

# Building IEEE 1451.2 Smart Transducer Interface Modules (STIMs)

Ronald D. Smith  
Telemonitor, Inc., Columbia, Maryland

## 1. ABSTRACT

This paper describes the design considerations, implementation techniques, and common problems encountered when building IEEE 1451.2 Smart Transducer Interface Modules (STIMs).

## 2. INTRODUCTION

Implementing an interface standard can be a difficult task. The goal of this paper is to make that task easier by describing how to design and build a STIM. The first section will discuss the hardware design of a STIM, including the serial interface protocol and electrical timing requirements. The second section will discuss the software design of a STIM, including design goals for the communication subroutine and guidelines for the required functionality of an IEEE 1451.2 compliant STIM. It is assumed that the reader has at least a working knowledge of the concepts and features of IEEE 1451.2.

## 3. HARDWARE

The basic hardware requirements for a STIM are a microcontroller with a synchronous serial port, an analog to digital converter (ADC), and non-volatile memory to store the Transducer Electronic Data Sheet (TEDS) information. Depending on the transducer, additional circuitry may be required for filtering or matching the signal to the input range of the ADC. Since many microcontrollers have built-in ADCs and have on-chip, non-volatile data storage, it is

possible to implement a STIM as a single chip design.

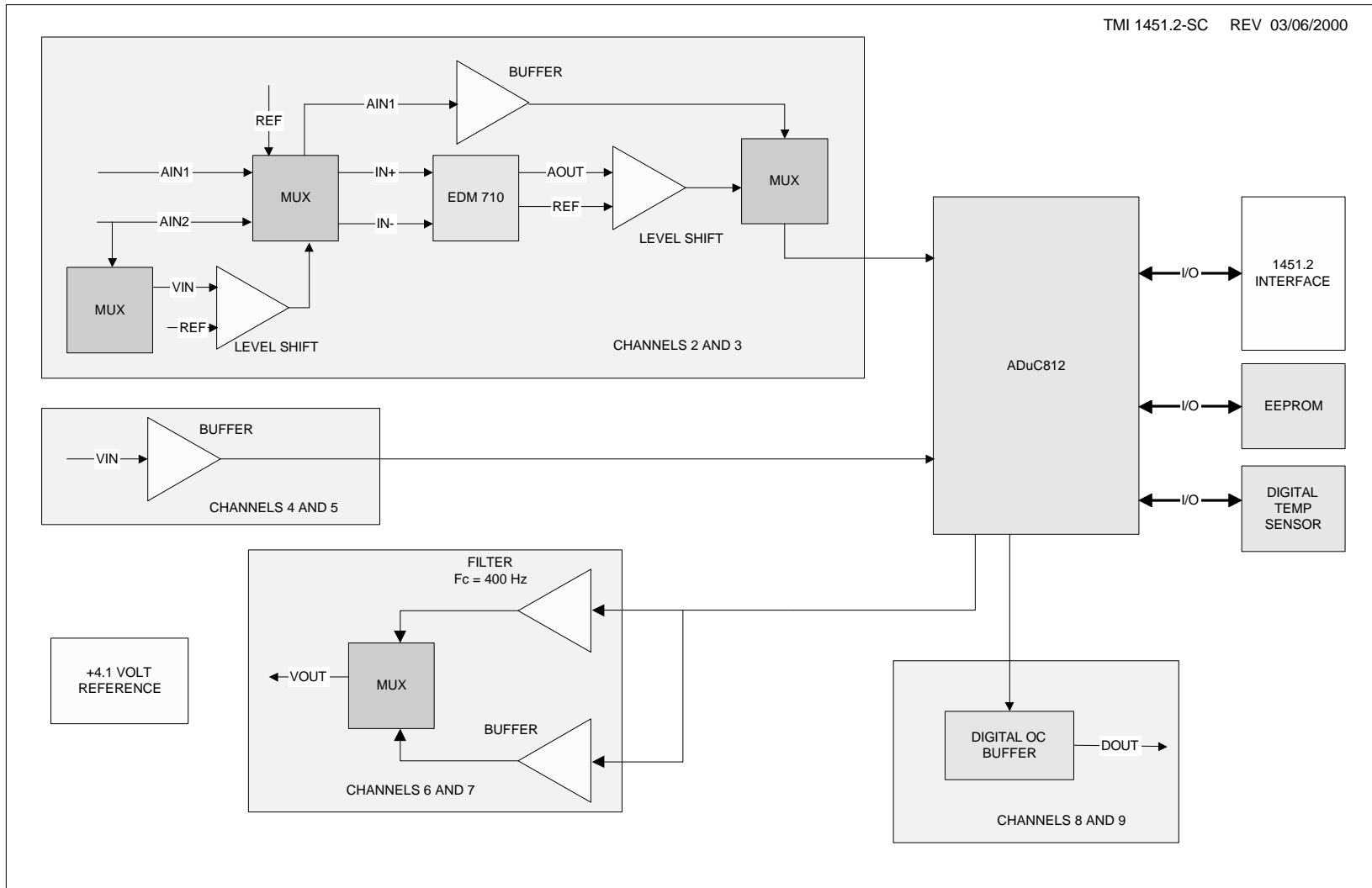
A block diagram of the TMI 1451.2-SC is shown in Figure 1. The TMI-1451.2-SC provides developers with a quick way to attach many resistive bridge or voltage type transducers to a STIM. It also has two analog output channels and two digital output channels. The Analog Devices ADuC812 Microconverter was chosen for this design since it incorporates both a 12-bit ADC, and a 12-bit DAC, as well as the synchronous serial port required to implement a STIM. In addition, the TMI 1451.2-SC provides full analog signal conditioning on two of its input channels.

Specifically, the EDM 710 sensor interface circuit designed by Electronics Development Corporation (Telemonitor's parent company) is used to provide a direct interface to resistive bridge sensors. It includes an anti-aliasing filter, gain amplifiers, and offset control, all programmable through a serial interface.

Of particular interest for this discussion is the addition of a separate voltage regulator for the analog power supply. This was added for two reasons: First, the linear regulator provides some isolation from the switching power-supply noise of the NCAP, and secondly it provides a fixed supply voltage for the analog interface circuitry. As described in section 6.5.1 of the IEEE 1451.2 standard, the STIM power supply can vary as much as .2 V from a nominal 5 V. This is not a problem for digital circuitry.

Figure 1 – TMI 1451.2-SC Block Diagram

TMI 1451.2-SC REV 03/06/2000



However, analog sensors typically have a sensitivity rating that varies depending on the supply voltage. So if a STIM is calibrated by a manufacturer using an NCAP having a 4.9V supply voltage, and then used by a customer with an NCAP having a 5.1V supply voltage, the calibration of the sensor may be affected. By providing a fixed analog supply voltage, the TMI-1451.2-SC avoids this problem and will maintain the correct calibration even when the NCAP power supply varies.

The Transducer Independent Interface (TII) is the IEEE 1451.2 specification for the communication link between a STIM and a NCAP. It is a synchronous serial protocol that includes handshaking and control signals. To implement the TII requires seven I/O lines on a microcontroller: three for the serial interface, three for the control signals, and one for the handshake signal. The TII is most easily implemented using a microcontroller with a built-in synchronous serial port, although the serial interface can be implemented in software. The IEEE 1451.2 standard does not require the serial data clock to have a constant frequency. This means that a bit bang serial interface is acceptable, and can be used with designs that can tolerate a low data transfer rate or that are extremely cost sensitive.

However, since many microcontrollers include a Serial Peripheral Interface (SPI) at little extra cost, the hardware serial port is the choice of most designers. Examples of available microcontrollers include the Microchip PIC16c6x series, the Motorola 6800 series, and the numerous 8051 derivatives. SPI is a synchronous serial protocol that is compatible with the TII serial protocol. However, there is one important difference: the TII is not designed to be multi-drop. The TII was designed to be a single master device (the NCAP) communicating

to a single slave device (the STIM). With the additional control signals specified by the TII, connecting multiple STIMs on the same bus requires more than just a slave select signal. A network of STIMs connected to a single NCAP is the focus of the proposed IEEE 1451.3 standard. However, at the 1999 ISA show, HP (now Agilent Technologies) demonstrated an IEEE 1451.2 NCAP that was capable of working with up to four separate STIMs. This was accomplished by using four independent TIIs.

The IEEE 1451.2 standard does not specify the type of connector used to implement the TII, but it does specify that the connector should be male for the STIM. The common connectors in use today are a 10-pin header and a 10-pin stacking connector. There are also some STIMs using a 15-pin high density DSUB, but those are limited to the STIM-in-a-connector series manufactured by Electronics Development Corporation. The TII signals are described in Table 1.

The TII requires that the STIM and the NCAP support powered insertion and extraction events, or hot-swaps. To support a hot-swap, the STIM must ensure that its input signals do not float at initialization. Typically, this means pull-up resistors on the DCLK, DIN, NIOE, and NTRIG signal lines. Likewise, the NCAP provides pull-ups on all of its input signals to ensure that the TII remains inactive at startup. In addition to the pull-up resistors, the signal and power lines should be protected from transients during connection. This is easily accomplished by the addition of a small capacitor to ground on each signal line and an inductor between the NCAP power output and the STIM power input. Short-circuit protection of the output signals is provided by a current-limit resistor in series with each of the output signals.

Table 1 – TII Pin Description

PIN NUMBER	SIGNAL NAME	DESCRIPTION	SIGNAL DIRECTION (STIM SIDE)
1	DCLK	Data clock	IN
2	DIN	Data input	IN
3	DOUT	Data output	OUT
4	NACK	Acknowledge	OUT
5	COMMON	Ground	IN
6	NIOE	Data transport active	IN
7	NINT	Interrupt	OUT
8	NTRIG	Trigger	OUT
9	POWER	+5V	IN
10	NSDET	STIM present	OUT

The NSDET signal is used by the NCAP to detect the presence of a STIM. It should be tied to the electrical ground of the STIM circuit board.

As the master device, the NCAP asserts data out on the falling edge of the clock, and latches data in on the rising edge of the clock. The STIM must also adhere to this timing. Keep in mind that although a full-duplex transfer is possible, the TII does not make use of this feature. Data is transferred in one direction only during any single operation. A typical read data transfer is shown in Figure 2. An explanation of the events follows the figure.

Notice in this transaction how the NACK signal changes state following each data transfer. This handshaking ensures that the STIM and the NCAP stay synchronized. Without it, the NCAP would not know when the STIM was ready and could easily begin the next byte transfer before the STIM processed the previous byte. Note that when an odd number of bytes is read, the state of NACK at the end of step 7 is negated so the NCAP does not wait for the STIM to end the transaction. However, the NCAP will not start another transaction before the End-Of-Frame Detection Latency time-out, as discussed

below. The operational hold-off for the TMI 1451.2-SC is about 500  $\mu$ s.

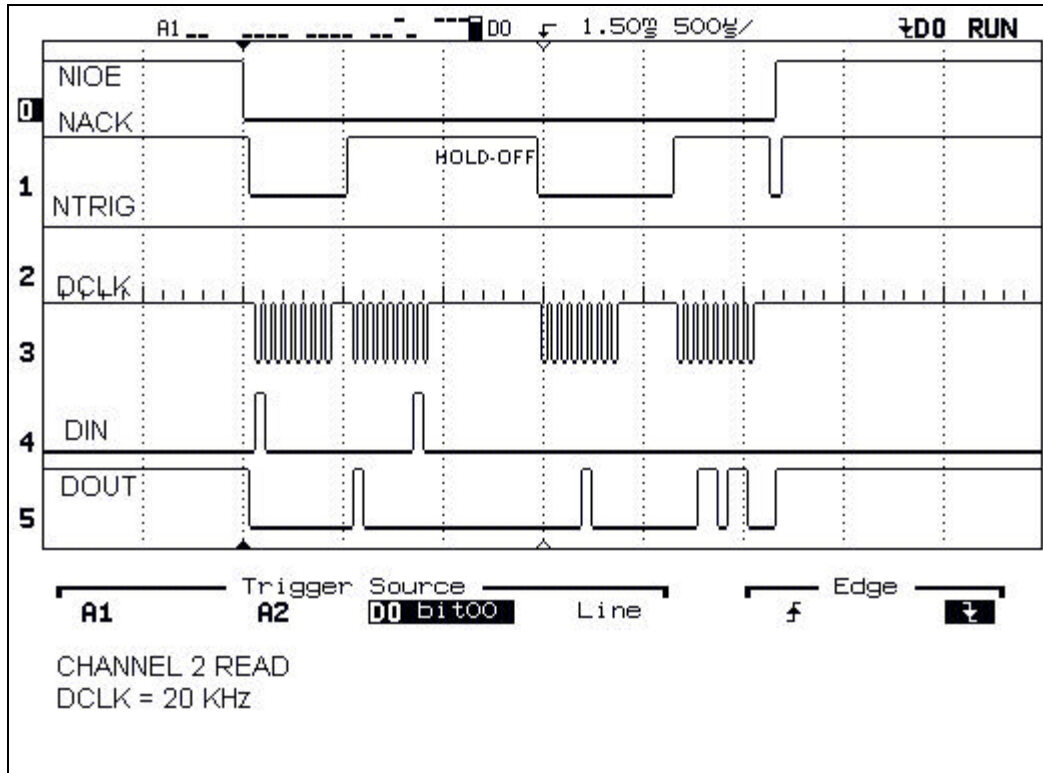
Of course, several things can go wrong during a transaction. What happens if the STIM never responds to the request for data? What happens if the NCAP reads or writes more data than the STIM is expecting? IEEE 1451.2 handles the first problem by setting timing limits in the TEDS for all data transfers between an NCAP and a STIM. The maximum time that the NCAP waits for the STIM to respond to a data transfer is called the Operational Hold-Off Time. This is the maximum time the NCAP will wait between bytes for the STIM to respond with the correct NACK signal (asserted or negated as required). The Operational Hold-Off Time applies to reads and writes to all function addresses except those dealing with transfer of TEDS information. Since the TEDS may be stored in slower, non-volatile memory, there is a separate hold-off time for those transfers. The timing parameters affecting data transfers are summarized in Table 2.

The second problem is handled mostly by software. The STIM must be able to read or write more (or less) data than it is expecting without crashing or hanging. In addition, the NCAP may remove the NIOE signal at any time

during the transfer. The STIM must detect this condition and end its own data transfer process. The timing parameter associated with aborted transfers, and odd-byte transfers, is called the End-Of-Frame Detection Latency. This is the maximum time that the STIM will take to detect

that the NCAP has removed NIOE and be ready to start a new data transfer. Obviously, this timing is very dependent on how the NIOE signal is monitored and how much other work the microcontroller is performing.

Figure 2 – Read Channel Timing



1. The NIOE signal is asserted by the NCAP to tell the STIM to get ready for a data transfer.
2. The NCAP waits for the STIM to assert the NACK signal.
3. The STIM asserts the NACK signal and the NCAP sends the function address to the STIM.
4. The NCAP wait for the NACK signal to be negated and then sends the channel number to the STIM.
5. The NCAP waits for the NACK signal to be asserted. When the STIM asserts NACK, the NCAP reads the first byte of data from the STIM.
6. The NCAP waits for the NACK signal to be negated and then reads the second byte of data from the STIM.
7. The NCAP negates NIOE and waits for the STIM to negate NACK.
8. The STIM negates NACK to end the transaction.

Triggering, the process used to tell a STIM to take a new data sample or update an output channel, has a more complex timing sequence than a data transfer. A typical trigger sequence is shown in Figure 3. Notice that the NCAP asserts the NTRIG signal but not the NIOE signal, and that the STIM asserts NACK some time after it detects NTRIG. There are several timing parameters associated with a trigger. The first parameter is the Channel Write Setup Time, which takes place following a write data transfer and prior to the assertion of NTRIG. This is how long it takes the STIM to get ready to update an actuator channel. The next parameter is the Channel Update Time. This is the maximum amount of time required for the STIM to read all of its input channels, update all of its output channels, and assert NACK in response to NTRIG.

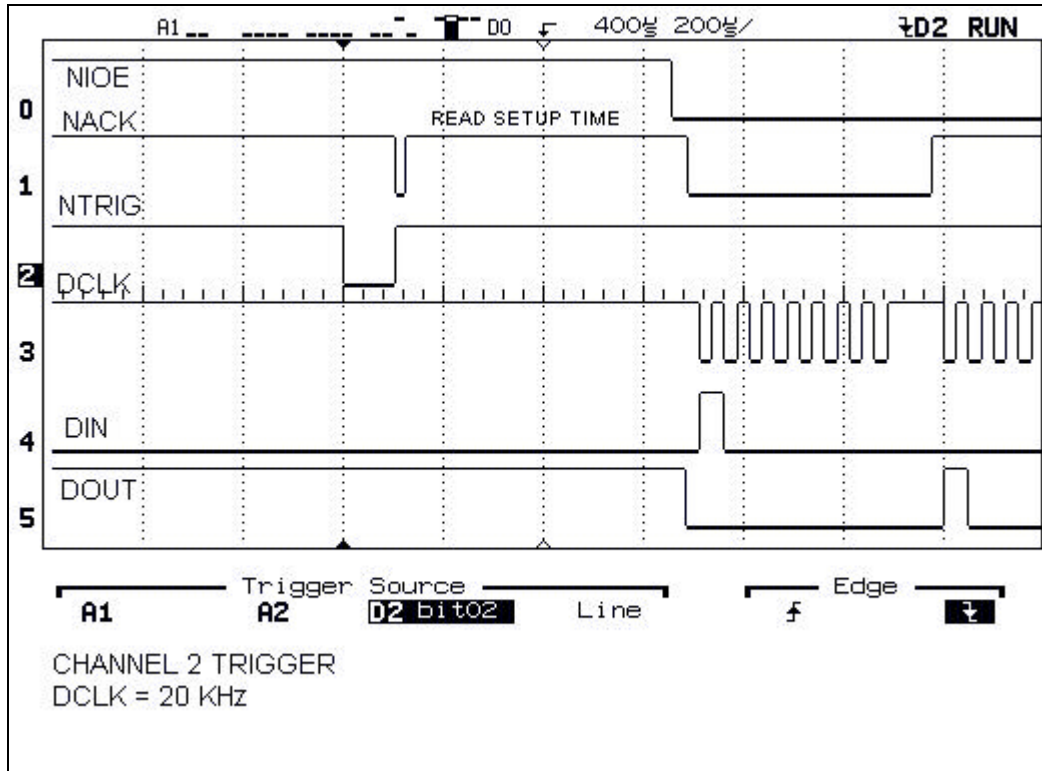
Finally, the Channel Read Setup Time is the minimum amount time required by the STIM to be ready for a read data transfer. For the TMI 1451.2-SC, the read setup time is roughly 550  $\mu$ s. The trigger timing parameters are summarized in Table 3.

All of these parameters can be tough to specify, and may vary depending on the amount of work the microcontroller is performing. For example, the TMI 1451.2-SC can be programmed to sample new data with each trigger or it can sample new data with a sample clock provided by the EDM 710. When sampling with the trigger, the channel update time is longer since the STIM is waiting for four ADC conversions to complete rather than simply reading the last converted values.

*Table 2 – Data Transfer Timing Parameters*

PARAMETER	DESCRIPTION	IEEE 1451.2 REFERENCE
STIM Handshake Time	Maximum time the STIM requires to negate acknowledge after the data transport ends. Measured from the rising edge of NIOE to the rising edge of NACK. Also applies to a trigger sequence. See Table 3.	5.1.3.19
End-Of-Frame Detection Latency	Maximum time the STIM requires to detect the end of a data transport. Following this time, the STIM should be ready to start a new transaction.	5.1.3.20
TEDS Hold-Off Time	Maximum time the STIM requires to acknowledge the transfer of a single byte. Measured from the last rising DCLK edge to the transition of NACK.	5.1.3.21
Operational Hold-Off Time	Same as above but for non-TEDS operations.	5.1.3.22
Maximum Data Rate	Maximum clock rate supported by the STIM. The current Agilent Technologies NCAP supports rates up to 10 Mbps.	5.1.3.23

Figure 3 – Trigger Channel Timing



The best method to determine the timing parameters is to set the initial value in the TEDS to a very high maximum, or a very low minimum. Then, armed with a logic analyzer, measure the worst case timing values for each of the signals. The worst case maximum values should attempt to cover things like interrupt service routines running during a trigger sequence, or background software tasks that may hold-off a data transfer. The worst case minimums should at least cover the time required by the STIM to read or write a data register. In most cases, the maximum values are more important since these values are used by the NCAP to determine timeouts. When the maximums are set too low, the STIM may work, but eventually a data transfer will take too much time and the NCAP will report an error condition.

#### 4. SOFTWARE

Most of the work required to build a STIM happens while writing the software for the microcontroller. Not that the hardware design isn't important, but the majority of the IEEE 1451.2 functionality comes from software. The software for a typical STIM consists of six major sections, as listed below. These functions comprise the basic set necessary for an operational STIM.

- Hardware initialization
- Reading sensor inputs
- Updating actuator outputs
- STIM/NCAP communications
- Command processing
- TEDS storage and retrieval

Table 3 – Trigger Sequence Timing Parameters

PARAMETER	DESCRIPTION	IEEE 1451.2 REFERENCE
STIM Handshake Time	Maximum time the STIM requires to negate acknowledge after the trigger sequence ends. Measured from the rising edge of NTRIG to the rising edge of NACK. Also applies to data transfers. See Table 2.	5.1.3.19
Channel Update Time	Maximum time the STIM requires to acknowledge a trigger. Measured from the falling edge of NTRIG to the falling edge of NACK.	5.2.3.21
Channel Write Setup Time	Minimum time required by the STIM after a write transaction but before a trigger. Measured from the rising edge of NIOE to the falling edge of NTRIG.	5.2.3.22
Channel Read Setup Time	Minimum time required by the STIM after a trigger but before a read transaction. Measured from the falling edge of NACK to the falling edge of NIOE.	5.2.3.23
Channel Sample Period	Minimum required time between trigger sequences. Typically, a function of how fast the STIM can take data samples.	5.2.3.24

The first section, hardware initialization, is the process of programming the input and output ports of the microcontroller, enabling timers, peripherals, and interrupt sources, and doing any required ADC or DAC initialization. In the TMI 1451.2-SC, this process includes programming the initial settings for the EDM 710 and setting the input multiplexers for the desired mode.

The next two sections, reading sensor input and updating sensor output, is the simplest part of the STIM software. Generally, the STIM will take a new ADC reading or write data to a DAC output when it receives a trigger signal. However, the ADC readings from the sensor

must be buffered since the NCAP will not actually read the data until after the trigger. Similarly, data is written to an actuator channel before the STIM is triggered, but the output should not change until the STIM is triggered.

The STIM/NCAP communications section is a large portion of the overall software design. The implementation of this function drives the remaining parts of the design. The communications protocol for IEEE 1451.2 is based on a memory mapped I/O device. It has registers that are read only and registers that are write only. As discussed in the hardware section, the NCAP asserts NIOE to start a

transaction. This functions like a chip-select. Then, the NCAP sends a 16-bit address comprised of an 8-bit function address and an 8-bit channel address. Each function address is a specific operation. In addition, the high-order bit of the function address is set if the operation is a read; the bit is clear if the operation is a write. Following the address, the STIM must respond with data for a read operation or receive data for a write operation. However, there is no guarantee of how much data the NCAP will read or write. The STIM can assume that the NCAP will read or write the amount of data shown in the TEDS for the channel, but that it is not a requirement. In addition, the NCAP may abort the transaction at any time.

Therefore, the incorrect way to design the communication function is to use loops that receive or send a fixed number of bytes. A better approach is to allow the serial interface to pace itself. At the end of each byte transfer, an interrupt service routine (ISR) can ready the next byte for input or output, and keep track of the number of bytes transferred. In addition, if the transfer exceeds the assumed size the ISR can send dummy data or discard any extra data. This requires some extra planning in the firmware, but it well worth the effort. The TMI 1451.2-SC firmware uses this approach for all of its communications.

Along with the physical aspects of transferring data with the NCAP, the STIM must also perform command processing. That is, interpreting the function address and performing the requested action. Section 4.4.3.2 of the IEEE 1451.2 standard lists the commonly used functional addresses. All STIMs should implement these commonly used functions to be considered compliant with the standard. However, it is not necessary to have the complete set for testing and debugging. The minimum set of functions for an operational STIM is listed in Table 4, in order of priority. Since the NCAP will not recognize a STIM without TEDS, it is important that the TEDS

functions be implemented first. The calibration TEDS are optional and can be done last.

The final section, TEDS storage and retrieval, is the most interesting part of the STIM software. The issue of TEDS storage is very much like the chicken and the egg controversy. A STIM must have TEDS to be properly recognized by the NCAP but when a STIM is first built the TEDS is invalid, so how do the TEDS get loaded? One option is to place the TEDS information into the firmware memory. This is a poor option for a number of reasons. Storing data in program memory is awkward, and it reduces the space available for the application code. In addition, it makes updating the TEDS difficult. This option is not recommended unless the STIM is very basic or cost is extremely important.

A better option is to store the TEDS in EEPROM, either internal to the microcontroller or external. In addition, if the EEPROM can be programmed directly by the STIM, then the TEDS can be updated at any time. However, there is one small problem: The NCAP will not recognize a STIM without valid TEDS. Fortunately, Agilent Technologies has addressed this problem by allowing certain operations to be performed on STIMs with blank TEDS. The TEDS is considered blank if the Meta-TEDS length is 0x00000000 or 0xFFFFFFFF. Since the blank state of an EEPROM is 0xFF, reading a STIM without any TEDS will return a length of 0xFFFFFFFF. This feature allows a new STIM to be loaded with its initial TEDS information.

Currently, there are two methods of loading TEDS in use by Telemonitor, Inc. and Agilent Technologies. The Agilent Technologies NCAP directly supports a block-loading mode that sends all of the TEDS information to the STIM using a single function address.

*Table 4 – Basic STIM Functions*

FUNCTION ADDRESS	NAME	COMMENTS
160	Read Channel TEDS	Meta-TEDS when channel=0.
161	Read Identification TEDS	Meta-Id TEDS when channel=0.
3	Write trigger channel	
128	Read transducer data	
0	Write transducer data	
192	Read Calibration TEDS	Not valid for channel 0. Optional.
193	Read Calibration-Id TEDS	Not valid for channel 0. Optional.

In contrast, STIMs built by Telemonitor, Inc. use a separate function address to write each type of TEDS. A separate loading program, such as the Telemonitor, Inc. TEDS Loader, is needed to allow the NCAP to support a STIM that uses this method. The function addresses for loading TEDS are listed in Table 5. Which method a STIM supports is a design choice. Telemonitor, Inc. chose to allow updating each individual type of TEDS because that is the intent of the IEEE 1451.2 standard. However, this method requires more design work and application code than a block-load.

An important consideration is when and by whom the TEDS will be updated. The IEEE

1451.2 standard mandates that end-users have the capability to update the calibration TEDS. However, the manufacturer's TEDS information should be write-protected. A STIM destined for the OEM market, like the TMI 1451.2-SC, can not have its TEDS write-protected by the STIM manufacturer because the manufacturer is not the final integrator of the STIM with the transducer. The STIM integrator, then, will require a method to write-protect the final TEDS information, while allowing end-user calibration updates. This may be more difficult with a STIM that uses the block-loading technique than with one that uses a section-loading technique.

*Table 5 – TEDS Loading Functions*

FUNCTION ADDRESS	NAME	COMMENTS
32	Write Channel TEDS	Write Meta-TEDS when channel=0.
33	Write Identification TEDS	Write Meta-Id TEDS when channel=0.
64	Write Calibration TEDS	Not valid for channel 0.
65	Write Calibration-Id TEDS	Not valid for channel 0.
112	Block load TEDS	Always channel 0.

## 5. SUMMARY

This paper described the hardware and software design of an IEEE 1451.2 compliant STIM. It presented the requirements for the electronics, covering both the interface specification and the electrical timing for that interface. Additional information about analog signal conditioning and physical connectors was discussed to give designers a feel for the real-life engineering issues that are not part of the IEEE 1451.2 standard. Furthermore, a brief overview of STIM software was presented covering the

major sections of STIM firmware. The software discussion included a description of the command protocol and suggestions for implementing the serial communications code. Finally, two practical methods of storing and retrieving TEDS data were presented.

*Ronald D. Smith is a senior Electronics Engineer with Telemonitor, Inc. He can be reached by voice at 410-312-6677 or by email at [rdsmith@telemonitor.com](mailto:rdsmith@telemonitor.com)*