I²MTC 2008 – IEEE International Instrumentation and
Measurement Technology Conference
Victoria, Vancouver Island, Canada, May 12–15, 2008

Sensor Network based on IEEE 1451.0 and IEEE p1451.2-RS232

Eugene Y. Song, Kang B. Lee
National Institute of Standards and Technology
100 Bureau Drive, MS# 8220
Gaithersburg, Maryland USA 20899-8220
Phone: (301) 975-6542, (301) 975-6604
E-Mail: ysong@nist.gov, kang.lee@nist.gov

Abstract *– The Institute of Electrical and Electronics Engineers (IEEE) 1451 family of standards defines a set of common communication interfaces for connecting smart transducers (sensors or actuators) to microprocessor-based systems, instruments, and networks in a network-independent environment. The IEEE 1451.2 working group is in the process of revising IEEE p1451.2 to work with the IEEE 1451.0-2007 standard. This paper proposes an architecture of sensor network based on the IEEE 1451.0 and IEEE p1451.2 to support the popular interfaces including Recommended Standard 232 (RS232), Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI), and Universal Serial Bus (USB). It also describes the development and implementation of such a network based on IEEE 1451.0 and p1451.2-RS232. The sensor network consists of a Network Capable Application Processor (NCAP) and a Transducer Interface Module (TIM) with an IEEE p1451.2-RS232 interface. The client-server communication protocols between the NCAP and the TIM have been developed based on the IEEE 1451.0-2007 standard and IEEE p1451.2-RS232 interface. Two case studies are presented as implementation examples. The work described provides a good foundation for updating the IEEE 1451.2-1997standard to be compatible with the IEEE 1451.0 standard.*

Keywords*: NCAP, TIM, IEEE 1451.0, IEEE p1451.2, TEDS, TII, RS232, SPI, UART, USB*

I. INTRODUCTION

The Institute of Electrical and Electronics Engineers (IEEE) 1451 family of standards defines a set of common communication interfaces for connecting smart transducers (sensors or actuators) to microprocessor-based systems, instruments, and networks in a network-independent environment [1]. This family of standards provides a set of protocols for wired and wireless distributed measurement and control applications. Figure 1 shows the architecture of the suite of IEEE 1451 standards. Basically the IEEE 1451 defines a Network Capable Application Processor (NCAP) and a Transducer Interface Module (TIM), and the interface between them. The NCAP consists of hardware and software systems that provide network gateway functions between the TIMs and the user network. The NCAP controls a TIM by means of a digital interface. The TIM can be a wired or wireless transducer module consisting of transducers (sensors and actuators), signal conversion and processing electronics, Transducer Electronic Data Sheets (TEDS), and a IEEE 1451.X communication module. IEEE 1451.X refers to a physical interface between the NCAP and the TIM. The TEDS contains the meta-data for smart transducers such as transducer identification, measurement range, location, calibration, user information, and more. The NCAP can access the TIM and pass the transducer information to the user network. There are three possible ways to access the NCAP through the user network: the IEEE 1451.1 protocol [2], the IEEE 1451.0 Hyper Text Transfer Protocol (HTTP) [3], and the proposed Smart Transducer Web Services (STWS) [4-5]. The IEEE 1451.0 HTTP protocol and STWS need eXtensible Modeling Language (XML) encoding for smart transducers.

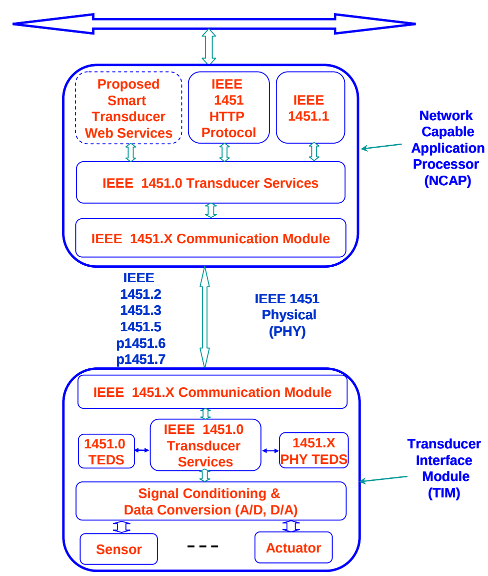


Figure 1. Architecture of the suite of IEEE 1451 standards.

The IEEE 1451.0 standard defines a common set of commands, TEDS formats, and communication protocols for the family of IEEE 1451 standards [3], which help to achieve data-level interoperability among the IEEE 1451 family of standards. The IEEE 1451 applications can communicate with the IEEE 1451.0 transducer services through the IEEE 1451.0 Transducer Services Interface on the NCAP. The IEEE 1451.X communication module on the NCAP can communicate with the IEEE 1451.X communication module on the TIM for transducer data and TEDS through the IEEE 1451.X physical interfaces such as IEEE 1451.2-1997 [6], IEEE 1451.3-2003 [7], IEEE 1451.5-2007 [8], IEEE p1451.6 [9], and IEEE p1451.7 [10].

The IEEE 1451.0 standard provides a unified interface for the IEEE 1451 family of smart transducers. In order for IEEE

1451 smart transducers to achieve plug-and-play, the NCAP and TIM must conform to the IEEE 1451.0 standard. The NCAP and TIM implementations must support the required functional specifications, message structures, commands, TEDS of the IEEE 1451.0 standard, one of the IEEE 1451.X physical media and the respective communications protocol [3]. Presently IEEE 1451.2-1997 is not compatible with the IEEE 1451.0 standard, but the IEEE p1451.2 working group is currently revising the standard (designed as IEEE p1451.2) to make it compatible with the IEEE 1451.0 standard. This paper mainly focuses on a proposed sensor network architecture based on IEEE 1451.0, IEEE p1451.2, and an IEEE p1451.2-RS232 interface.

II. RELATED WORK

IEEE 1451.2-1997 defined a 10-wire digital interface for connecting transducers to a microprocessor, where a transducer is a part of Smart Transducer Interface Module (STIM), and a microprocessor is a part of the NCAP. All communications between the TIM and the NCAP take place through the 10-wire digital interface that is known as the Transducer Independent Interface (TII). IEEE 1451.2-1997 also describes a TEDS and its data formats. The TII is basically an enhanced Serial Peripheral Interface (SPI) incorporating a synchronous serial interface along with other lines for triggering, handshaking, and detection of a plug-in module. It also defines read and write logic functions to access the TEDS and a wide variety of transducers. Data transfers are based on the SPI bit-transfer protocol. An IEEE 1451.2 compatible 10-wire TII has been implemented using the ADuC812** MicroConverter chip [11]. Similarly an IEEE 1451.2-1997 STIM module with a 10-wire TII was developed with a Microchip PIC16F877 microcontroller [12]. Many people have implemented IEEE 1451.2 with physical interfaces other than the TII. In one case, the TII was replaced with a 3-wire RS232 (Recommended Standard 232) serial interface [13]. Likewise a smart temperature sensor was developed using a RS232 interface for an IEEE 1451.2-1997 STIM [14]. An IEEE 1451.2 STIM consisting of one analog input (sensor) channel and one analog output (actuator) channel with a RS232 interface has been demonstrated [15]. A reconfigurable NCAP has been developed using a variety of popular interfaces for a STIM [16]. An IEEE 1451.2 STIM was implemented using a Universal Serial Bus (USB) [17]. All these interfaces are popular, but they are not compatible with the IEEE 1451.2-1997 and IEEE 1451.0-2007 standard. Thus the revised IEEE p1451.2 standard is proposed to support RS232, SPI, Universal Asynchronous Receiver/Transmitter (UART), and USB, in addition to TII.

III. PROPOSED ARCHITECTURE OF A SENSOR NETWORK BASED ON THE IEEE 1451.0 AND P1451.2

IEEE 1451.2-1997 defined a STIM-to-NCAP interface and a TEDS format for a point-to-point configuration. The standard describes a communication layer based on an enhanced SPI with additional hardware lines for flow control and timing. The standard is being revised to define a Transducer Interface Module (TIM) and to add support for some popular interfaces. The proposed architecture of a sensor network based on IEEE 1451.0 and p1451.2 with some popular interfaces is described in the following section.

A. Architecture of a Sensor Network based on IEEE 1451.0 and p1451.2

Figure 2 shows the architecture of a proposed sensor network based on IEEE 1451.0 and p1451.2. The NCAP is a device that can contain one or more communication modules such as the TII, RS232, SPI, UART, and USB. Each TIM contains a matching communication module, signal conditioning, analog-to-digital and/or digital-to-analog converters, and transducers. The NCAP can communicate with one or more IEEE p1451.2 TIMs through different specified communication modules. The NCAP can also be connected to a user network.

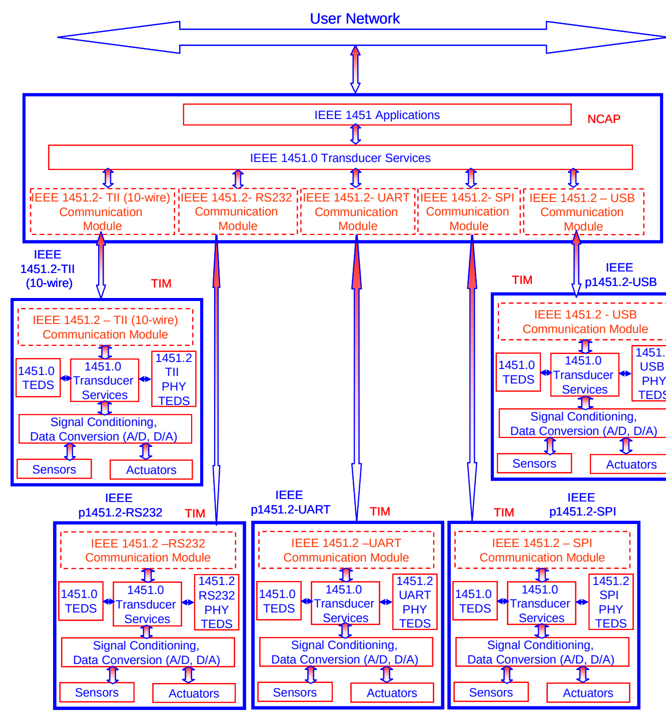


Figure 2. Architecture of sensor network based on IEEE 1451.0 and p1451.2.

The different communication modules of the IEEE p1451.2 are:

- TII consists of a set of signal wires connecting the TIM to the NCAP. It is intended to be used for point-to-point communication, power delivery, bus timing, and facilitating bi-directional communications of TEDS, sensor, and actuator data in various operational modes [6].

- RS-232 is a standard for serial binary data signals connecting between a Data Terminal Equipment (DTE) and a Data Circuit-terminating Equipment (DCE) [18].
- UART is a type of asynchronous receiver/transmitter. It is usually an individual integrated circuit used for serial communications over a computer or peripheral device serial port [19].
- SPI is a synchronous serial data link standard. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines [20].
- USB is a serial bus standard for interfacing devices. It was designed to allow peripherals to be connected using a single standardized interface socket and to improve plug-and-play capabilities by allowing devices to be connected and disconnected without rebooting the computer (hot swapping) [21].

B. Communication Protocols between NCAP and p1451.2 TIM

Figure 3 shows the client-server communications between the IEEE 1451 NCAP and the IEEE p1451.2 TIM. The NCAP sends a request to the TIM, and the TIM returns a response. Both the NCAP and TIM communicate with each other using the same IEEE p1451.2 communication module, for example, the IEEE p1451.2-RS232.

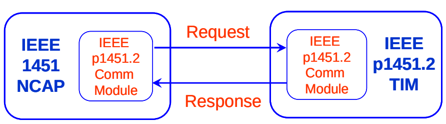


Figure 3. Communications between NCAP and p1451.2 TIM.

1) Common Commands to the IEEE p1451.2 TIM and TransducerChannel

The IEEE 1451.0 commands are divided into the standard and manufacturer-defined types. The command is divided into two octets. The most significant octet defines the class of the command. The least significant octet, called the function of the command, identifies the specific command within the class [3]. A TIM may generate a reply to a command under either of two circumstances. The first circumstance is when the command itself requires a reply, for example, the Query TEDS command. The second circumstance is when the status-event protocol is enabled. The standard commands, or control functions, allow commands to be sent to the TIM as a whole, or to each transducer channel that affects its state or operation. The standard command classes include common commands, transducer idle and operate, TIM idle and active. The common commands include Query TEDS, Read TEDS segment, Write TEDS segment, and Update TEDS.

2) Message Structure of IEEE 1451.0 Standard

The IEEE 1451.0 defined the structure of the messages sent across the Module Communications Interface (MCI) [3]. The order of transmission of the message is resolved to the octet level. A message consists of a group of octets transmitted sequentially.

a) Command Message Structure

The format of a command message is shown in Table 1 [3]. A command message consists of destination (TransducerChannel number), command class, command function, length of message, and command related octets (offset). The two examples of commands are described as follows. The first example is a command to read a TEDS, which includes the destination TransucerChannel number (two bytes xx xx), command class, command function, command length (two bytes xx xx), TEDS Type ID, and offset. The second example is a command to read a TransducerChannel (sensor) data, which includes the destination, command class, command function, length of command, and offset.

Table 1—Command message structure

1-Octet
7 6 5 4 3 2 1 0
Destination TransducerChannel Number (most significant octet)
Destination TransducerChannel Number (least significant octet)
Command Class
Command Function
Length (most significant octet)
Length (least significant octet)
Command dependent octets
….

```
xx xx 01 02 xx xx 03 00 00 00 00 (Read ChannelTEDS)
----- -- -- ----- -- -----------
 |    |  |     |  |        |----- Offset into TEDS
 |    |  |     |  |------------ TEDS type ID
 |    |  |     |-------------- Command Length
 |    |  |-------------------- Command Function(readTEDS)
 |    |----------------------- Command Class (commonCmd)
 |---------------------------- Destination (ChannelID)
```

```
xx xx 03 01 xx xx 00 00 00 00 (Read Channel Data)
----- -- -- ----- -----------
 |    |  |     |      |------- Offset
 |    |  |     |------------ Length of Command
 |    |  |------------------ Command Function (Read data)
 |    |--------------------- Command Class (XdcrOperate)
 |-------------------------- Destination (ChannelID)
```

b) Reply Messages

Reply messages are used to reply to a command. The format of a reply message is shown in Table 2 [3]. The reply message consists of a success/fail flag, length of message, reply related octets (offset), and transducer data or TEDS. For example, the first reply message is a reply to a read TEDS command, the second one is a reply to a read transducer channel data command. The two examples are described as follows. The first reply message includes a success flag, length of reply message, offset, and bytes of TEDS. The second one includes a success flag, length of reply message, offset, and bytes of sensor data.

Table 2—Reply message structure

1-Octet
7 | 6 | 5 | 4 | 3   2   1   0
Success/Fail Flag
Length (most significant octet)
Length (least significant octet)
Reply dependent octets
….

```
01 xx xx 00 00 00 00 YY … YY (Reply to Read TEDS)
-- ----- ----------- -------
 |   |        |          |---- TEDS bytes
 |   |        |------------- Offset into data
 |   |---------------------- Reply Length
 |------------------------- Success Flag

01 xx xx 00 00 00 00 ZZ ZZ (Reply to ReadChannelData)
-- ----- ----------- -----
 |   |        |         |----- bytes of transducer data
 |   |        |------------- Offset into data
 |   |---------------------- Reply Length
 |------------------------- Success Flag
```

IV. SENSOR NETWORK BASED ON IEEE 1451.0 and P1451.2-RS232

A. Architecture of a Sensor Network based on IEEE 1451.0 and p1451.2-RS232

Figure 4(a) shows the architecture of a sensor network based on the IEEE 1451.0 and p1451.2-RS232. The sensor network consists of one NCAP node and one TIM node. The NCAP communicates with the TIM through the IEEE 1451.0 and p1451.2-RS232 protocols using the client-server communication model. The client-server communications between two nodes have been implemented with a RS232 interface protocol. Data is sent as a time-series of bits through RS-232. Both synchronous and asynchronous transmissions are supported by the standard.

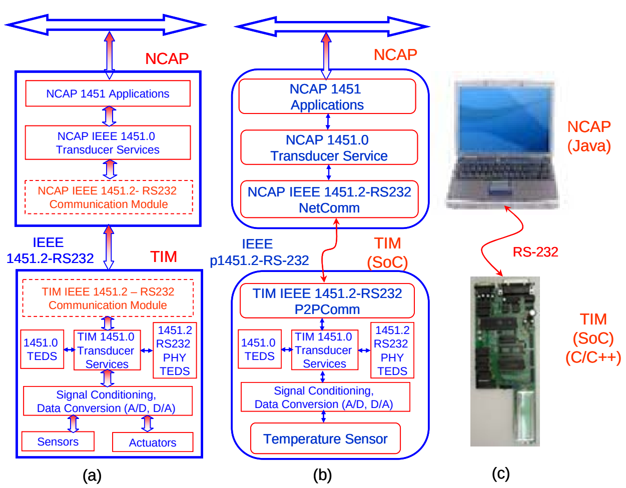


Figure 4. Sensor network based on IEEE 1451.0 and p1451.2-RS232.

B. Implementation of the Sensor Network based on IEEE 1451.0 and p1451.2-RS232

Figures 4(b) and (c) show the implementation of the sensor network based on the IEEE 1451.0 and IEEE p1451.2-RS232.

1) NCAP Implementation

The NCAP node implemented in a laptop was developed in Java based on the IEEE 1451.0 and proposed IEEE p1451.2-RS232 standards. The class diagram of the NCAP node is shown in Figure 5. The NCAP node consists of one `NCAPDot0Dot2IntegrationIApp` object, one `NCAPDot2RS232NetComm` object, and two help classes of IEEE p1451.2-RS232. The `NCAPDot0Dot2IntegrationApp` object implements `TransducerAccess`, `TedsManager`, `TimDiscovery`, `NetRegistration` and `NetReceive` interfaces of IEEE 1451.0. The `NCAPDot2RS232NetComm` object implementing the `NetComm` interface communicates with the IEEE p1451.2-RS232 TIM side using RS232 help classes. The class diagram of the NCAP node can be used to automatically generate the Java source code.

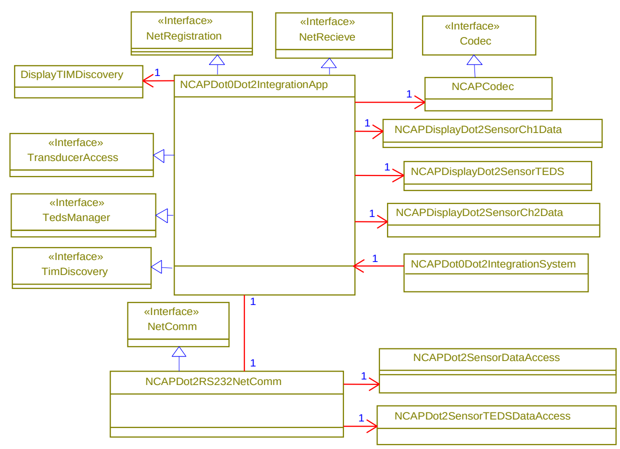


Figure 5. Class diagram of NCAP.

2) TIM Implementation

The TIM is developed and implemented in C++ on an 8052-based microprocessor. The TIM consists of a RS232 communication module, a temperature sensor, signal conditioning, and data conversion, and IEEE 1451.0 required TEDS such as PhyTEDS, metaTEDS, TransducerChannelTEDS, Calibration TEDS, and TransducerNameTEDS. The TIM can communicate with the NCAP through the IEEE p1451.2-RS232 communication module.

V. CASE STUDIES

A temperature sensor is used in a case study to read sensor and TEDS data and test the implementation of the sensor network based on IEEE 1451.0 and p1451.2. The two case studies are described in the following. The communications between the NCAP and TIM are implemented through RS232 and the command formats are implemented and described in section III.

A. Case Study 1: Read PhyTEDS

The PhyTEDS is a required TEDS. It includes detailed information of the IEEE 1451.X physical layer. The PhyTEDS octets are constant and read-only. The PhyTEDS types of the IEEE p1451.2 include the TII 10-wire (0), RS232 (1), UART (2), SPI (3), and USB (4) interfaces. The IEEE p1451.2 PhyTEDS consists of two parts: common physical TEDS data and specific physical layer TEDS data. The PhyTEDS of IEEE p1451.2-RS232 has been developed for the implementation tests.

1) Request of Reading PhyTEDS

The command to read the PhyTEDS is described as follows. The destination (0x00 0x00) means the TIM. The command class (0x01) means a common command. The command function (0x02) means to read the TEDS. The TEDS type ID (0x0D or 13) means the PhyTEDS.

```
00 00 01 02 00 05 0D 00 00 00 00 (Read PHYTEDS)
----- -- -- ----- -- -----------
  |    |  |   |    |      |------- Offset into TEDS
  |    |  |   |    |-------------- TEDS Type ID(PhyTEDS)
  |    |  |   |------------------- CommandLength (5)
  |    |  |----------------------- CommandFunction(readTEDS)
  |    |-------------------------- CommandClass (CommCmd)
  |------------------------------- Destination (TIM)
```

2) Response of Reading PhyTEDS

The header of the response message of reading PhyTEDS is as follow:

```
01 00 60 00 00 00 00 00 00 00 58 (Reply to Read TEDS)
-- ----- ----------- -----------
 |   |        |          |---- TEDS bytes
 |   |        |-------------- Offset into data
 |   |----------------------- Reply Length
 |--------------------------- Success Flag
```

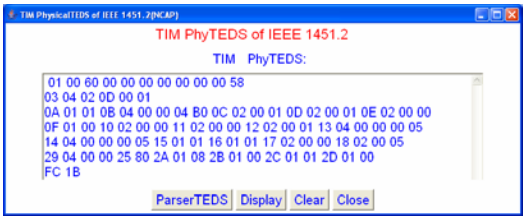
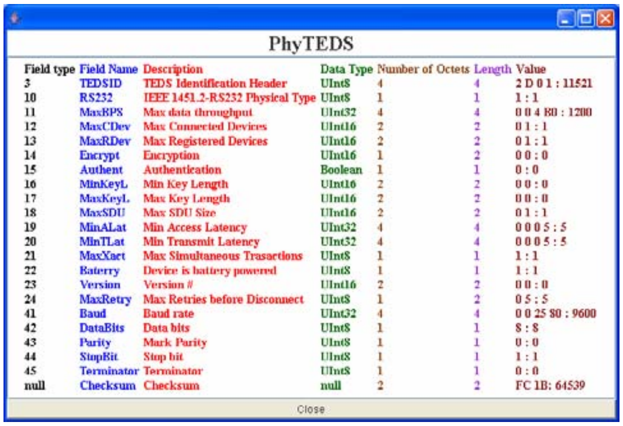


Figure 6. Screenshot of reading PhyTEDS.

Figure 6 shows a screenshot of reading physical TEDS. The first line in Figure 6 shows the header of the response message of reading the PhyTEDS. The other lines in the Figure 6 show the content of the PhyTEDS.

Figure 7 shows a screenshot of parsing the physical TEDS of the IEEE p1451.2-RS232 interface. The field types from 3 to 24 are common physical TEDS data, and the field types from 41 to 45 are RS232-specific TEDS data. The value (1) of the field type 10 means that the IEEE p1451.2 physical layer is RS232. The IEEE p1451.2-RS232 physical type has default parameters setup such as baud (9600), data bits (8), parity (0), stop bit (1), and terminator (0-CR/LF).



PhyTEDS

Field type	Field Name	Description	Data Type	Number of Octets	Length	Value
3	TEDSID	TEDS Identification Header	UInt8	4	4	2 D 0 1 : 11521
10	RS232	IEEE 1451.2-RS232 Physical Type	UInt8	1	1	1 : 1
11	MaxBPS	Max data throughput	UInt32	4	4	0 0 4 B0 : 1200
12	MaxCDev	Max Connected Devices	UInt16	2	2	0 1 : 1
13	MaxRDev	Max Registered Devices	UInt16	2	2	0 1 : 1
14	Encrypt	Encryption	UInt16	1	2	0 0 : 0
15	Authent	Authentication	Boolean	1	1	0 : 0
16	MinKeyL	Min Key Length	UInt16	2	2	0 0 : 0
17	MaxKeyL	Max Key Length	UInt16	2	2	0 0 : 0
18	MaxSDU	Max SDU Size	UInt16	2	2	0 1 : 1
19	MinALat	Min Access Latency	UInt32	4	4	0 0 0 5 : 5
20	MinTLat	Min Transmit Latency	UInt32	4	4	0 0 0 5 : 5
21	MaxXact	Max Simultaneous Trasactions	UInt8	1	1	1 : 1
22	Baterry	Device is battery powered	UInt8	1	1	1 : 1
23	Version	Version #	UInt16	2	2	0 0 : 0
24	MaxRetry	Max Retries before Disconnect	UInt8	1	2	0 5 : 5
41	Baud	Baud rate	UInt32	4	4	0 0 25 80 : 9600
42	DataBits	Data bits	UInt8	1	1	8 : 8
43	Parity	Mark Parity	UInt8	1	1	0 : 0
44	StopBit	Stop bit	UInt8	1	1	1 : 1
45	Terminator	Terminator	UInt8	1	1	0 : 0
null	Checksum	Checksum	null	2	2	FC 1B: 64539

Close

Figure 7. Screeshot of parsing PhyTEDS.

B. Case Study2: Read Transducer Channel Data

1) Request of Reading Sensor Data of Transducer Channel 1

The command to read the temperature sensor is described as follows. The destination (0x00 0x01) means channel 1 (temperature sensor). The command class (0x03) means transducer operation. The command function (0x01) means to read transducer channel data.

```
00 01 03 01 00 04 00 00 00 00 (Read Channel Data)
----- -- -- ----- -----------
  |    |  |   |       |------------ Offset
  |    |  |   |-------------------- Length of Command
  |    |  |------------------------ CommandFunction(Readdata)
  |    |--------------------------- CommandClass(XdcrOperate)
  |-------------------------------- Destination(ChannelID)
```

2) Response of Reading Sensor Data of Transducer Channel 1

The response message of reading channel data is as follows:

```
01 00 06 00 00 00 00 18 147 (Reply to ReadChannelData)
-- ----- ----------- -----
 |   |        |         |----- bytes of transducer data
 |   |        |-------------- Offset into data
 |   |----------------------- Reply Length
 |--------------------------- Success Flag
```

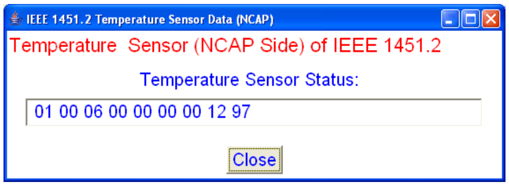


Figure 8. Screenshot of reading data of transducer channel 1.

Figure 8 shows a screenshot of reading the transducer channel data. The first seven numbers (0x01 0x00 0x06 0x00 0x00 0x00 0x00) are the header of the response message. The last two numbers, T1 and T2, are the data from the temperature sensor. They are presented in hexa-decimal numbers 0x12 and 0x97. The decimal equivalences are 18 and 151, respectively. The current temperature reading shown in Figure 8 is 297.4375 degrees Kelvin (or 24.29 degrees Celsius). The temperature is calculated as follows: Temperature in degrees Kelvin = ((T1 (in decimal) * 256) + T2 (in decimal)) / 16

VI. SUMMARY

The architecture of a sensor network based on the IEEE 1451.0 and p1451.2 standards is presented. A sensor network based on the IEEE 1451.0 and p1451.2-RS232 was developed to implement and test the communication protocols. The work described provides a good foundation for updating the IEEE 1451.2-1997 standard for compatibility with the newly approved IEEE 1451.0-2007 standard and ease of integration of IEEE 1451.0 and IEEE p1451.2.

ACKNOWLEDGEMENT

Thanks to Jon Geist and his associates of the EEEL Semiconductor Electronics Division of NIST for their contribution in this project by collaborating in the development of the IEEE p1451.2 serial interface between the IEEE 1451 NCAP and TIM.

****** Commercial equipment and software, many of which are either registered or trademarked, are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

1. Lee, K., "IEEE 1451: A Standard in Support of Smart Transducer Networking", Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, Baltimore, MD, May 1-4, 2000, Vol. 2, p.525-528.
2. IEEE 1451.1-1999, Standard for a Smart Transducer Interface for Sensors and Actuators-Network Capable Application Processor (NCAP) Information Model, The Institute of Electrical and Electronics Engineers, Inc. New York, June 1999.
3. IEEE 1451.0-2007, Standard for a Smart Transducer Interface for Sensors and Actuators– Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016
4. Eugene Song, Kang Lee, "An Implementation of Smart Transducer Web Services for IEEE 1451-based Sensor Systems", SAS 2007 - IEEE Sensors and Applications Symposium, San Diego, California, USA, 6-8 February 2007. pp.1-6
5. Eugene Song, Kang Lee, "Smart transducer web service based on IEEE 1451.0 standard", IMTC 2007 - Instrumentation and Measurement Technology Conference. WARSAW, POLAND, MAY 1-3, 2007
6. IEEE STD 1451.2-1997, Standard for a Smart Transducer Interface for Sensors and Actuators–Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH94566, Sept. 25, 1998.
7. IEEE 1451.3-2003, Standard for a Smart Transducer Interface for Sensors and Actuators Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, Dec. 29, 2003.
8. IEEE 1451.5-2007, Standard for a Smart Transducer Interface for Sensors and Actuators– Wireless Communication and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016.
9. The IEEE P1451.6 - Proposed Standard For A High-Speed CANopen-Based Transducer Network Interface For Intrinsically Safe And Non-Intrinsically Safe Applications. [Online]. Available: http://grouper.ieee.org/groups/1451/6/.
10. IEEE P1451.7 – Proposed Standard for a Smart Transducer Interface for Sensors and Actuators - Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats, (2007, August 30). [Online] Available http://standards.ieee.org/board/nes/projects/1451-7.pdf.
11. Conway, P. Heffernan, D. O'Mara, B. Burton, P. Miao, T, "IEEE 1451.2: An interpretation and example implementation", Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, Baltimore, MD, USA, vol.2, May 1, 2000, pp. 535-540
12. Jun Xu, Bo You and Quanli Li, "Implementation of an IEEE 1451 Smart Quartz Tuning fork Temperature Transducer for Real-time Distributed Measurement and Control System", Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China, June 21 - 23, 2006, vol.2, pp. 5411- 5416
13. L. Bissi, A. Scorzoni, P. Placidi, L. Marrocchi, M. Bennati, S. Zampolli, L. Masini, I. Elmi and G.C. Cardinali, "A Low-cost Distributed Measurement System based on Gas Smart Sensors for Environmental Monitoring", 1st International Conference on Sensing Technology, Palmerston North, New Zealand, November 21-23, 2005, pp. 301-306
14. Darold Wobschall and Wai Sing Poh, "Smart RTD Temperature Sensor with a Prototype IEEE 1451.2 Internet Interface", SIcod04 - Senson for Industry Conference, New Orleans, Louisiana, USA, 27-29 January, 2004, pp. 183-186
15. TMI Smart Transducer Module, [Online]. Available: http://www.telemonitor.com/data/tmi2-sa.pdf.
16. V. Kochana, K. Lee, R. Kochana,T, A. Sachenkoa, "Approach to improving network capable application processor based on IEEE 1451 standard", Computer Standards & Interfaces, Vol. 28, No.2, 2005, pp. 141–149
17. Helena Maria Ramos', Pedro M. Ramos', Pavel Paces, "Development of an IEEE 1451 Standard Compliant Smart Transducer Network with Time Synchronization Protocol", Instrumentation and Measurement Technology Conference - IMTC 2007, Warsaw, Poland, May 1-3, 2007, pp.1-6.
18. RS232, [Online]. Available: http://en.wikipedia.org/wiki/RS-232
19. UART, [Online]. Available: http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
20. SPI, [Online]. Available: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
21. USB, [Online]. Available: http://en.wikipedia.org/wiki/USB.