12.7 Data Rate Control

The System interface supports a maximum data rate of one doubleword per cycle. The data rate the processor can support is directly related to the secondary cache access time; if the access time is too long, the processor cannot transmit and accept data at the maximum rate.

The rate at which data is delivered to the processor can be determined by the external agent—for example, the external agent can drive data and assert **ValidIn*** every *n* cycles, instead of every cycle. An external agent can deliver data at any rate it chooses, but must not deliver data to the processor any faster than the processor is capable of receiving it.

The processor only accepts cycles as valid when **ValidIn*** is asserted and the **SysCmd** bus contains a data identifier; thereafter, the processor continues to accept data until it receives the data word tagged as the last one.

Data Transfer Patterns

A data pattern is a sequence of letters indicating the *data* and *unused* cycles that repeat to provide the appropriate data rate. For example, the data pattern **DDxx** specifies a repeatable data rate of two doublewords every four cycles, with the last two cycles unused. Table 12-6 lists the maximum processor data rate for each of the possible secondary cache write cycle times, and the most efficient data pattern for each data rate.

Table 12-6 Transmit Data Rates and Patterns

Maximum Data Rate	Data Pattern	Maximum Secondary Cache Access
1 Double/1 SClock Cycle	D	4 PCycles
2 Doubles/3 SClock Cycles	DDx	6 PCycles
1 Double/2 SClock Cycles	DDxx	8 PCycles
1 Double/2 SClock Cycles	DxDx	8 PCycles
2 Doubles/5 SClock Cycles	DDxxx	10 PCycles
1 Double/3 SClock Cycles	DDxxxx	12 PCycles
1 Double/3 SClock Cycles	DxxDxx	12 PCycles
1 Double/4 SClock Cycles	DDxxxxxx	16 PCycles
1 Double/4 SClock Cycles	DxxxDxxx	16 PCycles

In Tables 12-6 and 12-7, data patterns are specified using the letters **D** and **x**; **D** indicates a data cycle and **x** indicates an unused cycle. Figure 12-40 shows a read response in which data is provided to the processor at a rate of two doublewords every three cycles using the data pattern **DDx**.

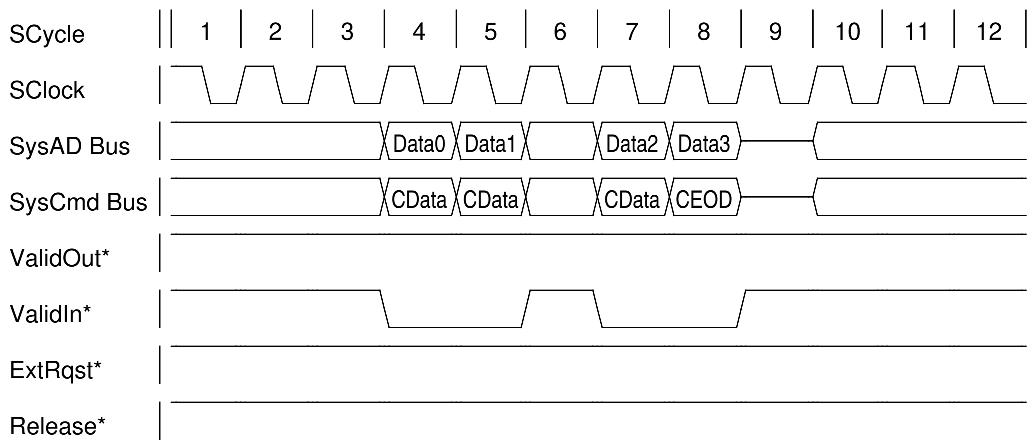

Figure 12-40 Read Response, Reduced Data Rate, System Interface in Slave State

Secondary Cache Transfers

The processor operates most efficiently if data is delivered in pairs of doublewords, since the secondary cache is organized as a 128-bit RAM array. The most efficient way of reducing the data rate is to deliver a pair of doublewords followed by some number of unused cycles, followed by another pair of doublewords. The secondary cache write cycle time should determine the rate at which this pattern is repeated. However, the processor accepts data in any pattern as long as the time between the transfer of any pair of odd-numbered doublewords is greater than, or equal to, the write cycle time of the secondary cache. Doublewords in the transfer pattern are numbered beginning at 0: the odd-numbered doublewords are the second, fourth, sixth, and so on.

Secondary Cache Write Cycle Time

Behavior of the processor is undefined if, based on the secondary cache write cycle time, data is delivered to the processor faster than the processor can handle it. Secondary cache write cycle time is defined as the sum of the parameters:

$$T_{Wr1Dly},\ T_{WrSUp},\ \text{and}\ T_{WrRc}$$

These parameters are defined in Chapter 9, Table 9-1.

The rate at which the processor transmits data to an external agent is programmable at boot time through the boot-time mode control interface. The transmit data rate can be programmed to any of the data rates and data patterns listed in Table 12-6, as long as the programmed data rate does not exceed the maximum rate the processor can handle, based on the secondary cache write cycle time. The behavior of the processor is undefined if a programmed transmit data rate exceeds the maximum the processor can support.

Figure 12-41 shows a processor write request in which the processor transmit data rate is programmed as one doubleword every two cycles, using the data pattern **DDxx**.

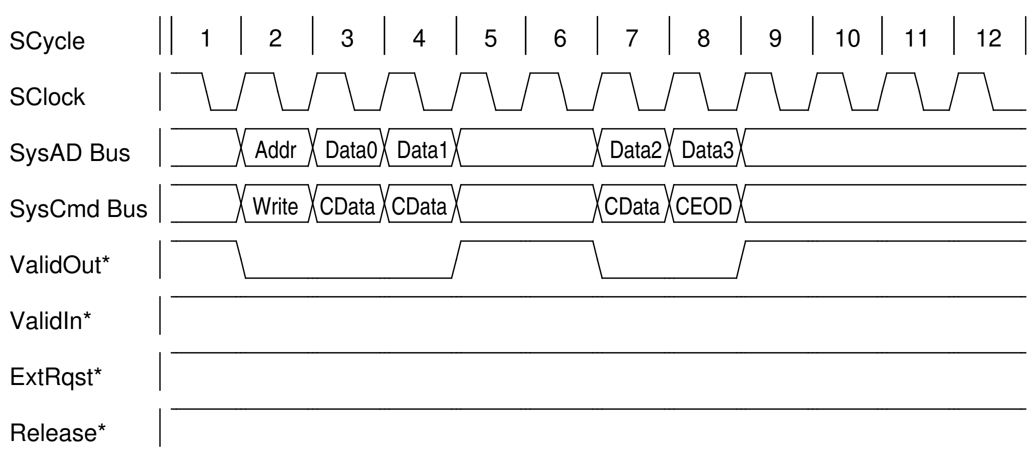

Figure 12-41 Processor Write Request, Transmit Data Rate Reduced

Table 12-7 shows the maximum transmit data rates for a given set of secondary cache parameters, based on a PClock-to-SClock divisor of 2. To find the maximum allowable secondary cache write cycle time and secondary cache access time, multiply the maximum secondary cache numbers for each pattern by:

(PClock_to_SClock_Divisor)/2

The minimum number for these parameters is always the minimum access time supported by processor.

Table 12-7 Maximum Transmit Data Rates

Secondary Cache Write Cycle Time	Maximum Data Rate	Best Data Pattern
1-4 PCycles	1 Double/1 SClock Cycle	D
5-6 PCycles	2 Doubles/3 SClock Cycles	DDx
7-8 PCycles	1 Double/2 SClock Cycles	DDxx
9-10 PCycles	2 Doubles/5 SClock Cycles	DDxxx
11-12 PCycles	1 Double/3 SClock Cycles	DDxxxx

Independent Transmissions on the SysAD Bus

In most applications, the **SysAD** bus is a point-to-point connection, running from the processor to a bidirectional registered transceiver residing in an external agent. For these applications, the **SysAD** bus has only two possible drivers, the processor or the external agent.

Certain applications may require connection of additional drivers and receivers to the **SysAD** bus, to allow transmissions over the **SysAD** bus that the processor is not involved in. These are called *independent transmissions*. To effect an independent transmission, the external agent must coordinate control of the **SysAD** bus by using arbitration handshake signals and external null requests.

An independent transmission on the **SysAD** bus follows this procedure:

1. The external agent requests mastership of the **SysAD** bus, to issue an external request.
2. The processor releases the System interface to slave state.
3. If the processor is being used with a secondary cache, the external agent issues a *secondary cache release external null request* to return ownership of the secondary cache to the processor.
4. The external agent then allows the independent transmission to take place on the **SysAD** bus, making sure that **ValidIn*** is not asserted while the transmission is occurring.
5. When the transmission is complete, the external agent must issue a *System interface release external null request* to return the System interface to master state.

System Interface Endianness

The endianness of the System interface is programmed at boot time through the boot-time mode control interface, and remains fixed until the next time the processor mode bits are read. Software cannot change the endianness of the System interface and the external system; software can set the reverse endian bit to reverse the interpretation of endianness inside the processor, but the endianness of the System interface remains unchanged.

12.8 System Interface Cycle Time

The processor specifies minimum and maximum cycle counts for various processor transactions and for the processor response time to external requests. Processor requests themselves are constrained by the System interface request protocol, and request cycle counts can be determined by examining the protocol. The following System interface interactions can vary within minimum and maximum cycle counts:

- spacing between requests within a cluster (*cluster request spacing*)
- waiting period for the processor to release the System interface to slave state in response to an external request (*release latency*)
- response time for an external request that requires a response (*external response latency*).

The remainder of this section describes and tabulates the minimum and maximum cycle counts for these System interface interactions.

Cluster Request Spacing

Processor internal activity determines the minimum and maximum number of unused cycles allowed between the requests within a cluster.

- The minimum number of unused cycles allowed between requests within a cluster is 0: in other words, the requests can be adjacent.
- The maximum number of unused cycles separating requests within a cluster varies depending on the requests that form the cluster.

Table 12-8 summarizes the minimum and maximum number of unused cycles allowed between requests within a cluster.

Table 12-8 Unused Cycles Separating Requests within a Cluster

From Processor Request	To Processor Request	Minimum Unused SClock Cycles	Maximum Unused SClock Cycles
Read	Update	0	2
Read	Write	0	2
Update	Write	0	2

Release Latency

Release latency is generally defined as the number of cycles the processor can wait to release the System interface to slave state for an external request. When no processor requests are in progress, internal activity—such as refilling the primary cache from the secondary cache—can cause the processor to wait some number of cycles before releasing the System interface. Release latency is therefore more specifically defined as the number of cycles that occur between the assertion of **ExtRqst*** and the assertion of **Release***.

There are three categories of release latency:

- Category 1: when the external request signal is asserted two cycles before the last cycle of a processor request, or two cycles before the last cycle of the last request in a cluster.
- Category 2: when the external request signal is not asserted during a processor request or cluster, or is asserted during the last cycle of a processor request or cluster.
- Category 3: when the processor makes an uncompelled change to slave state.

Table 12-9 summarizes the minimum and maximum release latencies for requests that fall into categories 1, 2, 3a and 3b. Note that the maximum and minimum cycle count values are subject to change.

Table 12-9 Release Latency for External Requests

Category	Minimum PCycles	Maximum PCycles
1	4	6
2	4	24
3a	0	See (3a), below
3b	0	See (3b), below

(3a) Read =
T_{dis}
+ 4- or 8-word Secondary cache write cycle time (depending upon Primary cache size)
+ 4-word Secondary cache write cycle time
+ Secondary cache line size
+ 16 PCycles

(3b) Read With Write Forthcoming
4-word Secondary cache Write cycle time
+ 4 PCycles

External Request Response Latency

The number of cycles the processor takes to respond to an external intervention request, read request, or snoop request, are referred to as the *intervention response latency*, *external read response latency*, or *snoop response latency*, respectively.

The number of latency cycles is the number of unused cycles between the address cycle of the request and the first data cycle of the response. Intervention response latency and snoop response latency are a function of processor internal activity and secondary cache access time. Table 12-10 summarizes the minimum and maximum intervention response latency and snoop response latency. Note that the latency values are subject to change.

Table 12-10 Intervention Response and Snoop Response Latencies

Maximum Secondary Cache Access	**Intervention Response Latency**		**Snoop Response Latency**	
	Min	**Max**	**Min**	**Max**
1-4 PCycles	6	26	6	26
5-6 PCycles	8	28	8	28
7-8 PCycles	10	30	10	30
9-10 PCycles	12	32	12	32
11-12 PCycles	14	34	14	34

External read response latency is a function of processor internal activity. Minimum and maximum external read response latency is 4 PCycles.

12.9 System Interface Commands and Data Identifiers

System interface commands specify the nature and attributes of any System interface request; this specification is made during the address cycle for the request. System interface data identifiers specify the attributes of data transmitted during a System interface data cycle.

The following sections describe the syntax, that is, the bitwise encoding of System interface commands and data identifiers.

Reserved bits and reserved fields in the command or data identifier should be set to 1 for System interface commands and data identifiers associated with external requests. For System interface commands and data identifiers associated with processor requests, reserved bits and reserved fields in the command and data identifier are undefined.

Command and Data Identifier Syntax

System interface commands and data identifiers are encoded in 9 bits and are transmitted on the **SysCmd** bus from the processor to an external agent, or from an external agent to the processor, during address and data cycles. Bit 8 (the most-significant bit) of the **SysCmd** bus determines whether the current content of the **SysCmd** bus is a command or a data identifier and, therefore, whether the current cycle is an address cycle or a data cycle. For System interface commands, **SysCmd(8)** must be set to 0. For System interface data identifiers, **SysCmd(8)** must be set to 1.

System Interface Command Syntax

This section describes the **SysCmd** bus encoding for System interface commands. Figure 12-42 shows a common encoding used for all System interface commands.

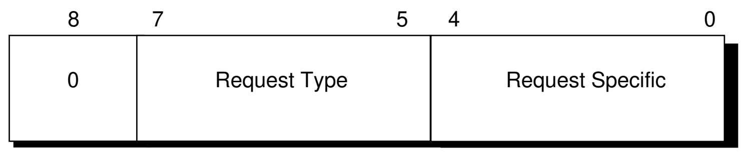

Figure 12-42 System Interface Command Syntax Bit Definition

SysCmd(8) must be set to 0 for all System interface commands.

SysCmd(7:5) specify the System interface request type which may be read, write, null, invalidate, update, intervention, or snoop; Table 12-11 lists the encoding of **SysCmd(7:5)**.

Table 12-11 shows the types of requests encoded by the **SysCmd(7:5)** bits.

Table 12-11 Encoding of SysCmd(7:5) *for System Interface Commands*

SysCmd(7:5)	Command
0	Read Request
1	Read-With-Write-Forthcoming Request
2	Write Request
3	Null Request
4	Invalidate Request
5	Update Request
6	Intervention Request
7	Snoop Request

SysCmd(4:0) are specific to each type of request and are defined in each of the following sections.

Read Requests

Figure 12-43 shows the format of a **SysCmd** read request.

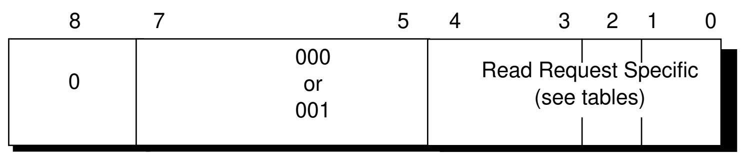

Figure 12-43 Read Request SysCmd *Bus Bit Definition*

Tables 12-12 through 12-14 list the encodings of **SysCmd(4:0)** for read requests.

Table 12-12 Encoding of SysCmd(4:3) *for Read Requests*

SysCmd(4:3)	Read Attributes
0	Coherent block read
1	Coherent block read, exclusivity requested
2	Noncoherent block read
3	Doubleword, partial doubleword, word, or partial word

Table 12-13 Encoding of SysCmd(2:0) *for Coherent and Noncoherent Block Read Request*

SysCmd(2)	Link Address Retained Indication
0	Link address not retained
1	Link address retained
SysCmd(1:0)	**Read Block Size**
0	4 words
1	8 words
2	16 words
3	32 words

Table 12-14 Doubleword, Word, or Partial-word Read Request Data Size Encoding of SysCmd(2:0)

SysCmd(2:0)	Read Data Size
0	1 byte valid (Byte)
1	2 bytes valid (Halfword)
2	3 bytes valid (Tribyte)
3	4 bytes valid (Word)
4	5 bytes valid (Quintibyte)
5	6 bytes valid (Sextibyte)
6	7 bytes valid (Septibyte)
7	8 bytes valid (Doubleword)

Write Requests

Figure 12-44 shows the format of a **SysCmd** write request.

Table 12-15 lists the write attributes encoded in bits **SysCmd(4:3)**. Table 12-16 lists the block write replacement attributes encoded in bits **SysCmd(2:0)**. Table 12-17 lists the write request bit encodings in **SysCmd(2:0)**.

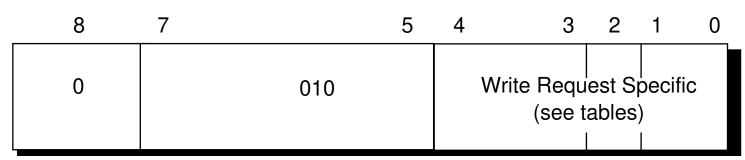

Figure 12-44 Write Request SysCmd *Bus Bit Definition*

Table 12-15 Write Request Encoding of SysCmd(4:3)

SysCmd(4:3)	Write Attributes
0	Reserved
1	Reserved
2	Block write
3	Doubleword, partial doubleword, word, or partial word

Table 12-16 Block Write Request Encoding of SysCmd(2:0)

SysCmd(2)	Cache Line Replacement Attributes
0	Cache line replaced
1	Cache line retained†
SysCmd(1:0)	**Write Block Size**
0	4 words
1	8 words
2	16 words
3	32 words

†The only time the processor sets this bit is if a Hit Writeback causes the processor to execute a write request (see Cache Write Policy in Chapter 11).

Table 12-17 Doubleword,Word, or Partial-word Write Request Data Size Encoding of SysCmd(2:0)

SysCmd(2:0)	Write Data Size
0	1 byte valid (Byte)
1	2 bytes valid (Halfword)
2	3 bytes valid (Tribyte)
3	4 bytes valid (Word)
4	5 bytes valid (Quintibyte)
5	6 bytes valid (Sextibyte)
6	7 bytes valid (Septibyte)
7	8 bytes valid (Doubleword)

Null Requests

Figure 12-45 shows the format of a **SysCmd** null request.

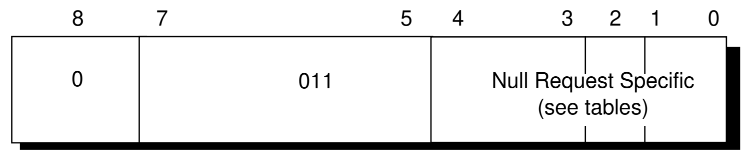

Figure 12-45 Null Request SysCmd *Bus Bit Definition*

Processor null write requests, System interface release external null requests, and secondary cache release external null requests all use the null request command. Table 12-18 lists the encodings of **SysCmd(4:3)** for processor null write requests. Table 12-19 lists the encodings of **SysCmd(4:3)** for external null requests.

SysCmd(2:0) are reserved for both instances of null requests.

Table 12-18 Processor Null Write Request Encoding of SysCmd(4:3)

SysCmd(4:3)	Null Write Attributes
0	Null write
1	Reserved
2	Reserved
3	Reserved

Table 12-19 External Null Request Encoding of SysCmd(4:3)

SysCmd(4:3)	Null Attributes
0	System Interface release
1	Secondary cache release
2	Reserved
3	Reserved

Invalidate Requests

Figure 12-46 shows the format for an invalidate request, and Table 12-20 lists the encodings of **SysCmd(4:0)** for an external invalidate request.

SysCmd(4:0) are reserved on a processor invalidate request.

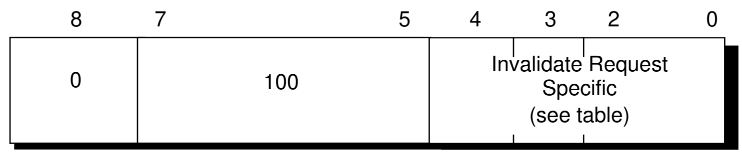

Figure 12-46 Invalidate Request SysCmd *Bus Bit Definition*

Table 12-20 Encoding of SysCmd(4:0) *for External Invalidate Requests*

SysCmd(4)	**Processor Unacknowledged Invalidate or Update Cancellation**
0	Invalidate or Update cancelled
1	No cancellation
SysCmd(3:0)	Reserved

Update Requests

Figure 12-47 shows the format for a **SysCmd** update request.

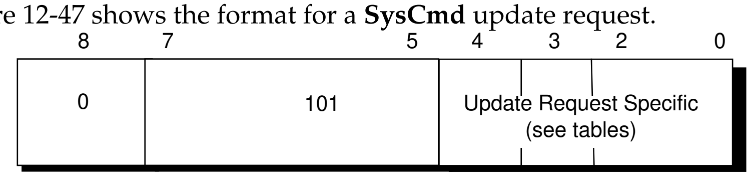

Figure 12-47 Update Request SysCmd *Bus Bit Definition*

Table 12-21 lists the encodings of **SysCmd(4:0)** for external update requests. Table 12-22 lists the encodings of **SysCmd(4:0)** for processor update requests. The remaining upper bits are the same for both processor and external update requests.

Table 12-21 Encoding of SysCmd(4:0) *for External Update Requests*

SysCmd(4)	Processor Unacknowledged Invalidate or Update Cancellation
0	Invalidate or Update cancelled
1	No cancellation
SysCmd(3)	**Update Cache State Change Attributes**
0	Cache state changed to shared
1	No change to cache state
SysCmd(2:0)	**Update Data Size**
0	1 byte valid (Byte)
1	2 bytes valid (Halfword)
2	3 bytes valid (Tribyte)
3	4 bytes valid (Word)
4	5 bytes valid (Quintibyte)
5	6 bytes valid (Sextibyte)
6	7 bytes valid (Septibyte)
7	8 bytes valid (Doubleword)

Table 12-22 Encoding of SysCmd(4:0) *for Processor Update Requests*

SysCmd(4)	Reserved
SysCmd(3)	**Update type**
0	Compulsory
1	Potential
SysCmd(2:0)	**Update Data Size**
0	1 byte valid (Byte)
1	2 bytes valid (Halfword)
2	3 bytes valid (Tribyte).
3	4 bytes valid (Word)
4	5 bytes valid (Quintibyte)
5	6 bytes valid (Sextibyte)
6	7 bytes valid (Septibyte)
7	8 bytes valid (Doubleword)

Intervention and Snoop Requests

Figure 12-48 shows the format of an intervention request; Figure 12-49 shows the format of a snoop request. Table 12-23 lists the encodings of **SysCmd(4:0)** for intervention requests; Table 12-24 lists the encodings **SysCmd(4:0)** for snoop requests.

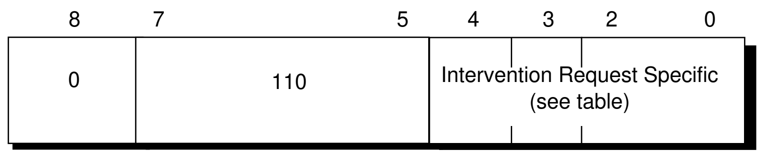

Figure 12-48 Intervention Request SysCmd *Bus Bit Definition*

Table 12-23 Encodings of SysCmd(4:0) *for Intervention Requests*

SysCmd(4)	Processor Unacknowledged Invalidate or Update Cancellation
0	Update or Invalidate cancelled
1	No cancellation
SysCmd(3)	**Response to Dirty or Exclusive State**
0	Return cache line data if in the dirty exclusive or dirty shared state
1	Return cache line data if in the clean exclusive or dirty exclusive state
SysCmd(2:0)	**Cache State Change Function**
0	No change to cache state
1	If cache state is clean exclusive, change to shared; otherwise no change to cache state
2	If cache state is clean exclusive or shared, change to invalid; otherwise no change to cache state
3	If cache state is clean exclusive, change to shared; if cache state is dirty exclusive, change to dirty shared; otherwise make no change to cache state
4	If cache state is clean exclusive, dirty exclusive, or dirty shared, change to shared; otherwise make no change to cache state
5	Change to invalid regardless of current cache state
6	Reserved
7	Reserved

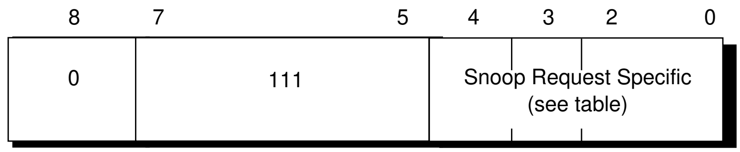

Figure 12-49 Snoop Request SysCmd *Bus Bit Definition*

Table 12-24 Encodings of SysCmd(4:0) *for Snoop Requests*

SysCmd(4)	**Processor Unacknowledged Update Cancellation**
0	Update cancelled
1	No cancellation
SysCmd(3)	Reserved
SysCmd(2:0)	**Cache State Change Function**
0	No change to cache state
1	If cache state is clean exclusive, change to shared state; otherwise make no change to cache state
2	If cache state is clean exclusive or shared, change to invalid state; otherwise make no change to cache state
3	If cache state is clean exclusive, change to shared; if cache state is dirty exclusive, change to dirty shared; otherwise make no change to cache state
4	If cache state is clean exclusive, dirty exclusive, or dirty shared, change to shared; otherwise make no change to cache state
5	Change to invalid regardless of current cache state
6	Reserved
7	Reserved

System Interface Data Identifier Syntax

This section defines the encoding of the **SysCmd** bus for System interface data identifiers. Figure 12-50 shows a common encoding used for all System interface data identifiers.

8	7	6	5	4	3	2 … 0
1	Last Data	Resp Data	Err Data	*See Note below*	Reserved	Cache State

Figure 12-50 Data Identifier SysCmd *Bus Bit Definition*

SysCmd(8) must be set to 1 for all System interface data identifiers.

> **NOTE: SysCmd(4)** is reserved for processor data identifier. In an external data identifier, **SysCmd(4)** indicates whether or not to check the data and check bits for error.

System interface data identifiers have two formats, one for coherent data and another for noncoherent data.

Coherent Data

Coherent data is defined as follows:

- data that is returned in response to a processor coherent block read request
- data that is returned in response to an external intervention request.

Noncoherent Data

Noncoherent data is defined as follows:

- data that is associated with processor block write requests and processor doubleword, partial doubleword, word, or partial word write requests
- data that is returned in response to a processor noncoherent block read request or a processor doubleword, partial doubleword, word, or partial word read request
- data that is associated with external update requests
- data that is associated with external write requests
- data that is returned in response to an external read request
- data that is associated with processor update requests.

Data Identifier Bit Definitions

SysCmd(7) marks the last data element and **SysCmd(6)** indicates whether or not the data is response data, for both processor and external coherent and noncoherent data identifiers. Response data is data returned in response to a read request or an intervention request.

SysCmd(5) indicates whether or not the data element is error free. Erroneous data contains an uncorrectable error and is returned to the processor, forcing a bus error. In the case of a block response, the entire line must be delivered to the processor no matter how minimal the error. The processor delivers data with the good data bit deasserted if a primary parity error is detected for a transmitted data item. If the system is in ECC mode, a secondary cache data ECC error is detected by comparing the values transmitted on the **SysAD** and **SysADC**.

SysCmd(4) indicates to the processor whether to check the data and check bits for this data element, for both coherent and noncoherent external data identifiers.

SysCmd(3) is reserved for external data identifiers.

SysCmd(4:3) are reserved for both coherent and noncoherent processor data identifiers.

SysCmd(2:0) indicate the data cache state to load the cache line, in response to processor coherent read requests for coherent data identifiers. **SysCmd(2:0)** also indicate the cache state for response data to an external intervention request, or for the data cycle issued in response to an external snoop request. **SysCmd(2:0)** are reserved for noncoherent data identifiers.

Table 12-25 lists the encodings of **SysCmd(7:3)** for processor data identifiers. Table 12-26 lists the encodings of **SysCmd(7:3)** for external data identifiers. Table 12-27 lists the encodings of **SysCmd(2:0)** for coherent data identifiers.

Table 12-25 Processor Data Identifier Encoding of SysCmd(7:3)

SysCmd(7)	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd(6)	**Response Data Indication**
0	Data is response data
1	Data is not response data
SysCmd(5)	**Good Data Indication**
0	Data is error free
1	Data is erroneous
SysCmd(4:3)	Reserved

Table 12-26 External Data Identifier Encoding of SysCmd(7:3)

SysCmd(7)	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd(6)	**Response Data Indication**
0	Data is response data
1	Data is not response data
SysCmd*(5)*	**Good Data Indication**
0	Data is error free
1	Data is erroneous
SysCmd(4)	**Data Checking Enable**
0	Check the data and check bits
1	Do not check the data and check bits
SysCmd(3)	Reserved

Table 12-27 Coherent Data Identifiers Encoding of SysCmd(2:0)

SysCmd(2:0)	Cache State
0	Invalid†
1	Reserved
2	Reserved
3	Reserved
4	Clean Exclusive
5	Dirty Exclusive
6	Shared
7	Dirty Shared

†This state also occurs if the line does not exist in the cache.

12.10 System Interface Addresses

System interface addresses are full 36-bit physical addresses presented on the least-significant 36 bits (bits 35 through 0) of the **SysAD** bus during address cycles; the remaining bits of the **SysAD** bus are unused during address cycles.

Addressing Conventions

Addresses associated with doubleword, partial doubleword, word, or partial word transactions and update requests, are aligned for the size of the data element. The system uses the following address conventions:

- Addresses associated with block requests are aligned to double-word boundaries; that is, the low-order 3 bits of address are 0.
- Doubleword requests set the low-order 3 bits of address to 0.
- Word requests set the low-order 2 bits of address to 0.
- Halfword requests set the low-order bit of address to 0.
- Byte, tribyte, quintibyte, sextibyte, and septibyte requests use the byte address.

Sequential and Subblock Ordering

The order in which data is returned in response to a processor block read request can be programmed to sequential ordering or subblock ordering, using the boot-time mode control interface. Appendix C has more information about subblock ordering. Either sequential or subblock ordering may be enabled, as follows:

- If sequential ordering is enabled on a block read request, the processor delivers the address of the doubleword at the start of the block. An external agent must return the block of data sequentially from the beginning of the block.
- If subblock ordering is enabled, the processor delivers the address of the requested doubleword within the block. An external agent must return the block of data using subblock ordering, starting with the addressed doubleword.

NOTE: Only R4000SC and R4000MC configurations (using a secondary cache) can be programmed to use sequential ordering.

For block write requests, the processor always delivers the address of the doubleword at the beginning of the block; the processor delivers data beginning with the doubleword at the beginning of the block and progresses sequentially through the doublewords that form the block.

During data cycles, the valid byte lines depend upon the position of the data with respect to the aligned doubleword (this may be a byte, halfword, tribyte, quadbyte/word, quintibyte, sextibyte, septibyte, or an octalbyte/doubleword). For example, in little-endian mode, on a byte request where the address modulo 8 is 0, **SysAD(7:0)** are valid during the data cycles.

12.11 Processor Internal Address Map

External reads and writes provide access to processor internal resources that may be of interest to an external agent. The processor decodes bits **SysAD(6:4)** of the address associated with an external read or write request to determine which processor internal resource is the target. However, the processor does not contain any resources that are *readable* through an external read request. Therefore, in response to an external read request the processor returns undefined data and a data identifier with its *Erroneous Data* bit, **SysCmd(5)**, set. The *Interrupt* register is the only processor internal resource available for *write* access by an external request. The *Interrupt* register is accessed by an external write request with an address of 000_2 on bits 6:4 of the **SysAD** bus.