

## **Device quarantines (simplified) as submitted to the p1394.1 committee**

Dr. David V. James, Sony  
3300 Zanker Road, MS SJ3C1  
San Jose, CA 95134-4591  
Phone: 408 955-6295  
FAX: 408 955-4591  
Email: dvj@alum.mit.edu

**October 12 1999**

**This contribution is one of several, presented for independent review.  
An overall contribution, which provides the context for multiple contributions,  
is also provided in BR047R08.**

Nodes must be signaled to discard their EUI-to-nodeID translations when dirty bus\_IDs are recycled.  
Since robust confirmed signals are not available, node's are quarantined at their source.  
Attempting to access resources results in a returned response error.  
The quarantine can then be retried.  
Possibilities for time-stamp values, to minimize the number of translation invalidations, can be explored.



# High performance Serial Bus bridges

## 1.6.6 Quarantined requester rejections

Nodes are sometimes “quarantined”, to prevent the use of stale EUI-to-nodeId translations. Requests from quarantines source nodes are rejected until the quarantine condition is released, as illustrated in figure 1 and figure 2. These quarantined remote-bus transaction are terminated with a distinctive-error indication by the destination-side portal.

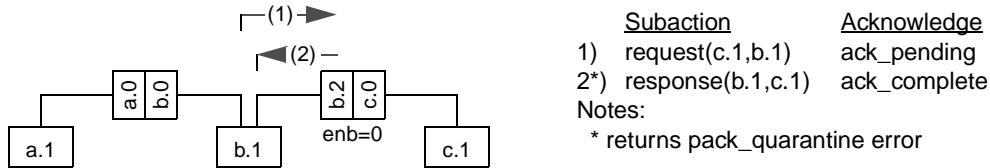


Figure 1—Quarantined transaction sequence

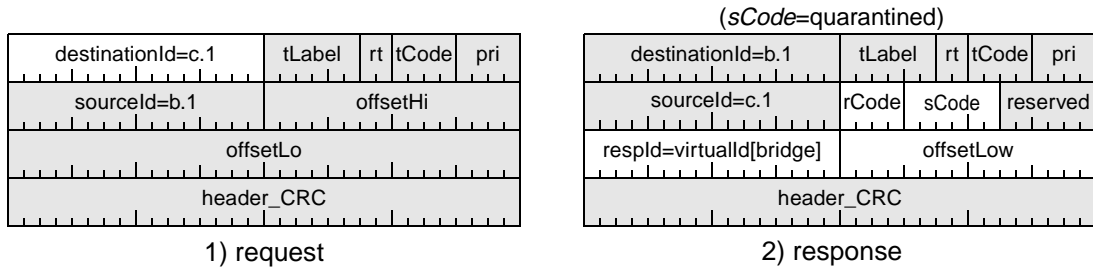


Figure 2—Quarantined subaction formats

A returned quarantine-rejection indication is expected to cause the bridge aware node to discard its (possibly) stale EUI-to-nodeId translations. Bridge-aware nodes are responsible for discarding these translations before clearing their quarantine condition; clearing of the quarantine condition is described in the following subclause.

### 1.6.7 Bridge-aware requester clears quarantine conditions

Bridge aware nodes clear their quarantine by sending a lock request to the affected portal, using a FirstLock transaction, as illustrated in figure 3 and figure 4.

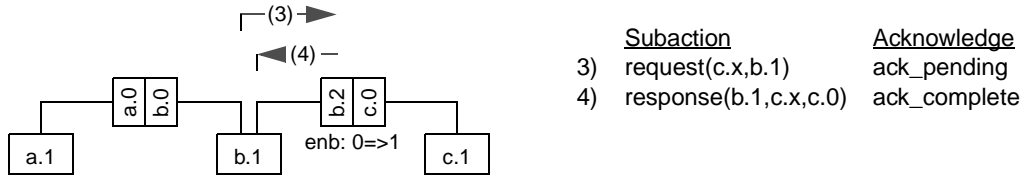


Figure 3—Bridge-aware requester clears quarantine condition

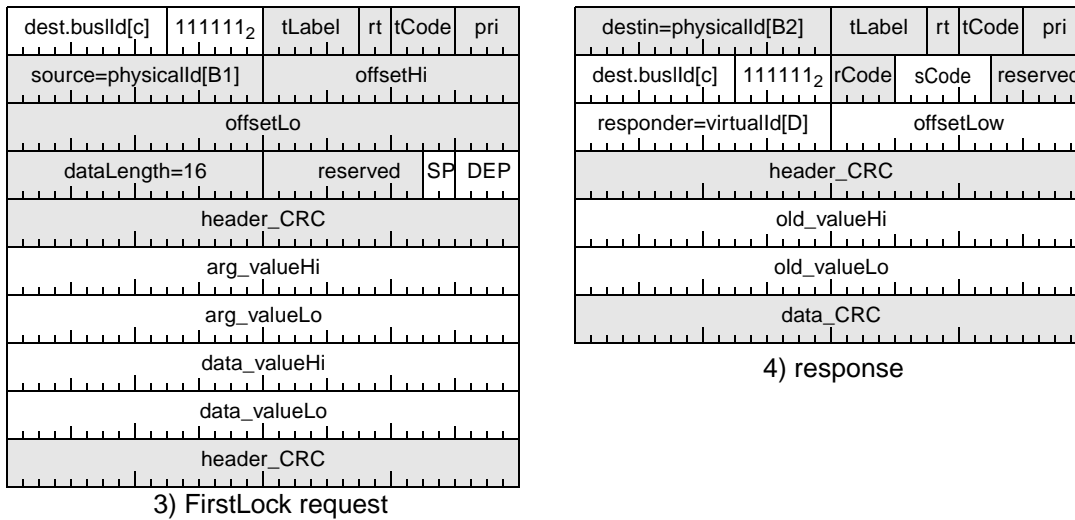


Figure 4—Bridge aware requester clears quarantine

Identifiers in the returned response indicate when the requester’s EUI-to-virtualId translations must be flushed before its remote-bus activity resumes.

TBD—Define the content of these packets (address and data values) in more detail. Alternatives include:

- a) Same format options as SerialBus and also support quadlets.
- b) Same format as shown above, for octlets only.
- c) Always transport and return the same number of bytes.

## 2. Packet routing

### 2.1 Asynchronous subaction routing

#### 2.1.1 Subaction routing model

This subclause specifies the behavior of bridge portals when they receive and transmit requests (read, write, or lock) and responses. These subactions are accepted based on their *destination\_ID* address, quarantined based on the *source\_ID* address, and forwarded to the adjacent bus, as illustrated in figure 5..

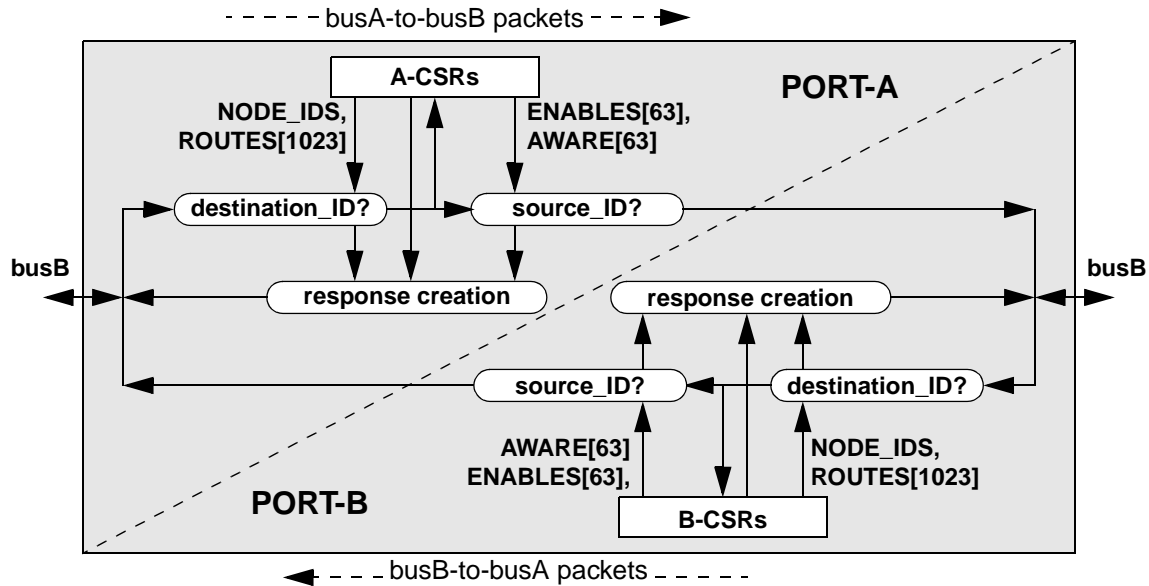


Figure 5—Asynchronous subaction processing

The request and response subactions that are forwarded in this fashion include the following: quadlet write request, block write request, write response, quadlet read request, block read response, lock request, and lock response.

There are 64 enable bits that (when at their initial zero value) quarantine local nodes by disabline the sending of request subactions from local-bus nodes, based on their *source\_Id* identifier. Another set of 64 *bridgeAware* bits selectes between legacy and bridge-aware processing, also based on the packet's *source\_ID* identifier.

When a source node is quarantined (its enable bit is zero), requests generated by that node are rejected with a quarantined indication. After a quarantined response indication is returned, a bridge aware node accepts the responsibility of discarding its (possibly stale) EUI-to-virtualId mappings before setting the bridge-resident enable bit and restarting the previously quarantined transaction.

A legacy device relies on the controller to clear its quarantined state, since legacy nodes were designed before this standard and are therefore not capable of restoring their quarantine state.

## 2.1.2 Consumer portal processing

### 2.1.2.1 Packet consumption (destination\_ID based)

A consuming portal eavesdrops on the bus and selectively accepts packets based on its *destination\_ID* address. These packet are (in concept) labeled to simplify following less time critical processing steps, as specified in table 1. This table applies to packets with valid header\_CRC values; other asynchronous packets shall be ignored..

**Table 1—Packet consumption (destination\_ID based)**

Inputs					Row	Consumer processing	
type	delta	busID	localID	routel		label	acknowledge
GASP	—	—	—	—	1	THROUGH	—na—
not GASP	—	3FF <sub>16</sub>	phyID	—	2	PORTAL	see table 2
		busId	phyID	—	3	VIRTUAL	
	1	busId	3F <sub>16</sub>	—	4	REJECT	
		3FF <sub>16</sub>	badID	—	5		
		busId	badID	—	6		
		diffId	—	FREE	7		
				DIRTY	8		
	—	diffId	—	FORW	9	THROUGH	

Notes:

route= ROUTES[destination\_ID.bus\_ID]  
 valid= SelfIDInfo[destination\_ID.local\_ID]  
 deltaPortal is 1 for delta portal, 0 otherwise

**Row 1:** GASP packets are accepted by all portals (excluding the transmitting portal).

**Row 2:** All portals shall accept physical-address packets directed to themselves.

**Row 3:** All portals shall accept virtual-address packets directed to themselves.

**Row 4:** The delta portal accepts packets directed to its aliased virtual-address.

**Row 5:** An delta portal shall accept packets to nonexistent fixed-bus addresses (these are later rejected).

**Row 6:** An delta portal shall accept packets to nonexistent local-bus addresses (these are later rejected).

**Row 7:** An delta portal shall accept packets to available remote-bus addresses (these are later rejected).

**Row 8:** An delta portal shall accept packets to retired remote-bus addresses (these are later rejected).

**Row 9:** A portals accepts packets that are destined for the adjacent bus.

### 2.1.3 Consumer acknowledge codes

The acknowledge for accepted request and response subactions depends on the consumer's conditions when the packet is observed, as specified in table 2.

**Table 2—Consumer acknowledge codes**

type	Condition	Row	Acknowledge
—	A data_CRC or data_length error (primary failure)	1	ack_busy_X
		2	ack_data_error
	A data_CRC or data_length error (secondary failure)	3	ack_data_error
	Overly long (data_length>max_length)	4	ack_type_error
	Currently insufficient queue space	5	ack_busy_X, ack_busy_A, ack_busy_B
request	Available queue space	6	ack_pending
response	”	7	ack_complete

**Row 1:** Packets with corrupted payloads are initially discarded; an ack\_busy\_X should be returned.

**Row 2:** Packets with corrupted payloads are initially discarded; an ack\_data\_error may be returned.

**Row 3:** Packets with consistently corrupted payloads are discarded; an ack\_data\_error shall be returned.

**Row 4:** Packets with overly large data\_length values are discarded; an ack\_data\_error shall be returned. In this context, overly large means larger than supported by bridge queue structures.

**Row 5:** Subactions are temporarily busied when the consuming portal has insufficient space. Busy indications specified by the 1394a dual-phase retry protocols shall be used.

**Row 6:** When request packets are accepted and an ack\_pending shall be returned.

**Row 7:** When request packets are accepted and an ack\_pending shall be returned.

### 2.1.4 Consumer portal relabeling (source\_ID based)

After acceptance, the packet's source\_ID determines how the accepted packets are processed at the consuming portal, as specified in table 3.

**Table 3—Consumer portal relabeling**

Inputs					Row	Revised subaction label		
label	firstCode	scope	aware	enabled		Request		Response
						Write	Nonwrite	
THROUGH	—	—	—	0	1	REJECTED		THROUGH
	firstRequest	—	—	1	2	PORTAL		—na—
	not firstRequest	local	0	1	3	POSTED	THROUGH	THROUGH
			1	1	4	THROUGH		THROUGH
			not local	—	—	5		
VIRTUAL	—	local	—	0	6	REJECTED		PORTAL
			1	7	PORTAL			
		not local	—	—	8			

Notes:

scope= (source\_ID.bus\_ID==NODE\_IDS.bus\_ID) ? local:!local;  
 enabled= ENABLE[source\_ID.local\_ID]  
 aware= AWARE[source\_ID.local\_ID]

**Row 1:** Virtual-address request packets are rejected when the local-bus source is quarantined.

**Row 2:** The FirstWrite and FirstRead requests are directed to the accepting portal.

**Row 3:** Nonquarantined legacy write requests are posted (create/return response and forward request).

**Row 4:** Nonquarantined bridge-aware requests and responses are delivered to the adjacent portal.

**Row 5:** Remotely sourced requests and responses are delivered to the adjacent portal.

**Row 6:** Virtual requests for the portal are rejected when the local-bus source is quarantined.

**Row 7:** Nonquarantined requests and responses are delivered to the portal.

**Row 8:** Remotely sourced requests and responses are delivered to the portal.

### 2.1.5 Producer portal processing

### 2.1.6 Producer portal routing

When the packet reaches the adjacent portal the packet's source\_ID is checked to determine when the packet shall be discarded. Packets may be delivered to the portal CSRs, may be discarded, or may be delivered to the transmitter, as specified in table 4.

**Table 4—Producing portal checking**

Inputs								Row	Delivery location	
ready	tCode	ext	route	destBus	port	destPhy	aware		Request	Response
no	—	—	—	—	—	—	—	1	delayed	
yes	GASP	—	FORW	—	—	—	—	2	transmitter	
		—	not FORW	—	—	—	—	3	discarded	
	request	final	—	thisBus	—	—	—	4	portal CSRs	
	request, response	—	—	thisBus	delta	3FF <sub>16</sub>	—	5		
					—	thisPhy	—	6		
					badPhy	0	7	rejected		
					diffPhy	0	8	transmitter	discarded	
					—			9	transmitter	

Notes:

```
#define ready (NODE_IDS.bus_ID==0X3FF)
#define route (ROUTES[source_ID.bus_ID])
#define destBus (destination_ID.bus_ID)
#define destPhy (destination_ID.local_ID)
#define thisBus (NODE_IDS.bus_ID)
#define thisPhy (NODE_IDS.local_ID)
```

**Row 1:** Subaction processing shall be delayed until *NODE\_IDS.bus\_ID* changes from its initial value.

**Row 2:** GASP packets are transmitted if their “response” would return in the opposing direction.

**Row 3:** GASP packets are discarded when passing through redundant routing paths.

**Row 4:** The FinalWrite and FinalRead requests are directed towards the destination-bus delta portal.

**Row 5:** The DeltaWrite/DeltaRead/DeltaSwap requests are directed to the destination-bus delta portal.

**Row 6:** Adjacent portal accesses shall be handled within the bridge and shall not appear on the bus.

**Row 7:** Nonexistent local-bus transactions are rejected internally and never appear on the bus.

**Row 8:** Returning response packets shall be discarded when addressed to legacy nodes.

**Row 9:** Other requests and responses are delivered to the adjacent bus.

Note that the GASP packet routing is based on the return route that would be taken by a “response”, if one were to be generated. When compared to routing based on the initial spanning-tree on/off information, routing broadcast GASP packets in this fashion has several benefits:

- 1) Reduced costs. One set of routing tables is sufficient to route directed as well as broadcast packets.
- 2) Deadlock free. Any ROUTE[] tables that are deadlock free for directed asynchronous subactions are also known to be deadlock free for broadcast GASP deliveries.
- 3) Efficient. This forwarding algorithm remains efficient and consistent when the initial ROUTE[] tables are adjusted by reconfiguration software.

NOTE—Although no additional cycle start routing tables are required: the ROUTE[] tables must either be shared (which required multiplexing hardware) or replicated (which increases the amount of bridge storage).

### 2.1.7 Request acknowledge errors

When ready for transmission, the processing of the request depends on pre-transmission conditions and (if transmitted) the acknowledgment code that is returned, as specified in table 5 Whenever a response is generated by a bridge portal, the virtualId of the bridge shall be returned in the response; the location of this field is specified in 7.2.

**Table 5—Subaction transmission processing**

Condition	Row	request processing		response action
		rCode	sCode	
ack_pending	1	discard		discard, errorCount+= 1
ack_complete	2	resp_complete	observed ack	discard
ack_data_error	3	resp_data_error	”	truncated
ack_conflict_error	4	resp_conflict_error	”	discard errorCount+= 1
ack_tardy	5			
ack_type_error	6			
ack_address_error	7			
timer>timeLimit	8	resp_conflict_error	ext_delay_error	
missing acknowledge	9	discard, errorCount+=1		
ack_busy_X, ack_busy_A, ack_busy_B	10	retried		

Note:  
timeLimit= (timeArrival +(request?Treq:Tres));

**Row 1:** An ack\_pending indicates the request was accepted and can be discarded by the producer.

**Row 2:** An ack\_complete completes the a write request or indicates the response was accepted.

**Row 3:** An ack\_data\_error terminates the transaction; a request changes to an error-reporting response. A response has its payload removed; resp\_data\_error and ext\_data\_error codes are inserted.

**Row 4: Row 5: Row 6: Row 7:** An `ack_conflict_error`, `ack_tardy`, `ack_type_error`, or `ack_address_error` completes the transaction; the request is used to produce an error-reporting response.

**Row 8:** An excessive delay terminates the transmission; a request changes to error-reporting response. Note that a bridge portal uses `Treq` and `Tres` times to determine when the retries shall be terminated, not the Serial Bus defined `retry-count` and `retry-time` registers.

**Row 9:** A missing acknowledge causes the producer to discard request and response subactions.

**Row 10:** Request and response subactions are normally retried when busy indications are returned; dual-phase retry protocols shall be used.

