A Centronics Based Transducer Independent Interface (TII) Fully Compliant with 1451.2 Std.

Helena Ramos[1], M. Pereira[1,2], V. Viegas[2], O. Postolache[1,2], P. Girão[1]

1. Instituto de Telecomunicações, DEEC, IST, Av. Rovisco Pais, 1049-001, Lisboa, Portugal
2. Escola Superior de Tecnologia, Instituto Politécnico de Setúbal, 2914-761, Setúbal, Portugal
hgramos@alfa.ist.utl.pt, joseper@est.ips.pt, vviegas@est.ips.pt, poctav@alfa.ist.utl.pt, psgirao@ist.utl.pt

Abstract – The paper reports the tests conducted on a dedicated Transducer Independent Interface (TII) to check its compliance with standard IEEE 1451.2 specifications. The interface was built using a synchronous data transfer based on the Serial Peripheral Interface (SPI) protocol integrated in a PIC microcontroller used as the Smart Transducer Interface Module (STIM) and the Centronics port of the PC, used as the Network Capable Application Processor (NCAP's TII port). The description of each line and the connections are also presented.
The working prototype of the STIM controlled by a NCAP module was implemented to connect several transducers without communication capabilities used in a water quality monitoring network. The NCAP, also described in the paper, is emulated in a PC and it was fully developed using LabVIEW 6.1

I. Introduction

The IEEE 1451 standard aims at simplifying transducer connectivity to existing networks. It introduces the concepts of Smart Transducer Interface Module (STIM), Transducer Independent Interface (TII) and Network Capable Application Processor (NCAP) [1-6]. The STIM can include from a single transducer to many channels of transducers and is controlled by a NCAP by means of a dedicated digital interface called the TII.
In order to connect several transducers without communication capabilities in a water quality network, a working prototype of a Smart Transducer Interface (STIM)[7-8] controlled by a Network Capable Application Processor (NCAP) module was developed.
The STIM was implemented with a low-cost microcontroller (PIC 16F877)[9-10] that includes an analogue to digital converter, digital input-output (I/O) ports and a built-in SPI communication port. Sensor elements (pH-Campbell CSIM11 and temperature-Pt100), and correspondent signal conditioning circuits were connected to the PIC unit through its I/O ports. A data memory EEPROM constitutes the Transducer Electronic Data Sheet (TEDS) as it stores the configuration data of each transducer. As, to our knowledge, there isn't in the market any off-the-shelf component that straightforwardly emulates a NCAP, a virtual NCAP was implemented. This NCAP includes a PC with Ethernet connection as the hardware component, and a software component fully developed in LabVIEW 6.1[11]. For the communication that occurs between the STIM and the NCAP a 10-wire Transducer Independent Interface (TII) is built using a synchronous data transfer based on the Serial Peripheral Interface (SPI) protocol integrated in the PIC microcontroller used as the STIM and the Centronics port of the PC as the NCAP's TII port.
This paper describes the Centronics based TII and the tests implemented to check the protocol and timing compliances of the developed TII with the standard requirements. The software developed to control data transfers across the interface is also described in detail.

II. Transducer Independent Interface Implementation

The physical interface between the STIM and the NCAP is the 10-wire bus Transducer Independent Interface (TII) compliant with the 1451.2 standard. Each line is described briefly in Table 1. The interface is implemented in the STIM with the Serial Peripheral Interface (SPI) port, integrated in the PIC16F877, plus some more additional control lines (as power, trigger and interrupt lines) and in the NCAP, using the Centronics port of the PC. The Data Register of the parallel port of the PC is used as an output, since it specifies the state of the lines that excite the PIC16F877, and the State Register is used to read the signals.

Table 1. Description of TII lines

Line	Driven by	Function
DIN	NCAP	Address and data transport from NCAP to STIM
DOUT	STIM	Data transport from STIM to NCAP
DCLK	NCAP	Positive-going edge latches data on both DIN and DOUT
NIOE	NCAP	Signals that the data transport is active and delimits data transport framing
NTRIG	NCAP	Performs triggering function
NACK	STIM	Serves: - trigger acknowledge; - data transport acknowledge
NINT	STIM	Used by the STIM to request service from the NCAP
NSDET	STIM	Used by the NCAP to detect the presence of the STIM
POWER	NCAP	5V power supply
COMMON	NCAP	Ground

In Figure 1 the connections between the PIC16F877 and the PC internal bus are depicted.

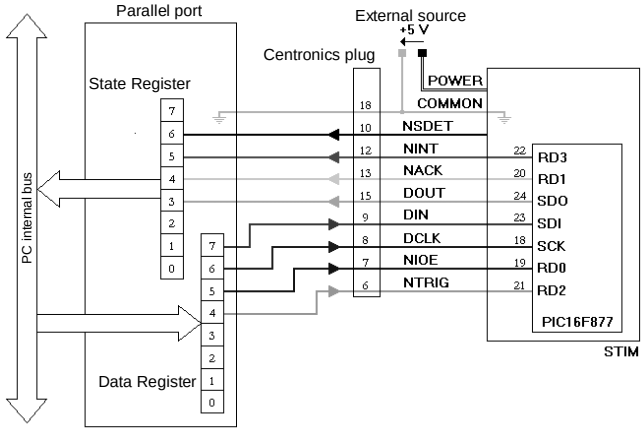


Figure 1. Connections between the STIM and the NCAP

A dynamic link library (DLL), `tii.dll` developed in C implements the set of functions used to control the parallel port of the PC that implements the communication protocol associated with the Transducer Independent Interface (TII). The functions implemented are described briefly in Table 2.

Table 2. Description of the functions implemented in the `tii.dll`

Function	Description
`Wait`	Pause temporisation
`WriteByte`	Series the bits through the parallel port beginning with the MSB
`ReadByte`	Reads the bits arriving in a serial manner from the STIM, beginning with the MSB
`WriteFrame`	Transmits a byte frame to the STIM
`ReadFrame`	Reads a byte frame coming from the STIM
`Trigger`	Triggers the STIM
`WriteActuactor`	Writes new values to the STIM outputs that can actuate a transducer
`ReadSensor`	Acquires new samples from the STIM
`ReadTEDS`	Reads the TEDS structures

III. Transducer Independent Interface Testing

Tests were conducted in the Transducer Independent Interface (TII) in order not only to evaluate the robustness of the data transfer, but also to check protocols and timing conformance of the interface.
The block diagram of the set-up used is presented in Figure 2. As mentioned before, the NCAP is emulated in the PC, the STIM is based in a PIC12F877, with a clock frequency equal to 4 MHz, and the physical

implementation of the TII is a 10-wire shielded cable connecting the Centronics port of the PC with the Serial Peripheral Interface (SPI) port, integrated in the PIC16F877 plus some additional control lines (power, trigger and interrupt lines).

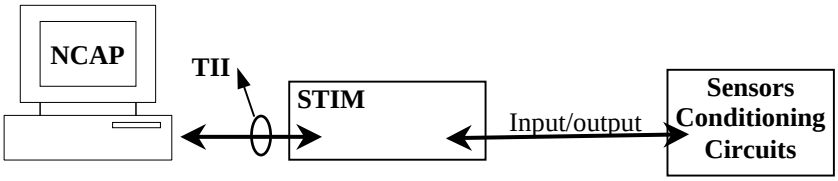
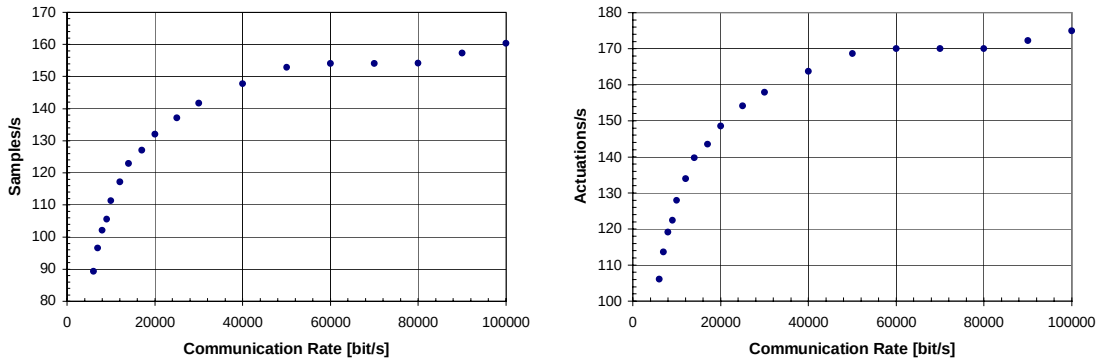


Figure 2. Set up used to test the TII.

Figure 3 shows how the number of acquired samples and the number of actuations per second depends on the communication rate between the STIM and the NCAP. For the experiments depicted in Fig. 3(a) an application commands the STIM to acquire an exact number of 10^4 samples. The acquisition rate is then computed as the ratio between the number of samples and the associated time interval. Several experiments were performed for different communication rates and Figure 3(a) presents the obtained results. Figure 3(b) presents the actuation rate computed as the ratio between the number of actuations performed and the time elapsed when the STIM actuates 10^4 times, for different communication rates between the STIM and the NCAP.



(a) (b)

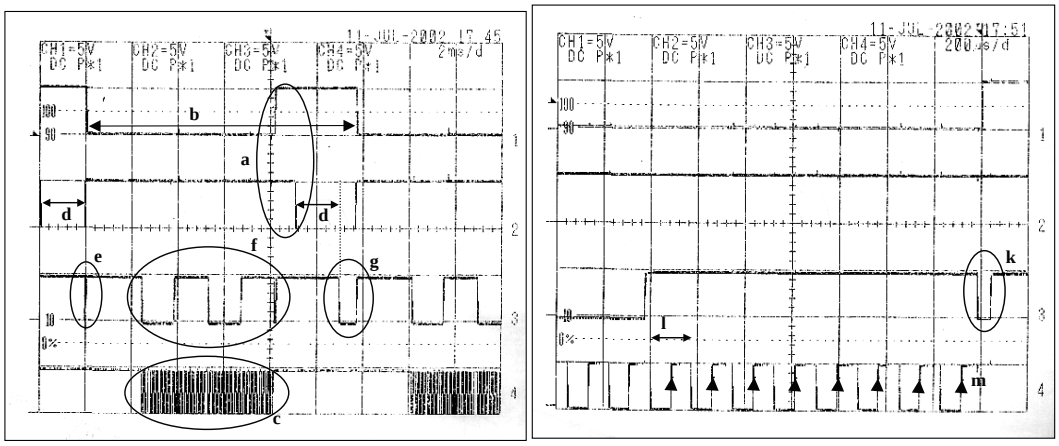
Figure 3. (a) Acquisition rate as a function of the communication rate between the STIM and the NCAP; (b) Actuation rate (actuations/s) as a function of the communication rate between the STIM and the NCAP.

For each of the tests presented above analogous Virtual Instruments (VIs) were built to emulate the NCAP controlling the STIM. These VIs, developed in LabVIEW, use a dedicate STIM driver library that includes the `tii.dll` to communicate with the STIM. These applications have two input variables: the communication rate between the STIM and the NCAP and the address of the channel to be sampled (or actuated). The outputs are the time elapsed to run a cycle with 10^4 acquired samples (or actuations) and the record of any error that may have occurred. This large number of samples was chosen in order to minimize the influence of the non-real time operating system that runs on the PC.

The methodology of the experimental procedure can be summarised in the following steps: (1) The computational load of the PC is minimized by deactivating all PC supplementary programs; (2) the STIM runs autonomously a program to acquire samples from a transducer (or to actuate it); (3) the VI that commands the STIM is started and it runs till the execution ends; (4) the time elapsed and any events that may occur are recorded; (5) steps (3) and (4) is repeated for different communication rates.

In order to evaluate protocol and timing conformity with IEEE 1451.2 standard, the Input Output Enable - NIOE, TRIgger - NTRIG, ACKnowledge - NACK and CloCK – DCLK lines were visualised in an oscilloscope for the two tests described before. The example depicted in Figure 4 is for the data acquisition test. In this figure it is clear that NIOE and NTRIG lines are never simultaneously active (0 V) because, as specified in the standard, it is not allowed to transfer data from and to trigger the STIM at the same time. As it is a data acquisition test, triggering is done before the data frame transmission. Other signal details are evident from the figure: (a) STIM trigger; (b) sampling interval (approximately 12 ms); (c) bit transfer rate between the STIM and the NCAP (6 kbit/s); (d) digitising time for a new sample and update of the STIM

internal state; (e) STIM trigger acknowledge; (f) STIM and NCAP exchange four bytes (NACK changes six times); (g) the same as in (e) but NCAP takes more time to deactivate NTRIG.



(a) (b)

Figure 4. Oscillogram of NIOE(ch.1), NTRIG(ch.2), NACK(ch.3) and DCLK(ch.4) lines for the data acquisition test: (a) dialogue between the STIM and the NCAP, (b) detail for the last byte of the transmitted frame (oscilloscope settings: horizontal sensitivity – 2 ms/div; vertical sensitivity – 5 V/div).

Labels (e) and (g) show a random behaviour in the time the NCAP takes to deactivate the NTRIG line. This is due to the non-real time behaviour of PC's Windows operating system that prevents the NCAP to take control over all the machine resources. Figure 4(b) details the last byte of the data frame to be transmitted: (k) NACK line changes to 0V to acknowledge the last bit of the frame and, when the frame ends, it turns back to 5V; (l) the clock period is approximately 167 μs (6 kbit/s); (m) the eight transitions marked form the last byte of the frame.

Figure 5 is the oscillogram obtained for one actuation test. The meaning of time labelled marks in this Figure is: (a) STIM triggering is done after the data frame transmission; (b) actuation period, less than 10ms; (c) communication rate between the STIM and the NCAP (6kbit/s); (d) time elapsed between one actuation and trigger acknowledge in the STIM; (e) trigger acknowledge; (f) there is a three bytes frame transmission which implies NACK changing four times.

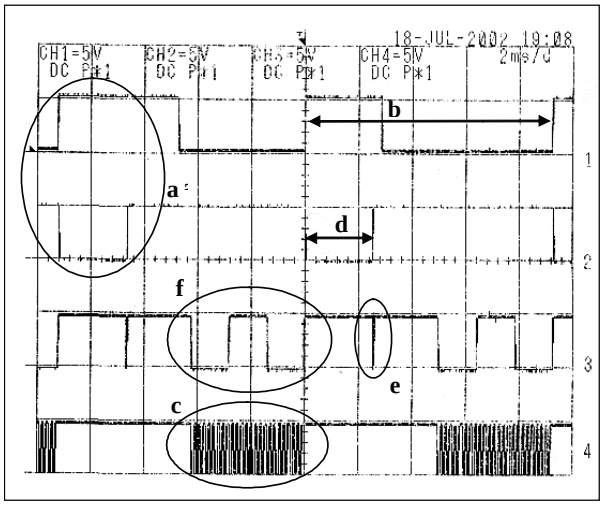


Figure 5. Oscillogram of NIOE(ch.1), NTRIG(ch.2), NACK(ch.3) and DCLK(ch.4) lines for the actuation test (oscilloscope settings: horizontal sensitivity – 2ms/div; vertical sensitivity – 5V/div).

Tests similar to the above described were performed in order to ckeck other timing requirements specified in subclause 6.4 of the IEEE Std.1451.2 for the data transport and trigger lines. As the standard establishes

that all STIMs and NCAPs shall support a common rate of 6 kbit/s, tests were conducted at this rate. For each of the tests reported a TII latency time is changed. Figure 6 presents the oscillograms obtained for the latency times referred in Table 3.

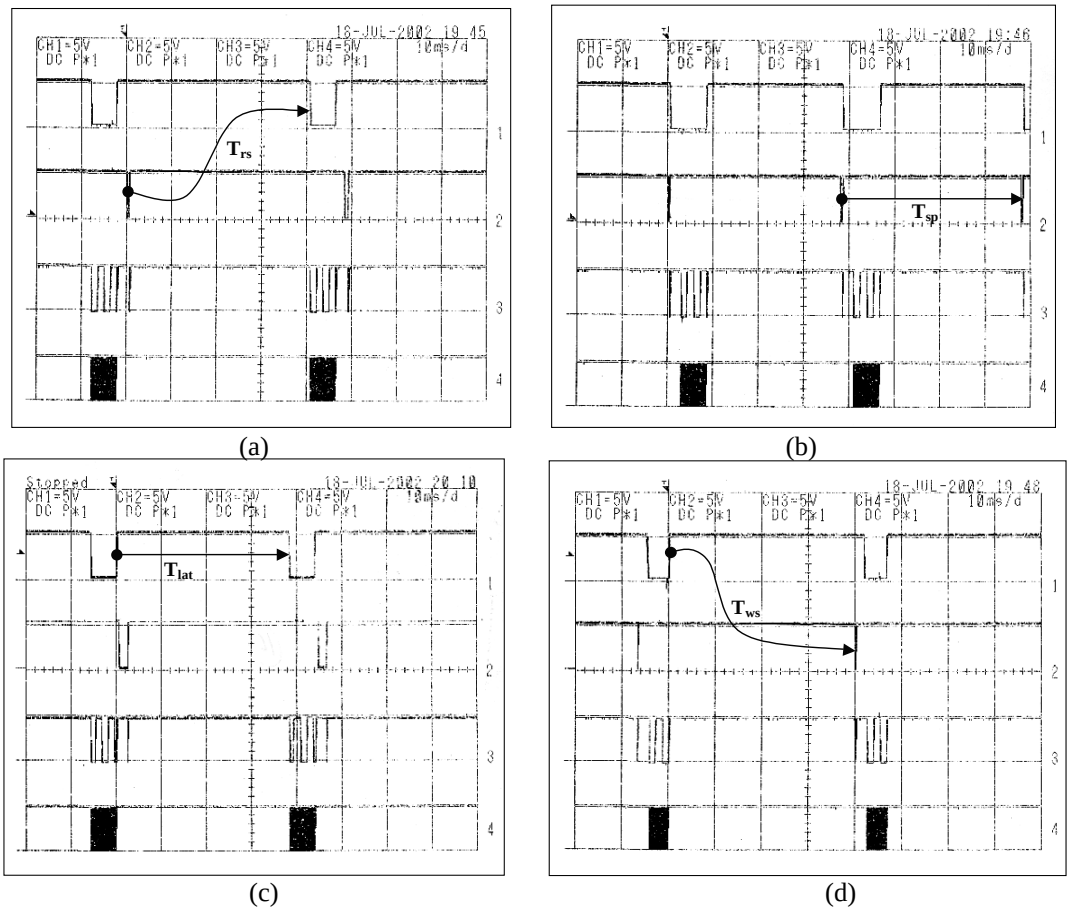


Figure 6. Oscillograms of NIOE(ch.1), NTRIG(ch.2), NACK(ch.3) and DCLK(ch.4) lines for obtained for the latency times referred in Table 3. Oscilloscope settings: horizontal sensitivity – 10 ms/div; vertical sensitivity – 5 V/div.

Figure 6(a) presents the timing relationships between data transfer involving the triggered channel and the triggering of that channel (T_{rs}=40ms is the time between the trigger and the following frame to be transmitted). Figure 6(b) depicts sampling periods (T_{sp}=40ms) and Figure 6(c) shows that frames are timely spaced by T_{lat}=40ms. By negating NIOE a minimum of T_{lat} seconds between data transfer frames the NCAP can ensure that the STIM detects the end of one frame and the beginning of another. Figure 6(d) shows the time required between two actuactions (T_{ws}=40ms).

Table 3. TII latency times and corresponding oscillogram

Latency time (ms)	sampling	Sampling period (T_{sp})	40	0	0	0
		Time to prepare for reading (T_{rs})	0	40	0	0
		Silence time (T_{lat})	0	0	40	0
	acquisition	Time to prepare for writing (T_{ws})	0	0	0	40
Oscillogram depicted in Figure 6			(b)	(a)	(c)	(d)

A test to quantify the error rate in the communication was also accomplished. The idea was to echo a bit pattern several times; if the pattern returned to the NCAP was not equal to the one that was sent, at least one error had occurred. When this test was performed no errors occurred for 10^6 data frames; errors only occurred when other programs were activated in the PC. These errors are generally associated with timeouts detected by the STIM driver software, and are caused by PC's computacional load (e.g. hard disk access).

IV. Conclusions

The Transducer Independent Interface implemented in the developed prototype proved to be fully compliant with the Std. IEEE 1451.2. It proved also to be a robust data transfer solution between the STIM and the NCAP. Even if some errors occur when the Windows operating system is accessing the hard disk, communications resume immediately after deactivating some applications or by increasing timeout intervals.

Some random behaviour in the time the NCAP takes to deactivate the NTRIG line was detected in our tests, as referred in the comments of Figure 4(a). This is caused also by Windows operating system that prevents the NCAP to take control over all the machine resources. All these problems can be solved or minimised by using a real-time operating system or a PC with more powerful hardware resources.

Acknowledgments

This work was supported in part by Portuguese Science and Technology Foundation PRAXIS XXI program POSI SFRH/BPD/11549/2002 and by the Project FCT PNAT/1999/EEI/ 15052. This support is gratefully acknowledged.

References

[1] IEEE Std. 1451.2-1997 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, New York, Sept.98.

[2] IEEE Std. 1451.1-1999 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model, New York, April 2000.

[3] Kang Kee, "IEEE 1451: A Standard in Support of Smart Transducer Interface Networking", Proceedings of the IMTC 2000, pp. 525-528, Baltimore, May 2000.

[4] L. Cámara, O. Ruiz, J. Samitier, "Complete IEEE-1451 Node, STIM and NCAP, Implemented for a CAN Network", Proceedings of the IMTC 2000, pp. 541-545, Baltimore, May 2000.

[5] J. Burch, J. Eidson, B. Hamilton, "The Design of Distributed Measurement Systems Based on IEEE 1451 Standards and Distributed Time Services", Proceedings of the IMTC 2000, pp. 525-528, Baltimore, May 2000.

[6] Kang Lee, "A Synopsis of the IEEE 1451 Standards for Smart Transducer Communication", http://ieee1451.nist.gov.

[7] Helena Ramos, M. Pereira, V. Viegas, O. Postolache, P. Girão, "A Virtual Instrument to Test Smart Transducer Interface", Proceedings of IMTC03, pp. 529-533, Vail, May 2003

[8] Helena Ramos, O. Postolache, M. Pereira, P. Girão, "An Application of the IEEE 1451.1 Correction Engine in an Integrated Sensing Structure", Proceedings of XVII IMEKO Congress, pp.609-613, Dubrovnik, June 2003.

[9] Brian O'Mara, Paul Conway, "Designing an IEEE 1451.2 Compliant Transducer", http://www.sensorsmag.com/articles/08000/46.shtml.

[10] John B. Peatman, "Design with PIC Microcontrollers", Prentice Hall, 1998

[11] "LabVIEW for everyone", Prentice Hall PTR, 1997.