Università degli studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Sviluppo di un editor in LabVIEW per Transducer Electronic Data Sheet (TEDS) conformi allo standard IEEE 1451

Development of a LabVIEW-based editor for Transducer Electronic Data Sheet (TEDS) IEEE-1451 compliant

LAUREANDO:

Federico Menegon - 578227 IF

RELATORE:

Prof.ssa Giada Giorgi

Padova, 27 Settembre 2010

Anno Accademico 2009/2010

Alla mia famiglia
e ad Eleonora.

Sommario

Il lavoro sviluppato in questa tesi riguarda lo sviluppo di un editor grafico per la scrittura e lettura di Transducer Electronic Data Sheet (TEDS) secondo il formato specificato dallo standard IEEE 1451.0 relativo agli Smart Transducer. Il rispetto dei formati imposti dallo standard IEEE 1451.0 garantisce la compatibilità dei TEDS creati con quelli invece redatti per mezzo di altri editor o programmi, evitando in questo modo l'insorgere di errori, come quelli dovuti al disallineamento dei dati in fase di lettura.

L'editor grafico è stato realizzato utilizzando come ambiente di sviluppo LabVIEW; questo ha permesso di creare facilmente una interfaccia utente intuitiva ed in grado di rispecchiare la struttura prevista dallo standard IEEE 1451.0 per i TEDS. Infine, il codice è stato sviluppato avendo, tra i vari obiettivi, quello della flessibilità. Questo ha consentito di creare un programma facilmente estendibile, in modo da poter inserire in futuro il codice necessario alla gestione di altri TEDS oltre a quelli attualmente implementati che sono rispettivamente: il Meta TEDS, il TransducerChannel TEDS ed il Calibration TEDS. Gli stessi requisiti di flessibilità sono stati considerati anche nello sviluppo dell'interfaccia grafica, ovvero del Pannello Frontale realizzato in LabVIEW, sempre al fine di consentire l'inserimento di altri tipi di TEDS senza avere la necessità di apportare modifiche al codice.

Indice

Capitolo 1

Introduzione allo standard IEEE 1451.0

Sviluppata nel corso degli ultimi dieci anni, dall'*Institute of Electrical and Electronics Engineers* (IEEE), la famiglia di standard IEEE 1451 descrive le caratteristiche e le modalità di accesso in rete a trasduttori differenti attraverso un insieme di procedure e comandi standardizzati per permettere ai trasduttori di connettersi a blocchi di elaborazione, sistemi di strumentazione e reti di controllo senza incorrere a problemi di compatibilità dovuti alle differenti scelte adottate dai costruttori di trasduttori per l'interfacciamento con quest'ultimi. Gli standard proposti all'interno della famiglia IEEE 1451 attualmente sono otto:

- **IEEE 1451.0**: definisce un insieme di basi e funzionalità comuni per tutti i membri della famiglia IEEE 1451, inoltre definisce comandi e *Transducer Electronic Data Sheet* (TEDS) per i trasduttori che adottano tale standard. Le funzionalità definite riguardano la configurazione ed il controllo dei TIM ed inoltre la scrittura e lettura dei TEDS oltre alla struttura di alcuni di essi;

- **IEEE 1451.1**: definisce il modello fisico e logico di un trasduttore intelligente, *smart trasducer*, descrivendo le sue strutture dati e il suo modo di funzionamento;

- **IEEE 1451.2**: definisce l'interfaccia di comunicazione e i TEDS per configurazioni *point-to-point*. Lo standard permette l'utilizzo di diffuse interfaccie

seriali come la UART (*Universal Asynchronous Receiver-Trasmitter*) e la USI (*Universal Serial Interface*);

- **IEEE 1451.3**: definisce l'interfaccia di comunicazione ed il TEDS per protocolli di comunicazione *multi-drop*;
- **IEEE 1451.4**: definisce una interfaccia mista per trasduttori analogici con modi di operare sia analogici e sia digitali. Esso descrive il modo di gestire i sensori tradizionali e come integrarli nella architettura di sistema dei sensori prevista da questa serie di norme;
- **IEEE 1451.5**: definisce l'interfaccia di comunicazione ed il TEDS per interfacce di comunicazione *wireless*, come ad esempio 802.11 (WiFi), 802.15 (Bluetooth), 802.15.4 (ZigBee) e la 6LowPAN. La parte relativa al trasduttore viene rinominata dallo standard *Wireless Transducer Interface Module* (WTIM);
- **IEEE 1451.6**: definisce l'interfaccia di comunicazione ed il TEDS per interfacce basate sulla rete ad alta velocità CANopen;
- **IEEE 1451.7**: definisce l'interfaccia di comunicazione ed il TEDS per sistemi RFID.

Lo standard IEEE 1451.0 introduce il concetto di trasduttore intelligente, detto anche *smart transducer*. Un trasduttore di questo tipo, oltre a possedere le normali funzionalità necessarie a fornire una corretta rappresentazione della quantità fisica in esame, possiede funzionalità che permettono di semplificare l'integrazione del trasduttore con una rete di sensori. Più precisamente lo *smart transducer* deve possedere le seguenti caratteristiche:

- auto-identificazione del trasduttore (*self-identification*);
- capacità di fornire una descrizione di se stesso (*self-description*);
- auto-diagnosi (*self-diagnosis*);

- auto-taratura (*self-calibration*);
- conoscenza della propria localizzazione e del suo riferimento temporale (*location-awareness* e *time-awareness*);
- capacità di elaborazione dei dati e di fornirli secondo uno formato standard (*data processing* e *standard-based data format*);
- capacità di generare segnali di allarme/notifica (*alert notification*);
- capacità di scambiare informazioni secondo un dato protocollo di comunicazione.

L'adozione dello standard IEEE 1451.0 da parte dei trasduttori consente di godere dei seguenti vantaggi:

- possibilità di definire una rete di trasduttori non vincolata a specifiche case costruttrici o standard proprietari;
- possibilità di definire un *data sheet* elettronico standardizzato (TEDS) contenete tutte le informazioni necessarie per gestire un trasduttore;
- adozione di un modello comune per gestire dati relativi al trasduttore, per le varie operazioni di controllo, configurazione e taratura e per le temporizzazioni;
- riduzione della possibilità di errore nella configurazione dei dispositivi stessi;
- possibilità di installare, aggiornare, sostituire o rimuovere i trasduttori nella rete con una modalità di tipo *plug-and-play* che richiede al gestore della rete uno sforzo minimo;
- possibilità di accedere ai trasduttori sia per mezzo di collegamenti via cavo, sia in modalità *wireless*, attraverso una vasta scelta di mezzi fisici.

Tali vantaggi permettono così di raggiungere lo scopo principale di questo standard, ovvero garantire l'interoperabilità fra i diversi membri della famiglia di standard IEEE 1451.

Il termine *trasduttore* si riferisce a quella classe di dispositivi che convertono una grandezza fisica da un dominio ad un altro e comprende perciò sia dispositivi di misura che permettono l'acquisizione di informazioni utili, indicati con il nome di *sensori*, sia dispositivi che, nel verso opposto, realizzano azioni di controllo e che generalmente vengono indicati con il nome di *attuatori*.

Il termine *Transducer Interface Module* (TIM) indica quel dispositivo che si occupa della gestione dei trasduttori, siano essi sensori oppure attuatori, presenti all'interno del dispositivo stesso, assieme a blocchi di elaborazione, necessari alla conversione e al condizionamento dei segnali, collegati ai trasduttori tramite *TransducerChannel*, ovvero canali di trasduzione che oltre a comprendere il trasduttore fisico comprendono anche tutta l'elettronica presente nel percorso che connette il trasduttore stesso al blocco di comunicazione, come ad esempio blocchi di amplificazione/attenuazione, filtraggio, conversione analogico/digitale ed altri circuiti di condizionamento. Grazie ai *TransducerChannel* è possibile individuare ogni singolo trasduttore presente all'interno del TIM.

Il termine *Network Capable Application Processor* (NCAP) indica un nodo di rete con funzionalità proprie di un nodo di comunicazione. Un NCAP si occupa della gestione della rete, operando da *gateway* tra l'utente della rete e i TIM con il quale si interfaccia per mezzo dell'interfaccia *Transducer Indipendent Interface* (TII), che può essere sia *wired* che *wireless*.

In Figura 1.1 è stato riportato lo schema di come i trasduttori, canali di trasduzione, TIM e NCAP sono collegati fra di loro.

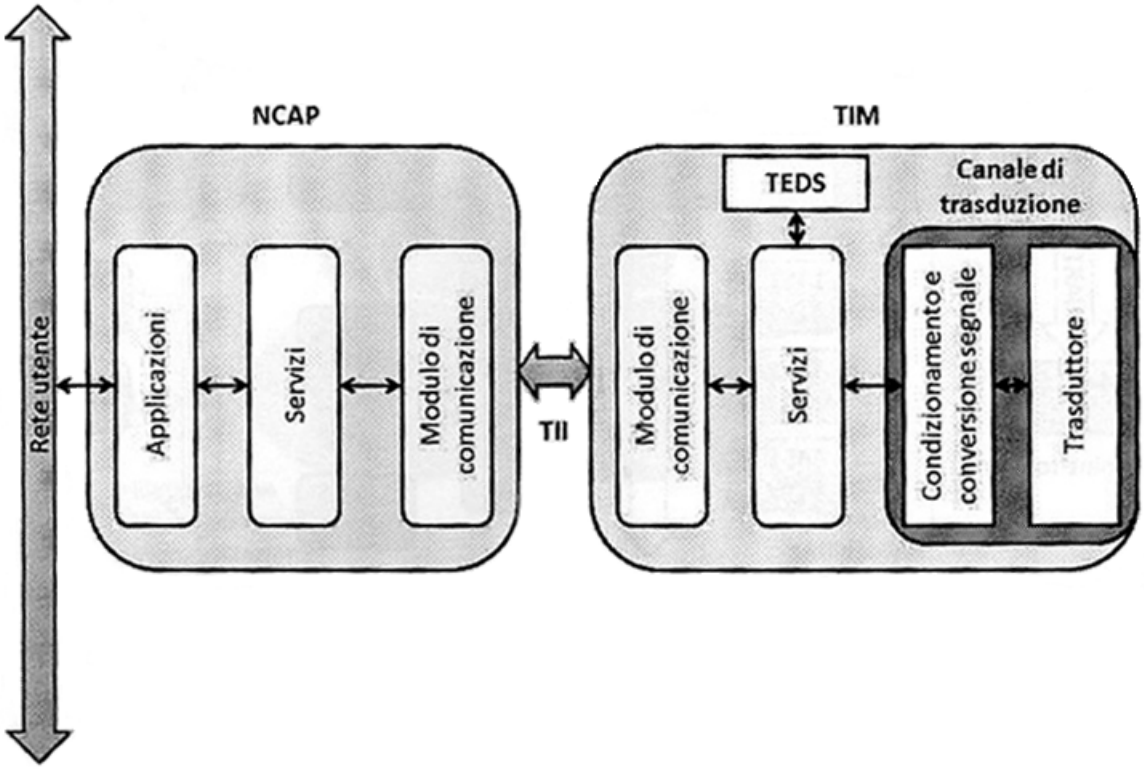

Figura 1.1: Schema di collegamento fra Transducer e rete utente

Lo standard IEEE 1451.0 definisce tre tipi di canali di trasduzione:

- **sensore**: dispositivo per la misurazione di un parametro della grandezza fisica applicata in ingresso. Un termometro è un ottimo esempio di sensore in quanto converte la temperatura di ingresso nel corrispondente valore numerico.

- **sensore ad eventi**: dispositivo per la rilevazione di un cambiamento di stato nella grandezza applicata in ingresso. Un comparatore è un sensore ad eventi in quanto confronta la grandezza fisica applicata in ingresso con una grandezza di riferimento e in uscita fornisce un livello logico basso se l'ingresso è superiore alla grandezza di riferimento, oppure valore logico alto se vale l'opposto.

- **attuatore**: dispositivo che produce un valore oppure un'azione in uscita in funzione al dato fornito in ingresso. Il termostato costituisce un attuatore che regola in modo opportuno un sistema di condizionamento a seconda del valore numerico ricevuto in ingresso.

Le basi di questa tesina sono incentrate sullo standard IEEE 1451.0 ed in particolare sulla definizione che esso fornisce della struttura dei Meta TEDS, TransducerChannel TEDS e Calibration TEDS, tre diverse classi di TEDS, ognuna specifica per un particolare tipo di trasduttore. I *Transducer Electronic Data Sheet*, meglio indicati con il nome di TEDS, rappresentano dei *data sheet* in formato elettronico specifici per qualunque tipologia di trasduttore. Essi memorizzano le informazioni riguardanti quel determinato trasduttore, come ad esempio il suo identificativo, il *range* di misura a cui può operare, l'accuratezza nelle misurazioni, i dati di taratura ecc, in modo del tutto simile alle informazioni che vengono scritte all'interno dei tradizionali *data sheet* forniti dai costruttori.

Capitolo 2

Transducer Electronic Data Sheet

I *Transducer Electronic Data Sheet*, come anticipato precedentemente, contengono le informazioni riguardanti il funzionamento di un canale di trasduzione. Tali informazioni vengono suddivise in blocchi e campi; l'ordine e la struttura relativa a questi campi viene definita all'interno della famiglia di standard IEEE 1451. Lo standard IEEE 1451.0 vuole che i TEDS vengano memorizzati in memorie non volatili o embedded, come ad esempio memorie EEPROM, situate all'interno dei TIM e accessibili da strumenti di misura o di controllo che li utilizzano per interfacciarsi con i trasduttori situati all'interno del TIM. Esistono però situazioni in cui tutto ciò non è possibile, per questo motivo lo standard permette di definire TEDS al di fuori dei TIM. Quando i TEDS vengono memorizzati al di fuori del TIM, come ad esempio in file binari, vengono indicati con il nome di *virtual TEDS*; questo è il caso dei produttori di trasduttori i quali non sapendo in quali TIM verranno installati, si limitano solamente a fornire il relativo *virtual TEDS*. Sarà poi responsabilità dell'utilizzatore associarlo al relativo TIM in modo da garantire sempre la disponibilità delle informazioni contenute all'interno del *virtual TEDS*. I TEDS di tipo virtuale estendono in questo modo i TEDS di tipo standardizzato per sensori e altre applicazioni, dove la memoria embedded potrebbe non essere disponibile.

La famiglia di standard IEEE 1451 definisce 15 classi di TEDS, ognuna per uno specifico compito; esse sono riportate in Tabella 2.1.

Tabella 2.1: Lista delle classi di TEDS

TEDS access code	**TEDS name attribute**	**Required/optional**
1	MetaTEDS	Required
2	MetaIdTEDS	Optional
3	ChanTEDS	Required
4	ChanIdTEDS	Optional
5	CalTEDS	Optional
6	CalIdTEDS	Optional
7	EUASTEDS	Required
8	FreqRespTEDS	Optional
9	TransferTEDS	Optional
10	CommandTEDS	Optional
11	TitleTEDS	Optional
12	XdcrName	Required
13	PHYTEDS	Required
14	GeoLocTEDS	Optional
15	UnitsExtension	Optional

Dalla Tabella 2.1 è possibile notare che ad ogni classe di TEDS è associato un attributo che indica se la classe è *Richiesta* oppure *Opzionale*, se *Richiesta* significa che tutti i TIM devono specificare obbligatoriamente al loro interno quel particolare TEDS. Le classi di TEDS sviluppate nell'editor in LabVIEW sono tre ovvero:

- MetaTEDS
- ChanTEDS
- CalTEDS

Tutte e tre le classi appartengono al sottoinsieme costituito da cosiddetti *binary TEDS*, ovvero TEDS scritti all'interno di file in codice binario; esiste poi un altro sottoinsieme, complementare al primo, che contiene le classi di TEDS di tipo *text-based*, le quali, a differenza dei *binary TEDS*, forniscono un meccanismo per i costruttori che permette l'inserimento di informazioni testuali all'interno degli *smart transducer*.

2.1 Formato generale dei TEDS

Come precedentemente anticipato, lo standard IEEE 1451.0 definisce l'esatta struttura che compone ogni classe di TEDS, questa struttura è uguale per tutte le classi e tipi di TEDS e si compone delle seguenti tre parti:

1. **TEDS length**: ogni TEDS come prima informazione memorizza, in una variabile di tipo UInt32 (4 byte), il numero totale di byte occupati dalle successive due parti del formato generale dei TEDS.

2. **Data block**: rappresenta una struttura gerarchica, diversa per ciascuna classe di TEDS, all'interno della quale vengono immagazzinate tutte le informazioni relative al trasduttore in esame. Ogni informazione viene vista come un singolo campo della struttura con un preciso identificativo (*ID field*) e posizione all'interno della struttura stessa.

3. **Checksum**: rappresenta una variabile di tipo UInt16 (2 byte) che serve a garantire la correttezza del TEDS in fase di lettura. Il calcolo del checksum è composto dalla somma di tutti i byte precedentemente immessi all'interno del file, inclusi anche quelli del campo *TEDS length*, successivamente si effettua il complemento ad uno del risultato ottenendo il valore di checksum del TEDS. È importante sottolineare che nel caso si dovesse verificare un overflow durante la somma di tutti i byte che compongono il TEDS, ad eccezione ovviamente degli ultimi due destinati ad ospitare il checksum, la regola da seguire è quella di ignorare l'overflow e procedere con la somma, una volta completata si considerano solo i 16 bit meno significativi. La seguente equazione riassume l'operazione necessaria al calcolo del checksum; la differenza tra il valore esadecimale 0xFFFF, corrispondente al numero 65 535, e la somma dei byte equivale ad effettuare il complemento ad uno di quest'ultima.

$$\text{checksum} = \text{0xFFFF} - \sum_{i=1}^{TotaleByte-2} \text{TEDSByte}(i)$$

In Tabella 2.2 viene rappresentato graficamente il formato generale con cui devono essere memorizzati i TEDS all'interno di memorie EEPROM oppure di file binari.

Tabella 2.2: Formato generale dei TEDS

Field	**Description**	**Type**	**# byte**
_	TEDS length	UInt32	4
1 to N	Data Block	Variable	Variable
_	Checksum	UInt16	2

Lo standard IEEE 1451.0 oltre a definire il formato generale dei TEDS specifica anche la struttura con la quale memorizzare ogni singolo campo all'interno del *Data Block*; questa struttura viene indicata con il nome di *TLV Structure* (Type/Length/Value) ed è composta da tre parametri:

1. **Type**: composto da un byte, questo parametro contiene un numero che funge da identificativo per il campo del TEDS a cui si riferisce.

2. **Length**: composto anch'esso da un byte, questo parametro specifica il numero di byte che il successivo parametro *Value* della struttura TLV andrà ad occupare all'interno del TEDS.

3. **Value**: questo parametro contiene l'informazione vera e propria, la sua grandezza può variare in base al tipo di campo e informazione a cui la struttura TLV si riferisce.

$$\text{Field} \Rightarrow \begin{cases} \textbf{T}\text{ype} \\ \textbf{L}\text{ength} \\ \textbf{V}\text{alue} \end{cases}$$

Lo standard infine precisa che, quando il parametro *Value* occupa due o più byte, si deve scrivere all'interno del TEDS prima il byte più significativo, poi i successivi fino a scrivere per ultimo il byte meno significativo, bisogna quindi adottare la notazione Big-Endian per la memorizzazione delle informazioni occupanti due o più byte. In Figura 2.1 viene riportato un esempio della notazione Big-Endian.

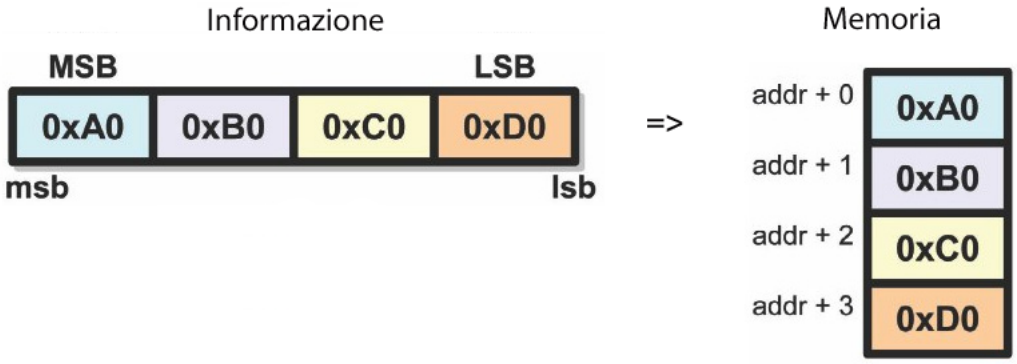

Figura 2.1: Notazione Big-Endian

2.2 Tipi di dato

Lo standard IEEE 1451.0 oltre a definire il formato generale dei TEDS e la struttura con cui memorizzare i campi al suo interno, definisce anche i tipi di dato utilizzati nei TEDS per la memorizzazione dell'informazione contenuta nel parametro *Value* della struttura TLV. Si elencano di seguito i tipi di dato previsti dallo standard e necessari all'interno dei MetaTEDS, ChanTEDS e CalTEDS:

Unsigned 8 bit integer

Simbolo: UInt8

Grandezza: 1 byte

Descrizione: Questo tipo di dati rappresenta un intero positivo da 0 a 255.

Unsigned 16 bit integer

Simbolo: UInt16

Grandezza: 2 byte

Descrizione: Questo tipo di dati rappresenta un intero positivo da 0 a 65 535.

Unsigned 32 bit integer

Simbolo: UInt32

Grandezza: 4 byte

Descrizione: Questo tipo di dati rappresenta un intero positivo da 0 a 4 294 967 295.

Signed 8 bit integer

Simbolo: Int8

Grandezza: 1 byte

Descrizione: Questo tipo di dati rappresenta un numero intero con segno che può assumere valori da -128 fino a 127.

Signed 16 bit integer

Simbolo: Int16

Grandezza: 2 byte

Descrizione: Questo tipo di dati rappresenta un numero intero con segno che può assumere valori da –32 768 fino a 32 767.

Signed 32 bit integer

Simbolo: Int32

Grandezza: 4 byte

Descrizione: Questo tipo di dati rappresenta un intero con segno da -2 147 483 648 a 2 147 483 647.

Single-precision real

Simbolo: Float32

Grandezza: 4 byte

Descrizione: Questo tipo di dati rappresenta un numero in virgola mobile a singola precisione la cui codifica è specificata dallo standard IEEE 754-1985.

Double-precision real

Simbolo: Float64

Grandezza: 8 byte

Descrizione: Questo tipo di dati rappresenta un numero in virgola mobile a doppia precisione la cui codifica è specificata dallo standard IEEE 754-1985.

TimeRepresentation

Grandezza: 8 byte

Descrizione: *TimeRepresentation* è una classe che definisce la rappresentazione del tempo. La struttura di questo tipo di dato è composta da

due parole di 4 byte ciascuna; la prima costituisce una variabile di tipo UInt32 che contiene il numero di secondi trascorsi a partire dalle ore 0:00:00 del 1 gennaio 1970, la seconda invece è anch'essa una variabile di tipo UInt32 il cui bit più significativo è interpretato come segno e i restanti 31 bit rappresentano invece il numero di nanosecondi (da 0 a 999 999 999) da sommare ai secondi prima di applicare il segno. Tale classe si suddivide in due sottoclassi:

- *TimeDuration*, utilizzata per specificare un intervallo di tempo piuttosto che un valore di tempo.
- *TimeInstance*, utilizzata per rappresentare un valore di tempo piuttosto che una durata temporale.

Physical Units

Simbolo: UNITS

Grandezza: 11 byte

Descrizione: *Physical Units* è una classe che codifica una qualunque unità di misura come il prodotto delle unità fondamentali del *Sistema Internazionale* (SI). Questa struttura codifica solamente gli esponenti di ciascuna unità, il prodotto resta implicito e alle unità non presenti viene assegnato esponente zero. Tale variabile deve essere vista come un insieme di undici variabili di tipo UInt8 che sono *Interpretation*, *Radians*, *Steradians*, *Meters*, *Kilograms*, *Seconds*, *Amperes*, *Kelvins*, *Moles*, *Candelas*, *Units Extension TEDS Access Code*.

Universal unique identification

Simbolo: UUID

Grandezza: 10 byte

Descrizione: UUID è un tipo di variabile che rappresenta l'identificativo di ciascun TIM il cui valore è univoco in tutto il pianeta. Tale va-

riabile è costituita da quattro parametri che sono *location* (42 bit), *manufacturer's* (4 bit), *year* (12 bit), *time* (22 bit).

Array of 16 bit unsigned integers

Simbolo: UInt16Array

Grandezza: variabile

Descrizione: Questo tipo di dati comprende un numero arbitrario di variabili di tipo UInt16.

Array of single-precision real numbers

Simbolo: Float32Array

Grandezza: variabile

Descrizione: Questo tipo di dati comprende un numero arbitrario di variabili di tipo Float32.

2.3 Classi di TEDS

Come già anticipato precedentemente la struttura gerarchica necessaria per scrivere un TEDS è definita dalla famiglia di standard IEEE 1451, in particolare lo standard IEEE 1451.0 definisce il formato per Meta TEDS, TransducerChannel TEDS e Calibration TEDS, Frequency Response TEDS, Transfer Function TEDS e infine Text-based TEDS. Di seguito verranno specificate solo le prime tre classi di TEDS che sono state implementate nell'editor sviluppato con LabVIEW.

2.3.1 Meta TEDS

Il MetaTEDS costituisce la prima classe di TEDS. Esso descrive le relazioni che intercorrono tra i canali di trasduzione di uno stesso TIM. Le funzioni principali del MetaTEDS sono quelle di permettere al TIM di impostare i valori di time-out consentendo al suo modulo di comunicazione di determinare se il relativo NCAP non sta rispondendo e quella di mettere a disposizione del livello di interfaccia tutte le informazioni necessarie per accedere a qualsiasi canale di trasduzione. La struttura precisa dei MetaTEDS viene indicata in Tabella 2.3:

Tabella 2.3: Struttura MetaTEDS

Field Type	**Field Name**	**Data Type**	**# byte**
Length	_	UInt32	4
0-2	[Reserved]	_	
3	TEDSID	UInt8	4
4	UUID	UUID	10
5-9	[Reserved]	_	
10	OHoldOff	Float32	4
11	SHoldOff	Float32	4
12	TestTime	Float32	4
13	MaxChan	UInt16	2
14	CGroup	_	_
20 21	GrpType MemList	UInt8 array di UInt16	1 NTc
15	VGroup	_	_
20 21	GrpType MemList	UInt8 array di UInt16	1 NTv

Tabella 2.3: continua nella prossima pagina

Tabella 2.3: continua dalla pagina precedente

Field Type	Field Name	Data Type	# byte
16	GeoLoc	_	_
24	LocEnum	UInt8	1
20	GrpType	UInt8	1
21	MemList	array di UInt16	NTv
17	Proxies	_	_
22	ChanNum	UInt16	2
23	Organiz	UInt8	1
21	MemList	array di UInt16	NTp
18-19	[Reserved]	_	
25-127	[Reserved]	_	
128-255	[Open to manufactures]	_	
Checksum	_	UInt16	2

Per una migliore comprensione della struttura del MetaTEDS e di come essa è organizzata, si riporta in Figura 2.2 la sua rappresentazione grafica.

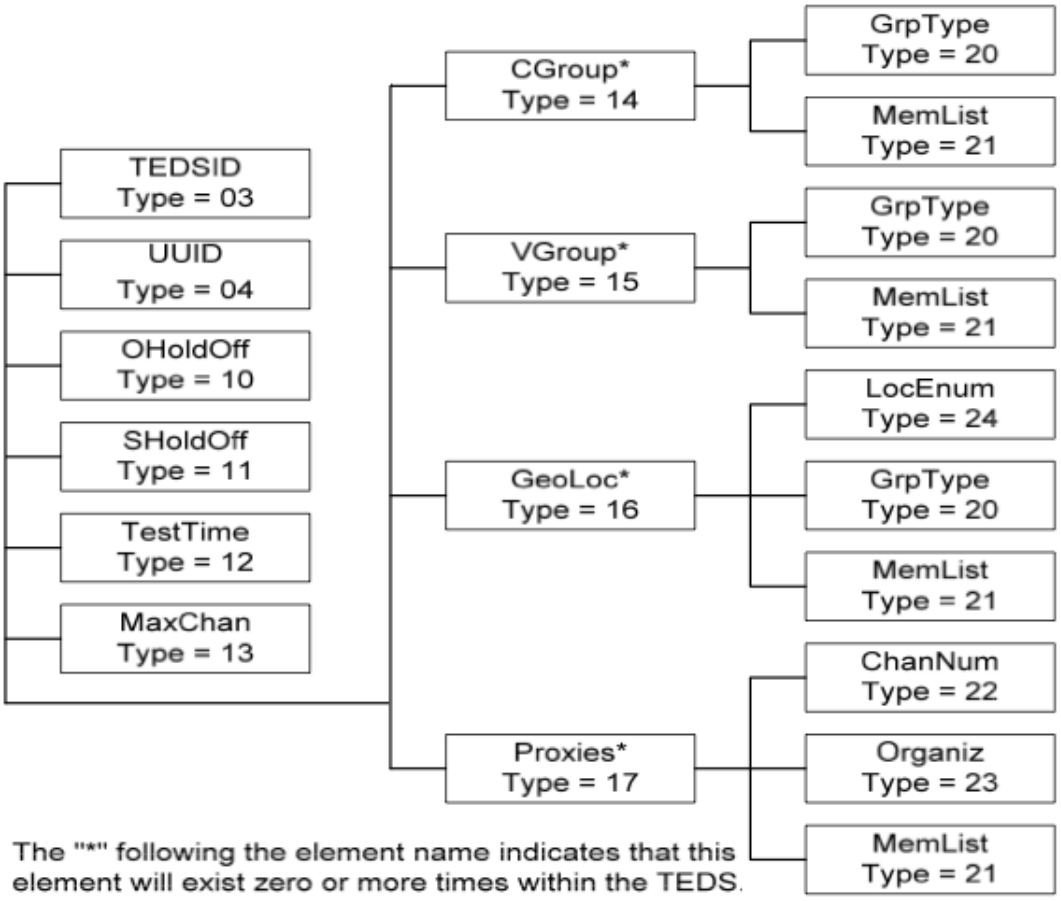

Figura 2.2: Struttura del MetaTEDS

TEDS Identification header - TEDSID

Il campo TEDSID è obbligatorio per qualunque tipo di TEDS ed è sempre il primo campo all'interno della struttura *Data Block* di ciascun TEDS. Tale campo è composto dai seguenti quattro parametri:

- **Family**, il quale occupa il primo byte del campo TEDSID e vale sempre 0 per indicare che il MetaTEDS appartiene alla famiglia zero dello standard IEEE 1451 ovvero alla standard IEEE 1451.0.

- **Class**, il quale occupa il secondo byte del campo TEDSID e indica il numero di *TEDS access code*[1] associato alla classe di appartenenza del TEDS; nel caso del MetaTEDS esso vale sempre 1.

- **Version**, il quale occupa il terzo byte del campo TEDSID e indica la versione del TEDS; il numero 0 sta ad indicare che è ancora in fase prototipale oppure che non è conforme allo standard; il numero 1 corrisponde alla versione originale di questo standard, i successivi numeri sono riservati invece a *release* future dello standard.

- **Tuple Length**, il quale occupa il quarto e ultimo byte del campo TEDSID e solitamente vale 1 per significare che ci sono 255 o meno byte complessivamente all'interno dei campi *Value* di ciascuna struttura TLV del TEDS.

Globally unique identifier - UUID

Il campo UUID rappresenta l'identificativo univoco di ciascun TIM. Esso occupa 10 byte ed è composto dai seguenti quattro parametri:

- **Location**, tale parametro rappresenta la posizione geografica attuale del TIM ed è formato da 42 bit. Il bit più significativo se asserito indica il Nord, se deasserito indica il Sud. I seguenti 20 bit più significativi indicano

[1] Vedi Tabella 2.1

la latitudine della posizione geografica. Il seguente bit più significativo se asserito indica l'Est, se deasserito indica l'Ovest. I rimanenti 20 bit indicano la longitudine della posizione geografica.

Tabella 2.4: Struttura parametro Location del campo UUID

1° bit MSB	**2° - 21° bit**	**22° bit**	**23° - 42 bit LSB**
North/West	**Latitude**	**Est/West**	**Longitude**

- **Manufacture**, composto da 4 bit, tale parametro identifica il costruttore in modo tale che la coppia di parametri *Location* e *Manufacture* possa identificare univocamente un costruttore di TIM.

- **Year**, composto da 12 bit, questo parametro indica l'anno corrente.

- **Time**, composto dagli ultimi 22 bit del campo UUID, rappresenta i secondi trascorsi a partire dall'inizio dell'anno; tale parametro viene scelto dal costruttore in maniera tale che la tupla composta dai parametri *Location*, *Manufacture*, *Year* e *Time* sia unica al mondo.

Operational time-out - OHoldOff

Questo campo contiene l'intervallo di tempo massimo, espresso in secondi, oltre il quale il ritardo tra la ricezione di un comando e la conseguente risposta che il dispositivo deve fornire, comporta la non riuscita della operazione. Questo tempo deve essere scelto più lungo del tempo necessario per cominciare a trasmettere una risposta nel peggior caso possibile, ovvero nel caso di ricezione del comando che richiede il più lungo tempo di esecuzione possibile, oppure maggiore del tempo necessario al dispositivo per prepararsi ad accettare il comando successivo quando non è necessario fornire nessuna risposta al comando ricevuto precedentemente. Questo periodo è utilizzato per la maggior parte dei comandi la cui esecuzione richiede un tempo inferiore a due cicli operativi e viene utilizzato per individuare i TIM che non rispondono ai comandi.

Slow access time-out - SHoldOff

Questo campo contiene l'intervallo di tempo massimo, espresso in secondi, oltre il quale il ritardo tra la ricezione di un comando e la conseguente risposta che il dispositivo deve fornire, comporta la non riuscita della operazione. Questo tempo deve essere scelto più lungo del tempo necessario per cominciare a trasmettere una risposta nel peggior caso possibile, ovvero nel caso di ricezione del comando che richiede il più lungo tempo di esecuzione possibile, oppure maggiore del tempo necessario al dispositivo per prepararsi ad accettare il comando successivo quando non è necessario fornire nessuna risposta al comando ricevuto precedentemente. Questo periodo è utilizzato per quei comandi che si sa richiederanno un lungo tempo di esecuzione, come ad esempio i comandi per la scrittura all'interno di una memoria non volatile, e viene utilizzato per individuare i TIM che non rispondono ai comandi.

Transducer module self-test time requirement - TestTime

Questo campo contiene il tempo massimo, espresso in secondi, richiesto per eseguire il *self-test*. Se il *self-test* non è implementato il campo deve valere 0. Il *self-test* corrisponde ad una procedura di auto-diagnostica che il TIM può fornire all'utilizzatore.

Number of implemented TransducerChannels - MaxChan

Questo campo contiene il numero di *TransducerChannels* implementati nel TIM. I *TransducerChannels* implementati devono essere numerati partendo dal numero uno e assicurandosi di non tralasciarne nessuno.

ControlGroups - CGroups

Lo standard IEEE 1451.0 permette di raggruppare più canali di trasduzione in relazione tra loro, questo campo risulta quindi utile a quei TIM multi-canale che necessitano di descrivere le relazioni tra i vari canali di trasduzione. Il campo *CGroup* definisce i gruppi di controllo, presenti all'interno del TIM, dove vi sono un

canale di trasduzione principale ed un insieme di canali di trasduzione complementari che forniscono informazioni aggiuntive sul funzionamento di quello principale oppure sono utilizzati per controllare alcuni suoi aspetti. Un esempio di gruppo di controllo si ha quando un attuatore viene utilizzato per impostare il valore di riferimento di un sensore analogico ad eventi. Tale campo rappresenta l'insieme di due sottocampi ovvero *GroupType* e *MemList*.

GroupType contiene un *enumeration*, ossia un valore numerico a cui viene associato un preciso significato, che può assumere un valore compreso tra 1 e 9.

MemList è costituito da un array di UInt16 di grandezza variabile contenente la lista di indirizzi dei canali di trasduzione, appartenenti al gruppo di controllo, con ordinamento predefinito. Significato e ordine dei valori di questi due sottocampi sono specificati in Tabella 2.5.

È molto importante sottolineare che possono esistere zero o più occorrenze del campo *CGroup*, ognuna con un valore del sottocampo *GroupType* differente.

Tabella 2.5: Definizione dei sottocampi GroupType e MemList per il campo CGroup

Group Type	**Member list order**	**Description**
1	1	Analog event sensor TransducerChannel.
	2	Analog input sensor TransducerChannel that measures the input value for the same input as the member 1 event sensor provides the state.
	3	Upper threshold embedded actuator TransducerChannel.
	4	Hysteresis embedded actuator TransducerChannel.
2	1	Sensor TransducerChannel (any type).
	2	High-pass filter embedded actuator TransducerChannel.
	3	Low-pass filter embedded actuator TransducerChannel.
	4	Scale factor embedded actuator TransducerChannel.
3	1	TransducerChannel (any type).
	2	Sample interval embedded actuator TransducerChannel.
4	1	Digital event sensor TransducerChannel.
	2	Digital input sensor TransducerChannel that measures the input value for the same input as the member 1 event sensor provides the state.
	3	Event pattern embedded actuator TransducerChannel.

Tabella 2.5: continua nella prossima pagina

Tabella 2.5: continua dalla pagina precedente

Group Type	**Member list order**	**Description**
5	1 2	Time interval sensor TransducerChannel. TransducerChannel number of the transducer that causes the output of the time interval sensor to be latched.
6	1 2	TimeInstance sensor TransducerChannel. TransducerChannel number of the transducer that causes the output of the TimeInstance sensor to be latched.
7	1 2-N	TransducerChannel number of an event sensor used to trigger other transducers. The remaining N entries give a list of TransducerChannels triggered by the event.
8	1 2	TransducerChannel (any type). Embedded time delay actuator TransducerChannel.
9	1 2	Location sensor TransducerChannel. TransducerChannel number of the transducer that causes the output of the location sensor to be latched.

VectorGroups - VGroups

Lo standard IEEE 1451.0 permette di raggruppare più canali di trasduzione in relazione tra loro, questo campo risulta quindi utile a quei TIM multi-canale che necessitano di descrivere le relazioni tra i vari canali di trasduzione. Il campo *VGroup* definisce i gruppi vettoriali, presenti all'interno del TIM, dove sono presenti gruppi di canali di trasduzione che però non sono legati fra di loro da una relazione di tipo gerarchica. Un esempio di gruppo di vettoriale è costituito da un accelerometro triassiale che comprende un sensore per ognuno dei tre assi X, Y e Z. Tale campo rappresenta l'insieme di due sottocampi ovvero *GroupType* e *MemList*.

GroupType contiene un *enumeration* che può assumere un valore compreso tra 1 e 7.

MemList è costituito da un array di UInt16 di grandezza prefissata contenente la lista di indirizzi dei canali di trasduzione, appartenenti al gruppo vettoriale, con

ordinamento predefinito. Significato e ordine dei valori di questi due sottocampi sono specificati in Tabella 2.6.

È molto importante sottolineare che possono esistere zero o più occorrenze del campo *VGroup*, ognuna con un valore del sottocampo *GroupType* differente.

Tabella 2.6: Definizione dei sottocampi GroupType e MemList per il campo VGroup

Group Type	**Member list order**	**Description**
1	1	x component of a right-hand rectangular spatial vector.
	2	y component of a right-hand rectangular spatial vector.
	3	z component of a right-hand rectangular spatial vector.
2	1	ρ component of a right-hand cylindrical spatial vector.
	2	ϕ component of a right-hand cylindrical spatial vector.
	3	z component of a right-hand cylindrical spatial vector.
3	1	r component of a right-hand spherical spatial vector.
	2	θ component of a right-hand spherical spatial vector.
	3	ϕ component of a right-hand spherical spatial vector.
4	1	Latitude component of a planetary coordinate system.
	2	Longitude component of a planetary coordinate system.
	3	Altitude component of a planetary coordinate system.
5	1	In-phase component of a two-dimensional vector.
	2	Quadrature component of a two-dimensional vector.
6	1	Red component in a color vector.
	2	Green component in a color vector.
	3	Blue component in a color vector.
7	1	Real component of a complex number.
	2	Imaginary component of a complex number.

Geographic location group - GeoLoc

GeoLoc implementa uno specifico tipo di *VectorGroup* utilizzato per fornire informazioni geografiche dinamiche. Tale campo rappresenta l'insieme di tre sottocampi ovvero *LocEnum*, *GroupType* e *MemList*.

LocEnum contiene un *enumeration* che può assumere un valore compreso tra 0 e 3 per indicare che tipo di informazioni geografiche il sottocampo *GeoLoc* fornisce.

GroupType contiene un *enumeration* che può assumere un valore compreso tra 1 e 4, con lo stesso significato che viene attribuito per il sottocampo *GroupType*

del campo *VGroup*[2].

Il sottocampo *MemberList* è costituito da un array di UInt16 di grandezza prefissata contenete la lista di indirizzi dei canali di trasduzione, appartenente al gruppo vettoriale, con ordinamento predefinito; i campi *GroupType* e *MemList* sono presenti solamente se il campo *LocEnum* assume valore 2 o 3.

È molto importante sottolineare che nel caso *LocEnum* valga 2 o 3 possono esistere zero o più occorrenze del campo *GeoLoc*, ognuna con un valore del sottocampo *GroupType* differente.

TransducerChannel proxies - Proxies

Un *TransducerChannel proxy* permette di combinare le uscite di più sensori oppure gli ingressi di più attuatori in una singola struttura. Un *TransducerChannel proxy* possiede un numero di *TransducerChannel* e può essere letto e scritto però non possiede le altre caratteristiche che caratterizzano un *TransducerChannel*, come ad esempio il possedere un TransducerChannel TEDS, un Calibration TEDS, un Transfer Function TEDS oppure un Frequency Response TEDS. Il *TransducerChannel proxy*, quindi, è un dispositivo creato per permettere di trattare una collezione di *TransducerChannel* come una singola entità. Tale campo rappresenta l'insieme di tre sottocampi, ovvero *ChanNum*, *Organiz* e *MemList*.

ChanNum contiene l'indirizzo del *TransducerChannel* utilizzato per indirizzare il *proxy*, in modo da poterlo usare, quando necessario, per leggere dati da un sensore del *proxy* oppure scrivere dati in un attuatore del *proxy*.

Organiz contiene un *enumeration* che può assumere uno dei valori elencati in Tabella 2.7 e indica il metodo con cui la collezione di campioni acquisiti da un sensore (oppure applicati ad un attuatore) in risposta ad un comando di trigger vengono combinati.

Il sottocampo *MemList* è costituito da un array di UInt16 di grandezza prefissata contenete la lista di indirizzi dei canali di trasduzione utilizzati per costruire il *proxy*.

[2]Vedi Tabella 2.6

È molto importante sottolineare che possono esistere zero o più occorrenze del campo *Proxies*.

Tabella 2.7: Valori assunti dal sottocampo Organiz del campo Proxies

Organiz	**Meaning**
1	Block method
2	Interleave method 1
3	Interleave method 2: This is the same as Interleave method 1 except that the list of interleaved TransducerChannels shall be preceded by a one-sample data set from another TransducerChannel or proxy. This is typically used for a time stamp identifying the time of the first sample in the combined data set.

In Figura 2.3 sono illustrati i tre metodi per combinare i dati quando questi vengono acquisiti/trasmessi dal/al *proxy*. Il *Block Method* permette ad ogni singolo *data-set* di possedere una propria lunghezza indipendentemente dagli altri *data-set*, mentre con l'*Interleave Method* tutti i *data-set* devono possedere la stessa lunghezza.

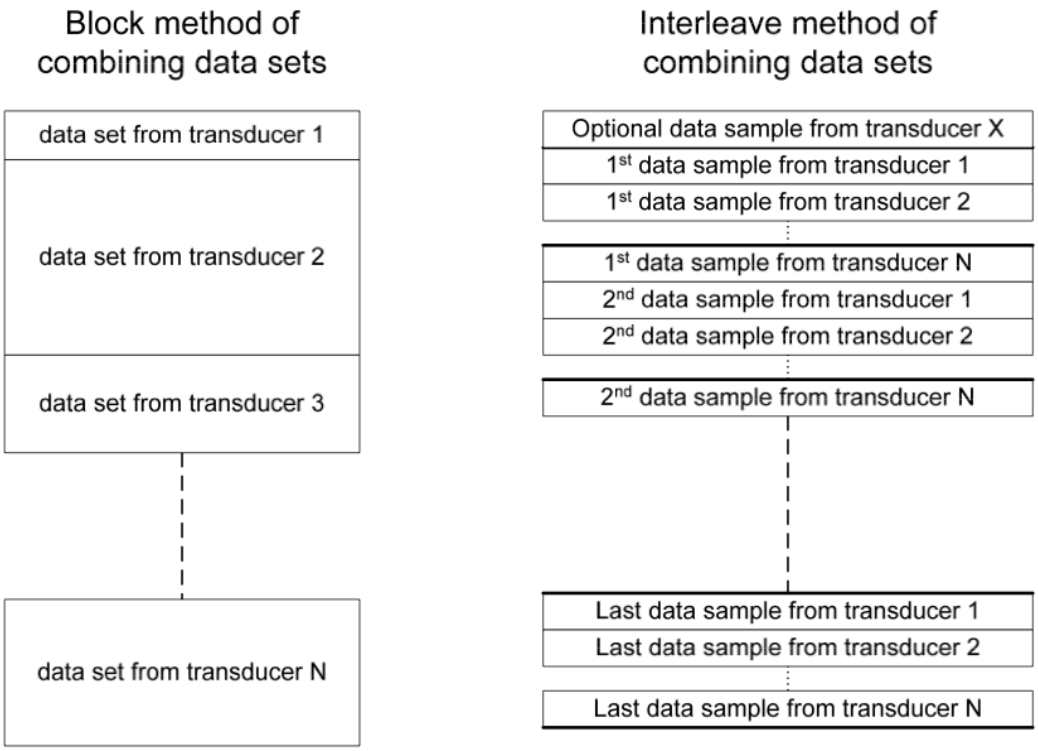

Figura 2.3: Metodi per combinare la collezione di dati

2.3.2 TransducerChannel TEDS

Il ChanTEDS costituisce la terza classe di TEDS. Esso fornisce informazioni dettagliate riguardo ad uno specifico canale di trasduzione, come ad esempio i parametri fisici che vengono misurati oppure controllati dal trasduttore, il *range* in cui il *TransducerChannel* opera, le caratteristiche dell'I/O digitale e le informazioni di temporizzazione. La struttura precisa dei ChanTEDS viene indicata in Tabella 2.8:

Tabella 2.8: Struttura del ChanTEDS

Field Type	**Field Name**	**Data Type**	**# byte**
Length	_	UInt32	4
0-2	[Reserved]	_	
3	TEDSID	UInt8	4
4-9	[Reserved]	_	
10	CalKey	UInt8	1
11	ChanType	UInt8	1
12	PhyUnits	UNITS	11
50	UnitType	UInt8	1
51	Radiants	UInt8	1
52	SterRad	UInt8	1
53	Meters	UInt8	1
54	Kilogram	UInt8	1
55	Seconds	UInt8	1
56	Amperes	UInt8	1
57	Kelvins	UInt8	1
58	Moles	UInt8	1
59	Candelas	UInt8	1
60	UnitsExt	UInt8	1
13	LowLimit	Float32	4
14	HighLimit	Float32	4
15	OError	Float32	4
16	SelfTest	UInt8	1
17	MRange	UInt8	1
18	Sample	_	_
40	DatModel	UInt8	1
41	ModLenth	UInt8	1
42	SigBits	UInt16	2
19	DataSet	_	_
43	Repeats	UInt16	2
44	SOrigin	Float32	4

Tabella 2.8: continua nella prossima pagina

Tabella 2.8: continua dalla pagina precedente

Field Type	**Field Name**	**Data Type**	**# byte**
45	StepSize	Float32	4
46	SUnits	UNITS	11
50	UnitType	UInt8	1
51	Radiants	UInt8	1
52	SterRad	UInt8	1
53	Meters	UInt8	1
54	Kilogram	UInt8	1
55	Seconds	UInt8	1
56	Amperes	UInt8	1
57	Kelvins	UInt8	1
58	Moles	UInt8	1
59	Candelas	UInt8	1
60	UnitsExt	UInt8	1
47	PreTrigg	UInt16	2
20	UpdateT	Float32	4
21	WSetupT	Float32	4
22	RSetupT	Float32	4
23	SPeriod	Float32	4
24	WarmUpT	Float32	4
25	RDelayT	Float32	4
26	TestTime	Float32	4
27	TimeSrc	UInt8	1
28	InPropDl	Float32	4
29	OutPropD	Float32	4
30	TSError	Float32	4
31	Sampling	_	_
48	SampMode	UInt8	1
49	SDefault	UInt8	1
32	DataXmit	UInt8	1
33	Buffered	UInt8	1
34	EndOfSet	UInt8	1
35	EdgeRpt	UInt8	1
36	ActHalt	UInt8	1
37	Direction	Float32	4
38	DAngles	2 Float32	8
39	ESOption	UInt8	1
61-127	[Reserved]	_	
128-255	[Open to manufactures]	_	
Checksum	_	UInt16	2

Per una migliore comprensione della struttura del ChanTEDS e di come essa è organizzata, si riporta in Figura 2.4 la sua rappresentazione grafica.

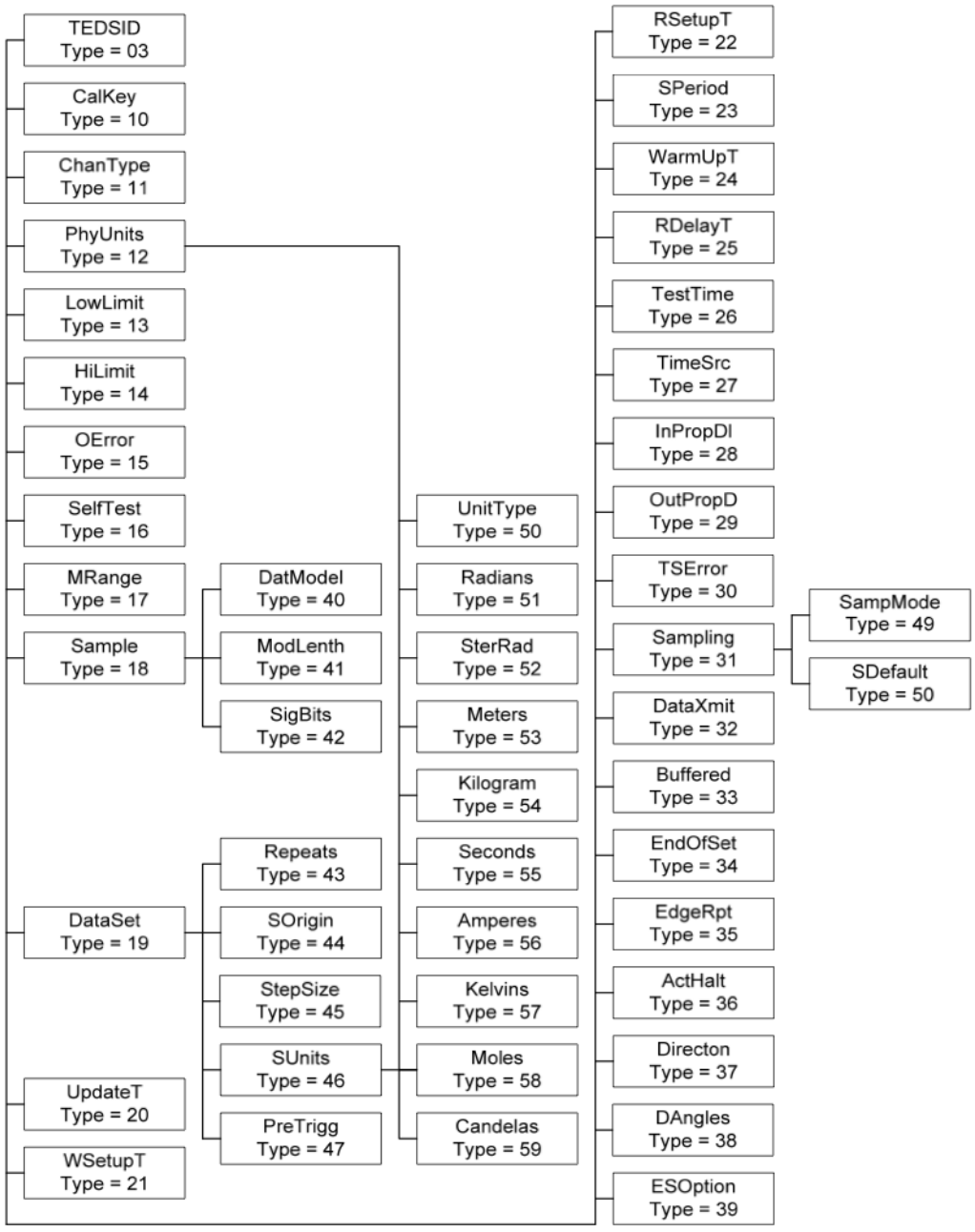

Figura 2.4: Struttura del ChanTEDS

Calibration key - CalKey

Il campo *CalKey* contiene un *enumeration* che può assumere un valore compreso fra 0 e 6 per indicare la capacità di calibrazione del *TransducerChannel*. I valori ed il loro significato vengono riportati in Tabella 2.9.

Tabella 2.9: Enumeration CalKey

Value	Calibration capability	Name
0	No calibration information needed or provided by the transducer module. This implies that there is no Calibration TEDS associated with this TransducerChannel. If an attempt is made to access the Calibration TEDS, the Calibration TEDSLength shall be zero.	CAL_NONE
1	Calibration information is provided as a Calibration TEDS. Correction is performed outside of the TIM.	CAL_SUPPLIED
2	Reserved.	-
3	Calibration information is provided but in a form that is not described in this standard. Correction is performed outside of the TIM.	CAL_CUSTOM
4	Calibration information is provided as a Calibration TEDS. Correction is applied in the TIM.	TIM_CAL_ SUPPLIED
5	Calibration information is provided as a Calibration TEDS, and is applied in the TIM. The calibration information is adjusted by a self-calibration capability.	TIM_CAL_SELF
6	Calibration information is provided but in a form that is not described in this standard. It may be a manufacturer-defined TEDS. Correction is performed in the TIM. If an attempt is made to access the Calibration TEDS, the Calibration TEDSLength shall be zero.	TIM_CAL_CUSTOM

TransducerChannel type key - ChanType

Il campo *ChanType* contiene un *enumeration* che può assumere un valore compreso fra 0 e 2 per indicare la tipologia di *TransducerChannel*, ovvero se è associato ad un sensore (0), un attuatore (1) oppure un sensore ad eventi (2).

Physical Units - PhyUnits

Il campo *PhyUnits* è utilizzato per definire l'unità di misura, appartenente al *Sistema Internazionale*, per indicare la grandezza fisica che viene misurata dal sensore oppure che controlla l'attuatore. Tale campo è un dato di tipo UNITS in cui i parametri *Radiants*, *Steradians*, *Meters*, *Kilograms*, *Seconds*, *Amperes*, *Kelvins*, *Moles* e *Candelas* indicano gli esponenti della relativa unità fisica necessari per calcolare l'unita di misura della grandezza fisica a cui si vuole riferire. Per esprimere gli esponenti, anche quelli negativi, lo standard impone che prima di esser memorizzati all'interno del TEDS l'esponente debba essere moltiplicato per 2 ed al risultato sommato 128, in questo modo è possibile rappresentare anche esponenti negativi all'interno di una variabile di tipo UInt8. Se per esempio l'unita di misura della grandezza fisica misurata da un sensore è il *Volt*, tale unità di misura può essere espressa attraverso le unità fisiche fondamentali del *Sistema Internazionale* secondo la seguente equazione:

$$\boxed{\text{Volt} = m^2 * kg * s^{-3} * A^{-1}}$$

Gli esponenti di *Meters*, *Kilograms*, *Seconds* e *Amperes* sono rispettivamente 2, 1, -3 e -1, mentre quello di tutte le altre unità fisiche vale 0. Di conseguenza i relativi parametri delle unita fisiche precedentemente elencate memorizzeranno rispettivamente i valori 132, 130, 122 e 126 mentre tutti gli altri assumeranno valore 128.

Design operational lower range limit - LowLimit

Nel caso di sensori questo campo contiene il più basso valore che il *TransducerChannel* può fornire dopo che un'eventuale correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite inferiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.

Nel caso di attuatori questo campo contiene il più basso valore che il *TransducerChannel* può accettare prima della correzione. Tale valore deve essere interpretato

attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite inferiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.

Design operational upper range limit - HiLimit

Nel caso di sensori questo campo contiene il più alto valore che il *TransducerChannel* può fornire dopo un'eventuale correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite superiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.
Nel caso di attuatori questo campo contiene il più alto valore che il *TransducerChannel* può accettare prima della correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite superiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.

Uncertainty under worst-case conditions - OError

Il valore all'interno di questo campo deve essere espresso attraverso l'unità fisica indicata nel campo *PhyUnits*. Tale campo è utilizzato per descrivere il livello di incertezza associata all'uscita del *TransducerChannel* nel peggior caso possibile tenendo conto di alcuni fattori come ad esempio le variazioni della tensione di alimentazione.

Self-test key - SelfTest

Tale campo è utilizzato per indicare se il *TransducerChannel* dispone di una funzione di *self-test*, ovvero di auto-diagnosi. *SelfTest* vale 0 se il *TrasducerChannel* non dispone di una funzione di *self-test*, mentre vale 1 se dispone di una funzione di *self-test*.

Multi-range capability - MRange

Il campo *MRange* specifica se eventuali trasduttori implementati nel TIM hanno la capacità di operare su più *range* differenti. Se esiste questa possibilità si rende necessario possedere un Commands TEDS che definisce i comandi utili per selezionare il *range* su cui si vuole operare.

Sample definition - Sample

Tale campo rappresenta l'insieme di tre sottocampi, ovvero:

- **DatModel**, descrive il modello dati, ossia il formato numerico con il quale il TIM opera in ingresso e uscita, da adottare quando si leggono i dati dal *TransducerChannel* oppure quando si scrivono dati nel *TransducerChannel*. Questo sottocampo contiene un *enumeration* che può assumere un valore compreso fra 0 e 7, il cui significato viene illustrato in Tabella 2.10.

Tabella 2.10: Enumeration DatModel

Value	**Based data type**	**Model**	**Constraint on data model length**
0	UInt8	N-octet integer (unsigned)	$0 \leq N \leq 8$
1	Float32	Single-precision real	$N = 4$
2	Double	Double-precision real	$N = 8$
3	UInt8	N-octet fraction (unsigned)	$0 \leq N \leq 8$
4	UInt8	Bit sequence	$0 \leq N \leq 8$
5	UInt8	Long integer (unsigned)	$9 \leq N \leq 255$
6	UInt8	Long fraction (unsigned)	$9 \leq N \leq 255$
7	TimeInstance	Time of day	$N = 8$

- **ModLenth**, contiene il numero di byte all'interno del modello dati con i limiti imposti dal valore della colonna *Constraint on data model length* in base alla rappresentazione scelta nel sottocampo *DatModel*. Se tale campo vale zero significa che il *TransducerChannel* non produce o utilizza dati.

- **SigBits**, contiene il numero di bit che sono significativi all'interno del modello dati in base alla rappresentazione scelta nel sottocampo *DatModel*.

Data set definition - DataSet

Tale campo rappresenta l'insieme di cinque sottocampi, ovvero:

- **Repeats**, contiene il numero massimo di ripetizioni dei valori del *Transducer-Channel* che possono esser prodotti oppure richiesti con un singolo segnale di trigger dopo l'inizio del campionamento. Ogni ripetizione rappresenta una misura addizionale o un valore di attenuazione prodotto o consumato dal *TransducerChannel* ad ogni evento del trigger. Se tale campo contiene un valore nullo (0) i sottocampi *SOrigin*, *StepSize* e *SUnits* devono essere ignorati.

- **SOrigin**, rappresenta il valore delle variabili indipendenti associate con il primo dato all'interno del *data-set* con unità di misura indicata dal sotto-campo *SUnits*. Se ad esempio nel sottocampo *SUnits* è specificato *Seconds* come unità di misura, il contenuto di questo sottocampo viene interpretato come il numero di secondi di ritardo prima dell'inizio di una acquisizione del *data-set*, ovvero della collezione di campioni acquisiti da un sensore (o applicati a un attuatore) in risposta ad un segnale di trigger.

- **StepSize**, contenere il minimo intervallo di tempo che il *TransducerChannel* può supportare.

- **SUnits**, contiene l'unità fisica a cui i precedenti campi fanno riferimento, memorizzata con le stesse regole del campo *PhyUnits*.

- **PreTrigg**, contiene il numero massimo di ripetizioni che possono essere campionate e memorizzate prima di un nuovo segnale di trigger quando il *TransducerChannel* sta operando in modalità *pre-trigger*.

TransducerChannel update time - UpdateT

Il campo *UpdateT* contiene l'intervallo di tempo massimo, espresso in secondi, che può trascorrere tra l'evento di trigger e l'arrivo del primo campione nell'insieme

di dati per il *TransducerChannel*. Questo intervallo di tempo deve essere calcolato assumendo che il contatore per la ripetizione dei dati sia impostato al valore specificato nel campo *Repeats*.

TransducerChannel write setup time - WSetupT

Il campo *WSetupT* contiene l'intervallo di tempo massimo, espresso in secondi, che può trascorrere tra la fine della scrittura del *data frame* e l'applicazione del trigger.

TransducerChannel read setup time - RSetupT

Il campo *RSetupT* contiene l'intervallo di tempo massimo, espresso in secondi, che può trascorrere tra la ricezione del segnale di trigger e l'istante in cui i dati sono disponibili per la lettura. Se il valore memorizzato all'interno di questo campo vale zero, allora il *TransducerChannel* può leggere i dati in qualsiasi momento dopo che ha ricevuto il primo segnale di trigger.

TransducerChannel sampling period - SPeriod

Il campo *SPeriod* definisce il minimo periodo di campionamento supportato. Per sensori e attuatori questo periodo è tipicamente limitato da tempo di conversione richiesto dai convertitori A/D e D/A. Nel caso di sensori ad eventi questo parametro rappresenta la minima risoluzione temporale con la quale il sensore riesce ad operare.

TransducerChannel warm-up time - WarmUpT

Il campo *WarmUpT* contiene il periodo di tempo, espresso in secondi, richiesto dal *TransducerChannel* per diventare operativo con una tolleranza che tiene conto della incertezza specificata all'interno del campo *OError*.

TransducerChannel read delay time - RDelayT

Il campo *RDelayT* contiene il massimo ritardo, espresso in secondi, che può presentarsi tra la ricezione del comando di lettura di un segmento dati del *TransducerChannel* e l'inizio della trasmissione del frame dati.

TransducerChannel self-test time requirement - TestTime

In questo caso se il campo *SelfTest* indica che nessun meccanismo di *self-test* è implementato allora questo campo deve essere ignorato, se invece *SelfTest* indica la presenza di un meccanismo di auto-diagnosi allora questo campo non deve essere ignorato. Il campo *TestTime* contiene il tempo massimo, espresso in secondi, richiesto per eseguire il meccanismo di *self-test*.

Source for the time of sample - TimeSrc

Il campo *TimeSrc* contiene un *enumeration* che può assumere un valore compreso fra 0 e 5, il cui significato viene illustrato in Tabella 2.11.

Tabella 2.11: Enumeration TimeSrc

Value	Name	Function
0	NoHelp	No facilities in the transducer module are available to support determination of the time that a sample was latched.
1	Incoming	The time interval between the receipt of the trigger and the latching of the sample is not measured. The default delay between the trigger and the latching of the sample is found in the incoming propagation delay through the data transport logic field of the TransducerChannel TEDS.
2	Outgoing	The time interval between the latching of the sample and when the data are transmitted is not measured. The default delay between the latching of the sample and the transmission of the sample is found in the outgoing propagation delay through the data transport logic field of the TransducerChannel TEDS. This is only useful when operation occurs in the immediate operation sampling mode.
3	MInterval	The time interval between the receipt of a trigger and the latching of the sample is measured using a time interval sensor. The Calibration TEDS for the time interval sensor provides the method of converting the output of the time interval sensor to time.

Tabella 2.11: continua nella prossima pagina

Tabella 2.11: continua dalla pagina precedente

Value	Name	Function
4	SInterval	The time interval between the receipt of the trigger and the latching of the sample is measured using a time interval sensor with a clock derived from a synchronization signal. The Calibration TEDS for the time interval sensor provides the method of converting the output of the embedded sensor to time. NOTE—This capability does not exist in all members of the IEEE 1451 family.
5	ToDSense	A TimeInstance Sensor supplies the time of day that the sample was processed. The characteristics of this sensor are defined in the TEDS for this TimeInstance sensor.

Incoming propagation delay through the data transport logic - InPropDl

Questo campo è richiesto se all'interno del campo *TimeSrc* è memorizzato il valore 1 a cui è associato la funzione *Incoming*, altrimenti il campo può essere omesso. Tale campo contiene un numero in virgola mobile a singola precisione che definisce il ritardo massimo, espresso in secondi, tra la ricezione del segnale di trigger dallo strato fisico del protocollo di comunicazione e l'acquisizione o l'applicazione di un campione.

Outgoing propagation delay through the data transport logic - OutPropD

Questo campo è richiesto se all'interno del campo *TimeSrc* è memorizzato il valore 2 a cui è associato la funzione *Outgoing*, altrimenti il campo può essere omesso. Tale campo contiene un numero in virgola mobile a singola precisione che definisce il ritardo massimo, espresso in secondi, tra la ricezione dell'ultimo campione del segnale e la trasmissione di un segnale proveniente dallo strato di trasporto.

Trigger-to-sample delay uncertainty - TSError

Tale campo contiene un numero in virgola mobile a singola precisione che definisce l'incertezza, espressa in secondi, del ritardo di default tra il trigger e il campione che viene preso da un sensore o applicato ad un attuatore.

Sampling attribute - Sampling

Il campo *Sampling* rappresenta l'insieme di due sottocampi, ovvero:

- **SampMode**, descrive quali modalità di campionamento sono supportate dal *TransducerChannel*, il quale può supportate una o più modalità. Nel caso esistessero più modalità di campionamento supportate si dice che il *TransducerChannel* è programmabile. Questo sottocampo è indispensabile per stabilire in che modo vengono acquisiti o forniti i dati da processare. Lo standard IEEE 1451.0 definisce ben cinque modalità di campionamento differenti che si distinguono fra loro per il modo di operare prima dell'arrivo del comando di trigger. Ogni modalità è associata ad uno specifico bit all'interno della variabile UInt8 che compone questo sottocampo, se il bit è asserito vuol dire che la modalità di campionamento è supportata, altrimenti no. Si riportano in Tabella 2.12 la relazione tra i bit del sottocampo *SampMode* e le modalità di campionamento.

Tabella 2.12: Enumeration SampMode

Bit	Definition
0 (lsb)	Trigger-initiated mode
1	Free-running without pre-trigger mode
2	Free-running with pre-trigger mode
3	Continuous Sampling mode
4	Immediate operation sampling mode
5	Reserved.
6	Reserved.
7 (msb)	Reserved.

Nella modalità *Trigger-initiated* il sensore/attuatore inizialmente è inattivo. La ricezione del comando di trigger avvia il processo di acquisizione dati nel caso del sensore oppure il processo di elaborazione dati nel caso di un attuatore. Il processo di campionamento continua finché tutti i campioni all'interno del data set sono stati processati. Nel caso di un sensore, tutte le misure effettuate sono di tipo *post-trigger*, dato che vengono attenute dopo

la ricezione del comando di trigger che ne abilità l'acquisizione e il successivo salvataggio.

Nella modalità *Free-running without pre-trigger* il sensore/attuatore è sempre attivo. Nel caso di un sensore la grandezza fisica applicata in ingresso viene continuamente campionata e convertita in un valore numerico. Dai valori ottenuti in questo modo vengono memorizzati solo quelli che seguono la ricezione del comando di trigger e il processo termina quando tutti i campioni del *data-set* sono stati processati. Anche in questa modalità tutte le misure campionate dal sensore sono di tipo *post-trigger*, in quanto ottenute tutte dopo aver ricevuto il comando di trigger che abilita il salvataggio.

La modalità *Free-running with pre-trigger* è disponibile solamente per i sensori ed estende la precedente modalità *Free-running without pre-trigger*, in questo caso infatti le misure acquisite dal sensore prima della ricezione del comando di trigger vengono salvate in una porzione del *data-set*, indicata col nome di *pre-trigger*, e gestita come un buffer circolare di grandezza prefissata, dove una volta riempito il buffer il campione più vecchio viene sovrascritto dal nuovo campione acquisito. Quando viene ricevuto il comando di trigger il salvataggio viene spostato sulla porzione rimanente del *data-set* fino al suo completamento.

La modalità *Continuous Sampling* è realizzabile solo con sensori/attuatori dotati di buffer di memoria multipli nei quali poter memorizzare i campioni. Questa modalità è simile alla modalità *Free-running without pre-trigger*, soltanto che l'elaborazione dei dati non termina quando un *data-set* è completato, ma continua sui successivi *data-set* disponibili. Nel caso di un sensore, una volta terminati tutti i buffer disponibili, vengono sovrascritti i campioni contenuti nel buffer meno recente. Nel caso di un attuatore, quando viene ricevuto il comando di trigger, si inizia a processare i dati contenuti nel buffer corrente, e una volta terminati si passerà a processare quelli contenuti nel buffer meno recente.

Nella modalità *Immediate operation sampling*, nel caso di un sensore, il comando di trigger corrisponde alla richiesta di lettura del *data-set*, in questo modo il sensore acquisisce un *data-set* e lo trasmette al nodo richiedente come risposta al comando di trigger ricevuto. Nel caso di un attuatore, il comando di trigger corrisponde alla richiesta di scrittura del *data-set*, in questo modo il *data-set* ricevuto assieme al comando di scrittura viene processato immediatamente dall'attuatore.

- **SDefault**, se solamente una modalità di campionamento è ammessa, la modalità di campionamento di default deve essere anche l'unica modalità di campionamento permessa. Nel caso di più modalità ammesse si deve scegliere fra una di esse. Per selezionare la modalità di default bisogna asserire il relativo bit associato come indicato in Tabella 2.12, mentre tutti gli altri bit devono rimanere deasseriti.

Buffered attribute - Buffered

Il campo *Buffered* descrive su quanti buffer il *TransducerChannel* può operare, contiene un *enumeration* che può assumere un valore compreso fra 0 e 3, il cui significato viene illustrato in Tabella 2.13

Tabella 2.13: Enumeration Buffered

Value	Description
0	TransducerChannel has no more than one buffer available.
1	TransducerChannel has multiple buffers available and may only be operated in the buffered mode.
2	TransducerChannel has multiple buffers available and may be operated either buffered or unbuffered. The unbuffered mode of operation is the default.
3	TransducerChannel has multiple buffers available and may be operated either buffered or unbuffered. The buffered mode of operation is the default.

End-of-data-set operation attribute - EndOfSet

Tale campo è richiesto se il *TransducerChannel* è un attuatore, e descrive le modalità di *End-of-data-set* che l'attuatore può supportare e contiene un *enumeration* che può assumere un valore compreso fra 0 e 4, il cui significato viene illustrato in Tabella 2.14.

Tabella 2.14: Enumeration EndOfSet

Value	Description
0	Not applicable.
1	This TransducerChannel holds the last value in the data set until another trigger is received.
2	This TransducerChannel recirculates through the last data set until another trigger is received.
3	This TransducerChannel may be operated with either the hold mode or the recirculate mode. These two modes are mutually exclusive. The hold mode is the default.
4	This TransducerChannel may be operated with either the hold mode or the recirculate mode. These two modes are mutually exclusive. The recirculate mode is the default.

Data transmission attribute - DataXmit

Il campo *DataXmit* può essere omesso nel caso di attuatori e descrive le modalità di trasmissione dati che il *TransducerChannel* supporta. Questo campo contiene un *enumeration* che può assumere un valore compreso fra 0 e 4, il cui significato viene illustrato in Tabella 2.15.

Tabella 2.15: Enumeration DataXmit

Value	Description
0	Reserved.
1	This TransducerChannel is only capable of being operated in the only when commanded modes.
2	This TransducerChannel is capable of being operated in the Streaming when a buffer is full or only when commanded modes.
3	This TransducerChannel is capable of being operated in the Streaming at a fixed interval or only when commanded modes.
4	This TransducerChannel is capable of being operated in the only when commanded, Streaming when a buffer is full or Streaming at a fixed interval modes.

Edge-to-report attribute - EdgeRpt

Tale campo è necessario per sensori ad eventi e descrive le modalità operative *Edge-to-Report* supportate dal sensore ad eventi. Questo campo contiene un *enumeration* che può assumere un valore compreso fra 0 e 7, il cui significato viene illustrato in Tabella 2.16.

Tabella 2.16: Enumeration EdgeRpt

Value	Meaning
0	Not applicable.
1	This event sensor only reports rising edges.
2	This event sensor only reports falling edges.
3	This event sensor reports on both edges.
4	Reserved.
5	This event sensor is capable of reporting both edges, but the default is to report rising edges.
6	This event sensor is capable of reporting both edges, but the default is to report falling edges.
7	This event sensor is capable of reporting both edges, and the default is to report both edges.

Actuator-halt attribute - ActHalt

Tale campo è necessario per gli attuatori e descrive le modalità operative *Actuator-Halt* supportate dall'attuatore. Questo campo contiene un *enumeration* che può assumere un valore compreso fra 0 e 3, il cui significato viene illustrato in Tabella 2.17.

Tabella 2.17: Enumeration ActHalt

Value	Meaning
0	Not applicable.
1	Halt Immediate.
2	Halt at the end of the data set.
3	Ramp to a predefined state.

Sensitivity direction - Directon

Tale campo indica quale direzione, su uno dei tre assi X, Y e Z, la sensibilità assume. Questo campo contiene un *enumeration* che può assumere un valore compreso fra

0 e 6, il cui significato viene illustrato in Tabella 2.18.

Tabella 2.18: Enumeration Direction

Value	Meaning
0	Not applicable.
1	+X
2	-X
3	+Y
4	-Y
5	+Z
6	-Z

Direction angles - DAngles

Tale campo definisce i due angoli che il *TransducerChannel* forma rispettivamente con il piano di riferimento e con la direzione di riferimento fissati dal costruttore. Il primo numero rappresenta il ρ delle coordinate cilindriche espresso in radianti, mentre il secondo numero rappresenta il ϕ delle coordinate cilindriche anch'esso espresso in radianti utilizzando, in entrambi i casi, la regola della mano destra per il calcolo dei valori.

Event sensor options - ESOption

Tale campo si applica solo ai sensori ad eventi e definisce l'abilità del modulo del trasduttore di rilevare e riportare le inconsistenze. Questo campo contiene un *enumeration* che può assumere un valore compreso fra 0 e 6, il cui significato viene illustrato in Tabella 2.19.

Tabella 2.19: Enumeration ESOption

Value	Meaning
0	Not applicable.
1	Pattern/threshold/hysteresis not changeable.
2	Changeable and inconsistencies detected.
3	Changeable and inconsistencies not detected.

2.3.3 Calibration TEDS

I CalTEDS forniscono le costanti di calibrazione necessarie per poter utilizzare i trasduttori, dopo che a questi è stato applicato il processo di correzione. Il processo di correzione consiste nell'applicare specifiche funzioni matematiche ai dati che transitano su uno o più *TransducerChannel* e/o ai dati forniti da programmi software. Per i sensori il processo di correzione consiste nell'applicare le costanti di calibrazione racchiuse all'interno del CalTEDS all'output del sensore in modo tale da convertire i dati in uscita nella forma desiderata dal sistema. Per gli attuatori invece, il processo di correzione viene applicato ai numeri forniti dal sistema per convertirli nella forma desiderata dall'attuatore. Il CalTEDS definisce due metodi di correzione, ovvero:

- **Linear Method**, ossia il metodo più utilizzato e descritto dall'equazione:

$$\boxed{y = mx + b = b + mx}$$

  Tale metodo si basa sul fatto che la maggior parte delle calibrazioni utilizza un singolo segmento lineare di conversione, così questo caso viene definito come un metodo a parte, che in effetti non è altro che un sottoinsieme del seguente metodo.

- **General Method**, che può essere utilizzato con qualsiasi tipo di funzione di correzione inclusa quella per il metodo lineare. In tutti i casi il metodo generale prevede l'esecuzione dei seguenti due passi:

  1. nel primo passo la curva di calibrazione viene suddivisa in più segmenti in modo da poter raggiungere l'accuratezza desiderata con il giusto grado per l'equazione del polinomio. Tale operazione viene indicata con il nome di segmentazione e nella maggior parte dei casi suddivide la curva di correzione in un unico segmento.
  2. nel secondo passo invece si definisce una curva polinomiale per adattare la risposta del trasduttore all'interno di ogni segmento.

La struttura del CalTEDS viene riportata di seguito in Tabella 2.20.

Tabella 2.20: Struttura del CalTEDS

Field Type	**Field Name**	**Data Type**	**# byte**
Length	_	UInt32	4
0-2	[Reserved]	_	
3	TEDSID	UInt8	4
4-9	[Reserved]	_	
10	LstCalDt	TimeIstance	8
11	CalInrvl	TimeDuration	8
12	SIConvrt	_	_
30	SISlope	Float32	4
31	Intrcpt	Float32	4
13	LowLimit	Float32	4
14	HiLimit	Float32	4
15	OError	Float32	4
16	OConvert	UInt8	1
17	IConvert	UInt8	1
20	LinOnly	_	_
21	XdcrBlk	_	_
40	Element	UInt16	2
41	ChanNum	UInt16	2
42	ChanKey	UInt8	1
43	Degree	UInt8	1
44	STable	_	_
45	OTable	Float32Array	
46	LoBndry	Float32Array	
47	HiBndry	Float32	4
22	CoefBlk	_	_
50	CellNum	UInt8	1
51	CoefSet	Float32Array	Variable
18-19	[Reserved]	_	
23-29	[Reserved]	_	
48-49	[Reserved]	_	
52-127	[Reserved]	_	
128-255	[Open to manufactures]	_	
Checksum	_	UInt16	2

Per una migliore comprensione d0ella struttura del CalTEDS e di come essa è organizzata, si riporta in Figura 2.5 la rappresentazione grafica della struttura CalTEDS.

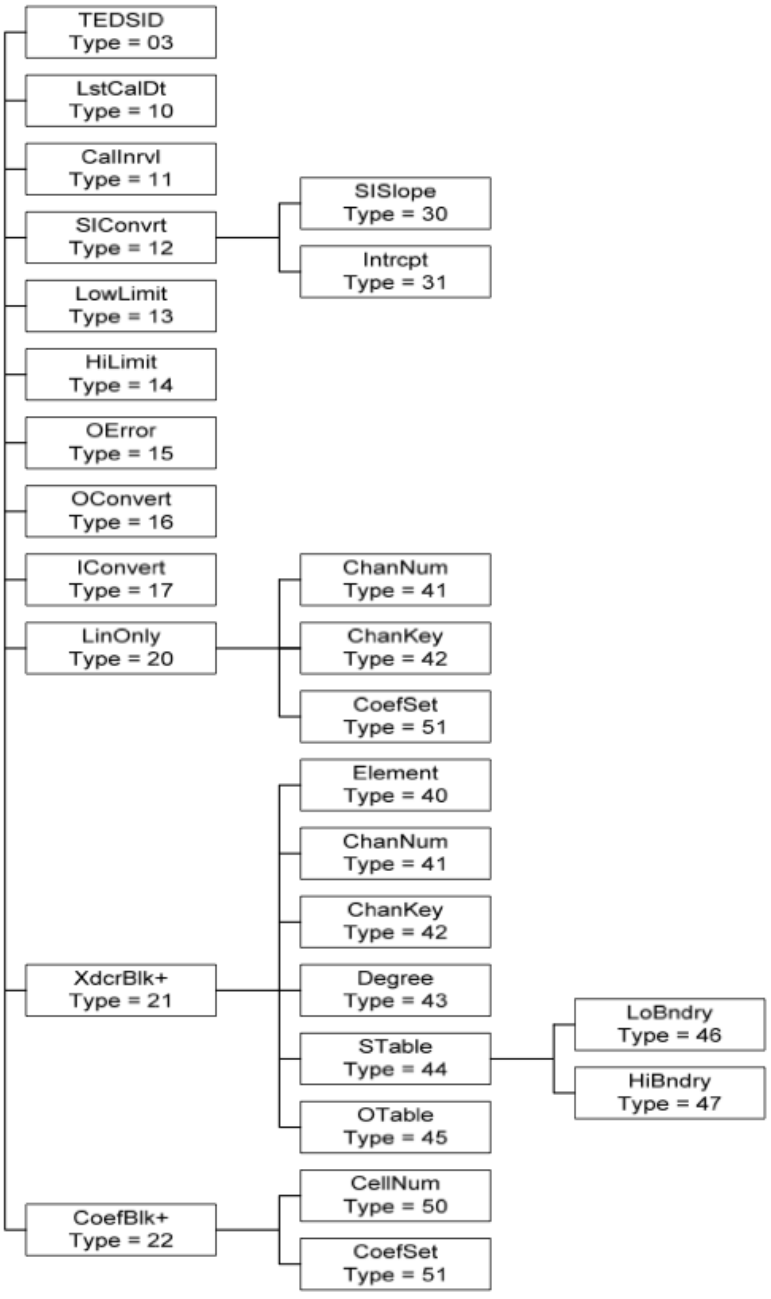

Figura 2.5: Struttura del CalTEDS

Last calibration date-time - LstCalDt

Questo campo fornisce l'ora e il giorno in cui il *TransducerChannel* è stato calibrato per l'ultima volta.

Calibration interval - CalInrvl

Il campo *CalInrvl* rappresenta la durata, espressa in secondi, per la quale il *TransducerChannel* può operare senza dover essere nuovamente calibrato, continuando così a produrre dati la cui incertezza è ancora quella specificata nel campo *OError*.

SI units conversion constants - SIConvrt

Il campo *SIConvrt* rappresenta l'insieme di due sottocampi, che sono:

- **SISlope**, il quale contiene il numero che se moltiplicato per il risultato ottenuto dal processo di correzione e sommato poi al numero che, espresso con le unità del *Sistema Internazionale*, esprime l'intercetta di conversione, memorizzato nel sottocampo successivo, darà come risultato un numero che rappresenterà il valore fisico corretto espresso con le unità del *Sistema Internazionale*.

$$\boxed{\text{SI} = \textbf{Slope} * \text{Value} + \text{Intercept}}$$

- **Intrcpt**, il quale contiene invece il numero che, se sommato al valore contenuto all'interno del sottocampo *SISlope* moltiplicato per l'output del processo di correzione, restituisce un numero che rappresenta il valore fisico espresso con le unità del *Sistema Internazionale*.

$$\boxed{\text{Value} = \frac{\text{SI} - \textbf{Intercept}}{\text{Slope}}}$$

Operational lower range limit - LowLimit

Nel caso di sensori questo campo contiene il più basso valore che il *TransducerChannel* può fornire dopo un'eventuale correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare

questo limite inferiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.
Nel caso di attuatori questo campo contiene il più basso valore che il *TransducerChannel* può accettare prima della correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite inferiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.

Operational upper range limit - HiLimit

Nel caso di sensori questo campo contiene il più alto valore che il *TransducerChannel* può fornire dopo un'eventuale correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite superiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.
Nel caso di attuatori questo campo contiene il più alto valore che il *TransducerChannel* può accettare prima della correzione. Tale valore deve essere interpretato attraverso le unità specificate dal campo *PhyUnits*. Se il valore dovesse superare questo limite superiore, esso potrebbe non essere compatibile con le specifiche del *TransducerChannel* imposte dal costruttore.

Uncertainty under worst-case conditions - OError

Il valore all'interno di questo campo deve essere espresso attraverso l'unità fisica indicata nel campo *PhyUnits*. Tale campo è utilizzato per descrivere il livello di incertezza associata all'uscita del *TransducerChannel* nel peggior caso possibile.

Post-conversion operation - OConvert

Il campo *OConvert* contiene un *enumeration* di valori compresi fra 0 e 5, il cui significato viene illustrato in Tabella 2.21, che identifica l'operazione matematica che deve essere applicata all'uscita del processo di correzione per produrre un valore nell'unità di misura specificata dal TransducerChannel TEDS.

Pre-conversion operation - IConvert

Il campo *IConvert* contiene un *enumeration* di valori compresi fra 0 e 5, il cui significato viene illustrato in Tabella 2.21, che identifica l'operazione matematica che deve essere applicata all'ingresso del processo di correzione per produrre un valore nell'unità di misura specificata dal TransducerChannel TEDS.

Tabella 2.21: Enumeration OConvert/IConvert

Value	Meaning
0	No pre- or post-conversion operation is required
1	Inversion: Apply 1/x
2	Log base 10: Apply log10(x)
3	Exponent Base 10: Apply 10x
4	Natural Log : Apply ln (x)
5	Exponent Base e: Apply ex

Linear conversion only - LinOnly

Nel caso il metodo di correzione prescelto fosse quello di tipo lineare allora questo campo non deve essere omesso, altrimenti si. Questo campo descrive tutte le costanti necessarie per un solo *TransducerChannel* quando il processo di conversione utilizza una sola sezione ed è di tipo lineare. Il campo *LinOnly* rappresenta l'insieme di due sottocampi, che sono:

- *ChanNum*, contiene l'indirizzo del TransducerChannel da cui prelevare l'ingresso per il processo di conversione.

- *ChanKey*, contiene un *enumeration* che assume solamente valore 0 o 1. Assume valore 0 se i valori di ingresso devono essere presi dal lato del transducer del processo di correzione, questo è il caso dei sensori, mentre assume valore 1 se i valori di ingresso devono essere presi dal lato dell'NCAP del processo di correzione e questo è invece il caso degli attuatori. In Firgura 2.6 viene riportato un esempio che consente di comprendere meglio il significato del sottocampo *ChanKey*. Il processo di correzione del *TransducerChannel 3* utilizza in ingresso i dati provenienti dal *TransducerChannel 1*. Questi da-

ti possono provenire da entrambi i lati del *TransducerChannel 1* (*Trasducer Side of Correction Process* e *NCAP Side of Correction Process*), la decisione spetta al sottocampo *ChanKey* nella descrizione del *TransducerChannel 1* nel Calibration TEDS del *TransducerChannel 3*.

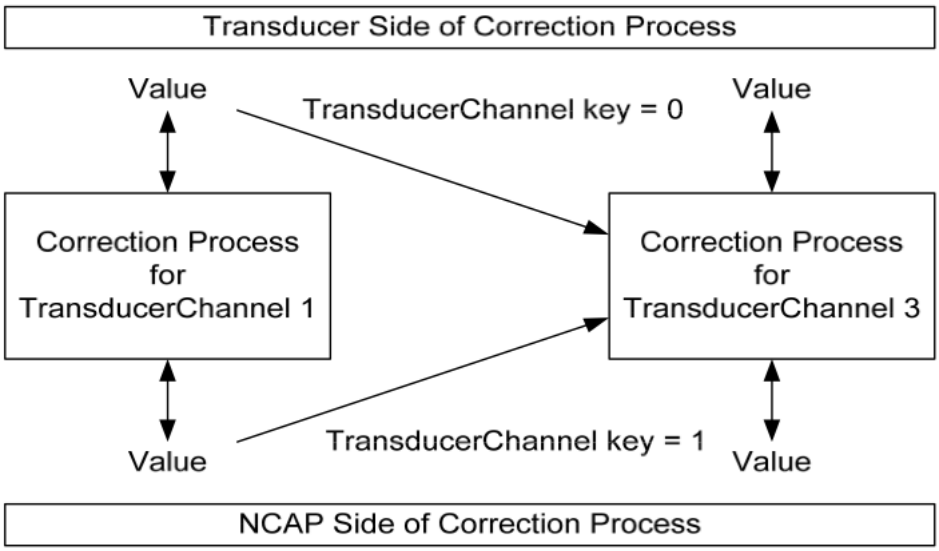

Figura 2.6: Applicazione del sottocampo ChanKey

TransducerChannel description - XdcrBlk

Nel caso il metodo di correzione prescelto fosse quello di tipo generale allora questo campo non deve essere omesso, altrimenti si. Questo campo descrive tutte le costanti, ed eccezione dei coefficienti di calibrazione, necessarie per un singolo *TransducerChannel* che partecipa al processo di correzione. Se più *TransducerChannel* sono utilizzati come ingresso del processo di correzione, allora ciascuno di questi ingressi devono possedere la propria descrizione all'interno del Calibration TEDS. Il campo *XdcrBlk* rappresenta l'insieme di sei sottocampi, ovvero:

- *Element*, tale campo è utilizzato per determinare l'ordine con il quale le celle sono numerate nel sottocampo *CellNum* del campo *CoefBlk*.
- *ChanNum*, contiene l'indirizzo del *TransducerChannel* da cui prelevare l'ingresso per il processo di correzione.

- *ChanKey*, contiene un *enumeration* che assume solamente valore 0 o 1. Assume valore 0 se i valori di ingresso devono essere presi dal lato del transducer del processo di correzione, questo è il caso dei sensori, mentre assume valore 1 se i valori di ingresso devono essere presi dal lato dell'NCAP del processo di correzione e questo è invece il caso degli attuatori.

- *Degree*, contiene la più alta potenza dell'ingresso del processo di correzione.

- *STable*, rappresenta l'insieme di due sottocampi, ovvero:

  - *LoBndry*, ossia un array uno-dimensionale contenente il limite inferiore per ciascun segmento del processo di correzione. In altre parole definisce i limiti del dominio in ordine ascendente.

  - *HiBndry*, contenente un valore che è più grande del più grande valore compreso nel dominio degli ingressi.

- *OTable*, tale campo rappresenta un array uno-dimensionale contenente un valore di offset per ciascun segmento dell'ingresso del processo di correzione.

Multinomial coefficient block - CoefBlk

Nel caso il metodo di correzione prescelto fosse quello di tipo generale allora questo campo non deve essere omesso, altrimenti si. Il campo *CoefBlk* rappresenta l'insieme di due sottocampi, ovvero:

- *CellNum*, contenente la numerazione delle celle da 0 a m.

- *CoefSet*, ossia un array di numeri a virgola mobile a singola precisione, contenente i coefficienti per ciascun termine dell'equazione utilizzata per il metodo generale di correzione.

Capitolo 3

Editor in LabVIEW per scrivere virtual TEDS

L'obiettivo finale di questa tesina è quello di produrre un editor, utilizzando il programma LabVIEW vers. 8.5, che faciliti la compilazione di TEDS virtuali rispettando i vincoli imposti dallo standard IEEE 1451.0, e salvi questi all'interno di file binari consentendo così in un secondo momento anche la loro lettura e modifica. Prima di addentrarci su come è strutturato il codice è bene soffermarsi un attimo su che cosa è esattamente LabVIEW. Esso è un ambiente di sviluppo grafico per la creazione di applicazioni di progettazione, controllo e test flessibili e scalabili. LabVIEW fa uso di un modello di programmazione, brevettato dalla *National Instruments*, che si basa sul flusso dati, in questo modo si semplifica notevolmente lo sviluppo di applicazioni, senza l'utilizzo dell'architettura sequenziale su cui i più recenti linguaggi di programmazione si basano. Il codice grafico, composto da diagrammi a blocchi e flussi di dati, conferisce una più elevata intuitività ai programmi permettendo inoltre di creare facilmente diagrammi a blocchi in grado di eseguire operazioni multiple anche in parallelo. In conclusione quindi si può pensare a LabVIEW come un linguaggio di programmazione grafica completo che include tutte le funzioni standard dell'ambiente generico di programmazione, come ad esempio strutture di dati, cicli di iterazione, gestione di eventi e addirittura programmazione orientata agli oggetti.

3.1 Funzionamento dell'editor

Lo scopo del file *TEDSEditor.vi* è quello di facilitare l'utente alla compilazione di un virtual TEDS che appartenga ad una delle tre classi implementate.

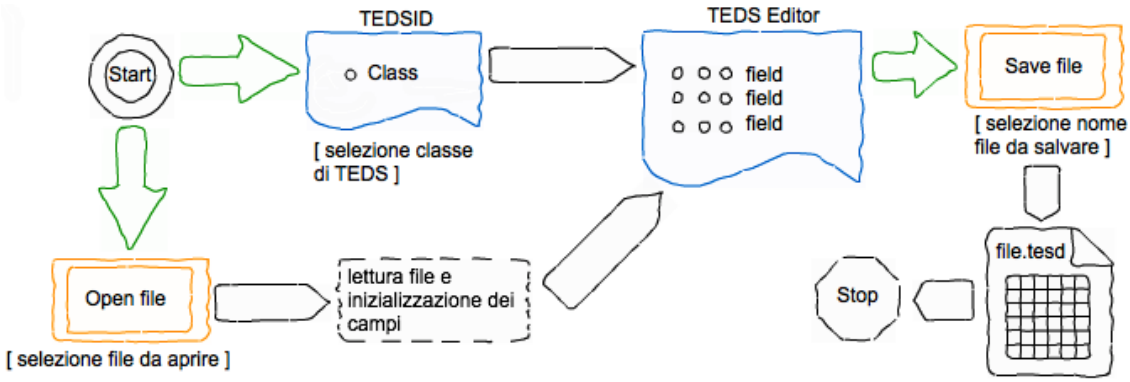

Figura 3.1: Diagramma di flusso del file TEDSEditor.vi

In Figura 3.1 è rappresentato il diagramma di flusso del file *TEDSEditor.vi*, in cui vengono evidenziate le due modalità operative dell'editor ovvero quella di default, dove tutti i campi dell'interfaccia sono inizializzati a zero e quella personalizzata che permette invece di inizializzare i campi dell'interfaccia con valori memorizzati all'interno di un TEDS precedentemente creato. Le frecce verdi rappresentano le scelte che l'utente può prendere durante l'utilizzo dell'editor, i doppi riquadri arancioni invece rappresentano l'interazione dell'utente con i due bottoni presenti nell'editor, mentre i gruppi blu di campi rappresentano l'interazione dell'utente con i campi messi a disposizione dall'interfaccia dell'editor per l'inserimento delle informazioni necessarie alla costruzione del TEDS. L'editor si presenta inizialmente come due sezioni, la prima, riportata in Figura 3.2 presenta due bottoni e quattro campi necessari per la costruzione del campo *TEDS identification header*.

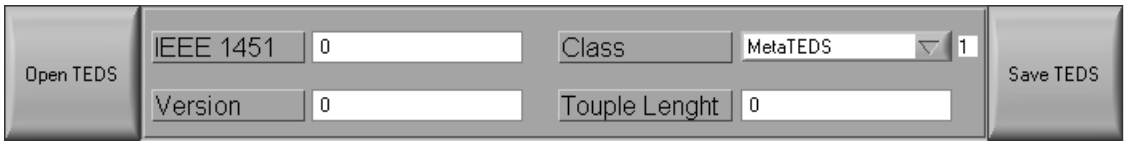

Figura 3.2: Sezione 1 dell'editor in LabVIEW

Il primo bottone, *Open TEDS*, consente di entrare nella modalità di inizializzazione personalizzata dei campi, una volta premuto infatti si presenterà una finestra di dialogo che chiederà di selezionare il file TEDS da aprire per la lettura dei valori di ciascun campo. In questa modalità l'editor riconosce automaticamente la classe a cui appartiene il TEDS selezionato e di conseguenza modifica l'interfaccia della seconda sezione per permettere all'utente di inserire tutte le informazioni necessarie che automaticamente vengono impostate per coincidere con quelle del file TEDS aperto precedentemente. Questa modalità è molto utile quando si vogliono creare due o più TEDS simili ma non uguali, oppure quando si vuole modificare un TEDS, magari a causa di un campo mal impostato, permettendo così di correggerlo rapidamente senza dover riscrivere l'intero TEDS.

La prima sezione dell'editor contiene poi quattro campi che servono per costruire il campo *TEDSID* comune a tutte le classi di TEDS e necessario per la loro identificazione. Di questi quattro campi il più importante risulta essere il campo *Class* il quale, essendo direttamente collegato alla seconda sezione dell'editor, consente la modifica automatica dell'interfaccia per adattarsi alla classe di TEDS selezionata, così da consentire l'inserimento di tutte le informazioni necessarie per la costruzione del TEDS. Utilizzando questi quattro campi si sceglie di operare nella modalità di default dove tutti i campi della seconda sezione dell'editor saranno inizializzati a zero per consentire la scrittura di un nuovo TEDS partendo da un file vuoto.

Secondo ed ultimo è il bottone *Save TEDS* il quale non corrisponde ad una modalità operativa bensì ad una funzione messa a disposizione dall'editor per il salvataggio del file TEDS. Se premuto appare una finestra di dialogo, come illu-

strato in Figura 3.3, che chiede dove e con che nome memorizzare il file TEDS con estensione *.teds*. La funzione di salvataggio è programmata per creare un nuovo file in caso non esista già un TEDS con lo stesso nome, oppure permette di sovrascrivere un TEDS già esistente semplicemente selezionandolo sulla finestra di dialogo, in questo caso però, prima di procedere con il salvataggio, l'editor avviserà l'utente per accertarsi di voler effettivamente sovrascrivere il TEDS selezionato.

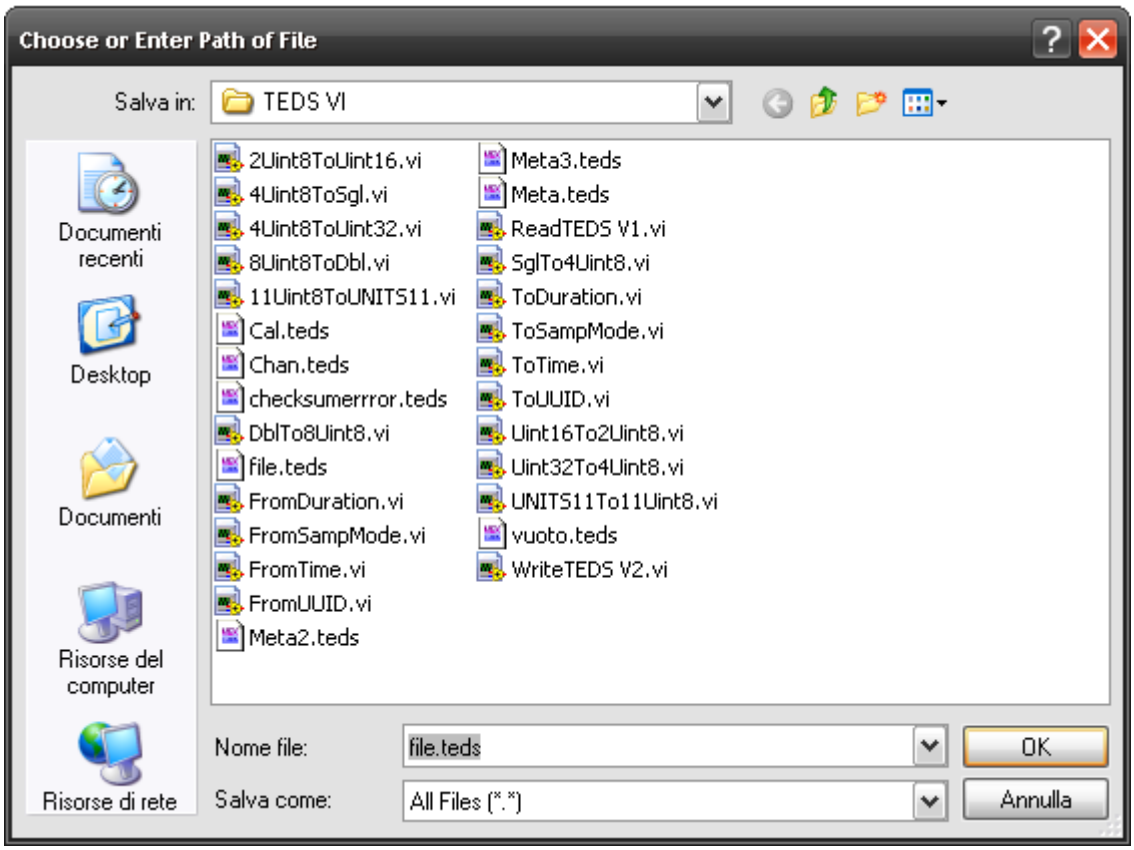

Figura 3.3: Finestra di dialogo per il salvataggio di un virtual TEDS

In Figura 3.4, 3.5 e 3.6 sono riportate alcune porzioni della seconda sezione dell'editor, che mostrano i campi di inserimento delle informazioni, una per ogni classe di TEDS implementata. La prima figura mostra un esempio di interfaccia del Meta TEDS, la seconda un esempio del TransducerChannel TEDS ed infine un esempio relativo all'interfaccia Calibration TEDS.

Figura 3.4: Esempio di interfaccia per Meta TEDS

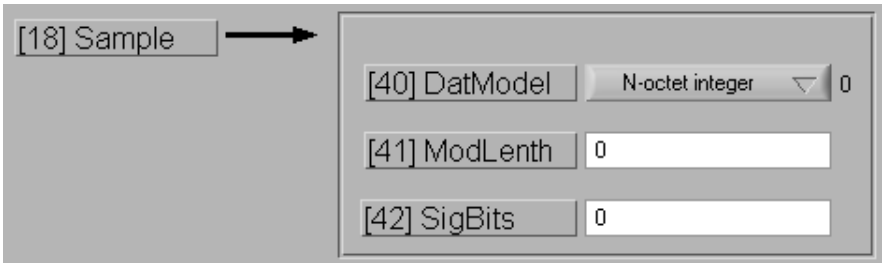

Figura 3.5: Esempio di interfaccia per TransducerChannel TEDS

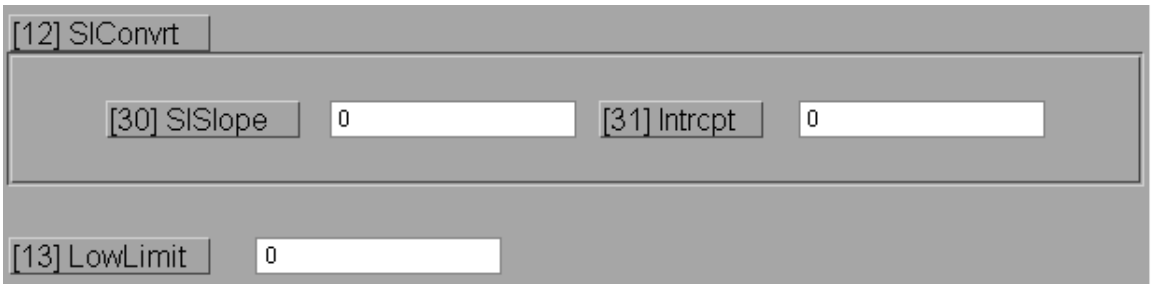

Figura 3.6: Esempio di interfaccia per Calibration TEDS

3.2 Struttura del codice

Il corpo principale dell'editor è costituito dalla *Case structure* necessaria per consentire l'inserimento di tutti campi richiesti dalla classe di TEDS selezionata. La *Case structure* è composta da quindici *case*, uno per ogni classe di TEDS e solamente il primo, il terzo e il quinto contengono i campi per la realizzazione rispettivamente delle classi MetaTEDS, ChanTEDS e CalTEDS. Gli altri campi sono lasciati vuoti per estensioni future ai rimanenti TEDS previsti dallo standard. La *Case structure* viene comandata dal sottocampo *Class* implementato all'esterno della struttura insieme ai sottocampi *IEEE 1451*, *Version* e *Touple Lenght* necessari per costruire il campo *TEDSID*. La *Case structure* principale raccoglie le informazioni necessarie alla costruzione del TEDS, una per ogni campo presente, e le inserisce ordinatamente all'interno di un array binario che verrà poi fornito come unico parametro di uscita della struttura. Accanto alla struttura principale ne esiste una seconda, composta da più *Case structure* nidificati, che si attiva solamente quando si decide di operare nella modalità di inizializzazione dei campi attraverso l'uso del bottone *Open TEDS*. Se attivata, una volta selezionato il file da aprire, questa seconda struttura provvede prima di tutto a verificare l'esistenza del file, successivamente ne calcola il checksum, se corretto procede con la lettura dei valori all'interno del file e a inizializzare con questi i campi della struttura principale compresi anche i sottocampi del campo *TEDSID* esterni ad essa.

3.2.1 Codice per il salvataggio dei TEDS

La funzione di salvataggio dei TEDS è realizzata con una serie di *Case structure* nidificati. Essi si rendono necessari per gestire le diverse situazioni, nel caso si voglia annullare l'operazione di salvataggio, oppure nel caso il file è già esistente e ci si vuole assicurare delle intenzioni dell'utilizzatore. L'unico parametro d'ingresso della funzione di salvataggio è l'array contenente tutti i valori dei campi del TEDS, che coincide con il parametro di uscita fornito dalla struttura principale dell'editor, memorizzati all'interno di strutture TLV come specificato dallo standard IEEE

1451.0. La funzione provvede a creare un file con il nome e la locazione specificata precedentemente nella finestra di dialogo, successivamente utilizza tale file come destinazione per scrivere in forma binaria il contenuto dell'array passato come parametro di ingresso, infine, una volta completata questa operazione, il file viene chiuso e l'operazione di salvataggio si conclude. In Figura 3.7 è riportato il codice LabVIEW per il salvataggio di un nuovo TEDS, mentre in Figura 3.8 è riportato quello per sovrascrivere un TEDS già esistente.

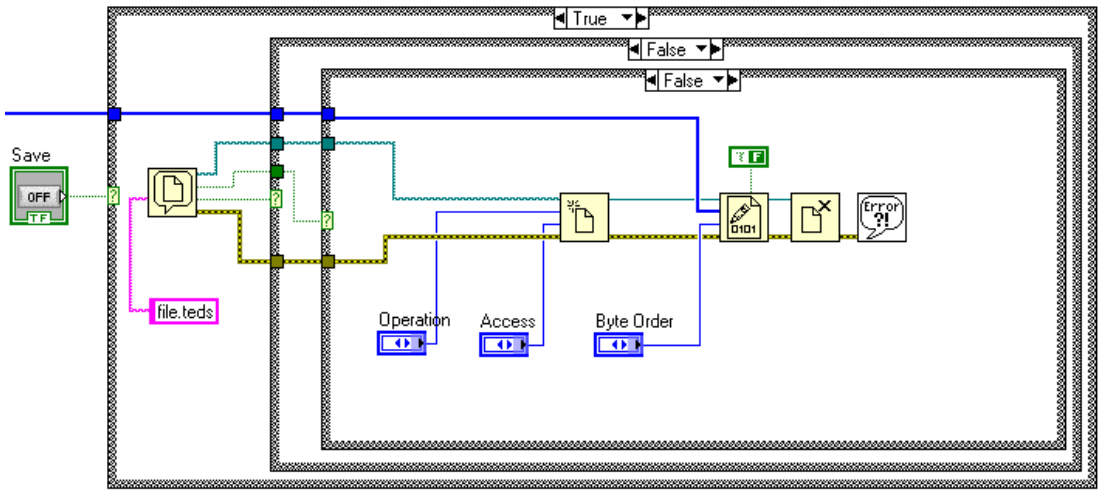

Figura 3.7: Codice per il salvataggio di un nuovo TEDS

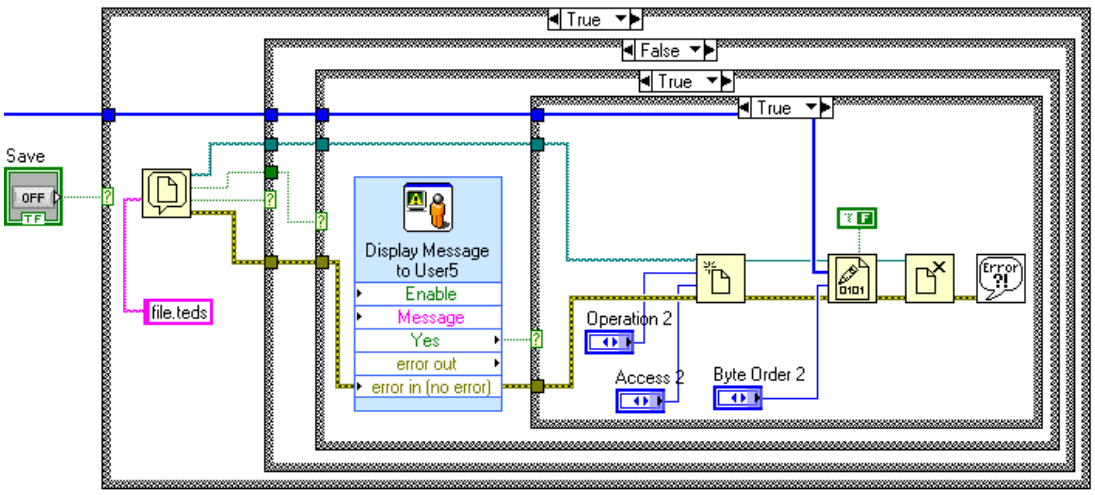

Figura 3.8: Codice per sovrascrivere un TEDS già esistente

3.2.2 Codice per il calcolo del checksum

La procedura di calcolo del checksum del TEDS viene utilizzata sia dall'editor di scrittura sia da quello di lettura di file TEDS. Nel primo caso è utilizzata per calcolare il checksum e successivamente memorizzare il risultato negli ultimi due byte del file, nel secondo caso è utilizzata per ricalcolare il checksum e confrontare il risultato con gli ultimi due byte del file; se i valori corrispondono allora significa che il file non è danneggiato. La realizzazione della procedura consiste in un ciclo *While* che itera N volte, dove N è il numero di byte occupati dal file meno i due byte dedicati a memorizzare il checksum, e calcola la somma totale dei valori contenuti in ciascun byte grazie all'utilizzo del meccanismo di *Shift register*, dopo di che si effettua la differenza tra la costante esadecimale 0xFFFF e il risultato ottenuto precedentemente ottenendo così il valore di checksum del file. Questa procedura non è altro che la realizzazione pratica, con gli strumenti messi a disposizione dall'ambiente di programmazione grafico LabVIEW, della formula fornita dallo standard IEEE 1451.0 per il calcolo, riproposta per questo qui di seguito.

$$\text{checksum} = \text{0xFFFF} - \sum_{i=1}^{TotaleByte-2} \text{TEDSByte}(i)$$

In Figura 3.9 è riportato il codice LabVIEW per il il calcolo del valore di checksum.

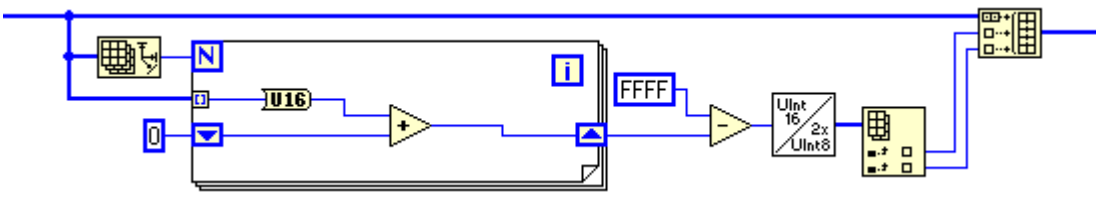

Figura 3.9: Codice per il calcolo del valore di checksum

3.2.3 Codice per la lettura dei TEDS

Oltre al file *TEDSEditor.vi*, è stato sviluppato un secondo programma all'interno del file *TEDSReader.vi* il quale permette la lettura dei *virtual TEDS*. Tale programma si basa sullo stesso codice utilizzato per consentire la modalità operativa di inizializzazione dei campi all'interno dell'editor, unica differenza sta su dove le variabili lette dal file binario vengono scritte. Nel caso dell'editor per la sola lettura dei TEDS, i valori vengono scritti all'interno di normali campi utilizzati come semplici indicatori e quindi disabilitati alla scrittura da parte dell'utente. Nel caso dell'editor per la compilazione dei TEDS i valori vengono scritti all'interno dei parametro *value* del *proprety node* associato a ciascun campo dell'interfaccia dell'editor, consentendo in questo modo di modificare i valori di inizializzazione di ciascun campo con quelli letti dal file binario.

Capitolo 4

Esempio di virtual TEDS

Il prodotto finale dell'editor per la creazione di *virtual TEDS* consiste in un file binario che contiene tutte le informazioni necessarie per potersi interfacciare con un dispositivo, ad esempio un *TransducerChannel*, memorizzate all'interno di una struttura gerarchica definita dallo standard IEEE 1451.0. Per comprendere meglio ciò che fisicamente è un *virtual TEDS*, e come effettivamente vengono memorizzate le informazioni al suo interno, viene ora riportato un esempio di creazione di un ChanTEDS e del relativo file binario che l'editor produrrà a seguito del salvataggio del TEDS.

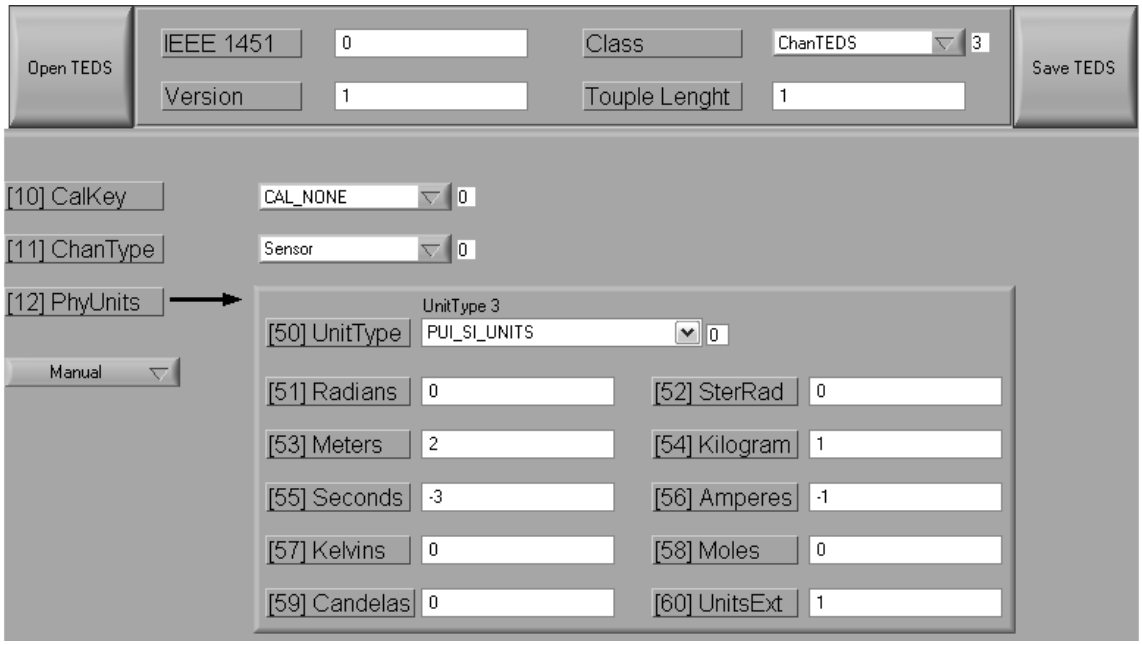

Figura 4.1: Esempio di creazione di un ChanTEDS

In Figura 4.1 vengono mostrati i valori immessi all'intero dei primi quattro campi di un TransducerChannel TEDS con l'editor sviluppato in LabVIEW. Qui sotto invece è riportato il contenuto del file binario prodotto dall'editor in seguito al salvataggio del file, dove ogni campo e sottocampo è stato distribuito su di una riga diversa per facilitare la comprensione, nella realtà il valori non sono separati da spazi o invii ma sono tutti consecutivi.

```
1   00 00 00 F4                  |  0  0  0  244
2   03 04 00 03 01 01            |  3  4  0  3  1  1
3   0A 01 00                     |  10  1  0
4   0B 01 00                     |  11  1  0
5   0C 21                        |  12  33
6       32 01 00                 |      50  1  0
7       33 01 80                 |      51  1  128
8       34 01 80                 |      52  1  128
9       35 01 84                 |      53  1  132
10      36 01 82                 |      54  1  130
11      37 01 7A                 |      55  1  122
12      38 01 7E                 |      56  1  126
13      39 01 80                 |      57  1  128
14      3A 01 80                 |      58  1  128
15      3B 01 80                 |      59  1  128
16      3C 01 01                 |      60  1  1
```

Il listato qui sopra riportato è suddiviso in due parti, quella a sinistra esprime il file binario con rappresentazione esadecimale, quella a destra invece con rappresentazione decimale. Come anticipato precedentemente in ogni riga è riportato un campo o un sottocampo, in particolare:

- **Riga 1:** Contiene la lunghezza complessiva del TEDS, ovvero 244 byte.
- **Riga 2:** Rappresenta il campo *TEDSID* all'interno della struttura TLV, il primo byte infatti vale 3 ed indica appunto l'identificativo del campo, il secondo byte vale 4 ed indica la dimensione del successivo parametro *value* che vale precisamente 00 03 01 01, il primo byte rappresenta il sottocampo *IEEE 1451*, il secondo il campo *Class*, il terzo il campo *Version* e infine il quarto che rappresenta il sottocampo *Touple Lenght*.
- **Riga 3:** Rappresenta il campo *CalKey* memorizzato all'interno della struttura TLV, possiamo facilmente vedere come questo campo abbia identificativo 10, dimensione 1 e valore 0.

- **Riga 4:** Rappresenta il campo *ChanType*, anche in questo caso, conoscendo la struttura TLV possiamo capire il numero di identificativo, 11, la sua dimensione, 1, e il valore contenuto, 0.

- **Riga 5:** In questo caso viene riportato solo l'identificativo e la dimensione della struttura TLV appartenente al campo *PhyUnits*, il campo *value* è suddiviso nelle successive 11 righe, una per ciascun sottocampo appartenente a *PhyUnits*.

- **Riga 6:** Sottocampo *UnitType* con identificativo 50, dimensione 1 e valore 0.

- **Riga 7 - 15:** Righe dedicate nel seguente ordine ai sottocampi *Radiants*, *SterRad*, *Meters*, *Kilogram*, *Seconds*, *Amperes*, *Kelvins*, *Moles* e *Candelas*. Per questi campi il valore da memorizzare viene prima moltiplicato per 2 e poi sommato a 128, per cui il valore decimale 128 in realtà corrisponde a 0, il valore 132 a 2, 130 a 1, 122 a -3 e infine 126 a -1.

- **Riga 16:** Sottocampo *UnitsExt* con identificativo 60, dimensione 1 e valore 1.

Capitolo 5

Conclusioni

Lo sviluppo di questa tesina si è basato sullo standard IEEE 1451.0; esso si è dimostrato chiaro e preciso per quel che riguarda la definizione di ogni singolo campo appartenente ad una delle tre classi di TEDS implementate in questo lavoro, ma purtroppo tralascia un pò la parte di documentazione relativa all'organizzazione dei campi all'interno del file binario che andrà a costituire il *virtual TEDS*.

Per quanto concerne l'editor in LabVIEW, esso pone solide basi per la costruzione di un editor più grande che possa supportare tutte le classi di TEDS, prestandosi quindi a futuri sviluppi e miglioramenti sia dal punto di vista delle funzioni offerte, sia dal punto di vista dell'interfaccia e quindi della sua usabilità. In particolare la struttura principale, composta da quindici *case*, di cui dodici vuoti, facilità l'implementazione di nuovi TEDS, che potrà essere ottenuta semplicemente implementando i campi del *Data Block* appartenente al TEDS all'interno del rispettivo *case* senza dover apportare altre modifiche al programma. Operazioni comuni infatti come per esempio il salvataggio, il calcolo del parametro *TEDS length* oppure il calcolo del checksum, essendo comuni a tutti i TEDS, non dovranno essere riscritte. Queste tre procedure infatti sono implementate al di fuori della struttura principale e operano indipendentemente dal tipo di TEDS che si vuole scrivere con l'editor. Per consentire l'uso della modalità di inizializzazione dei campi dell'editor, leggendo i valori da un TEDS precedentemente creato, è necessario che nel rispettivo *case* della struttura secondaria dell'editor si implementi il codice necessario per la lettura di ogni singolo campo del *Data Block*, questo codice potrà poi essere

riutilizzato per implementare il TEDS all'interno del programma *TEDS Reader* e perciò estendere le sue funzionalità con la lettura di una nuova classe di TEDS. Il riutilizzo del codice all'interno di entrambi i programmi è facilitato dall'utilizzo di numerose *SubVI*, soprattutto per quel che riguarda la scrittura di variabili su più byte e la lettura di una variabile memorizzata all'interno di più byte.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] L. Benetazzo, C.Narduzzi, *Dispense di Misure per l'automazione e la produzione industriale - Prima parte*, Padova, 14 ottobre 2008.

[2] L. Benetazzo, C.Narduzzi, *Dispense di Misure per l'automazione e la produzione industriale - Seconda parte*, Padova, 15 dicembre 2008.

[3] IEEE Instrumentation and Measurement Society, *IEEE Std 1451.0 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication, Protocols, and Transducer Electronic Data Sheet (TEDS) Formats*, IEEE, New York, 21 Settembre 2007.

[4] R. Bitter, T. Mohiuddin, M. Nawrocki, *LabVIEW Advanced Programming Techniques - Second Edition*, CRC Press - Taylor & Francis Group, Illinois, 2007.