

CONGRUENT SOFTWARE, INC.
98 Colorado Avenue
Berkeley, CA 94707
(510) 527-3926
(510) 527-3856 FAX

FROM: Peter Johansson
 TO: IEEE P1394.1 Working Group
 DATE: March 26, 1999
 RE: Bus ID enumeration in a net of bridged Serial Buses

Recent virtual node ID discussions in the working group have resolved most of the issues surrounding the assignment of 6-bit virtual IDs on a particular bus—but have left open questions about the assignment of the 10-bit bus ID. This memorandum proposes a method for the election of a singular ***prime portal*** from among all the bridge portals in a net; the prime portal is subsequently responsible for the unique enumeration (assignment of 10-bit bus ID) for all the buses in the net. These ideas were first worked out on the back of a napkin at an *ad hoc* design session in Huntington Beach and are the result of collaboration by a number of P1394.1 working group participants.

The graphic symbols illustrated by Figure 1 are used throughout the rest of the document and bear some explanation.

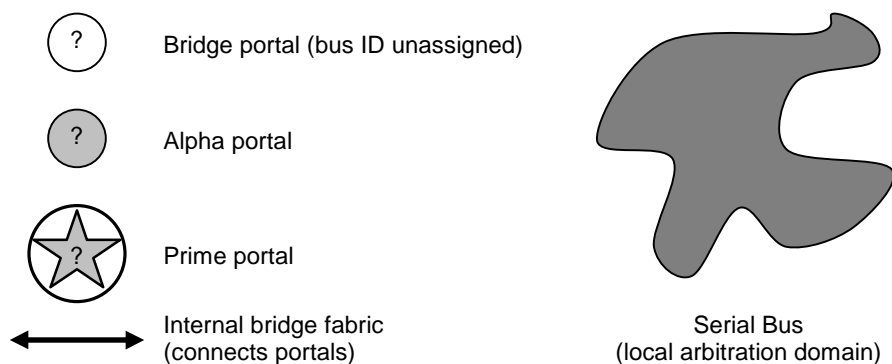


Figure 1 —Legend for bridge illustrations

A collection of two or more nodes that are joined by cables into a single arbitration domain with one root node (a local bus), is illustrated as an amorphous shape shaded gray. The details of the nodes present within the local bus are omitted because they are not relevant to the discussion. A degenerate bus that consists of only one node is never represented by such a shaded shape.

A **bridge portal**, one half of a bridge, is illustrated as a circle containing the enumerated bus ID of the connected bus. The structure of the bridge is shown by connecting two portals with a double-headed arrow that represents the **internal bridge fabric**. The fabric is not meant to represent any particular implementation; the two bridge portals may be co-located within a single enclosure or may be geographically separated. Of the two portals that form a bridge, one is always the **alpha portal**. When a bridge is part of a configured net, the alpha portal is the furthest of the two portals from the prime portal; it is shown shaded. The prime portal, a singular bridge portal within a net, is shown with a five-pointed star.

When a bridge is powered, it forms a degenerate Serial Bus net even if it is connected to no other devices. This isolated case is illustrated by Figure 2.

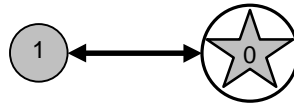


Figure 2 – Isolated bridge

The bridge shall configure itself so that one portal is the prime portal and the other is an alpha portal. The prime portal shall enumerate bus IDs for the two buses interconnected by the bridge, even though the buses are degenerate and consist of only their respective bridge portals. The bus IDs zero and one are illustrative, only; the prime portal may assign bus IDs according to an implementation-dependent algorithm.

The prime portal is also responsible to configure asynchronous routing tables for the net. Each portal has an inbound routing table with room for 1023 entries; a TRUE value for *route[bus_ID]* enables asynchronous subactions destined for *bus_ID* to be forwarded by the portal. The routing tables for an isolated bridge are simple and complementary, as illustrated by the table below.

<i>bus_ID</i>	Alpha portal	Prime portal
0	TRUE	FALSE
1	FALSE	TRUE
all other values	FALSE	

Although no other devices are present on either bus, the net is fully configured and available to route asynchronous transactions. As devices are connected to either bus, they are assigned a virtual node ID by their adjacent bridge portal and become immediately accessible *via* remote requests from devices on the other bus.

What happens when another bridge portal is added to a bus that already contains at least one bridge portal? Subsequent to the bus reset that follows insertion or power reset, all bridge portals shall execute a reconfiguration algorithm described below. The algorithm is explained in conjunction with Figure 3, which shows the connection of an isolated bridge to a configured net with an active bus. For the purposes of discussion, it is not important whether the bus with a *bus_ID* of one is degenerate (as shown) or populated.

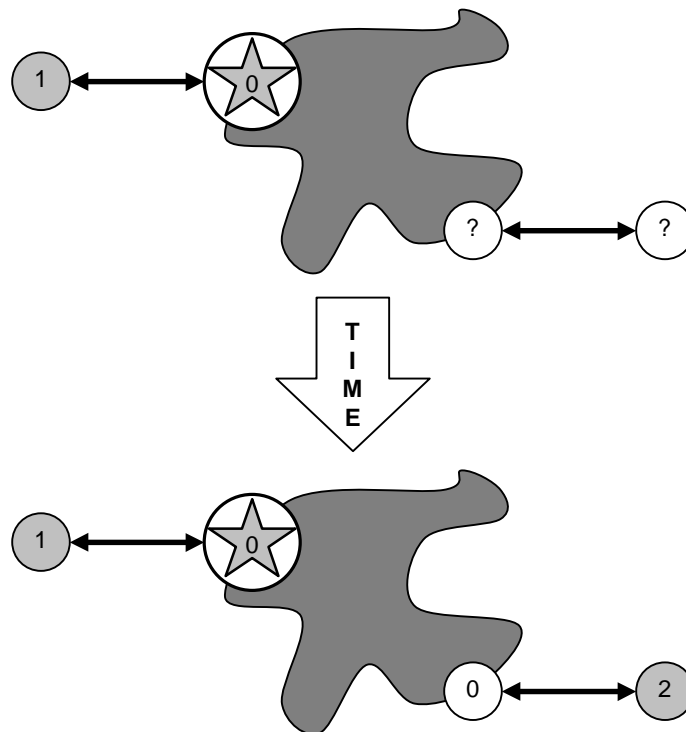


Figure 3 –Prime portal determination (new bridge portal)

A bridge portal that experiences a bus reset shall execute the following algorithm to determine the prime portal:

- a) During the self-identification process that follows bus reset, the portal shall monitor all self-ID packets in order to determine the *phy_ID* of all bridge portals connected to the bus;
- b) The newly connected portal shall interrogate¹ the other bridge portals to determine whether or not any identify themselves as alpha portals; and
- c) If there is one and only one alpha portal, the newly connected portal shall designate the bridge's other portal an alpha portal and request bus ID enumeration and routing table updates from the prime portal (whose virtual node ID has been obtained from the alpha portal).

In the example illustrated by , the alpha portal and the prime portal are one and the same—but the algorithm would be equally valid if the prime portal were the one on the degenerate bus. In either case, the prime portal assigns *bus_ID* two to the newly connected bridge's degenerate bus, confirms the identity of *bus_ID* zero to the newly connected portal and programs routing information for both portals.

More than one portal might identify itself as an alpha portal; this is the case when configured, multiple bus nets are joined. The algorithm is more complex, but is an extension of the simple procedure outlined above.

¹ The mechanism has yet to be determined, whether a simple read of CSR space or a command-based method. One idea is to define a new extended transaction code for lock that permits the exchange of information within a concatenated transaction.

The example below provides adequate complexity to explore the issues when two nets become connected. In Figure 4, two nets are about to be joined. Both nets consist of two buses, although the net on the right is shown with a degenerate bus both for the sake of simplicity and to illustrate the concept of weighted survival.

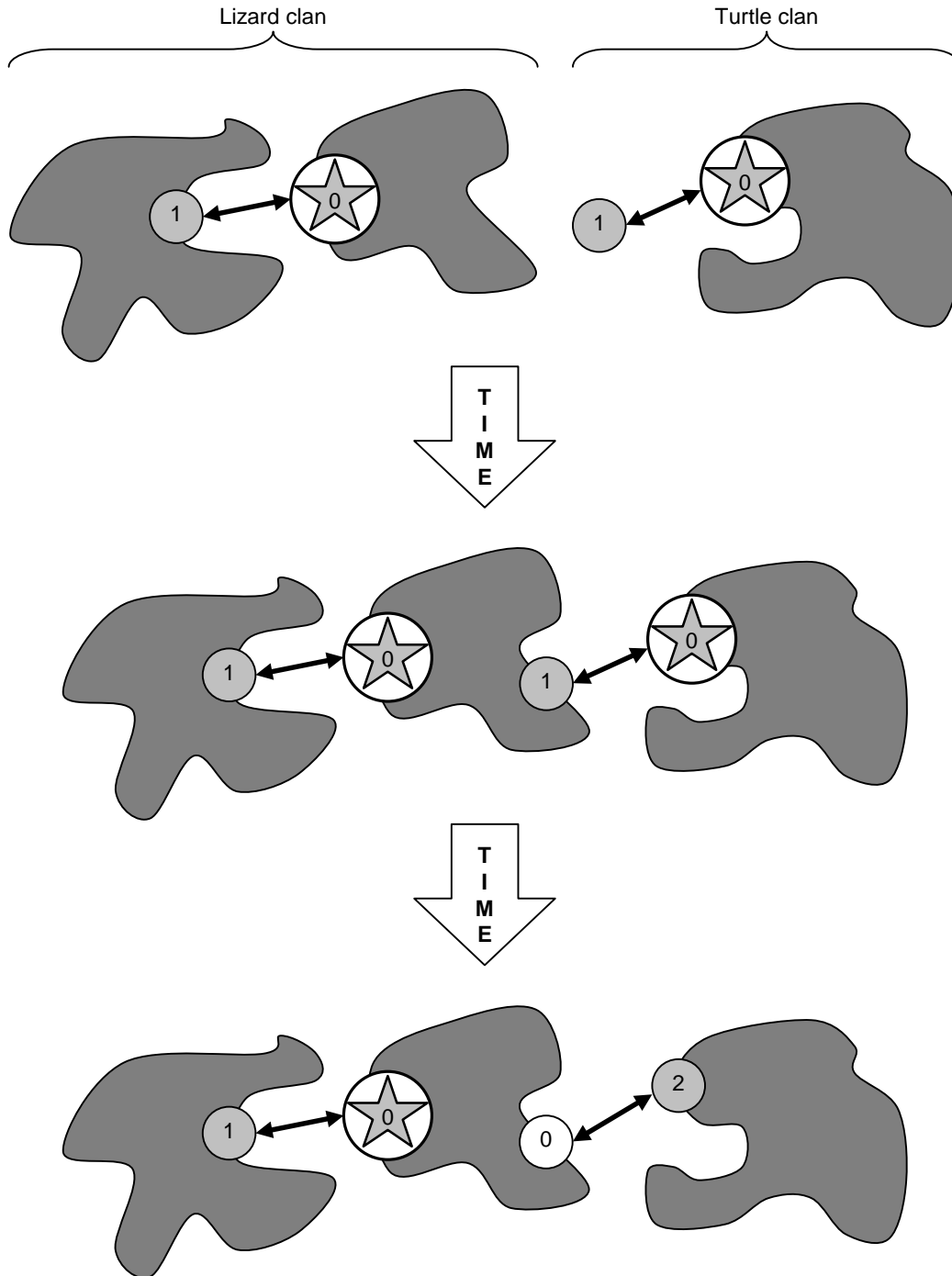


Figure 4 –Connecting two Serial Bus nets

Prior to the connection event, both nets contain a prime portal and a corresponding alpha portal. Assume that a similar bus ID enumeration method is employed by both prime

portals; this leads to an identical assignment of bus numbers (this will be useful to explain how duplicate bus IDs are resolved when two nets are joined).

At some point in time, a physical connection is made between a node in the left-hand net and a bridge portal belonging to the right-hand net. This condition is shown by the middle portion of Figure 4; a bus reset has occurred but the contradictory presence of more than one prime portal and duplicate bus IDs has not yet been resolved.

The algorithm described in greater detail below is responsible for the logical transition to the final state shown, in which there is once again only one prime portal and unique enumeration of bus IDs for the whole net. Before exploring the procedures, some new terminology may be helpful.

clan: A group of bridge portals that share the same prime portal. In a configured and stable net there is, of course, only one clan. When nets are joined, more than one clan may exist during the time it takes to resolve the identity of prime portal and reprogram routing tables.

dominant portal: A portal that is either an alpha or prime portal. Within a clan there is exactly one dominant portal on each bus that is part of the net. Portals that are neither alpha nor prime are called **subordinate portals**.

survivor clan: When two nets are joined, the survivor clan is least affected by the event; it retains the same enumerated bus IDs and may quickly resume asynchronous routing operations. This document proposes that the survivor clan be determined by a count of the number of buses within the net— with the exception that degenerate buses that consist of solely a bridge portal are not counted at all.

victim clan: The clan that may lose bus ID assignments and suffer temporary inability to route asynchronous subactions to some member buses when two nets are joined. Full operations are restored to former members of the victim clan as their routing tables are updated and they become members of the enlarged survivor clan.

The algorithm is set in motion by the bus reset that follows the insertion of the turtle clan bridge portal into a lizard clan bus. Upon detection of bus reset, all connected bridge portals stop forwarding asynchronous packets across the affected bus. A portal may transmit asynchronous packets whose bus ID is $3FF_{16}$ onto the just reset bus, since the portal has maintained an accurate map of virtual to physical IDs across the reset. Packets destined for other buses (*i.e.*, those whose bus ID is other than $3FF_{16}$) shall not be transmitted since more than one bridge portal might attempt to forward them. This cooperative behavior makes it unnecessary for bridge portals to interdict packets outbound from the bus, since none can appear at this point in time.

Dependent upon the MAX_RQ_FORWARD_TIME and MAX_RESP_FORWARD_TIME limits, an affected bridge portal may eventually be able to forward all of its pending subactions— unless the reconfiguration process described below consumes too much time.

As already described, the bridge portals observe the self-ID packets in order to determine the physical IDs of other bridge portals. As soon as asynchronous arbitration is permitted, the dominant bridge portals interrogate each other to determine the identify of the survivor clan.² Subordinate portals do not participate in this part of the algorithm. The survivor clan is chosen according to the number of buses contained within its net; this information is available in all bridge portals. If one or more clans contain the same number of buses, the EUI-64 of their prime portal is used as a tie-breaker. In order to provide a some randomization, the significance of the bits in the EUI-64 is reversed; the one with the largest magnitude wins and determines the identity of the survivor clan. In the example above, the lizard clan has two buses while the turtle clan has one; the lizard clan prime portal becomes the dominant portal responsible for the remainder of the algorithm.

In the preceding step, routing tables were collected from all dominant portals. This information is used subsequently to eliminate duplicate bus IDs from the routing tables of victim clan portals. At present, however, there are more pressing tasks for the survivor clan dominant portal: victim notification, a prerequisite for the safe resumption of asynchronous operations by the survivor clan portals.

Before any asynchronous packet destined for another bus may be transmitted on the just reset bus, the portal must be certain that at most one other portal on the bus will recognize the destination bus ID and transmit an *ack_pending* in response. The survivor clan dominant portal shall transmit a **victim notification** to all connected bridge portals that are not members of the survivor clan. Broadcast write requests shall not be used; it is essential to obtain positive confirmation that each victim portal has received the notification. The receipt of victim clan notification shall cause a portal to ignore all packets whose bus ID is not $3FF_{16}$.

Once all victim clan portals have been notified they no longer eavesdrop for packets to be transmitted off the bus: it is now safe for the survivor clan portals to resume all asynchronous routing operations. The survivor clan dominant portal shall transmit a **resume notification** to all connected bridge portals that are members of the survivor clan. Any asynchronous packets that have been held in suspense in survivor bridge portal queues may be transmitted, so long as the maximum forwarding time limit has not been exceeded.

At this point in time, operations in the lizard clan net are back to normal. In addition, the newly inserted turtle clan bridge portal has been assigned a virtual ID and is accessible to remote transaction requests from anywhere within the lizard clan buses. The situation is not as rosy for the turtle clan: all routing operations are suspended for turtle clan portals connected to the just reset bus.³ Clearly one could enumerate all of the buses in the turtle clan and assign unique bus IDs, but perhaps there is a faster method to resume partial operations.

² When two nets are joined, by definition there are at least two dominant portals.

³ If a larger turtle clan were illustrated in the example, portions of the turtle clan net would still be operational but the clan might be divided if the reset bus interdicted some of the possible routes.

Imagine a virtual net composed of the bus IDs in the turtle clan that are not duplicated in the lizard clan. There would be no confusion over routing if it were possible to safely integrate this virtual net into the lizard clan. In fact, the method might have considerable utility if a transient disconnection severed a configured net into two nets with no overlapping bus IDs. When the connection is restored there should be no necessity for bus ID reassignment. However, the method has to be robust and not susceptible to erroneous results if additional bus resets occur anywhere within the conjoined net before the merge is complete.

After all resume notifications have been transmitted to survivor clan portals, the survivor clan dominant portal communicates a **join request** to each victim clan portal. The join request conveys a number of parameters:

- the virtual node ID and EUI-64 of the prime portal;
- a list (likely represented as a bit map) of all the bus IDs that belonging to the survivor clan; and
- the bus ID of the bus to which the victim clan portal is connected.

Upon receipt of a join request, the victim clan portal shall initiates the following actions, to be performed by both portals of the victim clan bridge:

- a) The portal that received the join request shall eliminate from its own routing tables all duplicate bus IDs. This will prevent more than one bridge portal from acknowledging a packet intended for a remote bus. It shall also save the virtual node ID and EUI-64 of its new prime portal— but not yet make them visible to an outside observer;
- b) The victim portal on the just reset bus shall then convey the join request to the bridge's opposite portal and await a completion response;
- c) The opposite portal shall eliminate from its own routing tables all bus IDs duplicated in the survivor clan. If the bus ID of its connected bus is eliminated, it shall be marked "unassigned";
- d) The opposite portal shall communicate, by means of confirmed write transactions, the join request to all other portals on its bus. The local bus ID in the original join request shall be replaced with the bus ID of the opposite portal's connected bus, whether or not that bus ID is still assigned. Recipients of this modified join request shall individually execute this algorithm from its beginning. The effect of the join request is recursively propagated throughout the victim clan net;
- e) Once all the other bridge portals have acknowledged receipt of the join request, the opposite portal shall add all survivor clan bus IDs to its own routing table (since all of these survivor clan bus IDs are now accessible via this portal) and mark itself an alpha portal. At this point, the portal may accept packets destined for remote buses according to its routing tables. Last, the new alpha portal communicates a join response to the bridge's opposite portal;
- f) Upon receipt of a join response, the portal that received the join request shall now mark itself a subordinate portal, update the virtual node ID and EUI-64 of its new prime portal (switch its clan allegiance) and update the bus ID of its connected bus. At this time, all asynchronous routing operations may be resumed by the former victim portal; and

- g) The last action taken by the former victim portal is to transmit a join response to the dominant portal from which the join request was received. Most of the parameters of the join response are the same as received in the join request and they serve as confirmation that the bridge portal has changed its allegiance to the survivor clan. However, instead of a list of bus IDs that belong to the survivor clan, the former victim portal sends a copy of its own routing tables. The survivor clan dominant portal shall use this information to update its routing tables to reflect the addition of new bus IDs from the victim clan and shall then propagate this information to all other survivor clan portals throughout the net so they may do the same.

Once all of these operations (and their recursive offspring) have completed, the salvageable, unduplicated portion of the victim clan has been integrated into the survivor clan. The hope, not yet proven by any simulation efforts, is that this much can be accomplished within the maximum forward time limits for bridges. Whatever the actual time required, there now exist within the new, conjoined net zero or more buses with no assigned bus ID. For the sake of simplicity, the prime portal is the singular source for new bus ID assignment; this avoids the problems of synchronized, distributed databases of valid bus IDs. The process by which bus IDs are assigned may proceed in parallel with normal operation of the resumed net.

The alpha portal connected to a bus with no assigned bus ID is responsible to obtain a bus number from the prime portal. At the point where an alpha portal has communicated a join response to its opposite portal, a path exists from the alpha portal to the prime portal. The alpha portal shall transmit a **bus ID request** to the prime portal and await a response. When a bus ID is received from the prime portal, the alpha portal shall use join requests and responses to update all the routing tables in the net. Specifically:

- a) The alpha portal transmits a join request to all the subordinate portals on the connected bus. The list of bus IDs contains only the newly assigned bus number. Once receipt of the join request(s) is acknowledged, the alpha portal shall transmit a join response to the bridge's opposite portal (the join response(s) from the subordinate portals are unimportant and may be discarded when they arrive);
- b) The subordinate portals that receive the join request perform all of the operations previously described for the elimination of duplicate bus IDs;
- c) The opposite portal (a subordinate portal on its own bus) shall add the bus ID identified in the join response to its own routing tables and communicate the join response to the dominant portal on the bus to which it is attached; and
- d) Just as already described in the elimination of duplicate IDs, a dominant portal that receives a join response shall communicate the information to all bridge portals that belong to its clan (except the portal from which it received the join response).

There is no notification to the prime portal that the bus ID granted in response to a bus ID request has been successfully utilized. I assume that the prime portal undertakes periodic background garbage collection to sweep the net and verify that all the bus IDs it has granted are, in fact, in use.

There are, admittedly, many details yet to be fully described, but the fundamental operations are here. If the working group agrees that this concept is viable, I and the other contributors will continue to refine this proposal.