# Extended Address Event Representation Draft Standard v0.4

July 18, 2002

**Abstract**

Address Event Representation (AER) is a representation for the communication of asynchronous events between a collection of units doing computation in parallel. Although a large part of the intent of the AER standard is to allow communication between neuromorphic circuitry, we make no commitment to the details of the units involved: they may be neuromorphic integrated circuits, digital circuits, computer software emulations of neural systems, biological neurons, or standard computer programs. We shall generally use neuromorphic circuitry as an example medium when discussing issues in this document. That is not to be taken as prescriptive (or proscriptive).

## 1 AER Systems

An AER system comprises a collection of *units* which generate and consume *events*. Units in AER systems are typically clustered (for example, a collection of neuromorphic units on a single silicon substrate, or a collection of units in a computer simulation program). Each individual collection of units forms a *node* which communicates with other nodes (and hence units) via the protocol described here. Hence individual units inside a node are not constrained to use this protocol, but if they do not, then some translation circuitry is required to communicate off the node.

Each unit in a node has a locally unique address, and each node also has an address in its local *network* which is unique either relatively in some topological organisation (*i.e.* directions for how to move to get an event to the node) or absolutely in a more abstract organisation - to extend beyond this, each local network itself must also have a unique address which is only required when communicating from outside the network; this form of hierarchical organisation can be repeated. To send data to a target, a node transmits a message packet which consists of the address of its target or its source (*i.e.* the node address, and the local unit address if appropriate), and any associated information. In the default case there is no associated information and the message signals the occurrence of an event (*e.g.* a spike). The time at which the address is transmitted, together with the message including the address, completely determines the signal. The generation and transmission of a signal constitutes an *address event*.

A single address event in an AER system is ordinarily targeted at a single node or unit, and cannot be directly targeted at multiple units or nodes. To target multiple receivers with an event either multiple events must be sent, each with different targets, or a node must be targeted which can then retransmit multiple events to a local alias list, which would consist of a sequence of targets to which to relay an address event; these are called *target address events*. However, if a transmitter does not know (or wish to know) the target(s) to which the address event is to be delivered (for instance no older AER nodes have lookup tables which allow them to target address events) then it may also broadcast its own address on its local bus or link; such events are called *source address events*.

When an event is delivered to a receiver it may have various meanings depending on the structure of the message packet. For instance, an event might:

- provide synaptic input of a certain type to a neuromorphic neuron
- target an alias which signifies a broadcast to a particular set of receivers
- cause changes to an alias
- initiate changes in event reception (*e.g.* synaptic) parameters
- interrogate a node to find out its capabilities
- tell a node about the transmitter's capabilities
- be processed by a graphical user interface for display of system activity
- *etc.*

Most units and nodes are therefore both generators and consumers of events; some also relay events on to other nodes (*e.g.* routers). In some networks it may be possible to route events without a dedicated router (for instance where only one node exists which connects only to itself, between topologically organised nodes which use relative addressing or in a network where all nodes communicate on a common bus), however in general routers will be necessary to connect nodes to each other.

## 1.1 Definitions

**An AER system** is a collection of event generators and event consumers organised into nodes which communicate using the AER protocol.

**A node** is the element of address event transmission and reception in an AER system. It typically contains units which actually generate and consume the events (except in the case of a pure router) and it is typically the smallest element which uses the AER protocol itself.

**A network** is a collection of nodes connected together to allow them to communicate. If there are many nodes there will often be routers in the network designed expressly to carry messages between nodes efficiently.

**A unit** is the element of event generation and consumption in an AER system. It must form part of a node, through which it will transmit and receive its events in the form of address events.

**A target address event** is an event which contains at least an explicit description of the unit (or node) at which it is targeted. To deliver an address event of this type either the target must be on the transmitter's local link or bus, or there must be routers in the system to deliver the address event to the target's local link or bus.

**A source address event** is an event which is broadcast on a local bus or link by a node and which contains typically only the address of the source of the event. For an address event of this type to be delivered, either all potential recipients must be on the local bus or link, or there must be routers in the system capable of converting the source address event into one or more target address events which will then be passed to their destinations as above.

**An extended address event** is a target address event with a payload which contains additional information describing the event being transmitted.

See Section 3.2 for more details of the packet structure.

# 2 The Address Space

Event generators have unique addresses, taken from some address space. Local clusters in an AER system may contain spatial or topological structure, for example 2D arrays of units, and the clusters themselves may be spatially organised. The within-node and node address assignments may be made to reflect this spatial or topological structure either explicitly or by relative addressing.

Consider a local cluster of event generators forming a node - a neuromorphic chip, for example (or a computer program). Each event generator or consumer in the node will have a local address. If there are many such nodes, a unique address for each event generator or consumer can be created by combining a unique within-node address with a unique per-node address. This per-node address may include an address for the network it is on in the event of there being very large numbers of interconnected nodes. Such an address for a unit will be called the unit address. The address of the node itself will be called the node address. Note that the length of this node address will vary depending on the network configuration, and so it should be delineated by segmentation of the packet.

Frequently it will be the case that a consumer's unit address is also a event generator's unit address, for instance the unit address of a neuron's spike generation system may be the same as (or a subset of) that of its synaptic input system.

These observations determine the structure of the event address space.

## 2.1 Definitions

**A local node address** is a locally unique (either absolute or relative as discussed above) within-network address.

**A node address** is the concatenation of the local node address and a unique network address. From within the same network, the node address can be abbreviated to just the local node address.

**A local unit address** is the locally unique within-node address for a unit.

**A unit address** has two segments and concatenates the node address and the local unit address.

# 3 Extended Address Events

The default message type signifies the occurence of an event usually signifying a neuronal spike. However there may also be different classes of event generators within a given system. For instance, neuromorphic neurons and synapses may each be able to consume events which represent an instruction to change a specific parameter in the unit. The event type can then be signified by an extended command set communicated with the target unit or node address.

## 3.1 Extended Address Event Types

The various different types of extended address events are distinguished according to their meaning to the consumer. All such events are sent to a single target. At least the following types will be needed:

- delivery to an alias for retransmission / reporting spike production
- setting an alias parameter (*e.g.* an entry in a lookup table)
- setting a node or unit parameter (*e.g.* synaptic weight)

### 3.1.1 Definition

**An alias** is a consumer in a node which does not represent a real unit, but rather a list of other targets. This allows one address event to be targeted at a single alias which will then fanout to multiple consumers within a node (or a local network) to save global bandwidth.

## 3.2 The Message Packet

The address event itself is sent as a message packet containing two parts, a target address segment and a payload segment, divided as seen in Figure 1.

The actual allocation of space within the address segment depends on the local and global topologies. With a regular topological structure (*e.g.* a grid) it may consist of a relative address which simply gives directions to get to the target node and unit. In a star topology with a single router it might just be the target local node or unit address. In a more arbitrary topology any of a number of representations might be used to aid routing (*e.g.* a list of nodes to pass through for wormhole routing).

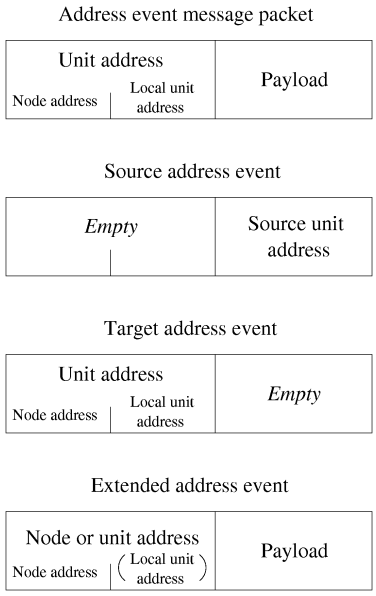


Figure 1: Address event packet layout

For the payload segment, in the case of a source address event, the segment can just contain the unit address of the generator; for an extended address event, however, the contents of the payload will of necessity vary depending on the nature of the event. The first part of the payload should describe the type of event being transmitted, followed by any additional information needed (*e.g.* a weight). Note that it should not be necessary to be able to parse this segment to relay it on to its target.

### 3.2.1 Definition

**The message packet** consists of a target node or unit address segment and a payload segment. By default both segments contain data, and the message represents an extended address event. If the target address segment is empty, then the payload segment must contain the source address and the event is a source address event. If the payload segment is empty then message packet represents a target address event. The layout of the packet can be seen more clearly in Figure 1.

# 4 Routing and communication protocols

No details on routing are included in this draft as they are too varied and still in a state of flux. Likewise essential implementation details such as low level communication protocols are too diverse to be standardised upon. However, an appendix should be added to the document giving reference examples.

# 5 Existing AER Implementations

There are currently implementations of two subsets of this protocol: the first, which contains only the payload segment in the description, consists only of source address events on a single local bus; the second, which contains only the target address segment consists of networks of nodes which can communicate only default events to one another. Hybrids and extensions of the two also exist.

# Appendices

## A Serial AER

### A.1 Wire protocol

The serial protocol is point-to-point and based on 6 symbols: $\{0, 1, Fs, Fe, Fd, x0\}$, where $d = \{0, 1\}$ are used to represent the actual data $F = \{Fs, Fd, Fe\}$ are used as syncronization and framing elements, while x0 is used as field separation elements. See section 2 for a physical description of these symbols.

To describe the protocol the following additional symbols will be used: $*$ indicates zero or more repetitions of the preceding elements, $+$indicates one or more repetitions, $\|$ indicates either the element on the right or the element on the left, parenthesis are used to group elements together.

### A.2 Data Stream

A data stream consists of a sequence of symbols $((Fs\|Fd)d + ((Fs\|Fd)\ d + \ ) * (x0\ d + \ ) * Fe)$ that is, every sequence is framed starting with an Fs or Fd and ending on an Fe symbol, and can have internal sub-frames specified by $\{Fs, Fd, x0\}$ symbols. Such choice for framing all data elements simplifies sincronization and state keeping at a small bandwith cost. Different data or payload elements are framed by x0 symbols. The data fields can be as large as needed for the individual data type, using powers of 2 for such sizing is not required (and not desired) to reduce implementation cost, however such choice could simplify communications with digital systems.

The minimum data stream, a simple AE will be $(Fs\ d + \ Fe)$ or $(Fd\ d + \ Fd\ d + \ Fe)$. A 2-component stream could be $(Fs\ d + \ x0\ d + \ Fe)$. That is $x0$ will serve as a data separation element. In general, all explicit addressing would be placed at the begining of the packet, and the highest level address will be the first address in the packet, such choice simplifies routing decisions and allows for wormhole-like routing.

### A.3 Packet types

The different packet types allowed are:

**Basic destination address event** $(Fd\ d + \ Fd\ d + \ (x0\ d + \ ) * Fe)$, allows for a clean hierarchical delimitation of addresses, and the simplification of routing elements, and routing tables. This scheme can be extended for multiple hierarchies of addresses.

**Basic source address event** $(Fs\ d + \ Fe)$, additional bits might be added at the begining of the address field, or removed from it by the individual routers.

**Extended address event** $(([Fs\|Fd]\ d + \ ) + x0\ d + \ (x0\ d + \ ) * Fe)$, for future compatibility and expandability, the first data field should contain some indication of the type or purpose of the packet, and later data fields, if present, the actual data being transfered.

An example of the flexibility of such framing scheme would be a destination/source event: $(Fd\,d+\ Fs\,d+\ (x0\,d+\ )*Fe)$, which would allow for simplified tunneling of source address events, with no particular restriction on the intervening routers.

A router element has to be able to make decisions based on the first address field, and after such decision has been made it might choose to transmit, consume, or discard the rest of the packet until the "frame end" ($Fe$) element has been processed. Such implementation would allow for 'dumb routers' to be forward compatible, and transparent, to any future expansion of the protocol. All packets types need not be supported at every level of the hierarchy, however, the behavior of the system should be defined for all unsupported packets, being a discard of the packet the preferable behavior.

## A.4 Addressing modes

**Binary tree addressing** is the prefered addressing mode for a serial implementation, each bit (or subsets of the bit field) in the address stream can be added/stripped by the individual routers. Such scheme is the most bandwidth/resource efficient for a bit-serial scheme, as routing decisions can be made based on each one of the arriving bits, thus requiring minimal packet buffering and delay.

**Relative addressing** is supported by modifying (incrementing or decrementing) the address section of the data stream. This requires the receiving unit to process the whole address before making a decision on its destination. To simplify implementation, the address should be sent with the least significant bit first.

**Direct addressing** — the address stream is passed unchanged, and each router has a comparator for its own address range. As before, the receiving unit has to process the whole address before making a decision.

Multiple addressing modes could be present in the same packet, and will remain compatible, as long as the addressing fields are logically kept separate from each other (i.e. $(Fd\,d+\ Fd\,d+\ Fd\,d+\ Fe)$). That is, a high level router only needs to be aware of its own addressing scheme.

## A.5 Physical implementations

### A.5.1 Communication channel

A channel is constructed with 3 signal lines, two for the actual data communication from the source (0,1), one for the acknowlege signal from the destination (ack). The actual signal levels and polarities would be dependant on the particular implementation, but would still remain compatible through the use of simple off-the shelf buffers and drivers.

A bi-directional communication link can be implemented with two independent uni-directional links. No provision has been made for bussed type implementation, as this would require almost as many wires as two independent links.

### A.5.2 Symbols

Figure 3 illustrates the different symbols to be implemented on the serial line assuming positive polarity logic, the meaning of the symbols is somewhat arbitrary, but must remain the same to allow for the interfacing of different systems without extra logic. These symbols can be translated to any line level implementation like 3-wire voltage-mode, 6-wire voltage-mode differential, or 6-wire current-mode differential.

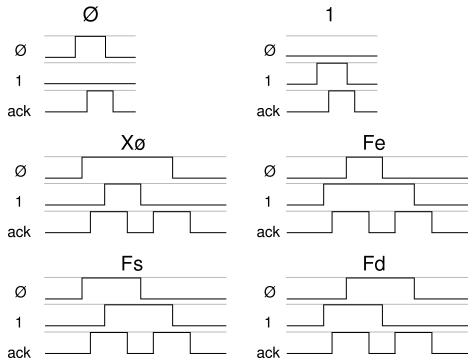


Figure 2: Symbols in serial AER

# B Word Serial AER

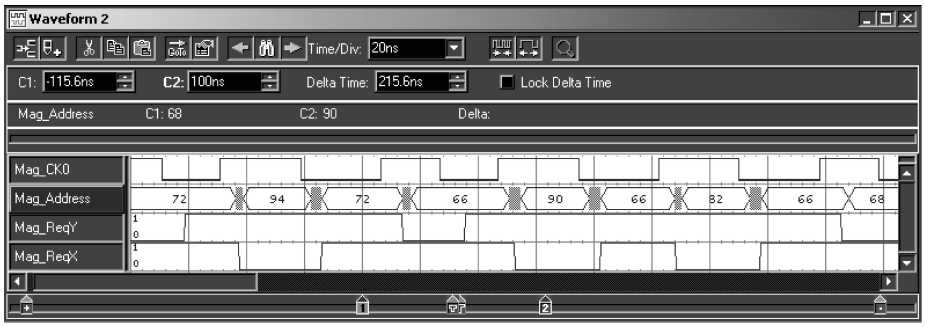


Figure 3: Self-Timed Word-Serial Data Transmission

*Cycle*: Transmitter asserts Mag_ReqY (active-high Y-Request) and outputs Mag_Address (row-address 72), which Receiver latches before it asserts Mag_CK0 (Acknowledge). Transmitter then asserts Mag_ReqX (active-low X-Request) and outputs Mag_Address (column-address 94), which Receiver latches before it takes Mag_CK0 low. At this point the transaction is complete, and the initial state is restored by taking Mag_ReqX high, followed by Mag_CK0, and then taking Mag_ReqY low, followed again by Mag_CK0. *Burst*: A single row address (66) and two column addresses (90 and 82) are transmitted; this burst is terminated the same way a single cycle is.