

**BR075R00 (submitted for p1394.1 committee vote):
Bus discovery**

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

December 1, 1999

**This contribution is one of several, presented for review and incorporation.
An overall contribution, which provides an overall context for this and other contributions, is presented in BR047R10.**

A simple protocol for supporting bus/node discovery is proposed.
This involves having portals keep the EUIs of attached nodes in accessible registers.
Although GASP-based protocols may be more efficient, their error recovery is not well defined, so this is needed.

1.3.5 Node discovery

Bus bridge tables contain sufficient information to discover the presence or absence of potential EUI addresses. Two types of tables are provided: an *ASYNC_ROUTE* table and an *EUI_PRESENCE* table. The 256-byte *ASYNC_ROUTE* table has 1024 two-bit entries that can be read to verify the presence or absence of specific busID addresses, as illustrated in figure 1. In this example, node a.1 reads the bus-bridge resident table to determine the presence of busID addresses {a,b,c}.

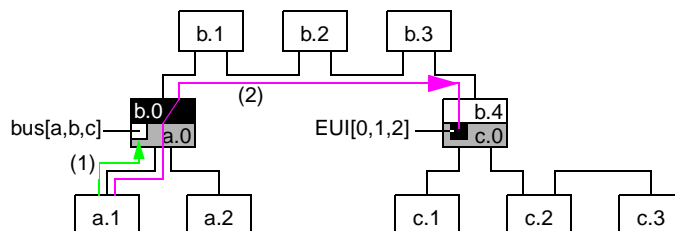


Figure 1—EUI discovery sequences

To fully discover which nodes are present, an additional FinalRead transaction is directed (2) to the *EUI_PRESENCE* registers on bus bridge portal *c.0*. The FinalRead transaction (see 1.11.2) allows this transaction to be directed to the final portal on a selected bus, without knowledge of that portal's phyID address.

The FinalRead transaction returns the EUI-64 identifiers corresponding to each of the 63-possible phyID addresses, and a redundant copy of the EUI-64 identifier corresponding to the alpha portal on that bus. This allows remotely located nodes to efficiently determine the identities of attached nodes and the phyID address of the alpha portal.

This discovery sequence works well when a bus number changes once or multiple times. The requester, after detecting the absence of a previous bus number, can simply check new bus numbers for the continued presence of its responder node.

9. Bridge state

9.1 CSR accessible state

9.1.1 Register offsets

The bridge implements a distinct set of registers, as described in this clause and listed in table 1.

Table 1—Bridge-specific registers

Offset address	Register name	Description
TBDa—TBDa+512	EUI_PRESENCE	EUIs of local-bus nodes
TBDb—TBDb+256	ASYNC_MAP	Asynchronous route, based on destination_ID
TBDc—TBDc+8	REMOTE_TIMEOUT	Split response timeout for remote-bus transactions
TBDd—TBDd+8	QUARANTINES	Quarantine bits
TBDe—TBDe+8	ERROR_COUNT	Cumulative error counter, for diagnostic purposes

9.1.2 EUI_PRESENCE tables

A bridge shall provide EUI_PRESENCE tables for identifying the nodes which are present, as illustrated in figure 2.

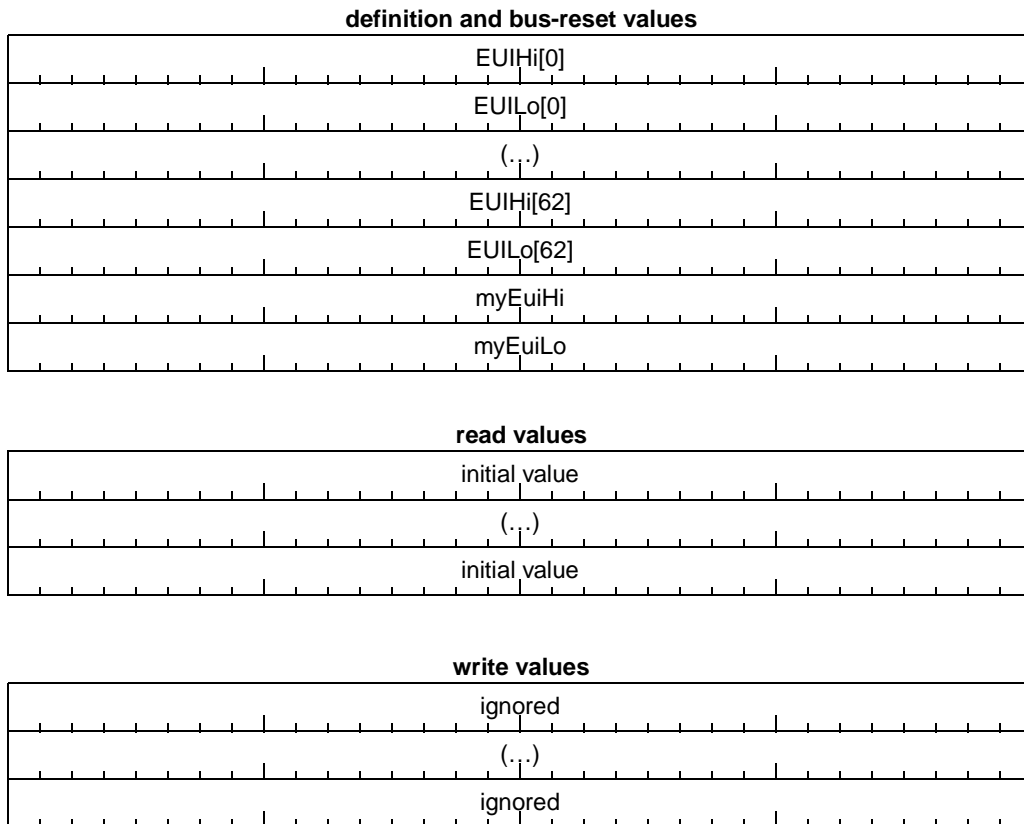


Figure 2—EUI_PRESENCE register formats

9.1.3 ASYNC_ROUTE routing tables

A bridge shall provide ASYNC_ROUTE tables for routing of asynchronous request and response subactions. The directed routing tables use 1024 2-bit fields to specify which asynchronous traffic is forwarded to the remote bus, based on its 10-bit *destination_ID* address, as illustrated in figure 3.

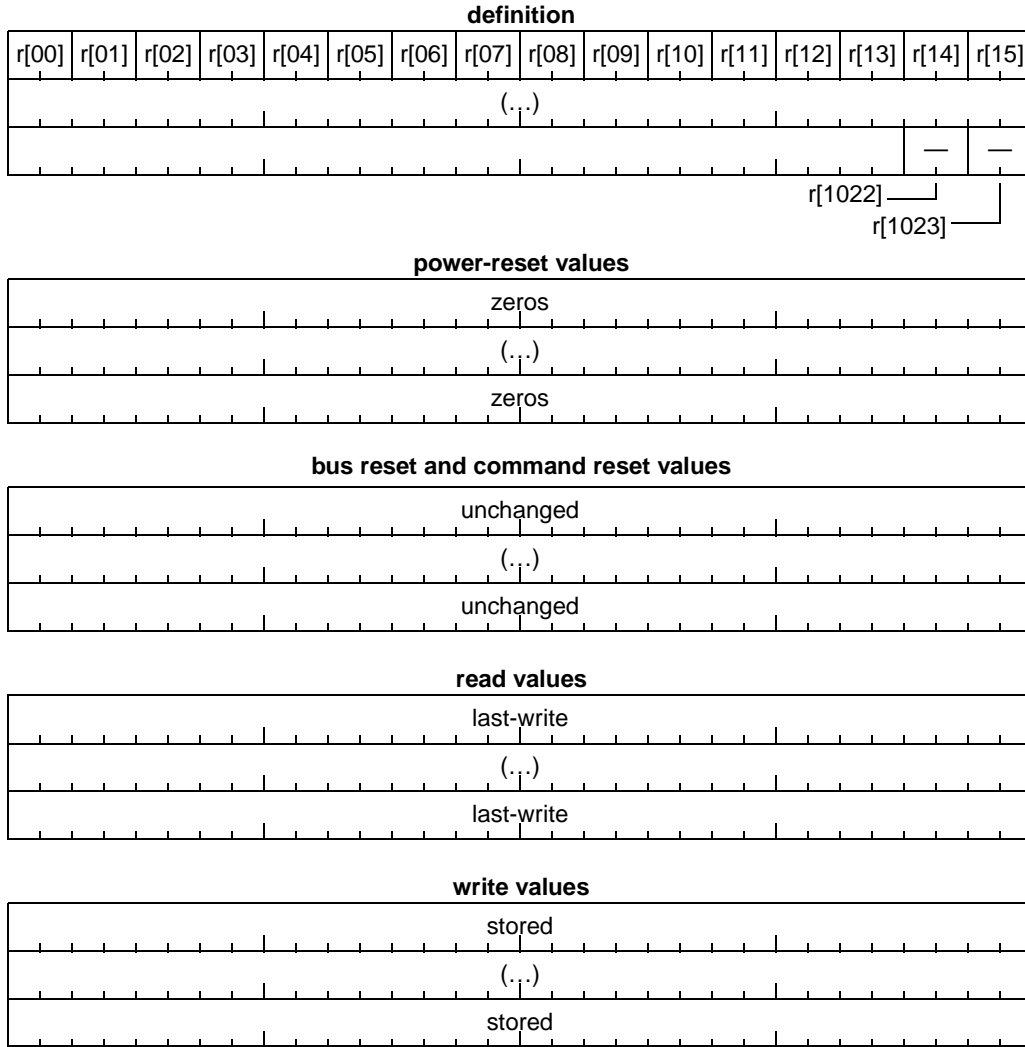


Figure 3—ASYNC_ROUTE register formats

Each of the *r[]* values specifies one of four routing possibilities, as listed in table 2. In a global sense, the term “valid” refers to either of the FORW or USED states; the term “invalid” refers to either of the DIRTY or FREE states, as noted in table 2.

Table 2—Route field r[] values

Validity	Value	Name	Description
invalid	0	FREE	Bus number not mapped, not recently used
	1	DIRTY	Bus number not mapped, but recently used
valid	2	USED	Bus number mapped, but through another portal
	3	FORW	Bus number mapped, through this portal

