

Serial Implementation of IEEE Standard 1451.2-1997 for Data Acquisition Applications

Communicating with IEEE 1451.2 STIMs Over an RS-485 Serial Link Facilitated Rapid Development of Customized Data Acquisition Systems

Robert N. Johnson
Telemonitor, Inc., Columbia, Maryland

1. INTRODUCTION

One of the most requested enhancements to the IEEE Standard 1451.2-1997 is a simpler interface than the present 10-pin Transducer Independent Interface (TII). Packaging the IEEE 1451.2 transactions in simple message packets and transmitting them over a serial interface has been suggested. One possible approach to this concept has been used for specialized data acquisition projects involving networks of smart sensors.

2. BACKGROUND

Several years ago we were requested to provide a network of smart sensors that could stream several channels of data from each of several observation points. Each point was to report tilt in two axes, strain in two directions, and temperature. There were to be up to 32 such observation points spread out over a distance of up to 2000 feet. Oh yes, and everything had to fit inside a pipe with an inside diameter of one inch.

We had experience with all of the sensors of interest using the IEEE 1451.2 STIM interfaces that had been included in the STIM Developer's Kit that was used in the early IEEE 1451.2 demonstrations. We repackaged those into a form factor that was appropriate for this application and developed modified software to capture the streaming data in a format convenient for the customer's data acquisition system.

This approach worked extremely well and was used on other custom data acquisition projects involving

networks of smart sensors. The recent interest in enhancing IEEE 1451.2, particularly the interest in developing an asynchronous serial interface, has revived interest in our previous experience. While our specific RS-485 network may not be appropriate for the serial version of IEEE 1451.2, the decisions that we made and our experience with the result should be of interest.

3. PHYSICAL LAYER

We selected RS-485 as the physical layer for several reasons: it is simple, there are inexpensive driver circuits readily available, it is multi-drop, and it can be used over the distance of interest. For our application we used a half-duplex version of RS-485 that requires only two lines for data. The tradeoff is that there can only be one transmitter at a time on the bus, so the hardware in each device connected to the bus has to be able to switch quickly between transmit and receive.

For development we used an RS-232C to RS-485 converter on the serial port of a PC. This was able to operate reliably at up to the maximum operating speed of the PC serial port, which is 115,200 baud. This was not fast enough for the final application however, so we used a PCI bus RS-485 adapter that operated at approximately 600k baud.

The cable included a total of five wires: power and ground, the two RS-485 data lines, and a hardware trigger to synchronize the data acquisition of all units. The protocol included a software trigger, so the hardware trigger was not always used.

4. DATA PACKETS

The nodes communicate using a variable length packet, consisting of the fields shown in Table 1. The maximum size of the packet, including the 2 header bytes, is 34 bytes. The minimum transmission is 6 bytes. There must always be at least 1 byte of data transmitted with each packet.

The header bytes are used to signal the start of a new message. It is required that each node sends 0x00 after transmission of 0xAA outside of the header (known as byte stuffing). Receiving nodes

must remove the 0x00 from the data. The extra 0x00 does not count as part of the data length or in the overall length of the transmitted packet. A 0xAA byte received outside of the header and not followed by 0x00 is invalid and the message will be discarded. In addition, at any time during the current message a 0xAA byte followed by 0x55 will signal the start of a new message. Each packet has a MOD-256 checksum that is calculated on packet fields 3-5. When a node receives a message with an invalid checksum, the message is discarded.

Table 1. Packet format.

Field #	Description	Value	Size (bytes)
1	Header	0xAA	1
2	Header	0x55	1
3	Node Identifier	0 - 255	1
4	Length of Data	1 - 29	1
5	Data		1-29
6	Checksum	0 - 255	1

5. NETWORKING

There is always a master node in the network. In addition, there must be at least one slave node. The master node does not have a node identifier and transmits control messages to the slave nodes. Each slave node responds to the control message by transmitting a response message to the master node. Only one slave node may transmit on the network at a time. However, the nodes timeshare the network in order to improve data throughput.

Each node has a time-slot that is based on its node address. The time-slots are not a fixed length, and shift as other nodes in the network transmit shorter or longer messages. Each node has a timer that is reset every time a data byte is received by a node. When the timer reaches zero, a counter is incremented. When the counter is equal to the address of the node, the node can transmit. When the node is done transmitting, or if the node does not have any data to transmit, then the counter also is incremented.

When the counter reaches the number of nodes in the network, it is reset to zero. In addition, whenever a node receives a valid message, the node counter is

set to the identifier of the sending node. This keeps all the nodes synchronized.

The time-slots do not guarantee any order for the transmission of messages on the network. Depending on how long a node takes to complete an operation it may lose its time-slot to another node putting the responses out of order in relation to the node addresses. That is, just because there are three nodes on the network does not mean that a response message will always be ordered node 1, node 2, node 3, etc. Depending on which node gets the network first, the other nodes will wait until that node is finished and then wait for their time-slot before transmitting. Each node has a fixed time-slot but may give up its slot if it has nothing to transmit. This allows a node with a higher address to start transmitting first leading to the out-of-order responses.

There is a fixed delay, called the site delay, between any two consecutive nodes transmitting. The site delay values are listed in table 2.

For maximum network data throughput, the nodes are assigned consecutive logical addresses. This avoids nodes waiting for unused time-slots. Also, the

master node sets the largest node address in the network since the default setting is 255.

5.1.1 Table 4 - Site Delay

Baud Rate	Site Delay
9600	1 ms
19200	600 us
38400	400 us
115200 and higher	200 us

6. NODE DISCOVERY

Software node identification allows any number of nodes, up to 255, to be attached to the network at any time. As long as the network is polled each time a node (or nodes) is added, the master node can assign consecutive logical addresses to each slave node in the network.

The basis for the software node identification is the bit-dominance protocol. The bit-dominance protocol relies on each node being assigned a unique identifier (UID). The UID is checked 1 bit at a time by every node in the network. A node with a '1' bit in the current bit location transmits during that bit time, a node with a '0' bit listens for other nodes to transmit. If a node "hears" another node transmitting, that node (the listening node) drops out of the current cycle. When all of the bits in the UID have been checked, there will be only 1 node remaining in the network (the isolated node). The isolated node is assigned an address and the cycle is repeated until all the nodes are identified.

The STIM interface nodes use a 32 bit UID, and the bits of the UID are checked MSB to LSB. The UID will never be zero. When a node has a '1' bit in the current location of its UID, the node transmits a break character (a logic low on the bus for 1 byte time). Any node that receives a break character drops out of the identification cycle. After 32 bits have been checked, the node will wait to be assigned an address if it determines that it is the last active node, or wait for the next identification cycle. Once a node is assigned an address, it will not participate in any other identification cycles. To repeat the identification process, the node must be reset (it is recommended that all nodes be reset prior to the node identification process).

The master node is required to initiate the identification process and assign addresses to the nodes. An identification cycle begins with the start ID command, and is followed by 32 check next UID bit commands. The master node will either receive a break character indicating that the current bit of the UID is 1, or receive nothing indicating that the current bit of the UID is 0. To ensure that all the nodes in the network have responded to the current cycle, the master node must wait at least the node site delay before examining the state of the current UID bit (refer to table 2). If the master node completes an identification cycle and the UID is 0x00000000 then either there are no nodes in the network (if the number of addresses assigned is zero) or all the nodes in the network are identified and the process is complete. Refer to figure 1 for a flowchart of the discovery and identification process.

To assign a node address, the UID the master node broadcast using the set node address command must match the UID of the isolated node. If the UID numbers do not match, then no acknowledge response will be sent to the master and the node will wait for the next identification cycle. The master node must repeat the cycle using the current node address (since it is unassigned) until the node address set command is acknowledged.

7. MESSAGES

A node will only accept messages addressed to it, unless the node identifier is set to 0. When the node identifier is set to 0, all the nodes in the network will perform the requested action. Usually, a node will not respond to a global message but there are two exceptions: the software global trigger command and the node identification commands.

There are two types of messages, control messages and response messages. A control message is sent from the master node to a slave node to initiate an action. A response message is sent from a slave node to the master node and reports the results of the requested function. A control message consists of a function code and its associated parameters. Likewise, a response message consists of a response code and its associated parameters.

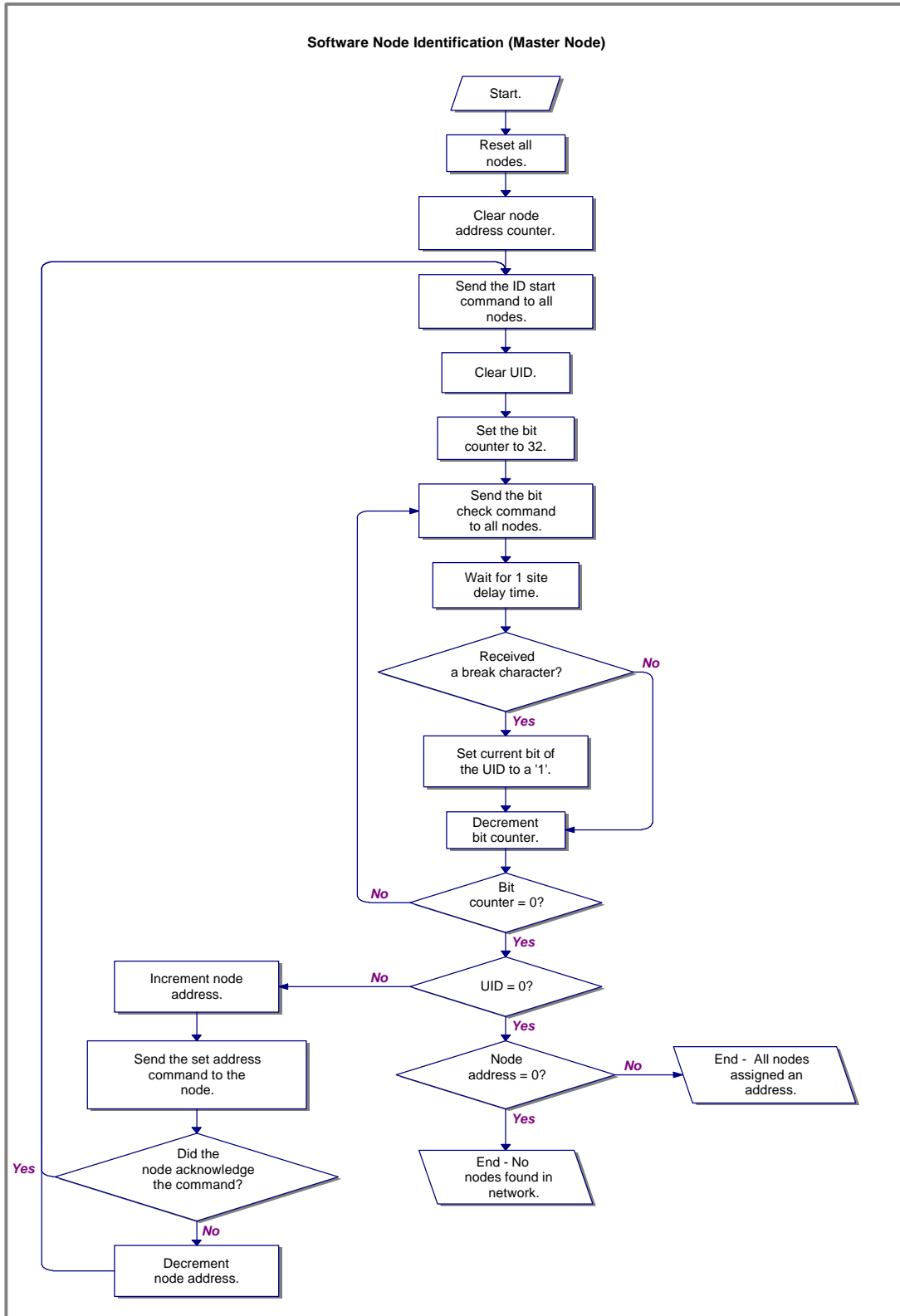


Figure 1. Discovery process flowchart.

In general, the master node should transmit the control message and then wait for the response message before issuing another control message. However, it is possible to request the system status, reset the system, abort a function, or power on/off the STIM module without waiting for a response from any of the STIM I/O functions. This is useful to determine why a particular function did not complete or to reset the system to a known state.

A complete description of all the messages is beyond the scope of this paper. However, the messages include all IEEE 145.2 transactions and hardware control functions such as triggering. Most of the IEEE 1451.2 transactions can be handled simply by including the function address, channel, and data in the data field of the message packet.

8. SOFTWARE TRIGGER

If the local trigger function is sent as a broadcast message (node address 0), then all of the nodes in the network will respond with data from the triggered channel (it doesn't have to be the same channel depending on how the nodes are configured).

More than one node may respond with data and the order of the responses is not guaranteed. Depending on the delay required for the each STIM module to complete the trigger and data transfer, the responses may arrive at the master node in any order (for example, node 3, node 1, node 2, in a 3-node system).

9. HARDWARE TRIGGER

The global trigger signal is driven directly by the PC master node and therefore all of the nodes in the

network will trigger at the same time (or as close as possible to the same time). Only nodes that have the global trigger signal enabled will trigger. Enabling the global trigger does not affect the local trigger. As with software global triggering, the exact order of the responses is not guaranteed. Depending on the delay required for the each STIM module to complete the trigger and data transfer, the responses may arrive at the master node out of sequence. Also, the exact trigger time, as measured from the falling edge of NACK, is unknown.

10. SUMMARY

This written paper presents a summary of the serial implementation of IEEE 1451.2 that has been used successfully for multiple customized data acquisition projects. This protocol will be provided to the IEEE 1451.2 Working Group as background for their consideration of possible asynchronous serial protocols for implementation of an enhanced version of IEEE 1451.2.

Robert N. Johnson is the Chair of the IEEE P1451.0 Study Group and participates in several of the other IEEE P1451 Working Groups. He is president of Telemonitor, Inc., a company that was spun off from Electronics Development Corporation (EDC) to pursue smart sensor and remote monitoring applications. He can be reached by voice at 410-312-6621, or by e-mail at robertj@telemonitor.com