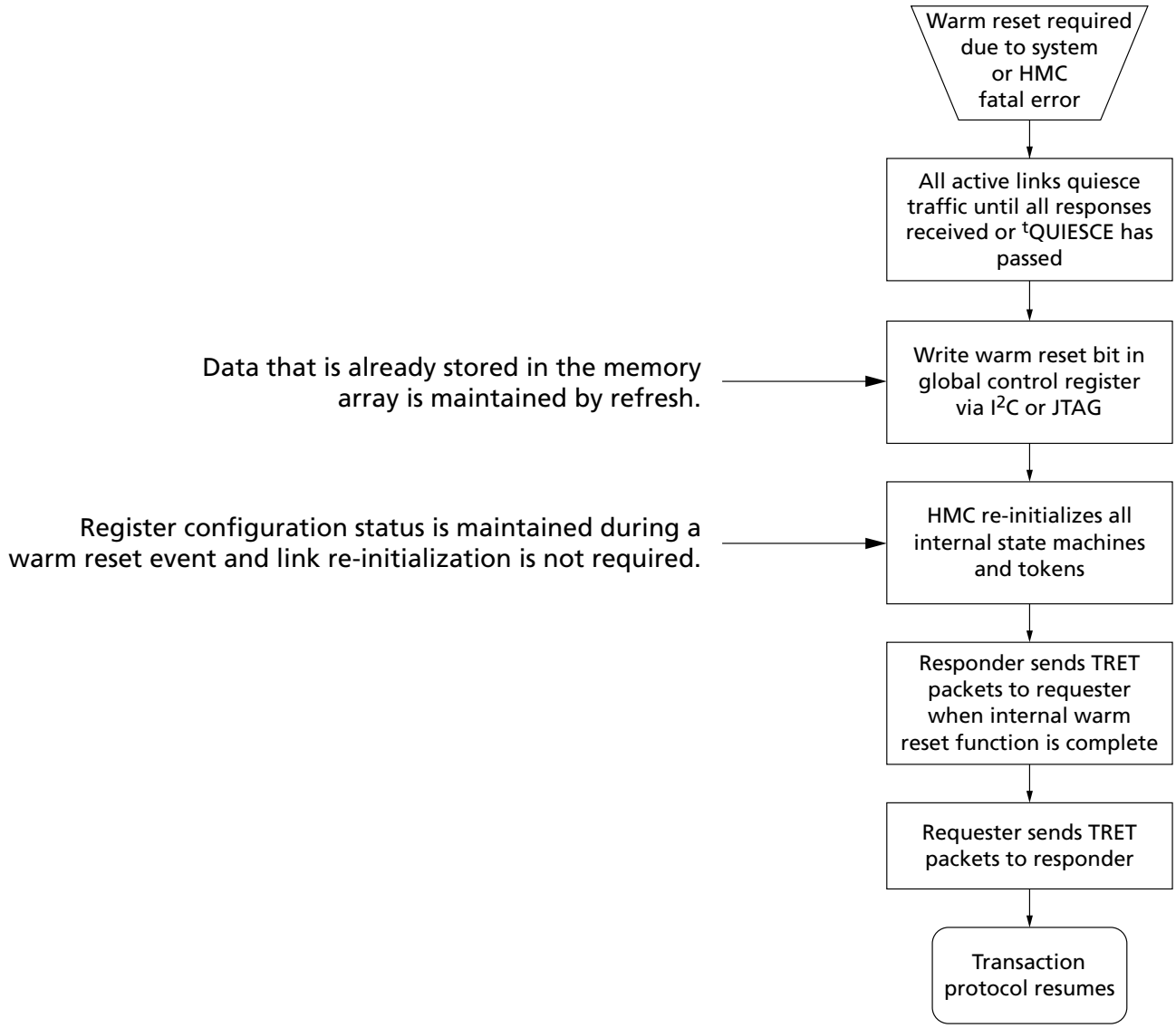
5. The blue link slave detects the IRTRY stream and counts the StartRetry flags. During this time it extracts the RRP and sends it to the green link master, which updates its retry buffer read pointer to point to the packet that had errored. When the IRTRY receive threshold is reached, the blue link slave sends a StartRetry pulse to the green link master.
6. The blue link slave continues to receive normal packets, forwarding them to its input FIFO, and extracts all RRPs, FRPs, and RTCs.
7. The green link master receives the StartRetry pulse that starts its retry sequence, enters into its LinkRetry_Begin state, finishes sending the current packet being transmitted from its output FIFO and then stops. That packet is the last packet that gets loaded into the retry buffer.
8. The green link master enters its LinkRetry_Init state and transmits a stream of IRTRY packets, each with the ClearErrorAbort flag set (FRP[1]=1). During this time the green link master is receiving valid FRPs from the blue link slave (via its retry pointer collection register), and it embeds those into the RRP field of every IRTRY packet.
9. The green link slave, currently in Error Abort mode, detects the stream of IRTRY packets and counts the ClearErrorAbort flags. When it reaches the IRTRY receive threshold, it clears Error Abort mode, begins extracting the RRPs from the IRTRY packet(s) and sends them to the blue link master.
10. The green link master enters its LinkRetry_FLIT state and transmits the pending packets from its retry buffer until its read pointer equals its write pointer. It continues to embed the latest value of its retry pointer collection register into the RRP field of each of the pending packets being retransmitted. As the pending packet stream completes from the retry buffer, the green link master switches to new packets waiting in its output FIFO.
11. The green link slave receives the retried packet stream and resumes the sequence check (along with CRC check). The first retried packet received should have a sequence number +1 greater than the last successfully received packet before the link error occurred. All successfully received retried packets, followed by new packets, will be forwarded to the link input FIFO.

Warm Reset

Warm reset is used to clear and restart the transaction layer without affecting the state of the link. The need for a reset event is signaled by a fatal error, in which case HMC will communicate through an Error Response Packet with a Fatal ERRSTAT code (if possible) and by the assertion of the FERR_N signal. Warm reset should not be performed prior to an INIT CONTINUE command. Warm reset will complete within ^{t}INIT.

To ensure that the transaction layer restarts properly following a warm reset, link traffic must be quiesced such that all requests have received a response or ^{t}QUIESCE has passed since the last request was issued. At that time, the host and HMC must be in the state described in Step 12 of the Power-On and Initialization section. All used links remain in a trained active state and will send NULL FLITs during warm reset. The host must not issue any requests until normal transaction protocol has resumed.

Figure 25: Warm Reset Flow Chart



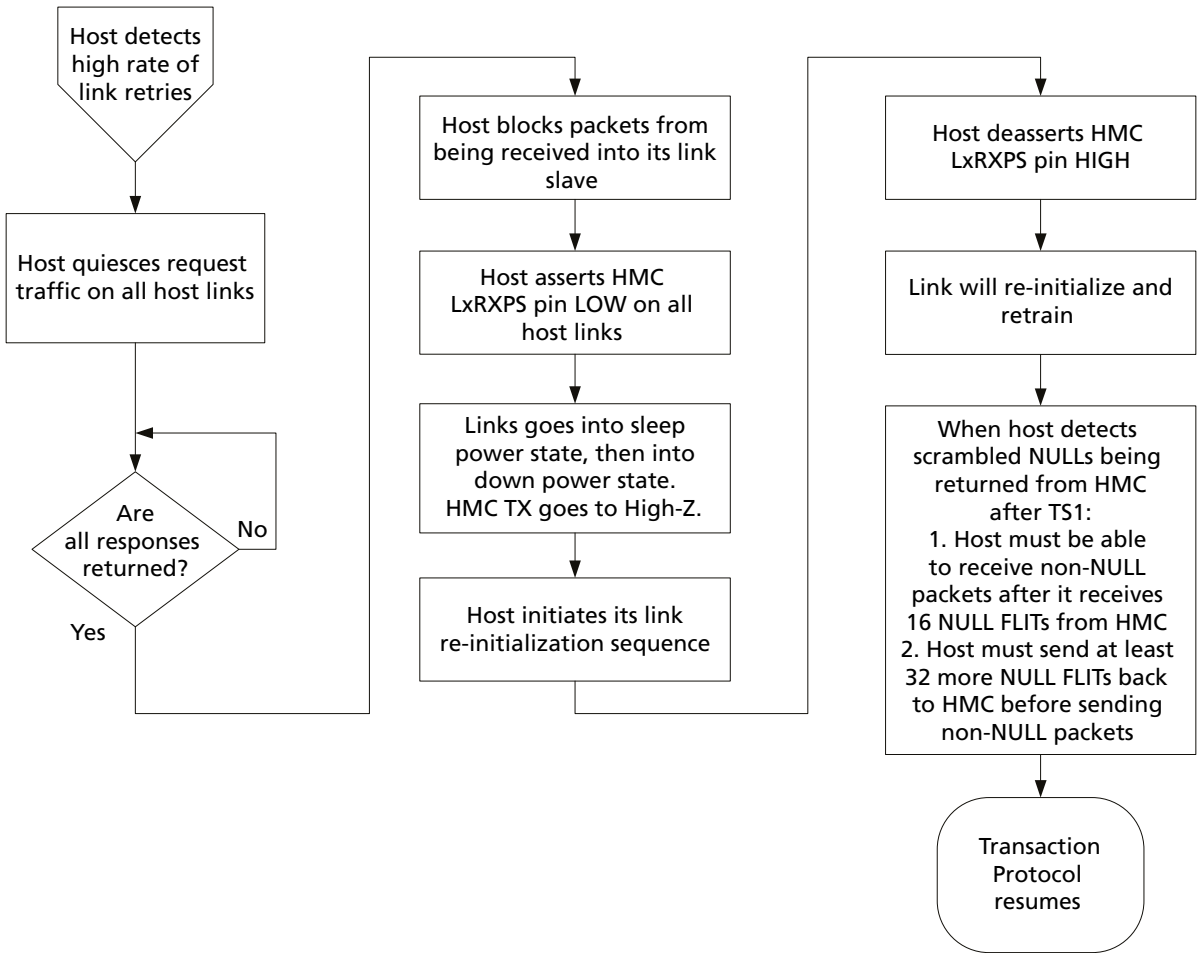
Retraining and Recovery

Retraining a Link With High Error Rate

If link bit errors are occurring at a higher than desired rate, but at a low enough rate that retry sequences are successful, the bit error rate may be reduced by retraining the link. The host can monitor the frequency of link retry occurrences for links connected directly to the host as by simply counting the occurrence of a link error detected on its end of the link (host RX errors) or count the receipts of Error Responses (ERRSTAT= Link Retry in Progress) as sent by the connected HMC. In chaining configurations where a link is not connected directly to the host, the host can monitor receipts of Error Responses

(ERRSTAT= Link Retry in Progress) that originate from the cube on either end of a link. The host can then initiate a retraining procedure when it determines the link error rate is above an acceptable threshold. The steps for retraining a device for these situations are outlined in Figure 26.

Figure 26: Retraining Due to Link With High Error Rate

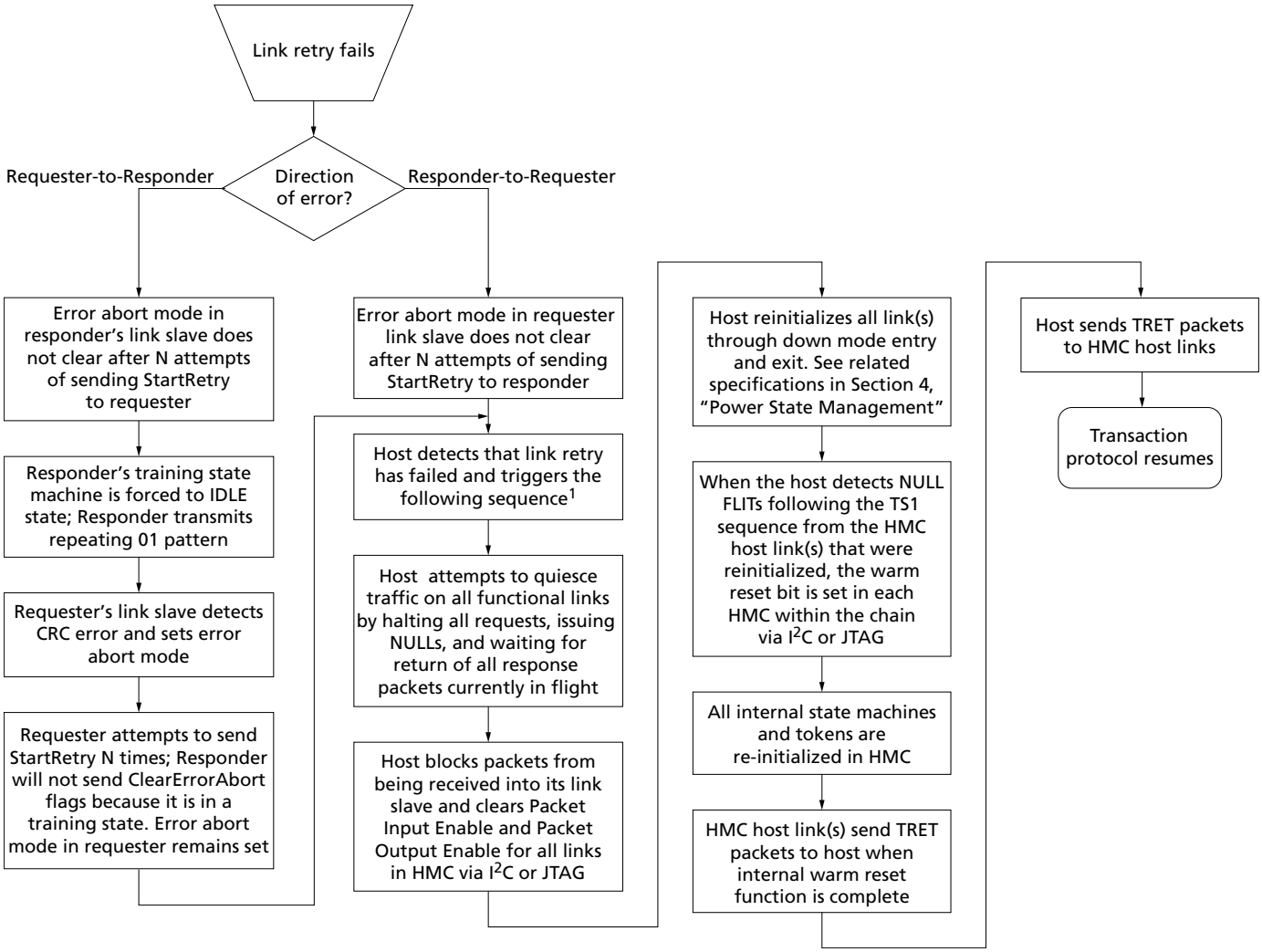


Note: 1. All links must be configured with the Inhibit Link Down field of the Link Control register set to '0' in order to enable successful retraining.

Host Recovery After Link Retry Fails

If HMC reaches the retry limit as set in the Link Retry Register, a fatal error will occur and recovery is required. In order for a host to recover from this state, retraining the SerDes, warm reset and reinitialization of the link are required. The specific flow and actions required are shown in Figure 27.

Figure 27: Host Recovery After Link Retry Fails



Notes:
1. Host detection of link retry failure on link not directly connected to host will occur via an Error Response packet from a downstream HMC.
2. All links must be configured with the Inhibit Link Down field of the Link Control register set to '0' in order to enable successful retraining.

Functional Characteristics

Packet Ordering and Data Consistency

There can be multiple packet-reordering points within HMC, including the following implementation examples:

1. The output of each link slave input buffer could reorder packets arriving from one link that are destined for different vaults. This would prevent a busy vault from blocking the packet flow out of the receive buffer and enable the flow of packets across the link to continue.

2. Each vault controller command queue could reorder packet execution to banks that are not busy, potentially optimizing the memory bus usage.

All reordering points within HMC will maintain the order of a stream of request packets from a specific link to a specific bank within a vault. This ensures that a stream of writes and reads within a bank maintains its order of execution at the memory. Under these conditions, the order of read response packets returned to the requester will be maintained with respect to other read response packets in this stream. Additionally, the order of write response packets with respect to other write response packets within this stream will be maintained under these conditions. However, there is a possibility that the relationship between the order of write responses with respect to read responses may not be maintained.

If a host sends a write request packet to a given address, followed by a read request packet to the same address, the read will always return the newly written data.

Packet ordering performed by HMC supports messaging or data-passing programming between two or more hosts with access to the same HMC. This common method of host-to-host messaging is known as producer-consumer programming and proceeds as follows:

1. Host 0 wants to pass a message/data to host 1. Host 0 writes memory location A (this could be multiple locations) with the message/data.
2. Upon receiving all expected successful write responses for the write(s) to location A, host 0 sets a flag in memory location B indicating that the message/data is available in location A.
3. Host 1 executes a loop, polling (that is, issuing read requests to) location B to observe the set flag.
4. Upon detecting the set flag in the read response for location B, host 1 issues a read request to location A.
5. Host 1 receives the read response for location A containing the newly-written data from host 0.

This scenario proceeds as defined if host 0 waits for the successful write response for the write to location A before issuing the write request packet to location B. The write response is generated at the vault controller after the write request is fully executed and is identified with the same tag as the write request. Locations A and B can be any memory locations. This scenario is not supported with the use of posted write requests.

Data Access Performance Considerations

Due to the internal 32-byte granularity of the DRAM data bus within each vault in HMC, inefficient utilization of this bus occurs when starting or ending on a 16-byte boundary rather than a 32-byte boundary. An extreme example of this would be the host issuing a series of 16-byte read requests. Each read request would fetch 32 bytes from the DRAM and return half of the data in the response packet, throwing away the other 16 bytes of data. For bandwidth optimization it is advantageous to issue requests with 32-byte granularity.

If the host issues a series of commands that access portions of the same data block (as defined by the maximum block size) rather than issuing one request accessing the entire data block, two performance disadvantages exist:

1. The probability of bank conflicts will increase in the accessed vault.
2. The multiple request and response packets have additional header/tail overhead on the link.

Because of this, commands with larger data sizes should be used. For example, a host defines the maximum block size as 128-bytes and issues four consecutive 32-byte requests to the same block location (instead of issuing a 128-byte request to a specific location). The cube will send these four independent requests to the specific bank, opening and closing the row in the same bank four times, thereby reducing bandwidth utilization and increasing latency when compared to a single 128-byte request being issued.

To maximize bandwidth utilization and reduce system latency the following should be considered:

1. Request data block sizes that are as large as possible, up to the configured maximum block size, that meet the requirements of the system.
2. Configure the maximum block size based on the most frequently requested data size. If the most frequently requested data size does not equal any of the possible maximum block sizes (32B, 64B, or 128B), select the smallest block size that includes the most frequently requested data size. For example, if a system most often requires 48B data transfers, 64B should be chosen as the maximum block size.

Vault ECC and Reference Error Detection

Memory READ and WRITE operations have a parity bit added to the command and address signals that are sent from each vault controller to the memory partitions and banks associated with each controller. If the bank receiving a request determines that there is a parity error, the reference is retried.

Data is protected in memory using ECC. The ECC should provide correction of a single-bit error as well as all data bits that are transmitted over a single, failed TSV between the memory and the logic base.

Memory locations are initialized with correct ECC data patterns before the host begins using the locations. This eliminates a write-before-read requirement to avoid SBEs and MUEs.

Refresh

DRAM refresh is handled internally within HMC by the vault controllers. As an example implementation, every vault controller could have a unique refresh rate for each of its bank groups (where a bank group is a set of banks within one memory die). An implementation may also choose to vary refresh rates dynamically, driven by conditions such as temperature and/or detected memory error rates.

Scrubbing

Scrubbing is used inside the cube to mitigate soft failures in memory. Two categories of scrubbing are provided in each vault controller:

- **Demand scrubbing:** If a correctable error is detected in the read data of a read request, the data is corrected and returned to the host in the response packet. A scrubbing cycle is invoked that writes the corrected data back into the memory array.
- **Patrol scrubbing:** This form of scrubbing is performed by the cube refresh logic. If a correctable error is detected, a scrubbing cycle is invoked that writes the correct data back into the memory array.

The scrubbing cycle writes the corrected data immediately after the correctable error is discovered on the read. This READ-UPDATE-WRITE operation is atomic, meaning that

no other reference to the location will be possible between the read and the associated write. This is handled as a bank conflict within the vault controller command queue.

Response Open Loop Mode

During normal transaction layer packet flow, return token counts (RTCs) are returned in both directions on each link. This controls the flow of request packets in one direction and response packets in the other direction. Response open loop mode can be configured on the upstream links that are connected to the host. This mode eliminates the requirement for the host to send tokens back to the cube and results in two simplifications:

- The host link slave does not have to generate tokens when response packets are consumed.
- The host link master does not have to embed any RTCs into request packets being transmitted to HMC.

The response open loop mode will force the link master to send response packets to the host without the token requirements. (Other packet flow controls such as retry buffer full conditions and retry in progress will still suspend packet flow if necessary.)

HMC Gen2 Electrical Specifications

Absolute Maximum Ratings

Stresses greater than those listed may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions outside those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may adversely affect reliability.

Table 34: Absolute Maximum Ratings

All voltages are referenced to V_{SS}

Description	Parameter	Minimum	Maximum	Unit	Notes
Storage temperature	T_{STG}	–40	140	°C	1
Logic core source	V_{DD}	–0.4	1.15	V	
PLL sources	$V_{DDPLLA[3:0]}$, $V_{DDPLLB[3:0]}$, V_{DDPLLR}	–0.5	1.4	V	
Link termination sources	V_{TT}, V_{TR}	–0.5	1.5	V	
DRAM source	V_{DDM}	–0.4	1.7	V	
DRAM wordline boost source	V_{CCP}	V_{DDM}	2.75	V	2
JTAG, NVM, I^2C source	V_{DDK}	–0.4	1.95	V	
Link input signal pin voltage (LxRXP, LxRXN)	Link input signal pin voltage (LxRXP, LxRXN)	More positive of –0.1V or V_{TR} - 1.1V	V_{TR} + 0.3V	V	
Link output signal pin voltage (LxTXP, LxTXN)	Link output signal pin voltage (LxTXP, LxTXN)	More positive of –0.1V or V_{TT} - 1.1V	V_{TT}	V	

Notes:
1. HMC case temperature, nonoperating (no supply voltages applied).
2. V_{CCP} level must never drop below V_{DDM}, including during power-up/down or power failure events.

AC and DC Operating Conditions

The limits specified in the following table include both DC and AC components. Specifications within the table represent limits that must be met at HMC's package balls. All voltages are referenced to V_{SS}.

Table 35: Recommended Supply Operating Conditions

All voltages are referenced to V_{SS}

Description	Parameter	Minimum (V)	Nominal (V)	Maximum (V)	Notes
Logic core source	V_{DD}	0.873	0.9	0.927	1
Logic core source DC RMS	V_{DD_DC}	0.886	0.9	0.914	
Link TX termination source	V_{TT}	1.14	1.2	1.26	2
Link RX termination source	V_{TR}	1.14	1.2	1.26	2
Link PLLA source	$V_{DDPLLA[3:0]}$	1.14	1.2	1.26	2
Link PLLB source	$V_{DDPLLB[3:0]}$	1.14	1.2	1.26	2

Table 35: Recommended Supply Operating Conditions (Continued)

All voltages are referenced to V_{SS}

Description	Parameter	Minimum (V)	Nominal (V)	Maximum (V)	Notes
Intermediate frequency PLL source	V_{DDPLLR}	1.14	1.2	1.26	2
DRAM source	V_{DDM}	1.14	1.2	1.26	1
DRAM source DC RMS	V_{DDM_DC}	1.17	1.20	1.23	
DRAM wordline boost source	V_{CCP}	2.375	2.5	2.625	3
JTAG, NVM, I^2C source	V_{DDK}	1.45	1.5	1.7	

Notes:
1. Complete voltage range, including both AC and DC components.
2. In addition to restricting the overall voltage to the specified range, the AC noise amplitude must be less than 50mV peak-to-peak.
3. V_{CCP} level must never drop below V_{DDM}, including during power-up/down or power failure events.

Table 36: Leakage Currents

Test conditions: All voltage sources at nominal levels, $0 \le V_{IN} \le V_{DDK}$; Sink current into the device pin is positive (maximum); Source current out of the device is negative (minimum)

Description	Parameter	Minimum	Maximum	Unit	Notes
Input Leakage Current, Logic-only pins: P_RST_N, SCL, SDA, TMS, TCK, TDI, TRST_N.	I_{LLO}	–10	240	µA	
Input Leakage Current, Logic plus DRAM pins: LxRXPS, REFCLK_BOOT[1:0], REFCLK_SEL, CUB[2:0], and FERR_N (1)	I_{LLD}	–10	700	µA	

Note:
1. Only the maximum value applies to the open-drain FERR_N pin. The FERR_N pin in the HIGH state (open-drain) is pulled high externally, and the maximum value in the table represents the maximum leakage into the pin. When FERR_N is in the LOW state, an internal output driver circuit drives the pin low, so therefore the minimum leakage value does not apply.

Table 37: Operating Temperature Range

Description	Parameter	Minimum	Maximum	Unit	Notes
DRAM operating temperature	T_J	0	105	°C	
Logic operating temperature	T_J	0	110	°C	
Change in DRAM or logic operating temperature	Delta-T_J/Delta-time	–	10	°C/minute	

Values in the following table are the maximum current on each power supply rail during the time between power-on and the completion of ^{t}INIT. For information on generating power/current estimates for normal operation (after ^{t}INIT), refer to the HMC System Power Calculator and HMC Gen2 User Guide. Both are available on micron.com.

Table 38: Power-On Currents

Conditions: 105°C T_{J_dram}, 110°C T_{J_logic}, maximum voltage for each supply

Condition	Power Supply								Unit
	V_{DD}	V_{DDPLLA} + V_{DDPLLB}	V_{DDPLLR}	V_{TT}	V_{TR}	V_{DDM}	V_{CCP}	V_{DDK}	
PLL configuration and lock – During P_RST_N until completion of ^{t}INIT	8	2.60	0.1	2.14	3.60	0.57	0.20	0.04	A

HMC-15G-SR Physical Link Specifications

Physical Link Electrical Interface

HMC transmitter and receiver circuits are designed to allow for flexible integration with the host. Various supported configurations are shown in Termination Configurations below. Both HMC RX and TX circuits are meant to be connected to their respective host lanes through the use of 100Ω differential line impedance. The SR PHY supports channels consistent with existing standards (OIF CEI-11-SR, INCITS FC-PI-5, IEEE nAUI).

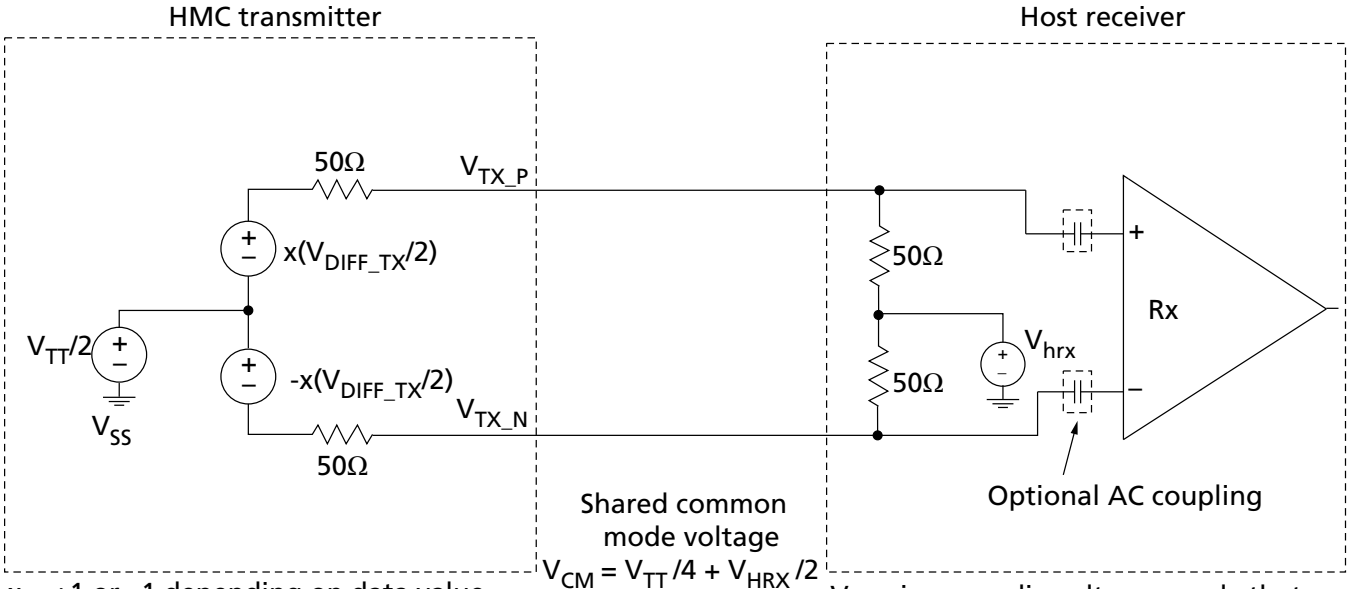
Termination Configurations

The impedance values shown are nominal. Host transmitter and receiver termination and signaling parameters may differ from those of HMC as long as HMC receiver parameters are still met as per the TX Signaling Parameters and RX Signaling Parameters tables.

HMC receivers are internally AC-coupled, removing the need for external AC coupling capacitors in the channel. However, as illustrated in the configuration in Figure 32 (page 88), HMC receivers can support channels that have external AC coupling present. The PHY Configuration ERI must be set appropriately, based on the use of external AC or DC coupling (see the HMC Register Addendum for further details). Baseline wander compensation circuitry within the receiver protects against the effects of long DC run lengths. HMC has been designed with its scrambling circuitry to minimize the probability of long DC run lengths. The host transmitter must meet the requirements found in the Lane Run Length Limitation section. Additional run length limit circuitry, used to avoid run lengths of longer than 85 UI at HMC's transmitter output, is also available (see Lane Run Length Limitation). Specific host AC-coupled receiver encoding requirements beyond this, such as 64b/66b encoding or 8b/10b encoding, are not supported.

Micron strongly recommends that HMC transmitter-to-host receiver paths have AC coupling capacitors, either internal or external to the host receiver. If AC coupling is not used, the system design must adhere to the specific conditions outlined in the examples below.

Figure 28: HMC TX Driving Host RX – Example #1



x = +1 or –1 depending on data value

V_{TT} = Transmit termination source (measured at HMC pin)

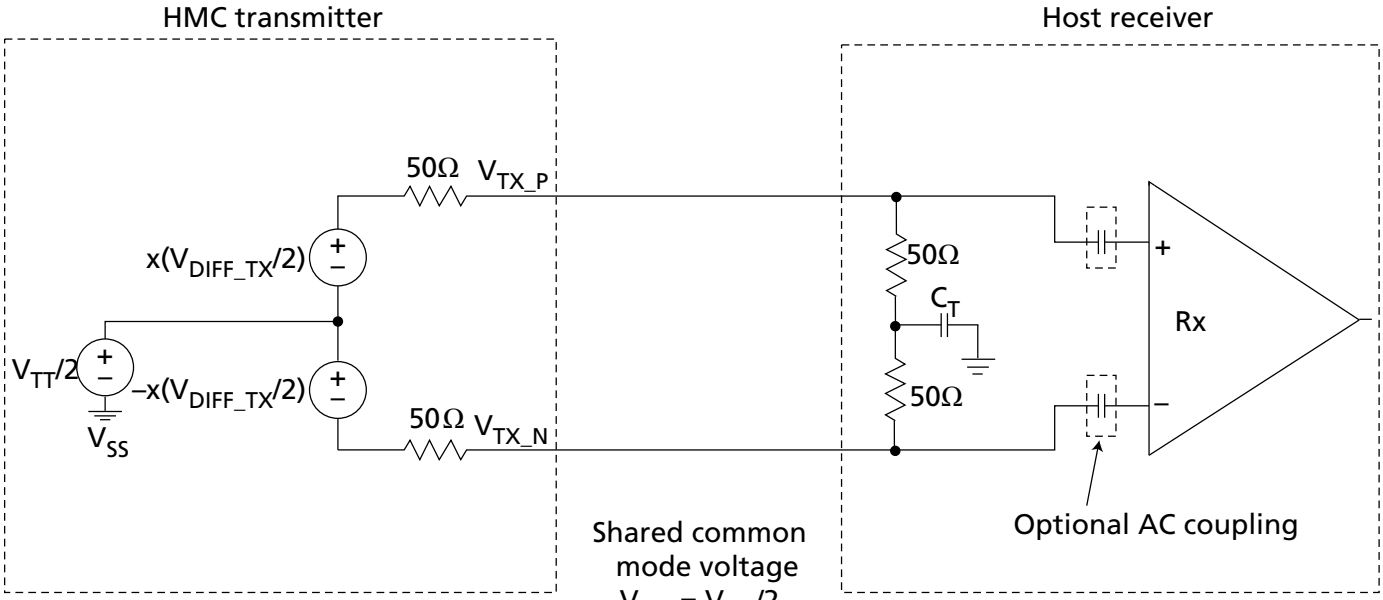
V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

V_{hrx} is an on-die voltage supply that must equal one of the following options:

1) V_{TT} /2 (±50mV tolerance)
2) V_{TT} - 200mV < V_{hrx} < V_{TT} - 100mV[1]

Note: 1. Implementation of Figure 28 (page 84) option #2 requires that the junction temperature of HMC logic die be restricted to a maximum of 100°C.

Figure 29: HMC TX Driving Host RX – Example #2



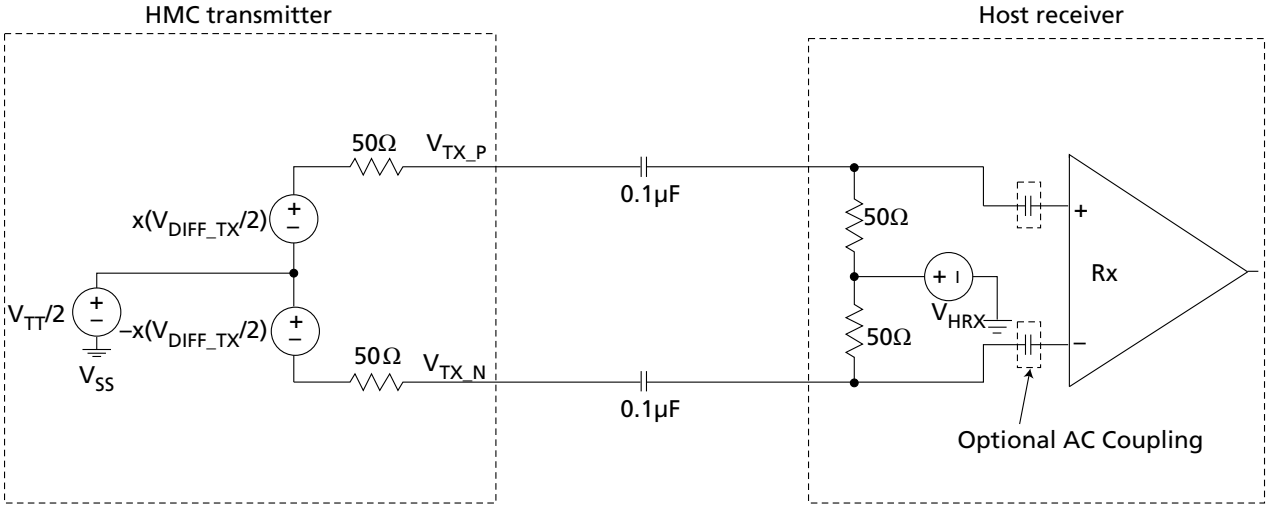
x = +1 or -1 depending on data value

V_{TT} = Transmit termination source (measured at HMC pin)

V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

Capacitor (C_T) recommended to control Rx common-mode return loss

Figure 30: HMC TX Driving Host RX With External AC Coupling Present



x = +1 or -1 depending on data value

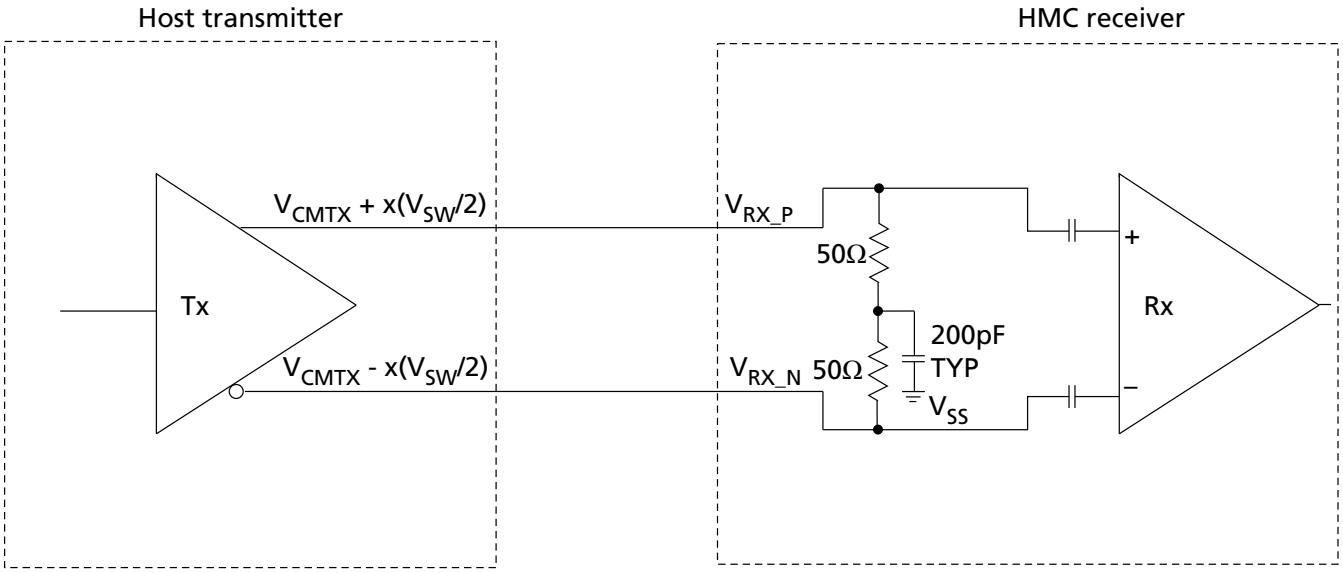
V_{TT} = Transmit termination source (measured at HMC pin)

V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

Tx common mode voltage (V_{CMTX}) = V_{TT}/2

Value of V_{HRX} depends on the host receiver design
Rx common mode voltage (V_{CMRX}) = V_{HRX}

Figure 31: Host TX Driving HMC RX Without External AC Coupling



x = +1 or –1 depending on data value

V_{CMTX} = Transmit common mode voltage

V_{SW} = Voltage swing

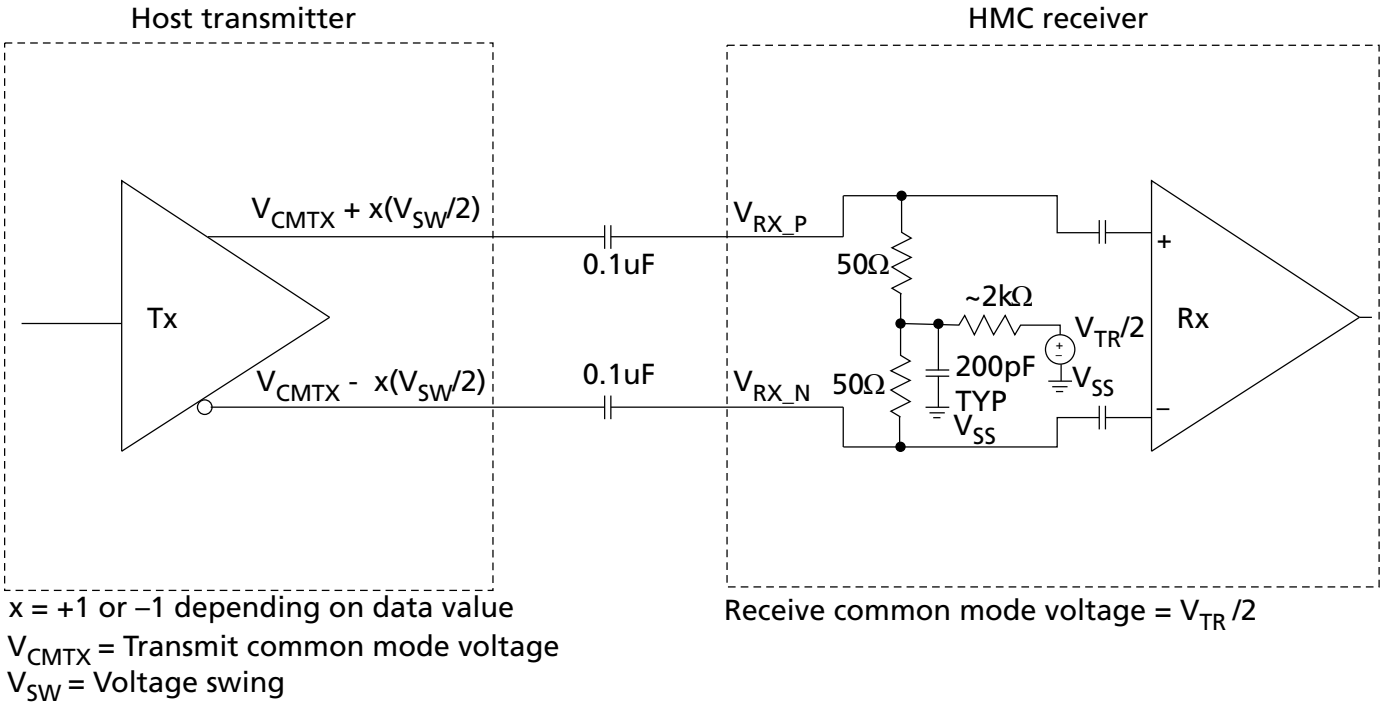
Receive common mode voltage = V_{TR} /2

Shared common mode voltage

$V_{CM} = V_{CMTX}$

Note: 1. TX output impedance is 50Ω nominal (not shown).

Figure 32: Host TX Driving HMC RX With External AC Coupling



Note: 1. TX output impedance is 50Ω nominal (not shown).

Equalization Schemes

HMC uses equalization schemes on both its transmitter and receiver to mitigate signal integrity issues that arise from media losses, crosstalk, and intersymbol interference.

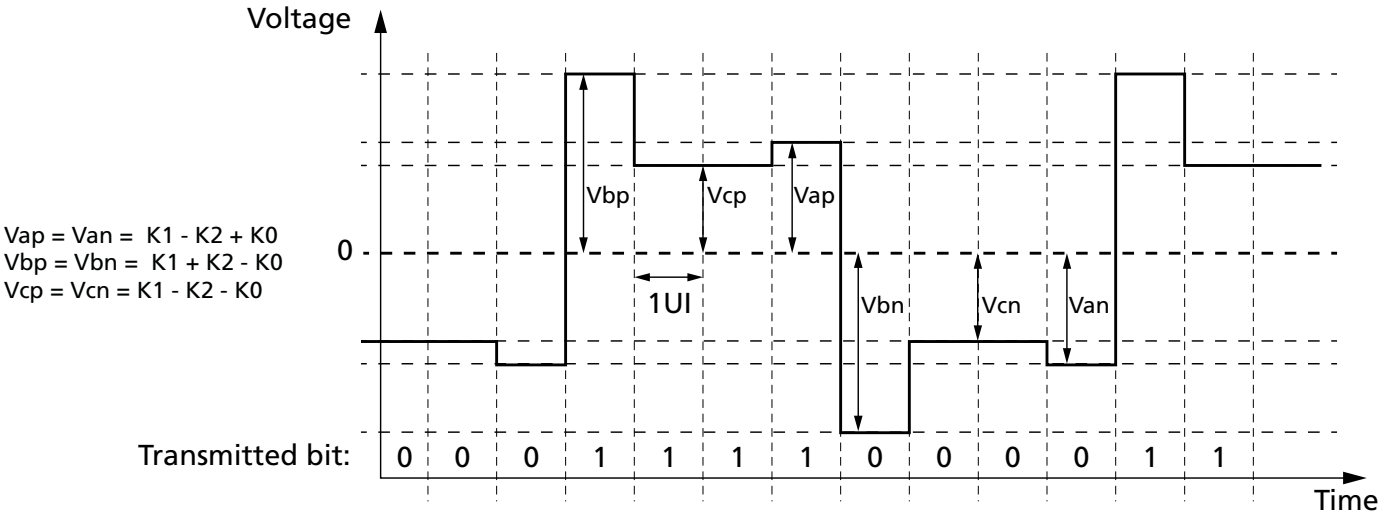
HMC's transmitter implements feed forward equalization (FFE) using a programmable three-tap, baud-spaced finite impulse response (FIR) driver with the following equation: $H(Z) = K0z^{+1} + K1z^{0} + K2z^{-1}$.

The driver amplitude K1 is fixed at 1000 mV to achieve the lowest current consumption. The relative weights of K0 to K2 are user-configurable to create a wide variety of transmitter FIR pulse shaping filters.

The receiver provides a powerful combination of an automatic gain control (AGC) amplifier with dynamic peaking control (DPC) plus a baud-spaced decision feedback equalizer (DFE) circuit that complements the transmitter equalization capability. The adaptation circuit examines the incoming serial stream and dynamically adjusts to maximize the internal eye opening.

Using the equalization capabilities of the high-speed SerDes cores, channel losses up to 12dB at the fundamental frequency are readily overcome and a BER <1E-12 can be achieved provided a fully compatible SerDes is used by the host. Lower BER may be achieved upon careful signal integrity analysis.

Figure 33: Digital Waveform With Pre-Tap (K0) and Post-Tap (K2)



Link Bit Rate

HMC link interface is synchronous on both the upstream and downstream lanes. It is not plesiochronous, therefore the source for REFCLK must be the same for both ends of the link. The choices for link bit rate are shown in the table below. Any fixed skew is allowable between the reference clock at the host and HMC. Any jitter variation below approximately 20 MHz is indistinguishable from sinusoidal jitter and will be subtracted from the amount allowed in Figure 34 (page 93).

Table 39: Synchronous Link Bit Rate Specifications

Parameter	Symbol	Test Conditions/Comments	Min	Typical	Max	Unit	Notes
Bit rate	BR10	^{f}REFCLK = 125.00 MHz	10.00	–	10.00	Gb/s	1
Bit rate	BR12.5	^{f}REFCLK = 125.00 MHz	12.50	–	12.50	Gb/s	1
Bit rate	BR15	^{f}REFCLK = 125.00 MHz	15.00	–	15.00	Gb/s	1
Bit rate range	BRR		0	–	0	ppm	2

Notes:
1. The link interface is synchronous. In this table, the bit rate is exactly 80, 100, or 120 times the reference clock frequency. See the Reference Clock Parameters table for alternative reference clock frequencies, as any of the three supported REFCLK frequencies can be used for any of the three supported bit rates.
2. Satisfying the 0 ppm BRR parameter requires that the reference clock be common at both ends of the link; between host and HMC or between two HMCs if chained. This does not refer to the reference clock frequency tolerance parameter specified in the Reference Clock Parameters table.

High-Speed Signaling Parameters

All parameters within Table 40 and Table 41 represent those of HMC. Host transmitter and receiver termination and signaling parameters may differ from those of HMC as long as the parameters within the tables are still met for HMC.

Table 40: TX Signaling Parameters

Parameter	Symbol	Test Conditions/Comments	Min	Typical	Max	Unit	Notes
Link supply TX termination voltage	V_{TT}	1.2V ±0.06V	1.14	1.2	1.26	V	
Differential peak-to-peak output voltage	V_{DIFF_TX}	$V_{DIFF\_TX} = 2 \times \lvert V_{TX\_P} - V_{TX\_N} \rvert$ Measured with slow square wave (64 zeros followed by 64 ones) with ideal 100Ω floating differential termination	860mV	–	$1.1 \times V_{TT}$	mV_{PP}	
Single-ended voltage (with respect to V_{SS}) on V_{TX_P}, V_{TX_N}	V_{TX_SE}	Active mode	$0.2 \times V_{TT}$	–	$0.8 \times V_{TT}$	mV	
Down or sleep mode output voltage	V_{TX_PD}		–	High-Z	–	mV	1
DC common mode output voltage	V_{TX_CM}	V_{TX_CM} = DC(avg) of $\lvert V_{TX\_P} + V_{TX\_N} \rvert$	–	See Note 4	–	mV	2
AC common mode noise	$V_{TX\_CM\_NZ\ SQUARE\ WAVE}$	1000mV differential peak-to-peak slow square wave output amplitude	–	–	50	mV_{PP}	
	$V_{TX\_CM\_NZ\ PRBS}$	PRBS data pattern	–	–	15	mV_{PRMS}	
Short circuit current	I_{SHORT_TX}	Output pins shorted to GND or each other	–50	–	50	mA	
Differential TX output rise/fall time	$^{t}TX\_RISE$, $^{t}TX\_FALL$	Measured from 20% to 80% into ideal 100Ω load	18	–		ps	
Differential transmitter resistance	R_{OD}		80	100	120	Ω	3
Single-ended transmitter resistance mismatch	R_{OSE}		40	50	60	Ω	3
Single-ended transmitter termination mismatch	$R_{TX\_DELTA\_AC}$	AC-coupled or DC-coupled ($V_{HRX} = V_{TT}/2$ ±50mV); Measured within a differential pair	–	–	5	%	
	$R_{TX\_DELTA\_DC}$	DC-coupled (V_{TT} - 200mV < V_{HRX} < V_{TT} - 100mV); Measured within a differential pair	–	–	25	%	4, 5

Table 40: TX Signaling Parameters (Continued)

Parameter	Symbol	Test Conditions/Comments	Min	Typical	Max	Unit	Notes
Output return loss – Relative to 100Ω differential system	$R_{LTX-DIFF}$	50 MHz – 2.8 GHz	–	–	–12	dB	
		2.8 GHz – 4.25 GHz	–	–	–8.15 + 13.3 × log (f/5.5 GHz)	dB	
		4.25 GHz – 7.5 GHz	–	–	–10	dB	
		7.5 GHz – 15 GHz	–	–	–10 + 1.21 × (f - 15 GHz)	dB	
		15 GHz – 16.5 GHz	–	–	–10 + 1.21 × (f - 15 GHz)	dB	
Common mode return loss – Relative to ideal 25Ω impedance	R_{LTX-CM}	50 MHz to 16.5 GHz	–	–	–6	dB	
Jitter generation, 15.0 Gb/s Jitter measurement bandwidth is F(bits/sec)/1667	DJ_{TX}	Measured at output ball	–	–	0.15	UI	
	DCD_{TX}	Measured at output ball; Considered part of and included within DJTX	–	–	5	%UI	
	RJ_{TX}	1E-12 BER (peak-to-peak jitter)	–	–	0.12	UI	
	TJ_{TX}	1E-12 BER		–	0.27	UI	
Output differential skew	^{t}TXSKEW_DIFF		–	–	7	ps	6
Lane-to-lane output skew at TX	$L_{TX-SKEW}$		–	–	200	ps	7

Notes:
1. The TX goes into a High-Z state and the output will float HIGH or LOW, depending on the termination. The TX output will nominally pull to $V_{TT}/2$ if external AC coupling is used (see Figure 30 (page 86)). The TX output voltage in all other cases will depend on the host receiver termination.
2. TX output common-mode voltage in a DC-coupled link is dependent on the type of termination used at the receiver. See Termination Configurations for supported termination topologies.
3. Refers to the resistive portion of the termination.
4. TX performance may degrade in DC-coupled mode. In the presence of poor RX common-mode return loss the AC common-mode noise can increase by 2x. The actual RX termination voltage will impact the single-ended output resistance matching.
5. Impedance calibration does not compensate for this behavior.
6. Output skew not including package skew. For package skew specifications, see the HMC Gen2 Package Delay spreadsheet on micron.com.
7. Measured at the crossing point of each differential pair.

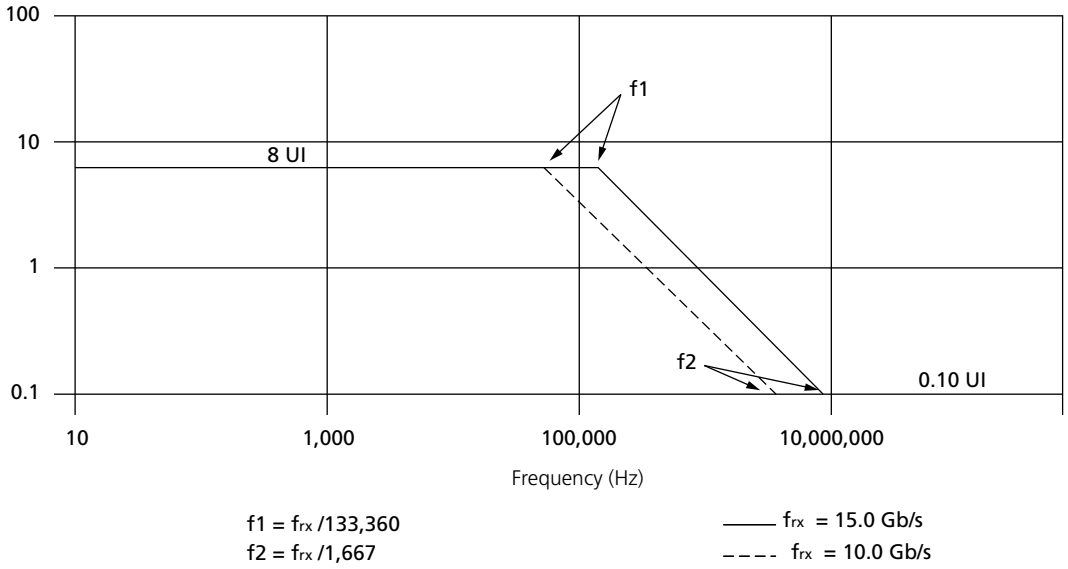
Table 41: RX Signaling Parameters

Parameter	Symbol	Test Conditions/Comments	Min	Typical	Max	Unit	Notes
Link supply RX termination voltage	V_{TR}	1.2V ±0.06V	1.14	1.2	1.26	V	
Differential peak-to-peak input voltage	V_{DIFF_RX}	$V_{DIFF\_RX} = 2 \times \lvert V_{RX\_P} - V_{RX\_N} \rvert$ Measured with reference load	100	–	1400	mV	1
Single-ended voltage (with respect to V_{SS}) on D+, D-	V_{RX_SE}	V_{TR} = 1.2V	0	–	V_{TR} + 300	mV	
Common mode of the input voltage	V_{RX_CM}	V_{RX_CM} = DC(avg) of $\lvert V_{RX\_P} + V_{RX\_N} \rvert$	300	–	V_{TR}	mV	
Short circuit current, RX off	$I_{SHORT\ RX\_OFF}$		–100	–	100	mA	2
Short circuit current, RX on	$I_{SHORT\ RX\_ON}$		–100	–	100	mA	2
Differential input resistance	R_{ID}		80	100	120	Ω	
	R_{ISE}		40	50	60	Ω	
Single-ended transmitter resistance	R_{RX_DELTA}		–	–	5	%	
Differential input loss – Relative to 100Ω differential system	$RL_{RX\text{-}DIFF}$	FB/1667 to (3/4)FB	–8	–	–	dB	3
		(3/4)FB to FB	-8 +16.6 × log (f/(0.75 × FB))	–	–	dB	3
Common mode return loss – Relative to ideal 25Ω	$RL_{RX\text{-}CM}$	Measured over FB/1667 to 3/4FB	–6	–	–	dB	3
Jitter tolerance	DJ_{RX}	Comprised of ISI, DCD, reflections, and crosstalk (does not include SJ)	–	–	0.42	UI	4
	SJ_{RX}	Sinusoidal, high frequency (f > f2)	0.1	–	–	UI	4, 5, 8
		Sinusoidal, low frequency (f < f1)	8	–	–		
	TJ_{RX}	Total	–	–	0.65	UI	4
Input differential skew tolerance	RX_{SKEW_DIFF}		–	–	.25	UI	6
Lane-to-lane skew at RX	$L_{RX\text{-}SKEW}$	Lane-to-lane skew at a receiver that must be tolerated	–	–	16	UI	7, 8

Notes:
1. The $V_{DIFF\_RX\_min}$ is an estimate that will be revised after the benefit of the RX DFE has been fully assessed. $V_{DIFF\_RX\_max}$ value is based on a low-frequency data pattern (16 ones followed by 16 zeros).
2. The receiver input maximum voltage ratings in the Absolute Maximum Ratings table cannot be exceeded for extended periods of time without affecting device reliability.
3. Half-baud frequency, FB, specified in GHz.

4. The RX decision feedback equalization (DFE) can effectively open a "closed" eye signal at its inputs, to achieve BER of 1E-12 or better. These jitter tolerance specifications correspond to the minimum capability of the RX when DFE is disabled for debug purposes. Assessment of DFE capabilities and BER for a given host TX and channel model can be performed using IBIS-AMI receiver models.
5. See Figure 34 (page 93).
6. Receiver differential skew as specified at the die. Host TX SERDES output skew, host package skew, PCB skew, and HMC package skew must be accounted for in the total budget. For the delay contribution in HMC Gen2 packaging, see the HMC Gen2 Package Delay spreadsheet on micron.com.
7. Lane-to-lane skew contributes directly to memory latency, so minimizing skew between RX lanes is recommended.
8. Any reduction in SJ_{RX} below the maximum allowed can be added directly to the lane-to-lane skew budget. In other words, LRX_skew + SJRX = 24 UI, but SJ_{RX} must not exceed 8 UI.

Figure 34: Receiver Sinusoidal Jitter Tolerance



Non High-Speed Link Parameters

The parameters in this section are for the I/O and timing aspects of HMC not covered in HMC-15G-SR Physical Link Specifications. Micron recommends running a simulation to analyze and validate your system's signal integrity for the non high-speed link I/O. Simulations can be done using the HMC IBIS model available on micron.com.

Table 42: Initialization and Bootstrap Parameters

Description	Symbol	Min	Typ	Max	Unit	Notes
Power supply ramp time	^{t}DD	–	–	200	ms	
Power supply slew rate	V_{DVDT}	–	–	10	V/ms	
Assertion time for P_RST_N	^{t}RST	20	–	–	ns	
P_RST_N slew rate	$V_{RSTDVDT}$	0.1	–	–	V/ns	
P_RST_N, REFCLKSEL, REFCLK_BOOT*, and CUB* input LOW voltage	V_{IL_LSIO}	V_{SS} - 0.4V	V_{SS}	V_{SS} + 0.5V	V	
P_RST_N, REFCLKSEL, REFCLK_BOOT*, and CUB* input HIGH voltage	V_{IH_LSIO}	V_{DDK} - 0.5V	V_{DDK}	Lower of V_{DDK} + 0.2V, or 1.7V	V	1
P_RST_N, REFCLKSEL, REFCLK_BOOT*, FERR_N, RXPS*, I^2C, JTAG, and CUB* pin leakage current	I_{LEAK}	-10	–	10	µA	
PLL and register configuration time	$^{t}INIT$	–	–	40	ms	
INIT continue ERI completion time	$^{t}INITCONTINUE$	220	–	1200	ms	2, 3
INIT continue to valid output (PRBS)	$^{t}PRBS$	–	14	20	ms	
Link receiver-phase acquisition	$^{t}RESP1$	–	5	7	ms	4
FLIT synchronization time	$^{t}RESP2$	–	–	1	µs	
NULL synchronization time	$^{t}RESP3$	10	50	200	ns	

Notes:
1. The maximum voltage for V_{IH_LSIO} will be whichever is the lower value of V_{DDK} + 0.2V or 1.7V. Example: if V_{DDK} is 0V, the I/Os should have signals no higher than 0.2V; or, if V_{DDK} is powered at 1.6V, the maximum signal level would be 1.7V (not 1.6V + 0.2V = 1.8V)
2. $^{t}INITCONTINUE$ represents the range of time that the INIT continue ERI function may take to complete. All aspects of link training on all active links must be completed before the INIT continue ERI completes, and therefore the minimum time must be used as the upper bound for link training time between the host and HMC. If a violation occurs, a link critical error status will be returned upon completion of INIT continue ERI.
3. Beginning with firmware revision 0.9B, $^{t}INITCONTINUE$ can be extended in order to wait for link training to complete prior to timing out and returning a link critical error status. See HMC Gen2 User Guide Register Definitions INIT Continue ERI section for details.
4. Link receiver phase acquisition times vary based on the time needed to acquire DFE synchronization. Better signal integrity on the lanes will reduce synchronization time.

Table 43: Link Power Management Parameters

Parameter Description	Symbol	Min	Max	Unit	Comments
Voltage Parameters					
LxRXPS input LOW voltage	V_{IL_RXPS}	–	V_{SS} + 0.5V	V	V_{SS} measured at HMC package ball

Table 43: Link Power Management Parameters (Continued)

Parameter Description	Symbol	Min	Max	Unit	Comments
LxRXPS input HIGH voltage	V_{IH_RXPS}	V_{DDK} - 0.5V	–	V	V_{DDK} measured at HMC package ball
LxTXPS output LOW voltage	V_{OL_TXPS}	–	V_{SS} + 0.1V	V	I_{OL} = 1mA
LxTXPS output HIGH voltage	V_{OH_TXPS}	V_{DDK} - 0.1V	–	V	I_{OH} = 1mA
LxTXPS output impedance		20	60	Ω	
Timing Parameters					
LxRXPS slew rate	$V_{RXPSDVDT}$	0.1	–	V/ns	
LxTXPS rise time with 2pF load	^{t}RISE_2pF_TXPS	120	540	ps	
LxTXPS rise time with 5pF load	^{t}RISE_5pF_TXPS	250	750	ps	
LxTXPS rise time with 10pF load	RISE_10pF_TXPS	530	1200	ps	
Power state transition timing	^{t}PST	–	80	ns	Delay from the transition of a single link's LxRXPS signal to the transition of the corresponding LxTXPS signal when no other link's LxRXPS signal is changing. See Figure 15 (page 30) for timing of multiple-link LxRXPS events
LxRXPS setup timing to enter down mode	^{t}IS	10	–	ns	Setup time of LxRXPS to P_RST_N transition from LOW to HIGH
Simultaneous power transition stagger	^{t}SS	–	525	ns	Entry/exit time from a state during simultaneous link power state transitions
Sleep mode entry	^{t}SME	–	200	ns	Time from transition LxTXPS LOW to sleep mode entry
Time to transition links from sleep mode to down mode	^{t}SD	–	160	µs	Measured from last link to take LxTXPS LOW to down mode Entry of all links
Time required to stay in down mode	^{t}DOWN	1.0	–	ms	Measured from last LxRXPS LOW to first LxRXPS HIGH
Time required to stay within active mode	^{t}OP	1.0	–	ms	Measured by LxTXPS HIGH time
HSS PLL self calibration timing	^{t}PSC	–	25	ms	Required after returning to active mode from down mode
Low power exit timing: transition of LxRXPS to RX PRBS	^{t}RXD	–	50	µs	Delay from transition of responder's LxRXPS to receiving PRBS at its RX
Low power exit timing: transition of LxTXPS to TX PRBS	^{t}TXD	–	50	µs	Delay from transition of responder's LxTXPS to its TX sending PRBS
Low power exit timing: TX PRBS to RX NULL	^{t}SigDet	–	50	µs	Delay from responder's TX sending PRBS to receiving NULL FLITs at its RX
Quiesce timeout	^{t}QUIESCE	20	–	µs	Wait time for in-flight transactions to complete after halting all request transactions from requester

Table 44: I²C Parameters

Parameter Description	Symbol	Min	Max	Units	Notes
Voltage Parameters					
SCL/SDA input LOW voltage	V_{IL_I2C}	V_{SS}- 0.4	V_{SS} + 0.45	V	1
SCL/SDA input HIGH voltage	V_{IH_I2C}	V_{DDK} - 0.45	V_{DDK} + 0.2	V	1
SCL/SDA pin capacitance	C_{I2C}	–	6	pF	
Timing Parameters					
SCL frequency	f_{I2C}	0	400	KHz	

Note: 1. As measured at HMC package ball.

Table 45: JTAG Parameters

Parameter Description	Symbol	Min	Max	Units	Notes
Voltage Parameters					
JTAG input LOW voltage	V_{IL_JTAG}	V_{SS} - 0.4V	V_{SS} + 0.5V	V	1
JTAG input HIGH voltage	V_{IH_JTAG}	V_{DDK} - 0.5V	V_{DDK} + 0.2V	V	1
JTAG output LOW voltage	V_{OL_JTAG}	–	V_{SS} + 0.1V	V	2
JTAG output HIGH voltage	V_{OH_JTAG}	V_{DDK} - 0.1V	–	V	3
Timing Parameters					
Clock cycle time	^{t}THTH	10	–	ns	4
Clock frequency	^{t}TF	–	100	MHz	4
Clock LOW time	^{t}TLTH	5	–	ns	
Clock HIGH time	^{t}THTL	5	–	ns	
TDI valid to TCK HIGH	^{t}DVTH	2.5	–	ns	
TCK HIGH to TDI invalid	^{t}THDX	2.5	–	ns	
TMS, capture setup	^{t}MVTH	2.5	–	ns	
TMS, capture hold	^{t}THMX	2.5	–	ns	

Notes:
1. As measured at HMC package ball.
2. I_{OL} = 1mA
3. I_{OH} = 1mA
4. Minimum clock cycle time (maximum clock frequency) for boundary scan register is 50ns (20 MHz) when boundary scan register is active.

Table 46: Reference Clock Parameters

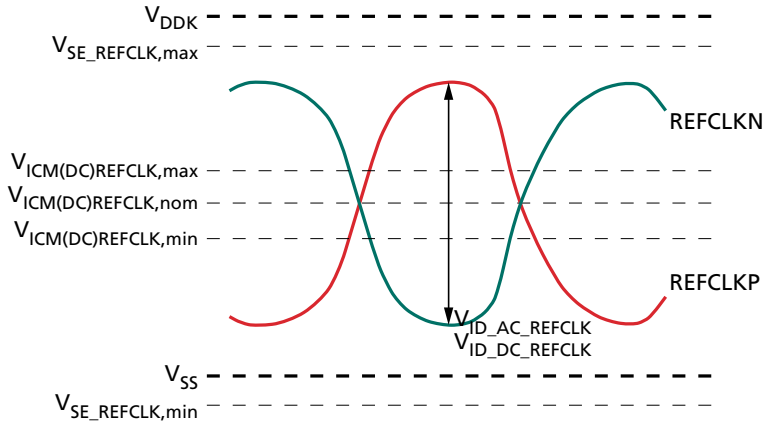
Parameter Description	Symbol	Min	Nom	Max	Unit	Notes
Reference clock input frequency	^{f}REFCLK	–	125, 156.25, 166.67	–	MHz	1
Reference clock frequency drift tolerance	^{f}TOL _REFCLK	–100	–	100	ppm	
Reference clock duty cycle tolerance	^{t}DCD_REFCLK	45	50	55	%	
Rise/Fall time for 125 MHz reference clock (10%–90%)	^{t}RISE_REFCLK_125	–	–	640	ps	

Table 46: Reference Clock Parameters (Continued)

Parameter Description	Symbol	Min	Nom	Max	Unit	Notes
Rise/Fall time for 156.25 MHz reference clock (10%–90%)	$^{t}RISE\_REFCLK\_156$	–	–	512	ps	
Rise/Fall time for 166.67 MHz reference clock (10%–90%)	$^{t}RISE\_REFCLK\_166$	–	–	480	ps	
Single-ended input voltage	V_{SE_REFCLK}	V_{SS} - 0.1	–	V_{DDK} - 0.1	V	
AC-Coupled Reference Clock Inputs						
Input differential voltage	$V_{ID\_AC\_REFCLK}$	200	–	800	mV	2
Internal differential termination resistance	$R_{ID\_AC\_REFCLK}$	90	100	110	Ω	3
AC coupling capacitance (external DC blocking capacitor)	$C_{REFCLK\_DC\_Block}$	–	0.1	–	µF	
DC-Coupled Reference Clock Inputs						
Input differential voltage	$V_{ID\_DC\_REFCLK}$	200	–	800	mV	
Common-mode voltage	$V_{ICM\_DC\_REFCLK}$	680	750	900	mV	
Internal parallel termination resistance	$R_{ID\_DC\_REFCLK}$	45	50	55	Ω	4
Reference Clock Phase Noise Requirements						
Offset from nominal input frequency	100Hz	–	–	–85	dBc/Hz	
	1 kHz	–	–	–97	dBc/Hz	
	10 kHz	–	–	–97	dBc/Hz	
	100 kHz	–	–	–114	dBc/Hz	
	1 MHz–1 GHz	–	–	–126	dBc/Hz	

Notes:
1. Only the specified nominal frequencies are supported for the reference clock; spread spectrum clocking (SSC) is not supported.
2. If input differential voltage exceeds $V_{ID\_AC\_REFCLK}$, the use of attenuation resistors on the board are required.
3. AC-coupled input mode uses internal calibrated differential termination.
4. DC-coupled input mode uses internal calibrated termination to V_{SS}.
5. The Shutdown ERI must be run prior to the reference clock being reset or brought out of specification.

Figure 35: Requirements for Reference Clock Signals

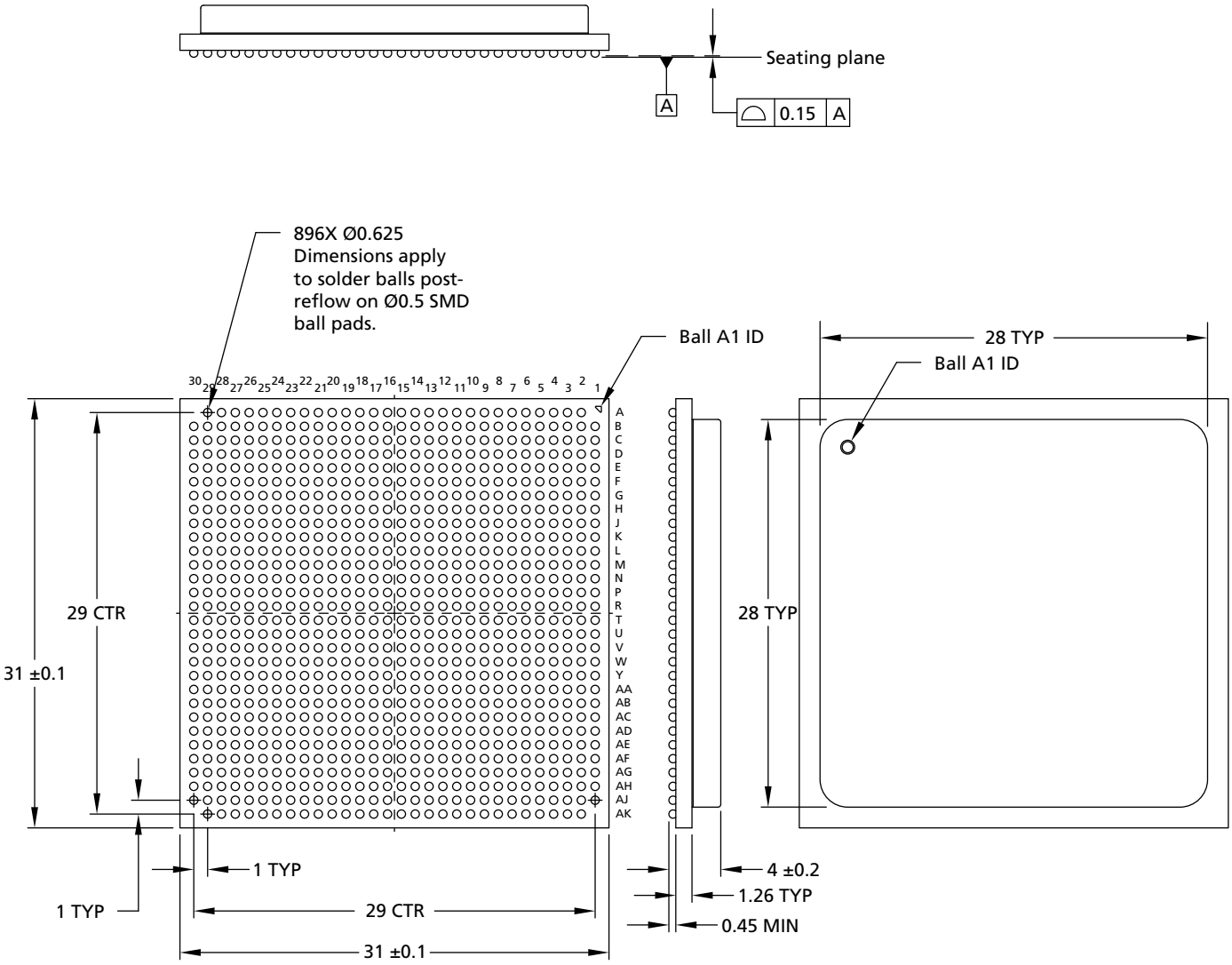


Impedance Calibration

HMC includes internal calibration circuitry to maintain tight termination impedance tolerance. This circuitry auto-calibrates and requires no action from the user other than the placement of 200Ω high-precision resistors (0.1% tolerance recommended) between EXTRESTP and EXTRESTN as well as one between EXTRESBP and EXTRESBN.

Package Dimensions

Figure 36: HMC Package Drawing for 4-Link (31 x 31 mm) HMC Gen2 Device



Notes:
1. All dimensions are in millimeters.
2. Package balls are SAC405 (95.5% Sn, 4% Ag, 0.5% Cu).
3. For recommended soldering parameters and reflow guidelines, see Micron technical note TN-00-15.
4. Maximum compressive load: 219 N.
5. This package is not hermetically sealed.

Figure 37: HMC Ballout for 4-link (31 x 31 mm) HMC Gen2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A		VSS	L2TXP[4]	L2TXN[4]	VSS	VSS	L2RXP[8]	L2RXN[8]	VSS	VSS	L2TXP[13]	L2TXN[13]	VSS	VSS	L0RXN[2]	L0RXP[2]	VSS	VSS	L0TXN[6]	L0TXP[6]	VSS	VSS	L0RXN[3]	L0RXP[3]	VSS	VSS
B	VSS	VSS	VSS	L2TXP[14]	L2TXN[14]	VSS	VSS	L2RXP[12]	L2RXN[12]	VSS	VSS	L2TXP[12]	L2TXN[12]	VSS	VSS	VSS	VSS	L0TXN[7]	L0TXP[7]	VSS	VSS	L0RXN[7]	L0RXP[7]	VSS	VSS	L0TXN[5]
C	L2RXP[1]	L2RXN[1]	VSS	VSS	L2TXP[0]	L2TXN[0]	VSS	VSS	L2RXP[13]	L2RXN[13]	VSS	DNU	L2TXP[8]	L2TXN[8]	VSS	VSS	L0TXN[3]	L0TXP[3]	VSS	VSS	L0RXN[6]	L0RXP[6]	VSS	DNU	L0TXN[11]	L0TXP[11]
D	VSS	L2RXP[0]	L2RXN[0]	VSS	VSS	L2TXP[15]	L2TXN[15]	VSS	VSS	L2RXP[14]	L2RXN[14]	VSS	VSS	L2TXP[9]	L2TXN[9]	L0TXN[2]	L0TXP[2]	VSS	VSS	L0RXN[5]	L0RXP[5]	VSS	VSS	L0TXN[4]	L0TXP[4]	VSS
E	DNU	VSS	L2RXP[4]	L2RXN[4]	VSS	VSS	L2TXP[11]	L2TXN[11]	VSS	VSS	L2RXP[15]	L2RXN[15]	VSS	VSS	VSS	VSS	VSS	VSS	L0RXN[4]	L0RXP[4]	VSS	VSS	L0TXN[0]	L0TXP[0]	VSS	VSS
F	VSS	VSS	VSS	L2RXP[5]	L2RXN[5]	VSS	VSS	L2TXP[10]	L2TXN[10]	VSS	VSS	L2RXP[10]	L2RXN[10]	VSS	L2RXP[9]	L2RXN[9]	VSS	L0RXN[1]	L0RXP[1]	VSS	VSS	L0TXN[1]	L0TXP[1]	VSS	VSS	L0RXN[14]
G	L2TXP[5]	L2TXN[5]	VSS	VSS	L2RXP[6]	L2RXN[6]	VSS	VSS	L2TXP[1]	L2TXN[1]	VSS	VSS	L2RXP[11]	L2RXN[11]	VSS	VSS	L0RXN[0]	L0RXP[0]	VSS	VSS	L0TXN[10]	L0TXP[10]	VSS	VSS	L0RXN[13]	L0RXP[13]
H	VSS	L2TXP[6]	L2TXN[6]	VSS	VSS	L2RXP[7]	L2RXN[7]	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	L0RXN[12]	L0RXP[12]	VSS
J	EXTRESTP	VSS	L2TXP[7]	L2TXN[7]	VSS	VSS	VSS	VTR	VSS	VTT	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VSS	VTR	VSS	VTR	VSS	VSS	VSS	VSS
K	VSS	EXTRESTN	VSS	L2TXP[3]	L2TXN[3]	VSS	VSS	VSS	VTT	VSS	VTT	VDDPLLB2	VDDPLLA2	VDDPLLB2	VDDPLLA2	VDDPLLB0	VDDPLLA0	VDDPLLB0	VDDPLLA0	VSS	VTT	VSS	VTT	VSS	VSS	L0TXN[8]
L	L2RXP[3]	L2RXN[3]	VSS	VSS	L2TXP[2]	L2TXN[2]	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTR	VSS	VSS	L0TXN[9]	L0TXP[9]
M	VSS	VSS	L2RXP[2]	L2RXN[2]	VSS	VSS	DNU	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	TST_VSS[1]	VSS	VSS
N	REFCLK_BOOT[0]	VSS	VSS	VSS	VSS	L2RXPS	DNU	DNU	VDD	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	L0TXPS	DNU	VSS
P	DNU	L3RXPS	L3TXPS	DNU	DNU	TST_VSS[1]	TDI	VSS	DNU	DNU	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	CUB[1]	DNU	DNU
R	DNU	VSS	DNU	VDDK	TMS	VDDK	TRST_N	DNU	VSS	VDDK	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	**TST_VSS[1]**	VSS	TST_VSS[1]	VSS	TST_VSS[1]	VSS	TST_VSS[1]
T	TDO	VSS	TST_VSS[1]	VSS	TCK	VSS	DNU	TST_VSS[1]	VDD	VDD	VSS	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	SDA	VDDK	TST_VSS[1]
U	DNU	L2TXPS	DNU	DNU	FERR_N	TST_VSS[1]	DNU	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VCCP	VSS	DNU
V	TST_VSS[1]	VSS	VSS	VSS	VSS	REFCLKSEL	DNU	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	DNU	L1TXPS	CUB[0]	VSS
W	VSS	VSS	L3RXP[2]	L3RXN[2]	VSS	VSS	CUB[2]	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VSS	VSS	VSS
Y	L3RXP[3]	L3RXN[3]	VSS	VSS	L3TXP[2]	L3TXN[2]	VSS	VSS	VTR	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	L1TXN[9]	L1TXP[9]
AA	VSS	EXTRESBN	VSS	L3TXP[3]	L3TXN[3]	VSS	VSS	VTT	VSS	VTT	VSS	VDDPLLB3	VDDPLLA3	VDDPLLB3	VDDPLLA3	VDDPLLB1	VDDPLLA1	VDDPLLB1	VDDPLLA1	VTT	VSS	VTT	VSS	VSS	VSS	L1TXN[8]
AB	EXTRESBP	VSS	L3TXP[7]	L3TXN[7]	VSS	VSS	VSS	VSS	VTR	VSS	VTR	VSS	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VTT	VSS	VTR	VSS	VSS	VSS
AC	VSS	L3TXP[6]	L3TXN[6]	VSS	VSS	L3RXP[7]	L3RXN[7]	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	L1RXN[12]	L1RXP[12]	VSS
AD	L3TXP[5]	L3TXN[5]	VSS	VSS	L3RXP[6]	L3RXN[6]	VSS	VSS	L3TXP[1]	L3TXN[1]	VSS	VSS	L3RXP[11]	L3RXN[11]	VSS	VSS	L1RXN[0]	L1RXP[0]	VSS	VSS	L1TXN[10]	L1TXP[10]	VSS	VSS	L1RXN[13]	L1RXP[13]
AE	VSS	VSS	VSS	L3RXP[5]	L3RXN[5]	VSS	VSS	L3TXP[10]	L3TXN[10]	VSS	VSS	L3RXP[10]	L3RXN[10]	VSS	L3RXP[9]	L3RXN[9]	VSS	L1RXN[1]	L1RXP[1]	VSS	VSS	L1TXN[1]	L1TXP[1]	VSS	VSS	L1RXN[14]
AF	DNU	VSS	L3RXP[4]	L3RXN[4]	VSS	VSS	L3TXP[11]	L3TXN[11]	VSS	VSS	L3RXP[15]	L3RXN[15]	VSS	VSS	VSS	VSS	VSS	VSS	L1RXN[4]	L1RXP[4]	VSS	VSS	L1TXN[0]	L1TXP[0]	VSS	VSS
AG	VSS	L3RXP[0]	L3RXN[0]	VSS	VSS	L3TXP[15]	L3TXN[15]	VSS	VSS	L3RXP[14]	L3RXN[14]	VSS	VSS	L3TXP[9]	L3TXN[9]	L1TXN[2]	L1TXP[2]	VSS	VSS	L1RXN[5]	L1RXP[5]	VSS	VSS	L1TXN[4]	L1TXP[4]	VSS
AH	L3RXP[1]	L3RXN[1]	VSS	VSS	L3TXP[0]	L3TXN[0]	VSS	VSS	L3RXP[13]	L3RXN[13]	VSS	DNU	L3TXP[8]	L3TXN[8]	VSS	VSS	L1TXN[3]	L1TXP[3]	VSS	VSS	L1RXN[6]	L1RXP[6]	VSS	DNU	L1TXN[11]	L1TXP[11]
AJ	VSS	VSS	VSS	L3TXP[14]	L3TXN[14]	VSS	VSS	L3RXP[12]	L3RXN[12]	VSS	VSS	L3TXP[12]	L3TXN[12]	VSS	VSS	VSS	VSS	L1TXN[7]	L1TXP[7]	VSS	VSS	L1RXN[7]	L1RXP[7]	VSS	VSS	L1TXN[5]
AK		VSS	L3TXP[4]	L3TXN[4]	VSS	VSS	L3RXP[8]	L3RXN[8]	VSS	VSS	L3TXP[13]	L3TXN[13]	VSS	VSS	L1RXN[2]	L1RXP[2]	VSS	VSS	L1TXN[6]	L1TXP[6]	VSS	VSS	L1RXN[3]	L1RXP[3]	VSS	VSS
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Notes:
1. Pins used for internal test purposes and are not part of power delivery network. TST_V_{SS} pins
2. Ballout represents an X-ray view, looking through package down to the balls.

Appendix A: Glossary of Terms

Table 47: Glossary of Terms

Term	Definition
ADR or ADRS	Address
BIST	Built in self test
DFE	Decision feedback equalizer
Downstream	Away from the host
EAM	Error abort mode
ERI	External register interface: Registers that can indirectly access HMC configuration and status registers.
FIFO	First in, first out
Forward	From the point of view of the link master, "forward" means in the direction that the link master is driving on this side of the link.
FRP	Forward retry pointer
HMC	Hybrid memory cube
Host link	HMC link configuration that uses its link slave to receive request packets and its link master to transmit response packets.
SerDes	High-speed SerDes
Lane	A pair of differential signal lines, one in each direction (transmitters and receivers); multiple lanes are combined to form links.
LFSR	Linear feedback shift register
Link	A fully-duplexed interface connecting two components, implemented with SerDes lanes that carry system commands and data.
LSB	Least significant bit
MCE	Multi-bit correctable error
MI	Memory interface
MSB	Most significant bit
MUE	Multiple uncorrectable errors
Partition	Portion of a memory die that is connected to and controlled by a single vault controller
Pass-through Link	HMC link configuration that uses its link master to transmit the request packet toward its destination cube and its link slave to receive response packets destined for the host processor.
Quadrant	The four local vaults associated with a given link that do not require routing across the crossbar switch.
Requester	Represents either a host processor or an HMC link configured as a pass-through link. A requester transmits packets downstream to the responder.
Responder	Represents an HMC link configured as a host link. A responder transmits packets upstream to the requester
Return	From the point of view of the link master, "return" means in the direction on the other side of the link that the local link slave is receiving
RMW	Read-modify-write sequence
RRP	Return retry pointer
SBE	Single bit error

Table 47: Glossary of Terms (Continued)

Term	Definition
TSV	Through-silicon via: Electrical connection through a die to enable die stacking without wires or spacers, providing excellent electrical properties
Upstream	Toward the host
Vault	Independent memory controller and local memory stacked directly above, in a cube

Revision History

Rev. G – 04/2017

- Changed from self refresh to down mode
- Added package notes : not hermetically sealed and compressive load
- Added token return timing and host allowable delay for half link usage
- Updated Pin Description for the FERR_N pin
- Renamed table title “DC Electrical Characteristics” to “Recommended Supply Operating Conditions”, removed the Ground specification and added "all voltages are referenced to VSS" to description
- Renamed section title “DC Operating Conditions" to “AC and DC Operating Conditions”
- Removed the operating temperature ranges from the Absolute Maximum Ratings table, and put into a new table “Operating Temperature Range” under the AC and DC Operating Conditions section. Also removed redundant footnote.
- Updated Operating Conditions to include explicit DC ranges for V_{DD} (V_{DD_DC}) and V_{DDM} (V_{DDM_DC})
- Added new table “Leakage Currents” which includes Input Leakage (I_L) and FERR_N Leakage ($I_{FERR\_N\_L}$)
- Renamed "Maximum Current Conditions" table name to "Power-On Currents"
- Removed specification for current during down mode, redirecting to power calculator
- Added DINV field and removed "note 3" on CRC field in "Valid Field and Command Summary" table
- Removed I^2C and JTAG Interface sections. Added a reference to the User Guide for details related to the JTAG/I^2C Interfaces
- Added I^2C to the features section on the top page

Rev. F – 05/2016

- Removed references to NGK and NFH (8H) devices.
- Modified ^tSD from 150 µs to 160 µs

Rev. E – 03/2016

- Added 010xxx and 1011xx as Reserved commands in the Request Commands table.
- Corrected 2-link (16x19.5) package pinout file due to error on pin AD17 (previously indicated VTR, actually VTT).
- Removed part code NGF which was missed in one location in Rev. D edits.
- Corrected the P_RST_N voltage levels in the Power-On and Initialization section to match the voltages in the Initialization and Bootstrap table.
- Added a Cold Reset and Power-Off Considerations section.
- Added REFCLK_BOOT* and CUB* voltage requirements to the Initialization and Bootstrap Parameters table, combined with P_RST_N and REFCLKSEL.
- Added Change in operating temp to Absolute Maximum Ratings table.
- Added tINITCONTINUE values and info about timeout extensions.
- Updated SERDES RX and TX skew values to align with SERDES documentation updates.

- Updated IRTRY and PRET sections to indicate fields ignored by the HMC rather than forced to 0. This represents the same expected behavior in the host, but makes actual implementation in the HMC more clear.
- Added footnote to the Reference Clock table: The Shutdown ERI must be run prior to the Reference Clock being reset or brought out of specification.
- Updated tSS maximum to 525 ns.
- Clarified input (RX) and output (RX) differential skew notes and updated values to reflect this clarification.
- Added footnote to the DC Characteristics table for VDD & VDDM: Following the start of the INIT Continue ERI, the DC level of this supply must be kept within +/-(TBD)%.
- Removed recommendation for VCCP/VDDK ordering, as this has been resolved.

Rev. D – 09/2015

- Changed the top title and Options from HMC-15G-SR to HMC Gen2.
- Added one more state to Initialization Flow Diagram to ensure host checks for end of INIT Continue.
- Updated ERRSTAT table to clarify that Link Errors are detected and initiated by the Link Slave.
- Updated retraining section with new flow diagram and differentiation between high rate of retry and too many retry attempts.
- Changed Atomic rollover limits back to 2^63 - 1 and 2^127 - 1 respectively (improperly changed in Rev C).
- Clarified that ^{t}PST applies in the case of sleep entry (now agrees with descriptions and diagrams for both sleep and down mode).
- Made INIT Continue wording more consistent throughout.
- Added P_RST_N and REFCLKSEL input voltage levels to Low Speed IO section.
- Updated appropriate part number to reflect logic revision 2.
- Added "Poisoned Packet Rules" section for benefit of host design.
- Overhauled section on packet integrity to address all data integrity.
- Updated Transaction Layer section with respect to data integrity.
- Updated text to reflect K0 and K2 terminology to match other documents and noted K1 fixed amplitude reasoning.
- Added figure showing TX pre and post tap emphasis effect.
- Added note to pin connections description regarding PLLA and PLLB usage.
- Swapped ERRSTAT 0x01 and 0x7E (temperature thresholds) to properly reflect what these thresholds represent.
- Updated the description of Reliability Threshold ERRSTAT to be more descriptive.
- Removed part code NGF.
- Updated warm reset section to include "tQUIESCE" and make wording more clear.
- Added tQUIESCE parameter to Power State Management Timing parameters table.
- Fixed scrambler example, column 0 which was off by one value.
- Added notes in retraining and power state sections regarding the ability to inhibit link down mode.
- Updated Figure 9 to agree with Figure 10 (and specification) with respect to requester NULL transmission time.
- Added new section titled "Token Return Loop Time".

- Updated flow chart for link retraining upon high BER to agree with text.
- Removed tolerances for power supplies in the pin descriptions table (covered in voltage specifications section).
- Made note in "Lane Reversal" section that half links utilize lanes 0-7 and reverse accordingly.
- Removed typical value of t_{INIT}.
- Removed enumeration of "tst_vss" fixed superscript footnote reference.
- Added Lane Reversal/Lane Polarity figure.
- Added additional supported Chaining topologies.

Rev. C – 11/2014

- Converted to CMS.
- Added revision number for Rev2 devices to the IDCODE table (Identification Register Definitions).
- Consolidated voltage specifications for non-high speed I/O
- Updated Retry Pointer Loop Time implementation example table.
- Modified Atomic wording to fix note on carry-out exceeding limits. Used to read "exceeds 2^63 and 2^127 but should have read (and now reads) 2^64 - 1 and 2^128 - 1
- Updated CUB Request Packet Header definition, as this field is used regardless of chaining usage.
- Updated section on Link Retry to clarify when a retry error packet is being sent.
- Updated Fatal Error ERRSTAT table to meet HMCC 1.1 protocol specification
- Updated JTAG timing and electrical characteristic names for consistency with similar parameters.
- Added I^2C electrical and timing section.
- Modified Table 42 to remove reference to Link receiver-phase acquisition with no DFE (tRESP1) as DFE shall be left on.
- Changed name of tRESP1_DFE to simply tRESP1.
- Added specification for tINITCONTINUE
- Updated AC specifications for REFCLK.
- Specified REFCLK internal termination values.
- Removed TX Amplitude references, as it is fixed at 1V.
- Added RXPS and TXPS slew and drive characteristics.
- Clarified auto-adaptation for DFE and DPC.
- Updated with improved TX output skew specification.
- Updated initialization timing requirements.
- Added note on package ball composition and reflow guidelines.
- Removed note referencing VDDK must be connected to VDDK.
- Clarified CUB field usage and consequences of not issuing proper CUB field in header.
- Clarified section on reordering.
- Updated I^2C section for clarity.
- Noted correct PRBS algorithm used by scrambler.
- Fixed descrambler polynomial.
- Added note for use of in-band access for warm reset.

- Added note about warm reset must be during runtime.
- Major update to power section, including reference to power estimation spreadsheet.
- Added specification for current during reset.
- Added TST_Vss pins to Table 2
- Updated warm reset flow chart to note in-band behavior requirements.
- Added note regarding AC noise on analog supplies.
- Modified Power on Initialization section to properly indicate INIT Continue ERI must be run (previously noted "Init continue bit").
- Modified Power on Initialization section to note consequences of not sending NULL flits when INIT Continue is run.
- Made note 4 regarding NULL Flits from downstream link during transition into sleep mode an explicit requirement rather than implied requirement.
- Reduced tSME to less restrictive 200 ns.
- Filled in power specifications for time P_RST_N until tINIT.
- Fixed error in JTAG CFG_RD flow chart
- Properly clarified TAG field when either read response or error response is possible.
- Clarified title in "Retry Buffer Full Period" table to properly indicate that not all are 256-FLIT retry buffer full period.
- Removed notes on ^{t}RXD and ^{t}SidDet that indicated only applicable to cube to cube links.
- Added recommendation that V_{DDK} ramp with or before V_{CCP}.
- Changed "closed-bank" to "closed-page" to reflect industry standard terminology.
- Added reference to HMC Register Addendum in MODE READ and WRITE section.
- Added rise and fall time requirements for REFCLK.
- Updated notes in Synchronous Link Bit Rate Specifications to clarify that any of the three valid REFCLK frequencies will work for any of the the three valid bit rates.

Rev. B – 1/2014

- Updated package codes.
- Updated Figure 4.
- Added text and the Scrambler Example table.
- Updated Chaining section:
  - Deleted last sentence of first paragraph
  - Added last sentence to third paragraph
  - Added last paragraph and Figure 7
- Updated Power-On and Initialization section:
  - Added last two sentences to first paragraph
  - Updated steps 3, 4, 5, and 6
  - Updated/Added steps 11, 12, and 13
  - Updated HMC Initialization Flowchart and Initialization Timing figure
- Updated Sleep Mode Entry and Exit (Single Link Only) figure:
  - Updated Note 2
  - Added Note 4
- Updated Transaction Layer section:

  - Added FLIT introductory text and example tables
  - Deleted last sentence of second paragraph of section 9.1.4
  - Added sentence toward the end of the second paragraph in section 9.3
  - Updated ERRSTAT[6:0] Bit Definitions table
  - Updated second sentence of section 9.9
  - Changed two instances of “Vendor specific” to “Reserved” in Transaction Layer – Request Commands table
  - Updated text in last paragraph of Section 9.10.4
  - Changed “NA” to “1” in Flow Commands table
  - For the Valid Field and Command Summary Table:
    - Added CUB row
    - Changed values for DLN row
    - Added Note 7
- Updated Configuration and Status Registers:
  - Updated/deleted text in second and third paragraphs
  - Deleted notes from Configuration and Status Register Access figure
  - Deleted all tables at the end of the section
- Updated Link Retry section:
  - Added/updated text in section 11.2.1
  - Deleted text from first paragraph in section 11.3.3
- Updated Host Recovery after Link Retry Fails figure.
- Added Note 5 to JTAG Voltage and Timing Parameters table.
- Updated HMC-15G-SR Physical Link Specifications section:
  - Updated second paragraph in section 18.1.1
  - Updated/Corrected/Added information in TX Signaling Parameters table
  - Added information in Signaling Parameters table
  - Updated Receiver Sinusoidal Jitter Tolerance figure
  - Removed row 14 and changed the last three TBDs to 50µs in Link Power Management Parameters table
- Updated Packages for HMC-15G-SR Devices section
  - Updated both package figures
  - Updated the HMC Ballout for 4-link HMC-15G-SR figure
- Corrected Figure 6 title (1/3/2014)

Rev. A – 1/2013

- Initial release.

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-4000
www.micron.com/products/support Sales inquiries: 800-932-4992

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for micron ***manufacturer:***

Other Similar products are found below :

MTFC4GACAJCN-1M WT TR MT18JSF1G72PDZ-1G6E1 MTFDDAK960MAV-1AEA2AAYYES M29W400DT55N6E M25P32-VME6G MT29F1G16ABBDAH4-ITX:D TR MTFDHAL7T6TDP-1AT1ZABYY MTFDDAA240MBB-2AE1ZABYY MTFDDAK1T9TDD-1AT1ZABYY MTFDDAK3T8TDT-1AW1ZABYY MTFDDAK3T8TDS-1AW1ZABYY MT47H32M16NF-25E IT:H EDW4032BABG-70-F-D MT47H32M16NF-25E IT:H TR MT40A512M16LY-075:E MT25QL128ABA1ESE-MSIT TR MTFDDAV256TBN-1AR12ABYY MTFDDAK7T6TDS-1AW15ABYY MTFDDAK960TDT-1AW1ZABYY MT48LC8M16A2P-6A:G LJDTT8GB-000-617 MT16KTF1G64AZ-1G4E1 MTFC32GAKAEEF-AIT TR MT40A512M8SA-062E:F MTFDDAK3T8TDT-1AW16ABYY MTFDDAK2T0TDL-1AW1ZABYY MT29F32G08CBADAWP:D MT29F4G08ABADAH4:D TR MTFC8GAKAJCN-1M WT MTFDDAC512MAM-1K1 MT41K512M8DA-107 XIT:P TR MT28EW01GABA1HJS-0SIT TR MTFDHAL15T3TDP-1AT1ZABYY MT40A2G16SKL-062E:B UF25B100 MTFDDAK960TDT-1AW16ABYY MT40A512M8RH-083E:B MTFDHAL7T6TCT-1AR1ZABYY MTFDHAL3T2TDR-1AT1ZABYY MTA36ASF4G72PZ-2G9E2 MTFDHBK256TDP-1AT12AIYY MT47H64M16NF-25E XIT:M TR MT47H64M16NF-25E:M TR MTFDDAK064MBD-1AH12ITYY MT46H64M16LFBF-5 AIT:B TR MT29F1G08ABAFAWP-ITE:F MTFDHAL12T8TDR-1AT1ZABYY MTFDHBK1T0TDP-AAT12AIYYES N25Q064A13EF640E MT25QU01GBBB8ESF-0AAT TR