Capitolo V

Progettazione e realizzazione

di alcuni Moduli Test

La progettazione del Test Bench per gli SCM di ANTARES descritta nel capitolo precedente costituisce la base per la realizzazione di questo apparato. Alla sua costruzione il Gruppo di Bari sta procedendo, mentre nell'economia del presente lavoro di tesi è parso opportuno completare la progettazione stessa con la realizzazione in laboratorio di:

- la procedura di comunicazione dei moduli Acoustic Transponder e Pressure Sensor;
- la procedura di comunicazione dei moduli Laser Beacon e SPM;
- la procedura di controllo delle caratteristiche di un segnale periodico (ad esempio il segnale di clock).

Sono tre componenti qualificanti il dispositivo in realizzazione i cui risultati sono riportati in questo capitolo.

Questo capitolo è dunque dedicato alla descrizione del lavoro svolto per realizzare le parti del Test Bench indicate. Tali moduli sono stati implementati mediante l'uso dell'ambiente di programmazione grafico LabView.

5.1 Progettare con LabView

Come è stato illustrato nel precedente capitolo lo sviluppo della fase di test deve realizzarsi attraverso l'esecuzione di un programma da parte di un computer di test. Tale programma deve avviare le varie fasi di test secondo un ordine prestabilito, deve gestire la comunicazione dei dati da/verso l'SCM, deve permettere la conservazione di tutti i dati trattati e i risultati delle varie fasi e deve infine darne una rappresentazione grafica.

La soluzione scelta per soddisfare tutte le esigenze sopraelencate è quella dell'utilizzo di LabView. LabView è un ambiente di programmazione di tipo grafico prodotto da National Instruments, ed orientato principalmente allo sviluppo di applicazioni software per la gestione di strumenti di misura. La sua flessibilità è tale da permettere ai programmatori lo sviluppo di software di impiego più generale e non necessariamente limitato alla strumentazione numerica; di fatto LabVIEW si propone come valida alternativa ai compilatori tradizionali. Anche se questo nuovo editor trae molte delle sue caratteristiche dai linguaggi di programmazione, esso offre notevoli innovazioni. Queste risiedono essenzialmente nella possibilità di programmare non più con un linguaggio dalla sintassi specifica ma tramite "icone". Questa caratteristica permette a questo nuovo ambiente di sviluppo di essere più intuitivo per l'utente e quindi di rendere più semplice e più veloce la costruzione di programmi sia semplici che complessi. La potenzialità di questo ambiente di programmazione si basa principalmente sulla comunicazione a flusso di dati. Questa consente un continuo aggiornamento

dei dati forniti da una scheda di acquisizione al software oppure la possibilità di fornire dati ad un dispositivo hardware, come nel nostro caso, per l'automazione di un sistema. La programmazione a flusso di dati permette anche l'esecuzione in contemporanea di più operazioni matematiche complesse, rendendo molto più veloce il funzionamento del programma. Quest'ultima caratteristica dota il LabView di una funzionalità maggiore rispetto ai linguaggi di programmazione tradizionali che eseguono operazioni sequenzialmente .

LabView permette di creare degli strumenti virtuali (VI) di cui il monitor costituisce il pannello, e il computer stesso costituisce il cuore dello strumento, eseguendo le operazioni definite all'interno di uno schema a blocchi di definizione.

Per realizzare un generico programma si utilizzano alcune funzioni di libreria che sono i costituenti fondamentali degli strumenti virtuali: questa organizzazione del software permette di costruire un VI (Virtual Instrument) avente solo le funzionalità desiderate tralasciando quelle non necessarie in una determinata situazione.

LabView può essere usato per rilevare, durante una esperienza di laboratorio, il comportamento di alcune grandezze fisiche, in modo automatico, utilizzando un computer. In questo modo i dati potranno poi essere elaborati e visualizzati.

La scelta dell'ambiente LabView per la gestione e l'organizzazione sequenziale delle prove di test da effettuare e la realizzazione dei programmi che gestiscono i moduli costituenti il Test Bench può essere giustificata per i seguenti

motivi:

- possibilità di realizzare velocemente programmi che gestiscono una sezione di misura;
- ampia disponibilità di librerie predisposte per l'uso di strumentazione;
- possibilità di ottenere una struttura modulare dei programmi;
- possibilità di realizzare un'interfaccia grafica particolarmente accurata e di facile accesso per la rappresentazione dei dati.

5.1.1 Introduzione al software

Un generico VI LabView è costituito da due parti:

- il pannello frontale
- il diagramma a blocchi.

Il pannello frontale agisce da interfaccia con l'utente (figura 5-1) mentre il diagramma a blocchi definisce il principio di funzionamento dello strumento e corrisponde al codice sorgente di un programma eseguito utilizzando i metodi standard di programmazione. Il pannello frontale può essere costruito graficamente mediante componenti predefiniti selezionabili nella finestra Controls.

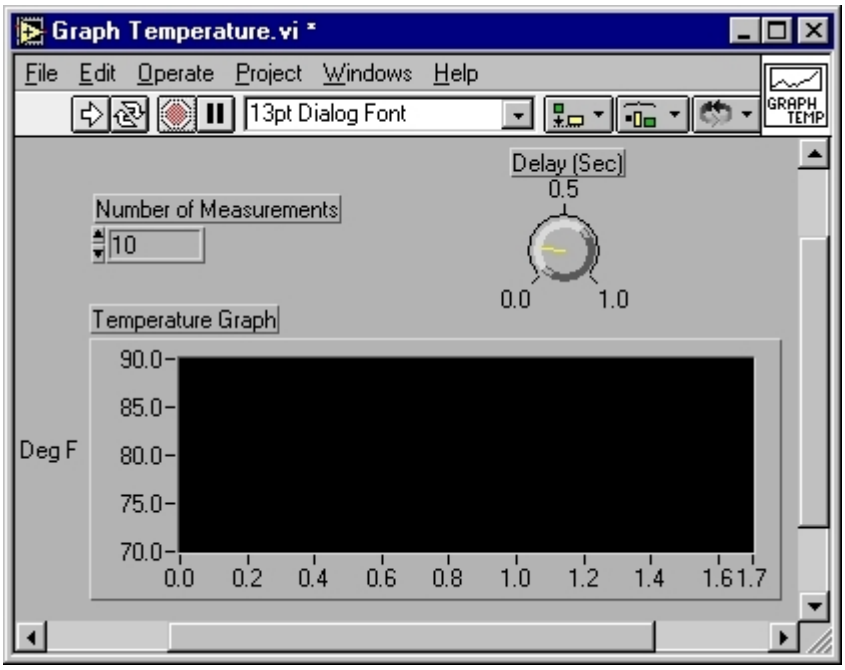

Figura 5-1. Immagine del pannello frontale di un generico strumento virtuale.

In questa finestra i vari oggetti sono rappresentati mediante dei simboli grafici e possono riguardare sia componenti meccanici (interruttori, deviatori) che elettronici (display digitali e analogici, registratori, ecc). Ciascuna icona presente nella finestra dei componenti predefiniti (figura 5-2) rappresenta un gruppo di oggetti di caratteristiche similari. Inoltre è possibile anche inserire dei componenti speciali (pulsanti particolari, indicatori grafici, ecc.), creati all'interno di Labview mediante un editor grafico oppure utilizzando un programma di grafica esterno.

A seconda della funzione prevista, gli oggetti inseriti in un pannello possono inoltre essere suddivisi in controlli e indicatori. Mentre i primi servono per immettere dei dati o impostare dei parametri, i secondi servono per visualizzare dei risultati. Per definire la caratteristiche di ciascun oggetto inserito nel pannello frontale, basta premere col pulsante destro all'interno dell'oggetto.

Sarà così possibile definire il tipo di dato (intero, decimale, booleano), l'intervallo entro cui esso può variare, l'unità di misura, ecc.

Figura 5-2. Finestra dei componenti predefiniti di LabView.

Il diagramma a blocchi sostituisce il codice sorgente di un programma creato con uno dei linguaggi tradizionali (figura 5-3).

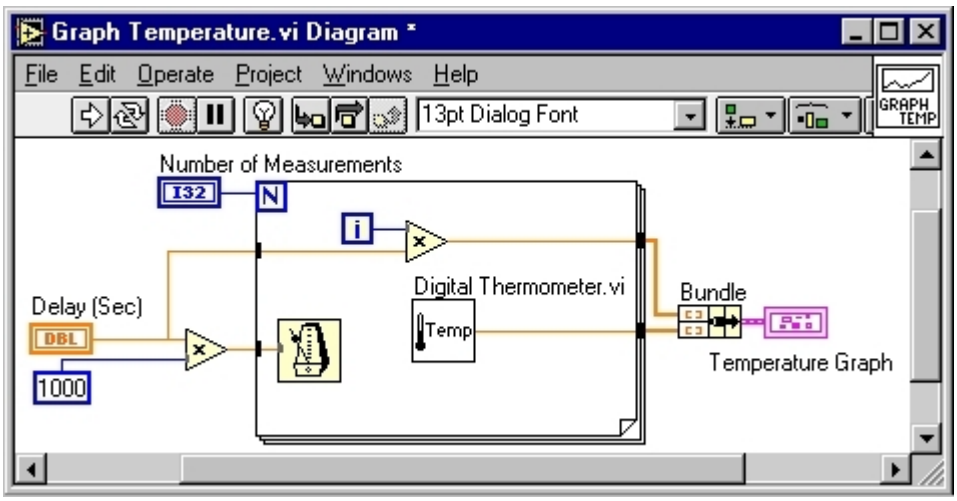

Figura 5-3. Diagramma a blocchi relativo al pannello frontale di figura 5-1.

Nel diagramma compaiono dei simboli grafici detti nodi, collegati tra di loro mediante delle linee o fili che rappresentano il percorso seguito dal segnale (o

dati). Ciascun nodo svolge una determinata funzione.

Il risultato di una determinata funzione viene inviato al nodo successivo tramite il filo che collega tra di loro i due nodi. Per inserire i vari nodi (o funzioni) che dovranno elaborare i dati provenienti dai controlli, si fa clic all'interno della finestra Functions (figura 5-4).

Figura 5-4. Finestra Function.

Le varie funzioni previste in LabView vengono rappresentate mediante diversi simboli grafici raggruppati per tipo (l'elenco rispetta l'ordine mostrato nella figura 5-4 da sinistra verso destra e dall'alto verso il basso):

- strutture corrispondenti alle istruzioni di ripetizione di un determinato gruppo di comandi (if, for, while, case ecc.);
- operazioni aritmetiche (somma, sottrazione, moltiplicazione, sommatoria,

reciproco, funzioni trigonometriche, ecc.);

- operazioni logiche (AND, OR, EOR, NOT, NAND, NOR, ecc.);
- operazioni su stringhe (lunghezza, concatenazione, estrazione, confronto, conversione, ecc.);
- operazioni su matrici e su vettori (costruzione di una matrice, sostituzione e inserzione di un elemento, valore minimo o massimo, ecc.);
- operazioni di confronto (uguale, diverso, maggiore, minore, ecc.);
- funzioni tempo (cronometro, intervallo di tempo, data e ora, ecc.);
- operazioni su file (memorizzazione e lettura dati);
- operazioni di comunicazione;
- gestione del collegamento con uno strumento via IEE488, RS232 o altro;
- gestione acquisizione dati da una scheda di acquisizione (configurazione scheda, ingresso e uscita analogica e digitale, ecc.);
- funzioni matematiche e statistiche per la elaborazione dei dati.

La finestra Tools contiene degli strumenti che permettono di eseguire delle operazioni di editing sugli oggetti presenti all'interno delle due finestre relative al pannello di controllo e al diagramma a blocchi (figura 5-5). Inoltre sono presenti anche dei simboli che servono per controllare il funzionamento di uno strumento.

Figura 5-5. Finestra Tools.

L'avvio di un programma può avvenire in modo 'run' (primo bottone in alto del pannello frontale di figura 5-1) o in modo 'run continuously' (secondo bottone), mentre l'arresto del programma e di tutte le sue funzioni avviene mediante un vistoso tasto di stop di colore rosso 'abort execution' (terzo bottone).

5.2 Comunicazioni del modulo Local Interfaces

La realizzazione del modulo Local Interfaces, come già è stato descritto nel precedente capitolo prevede la creazione di un certo numero di connettori che devono essere connessi all'SCM. Per favorire la chiarezza espositiva preferisco riepilogare tali connessioni:

- collegamento con il Pressure sensor che trasmette attraverso lo standard RS232;
- collegamento con il Sound Velocimeter che trasmette attraverso lo standard RS232;
- collegamento con l'SPM che trasmette attraverso lo standard RS485 con protocollo MODBUS;

- collegamento con il Laser Beacon che trasmette attraverso lo standard RS485 con protocollo MODBUS;
- collegamento con il Backplane dell'SCM che trasmette attraverso lo standard RS485 con protocollo MODBUS.

Tali connessioni devono essere tutte indipendenti in quanto, come già detto nel precedente capitolo, il test deve avvenire in completa autonomia e senza alcun intervento manuale nella modifica delle connessioni. Dunque mentre sarebbero necessarie solamente 2 connessioni (una con standard RS232 ed una RS485) nel caso in cui le connessioni possano essere modificate, occorrono invece almeno 5 connessioni per realizzare un test completamente autonomo.

5.3 Procedura di comunicazione dei moduli Pressure Sensor e Sound Velocimeter

5.3.1 Descrizione dei sensori

Ho ritenuto conveniente accorpare la descrizione e la realizzazione dei sensori Pressure Sensor e Sound Velocimeter in quanto hanno entrambi le stesse caratteristiche costruttive e di comunicazione.

Come si è avuto modo di osservare nei precedenti capitoli il Pressure Sensor è un modulo esterno all'SCM che permette la misura della pressione idrostatica. L'utilità di questo modulo è duplice, in quanto permette il

monitoraggio delle condizioni ambientali del telescopio e soprattutto permette un accurato calcolo delle posizioni degli OM attraverso il sistema di posizionamento acustico.

Il sensore è racchiuso in un involucro cilindrico in lega di titanio del diametro di 8 cm e lunghezza 31.6 cm (figura 5-6) e presenta 6 contatti di cui ne sono utilizzati 4: 2 per l'alimentazione (48 V con 50 mA di assorbimento) e 2 per la trasmissione RS232. Un cavo speciale, provvisto di contatti SUBCONN IL6F e SUBCONN IL6M (maschio e femmina), garantisce questi collegamenti tra il sensore e il modulo SCM.

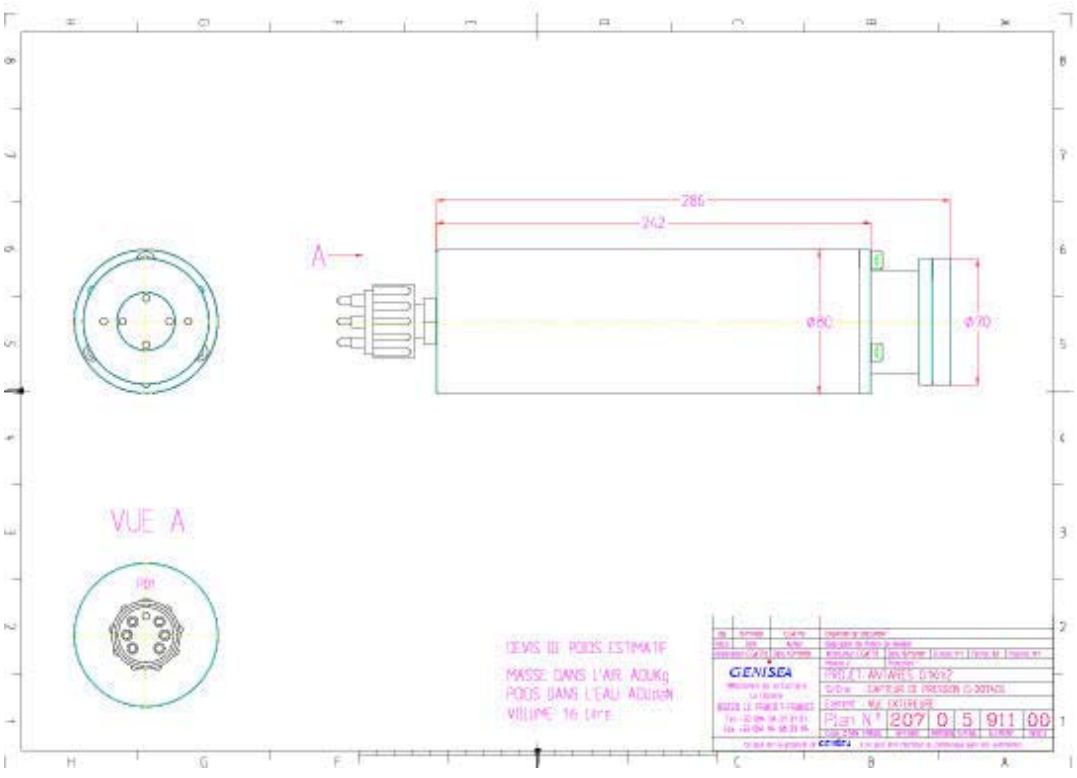

Figura 5-6. Disegno del Pressure Sensor.

Il Sound Velocimeter permette la misura della velocità del suono nell'acqua indispensabile per un accurato calcolo della forma assunta dalla

stringa. Il contenitore ha una forma simile a quella del Pressure Sensor come è possibile osservare dalla figura 5-7, e stesso diametro ma la lunghezza è maggiore (40,9 cm). Il numero dei contatti di collegamento e la loro funzione è identica a quella indicata per il Pressure Sensor.

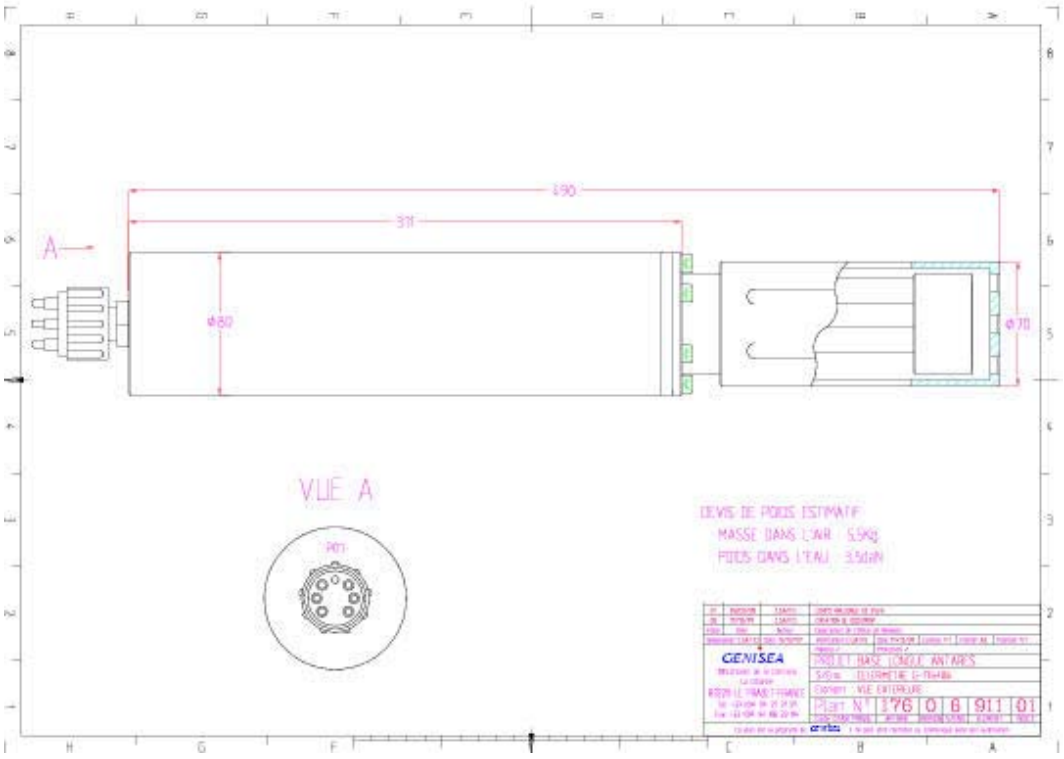

Figura 5-7. Disegno del Sound Velocimeter.

Le comunicazioni tra l'SCM ed i sensori avvengono attraverso questo protocollo:

Comando dell'SCM:

CR

la risposta del Pressure Sensor è di 15 caratteri:

:PR= xxx.xxx CRLF

la risposta del Sound Velocimeter è di 16 caratteri :

:CE= xxxx.xxx CRLF

dove:

xxx.xxx valore della pressione in bar nel range [0,250] per il Pressure Sensor

xxxx.xxx valore della velocità del suono nell'acqua in m/s nel range [1400,1600] per il Sound Velocimeter;

CR = Carriage Return che nel codice ASCII (American Standard Code for Information Interchange) è *$0D* ;

CRLF = Carriage Return, Line Feed che nel codice ASCII è *$0D $0A*.

La velocità del flusso di dati è di 19200 bytes/sec con frame di 8 bits, con 1 bit di start, 1 di stop e nessun controllo di parità.

5.3.2 Realizzazione dei simulatori

Le connessioni che simulano il funzionamento dei sensori descritti sono le più facili da realizzare in quanto porte RS232 sono disponibili su tutti i personal computer commerciali.

Dunque per simulare i sensori Pressure Sensor e Sound Velocimeter occorre collegare due cavi seriali RS232 provenienti dalle porte COM1 e COM2 ai connettori opportuni dell'SCM e rispondere nel modo indicato al comando descritto dell'SCM. Dunque la simulazione dei due sensori è fatta attraverso un software che riceve e verifica il comando proveniente dall'SCM e fornisce in risposta la stringa di caratteri su indicata. Ho personalmente prodotto il software necessario attraverso l'ambiente di programmazione LabView versione 5.1.

Il protocollo che permette la comunicazione di questi moduli con l'SCM, descritto in precedenza, si basa sulla trasmissione di caratteri ASCII dunque è stata necessaria l'implementazione di due software VI: uno che simuli la presenza dell'SCM e l'altro che simuli la presenza di uno dei moduli Pressure Sensor o Sound Velocimeter. Entrambi i software sono stati caricati sul medesimo PC però ciascuno interagente con una porta seriale differente. Quindi i comandi sono stati trasmessi da una porta seriale all'altra e viceversa. Dunque sono state collegate insieme le due porte attraverso un cavo seriale, con l'accortezza di invertire i segnali di trasmissione e ricezione (pin 2 e 3) come mostra la figura 5-8.

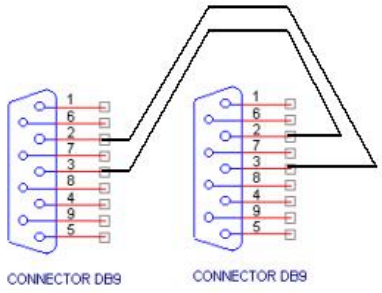

Figura 5-8. Connettori DB9 del cavo di comunicazione RS232 nel quale sono stati scambiati i segnali di ricezione e trasmissione ai pin 2 e 3.

Per la gestione di entrambi i software indicati è stato implementato un altro VI che avvia in tempi opportuni le due procedure. Dunque l'implementazione definitiva di questa parte del Test Bench sarà molto simile a quella implementata in quanto occorrerà solamente cambiare gli indirizzi delle porte seriali. Descrivo i programmi indicati nel paragrafo successivo.

5.4 Procedura di comunicazione dei moduli Laser Beacon e SPM

5.4.1 Descrizione del modulo Laser Beacon

La funzione del modulo Laser Beacon è stata ampiamente descritta nel capitolo III. In questo paragrafo perciò mi limito a descriverne le principali caratteristiche costruttive.

Il contenitore è in lega di titanio e di forma cilindrica, come mostra la figura 5-9, e le dimensioni sono: diametro 17 cm e lunghezza 75 cm.

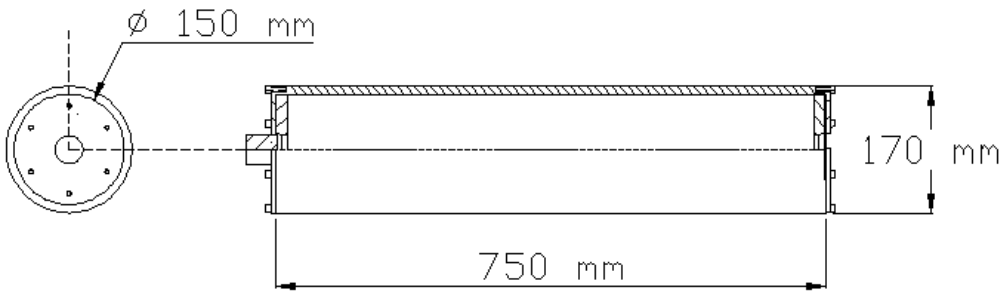

Figura 5-9. Descrizione geometrica del contenitore del Laser Beacon.

Il connettore è del tipo SUBCONN IL12F a 12 contatti di cui 8 sono utilizzati dai seguenti segnali:

2 per l'alimentazione (48 V);

2 per la comunicazione seriale;

2 per comandare l'accensione del laser;

2 per l'acquisizione del segnale proveniente dal PIN.

Il modulo possiede internamente delle schede elettroniche che permettono di svolgere alcune funzioni di monitoraggio e regolazione quali la

regolazione della tensione per le necessità delle schede, il controllo dell'alimentazione, l'accensione del laser, la misura di temperatura e umidità è inoltre prevista la possibilità di spegnere tutte le schede elettroniche e il laser tranne la scheda di comunicazione UNIV1. Il Laser Beacon è normalmente in questo stato di 'stand-by' finché non giunge una richiesta di calibrazione. La figura 5-10 mostra lo schema a blocchi di questa organizzazione.

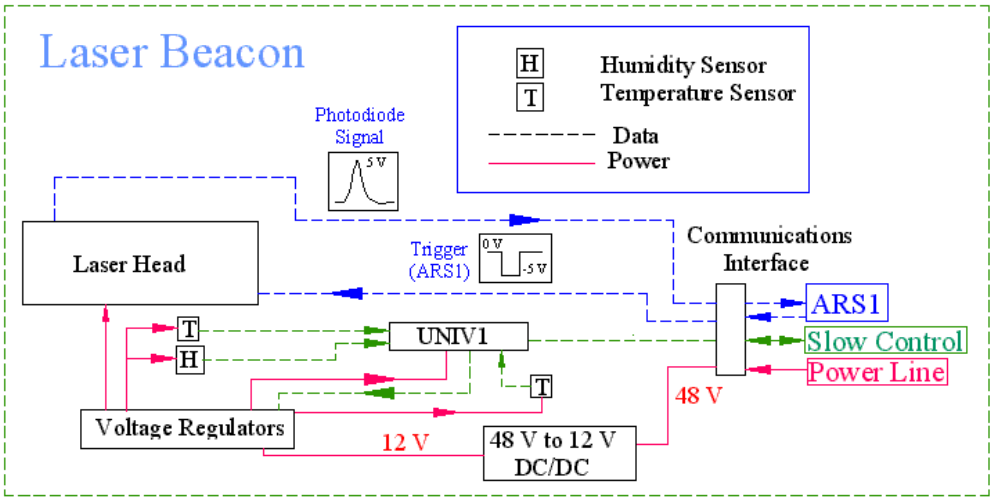

Figura 5-10. Descrizione dell'organizzazione delle schede elettroniche interne al Laser Beacon.

La comunicazione seriale avviene tramite standard RS485, ad una velocità di 19200 bytes/sec. Anche in questo caso le frames sono costituite da 8 bit con 1 bit di start, 1 di stop, e nessun controllo di parità. Il protocollo usato è il MODBUS con formato RTU (Remote Terminal Unit) [22].

5.4.2 Descrizione del modulo SPM

L'involucro dell'SPM è lo stesso utilizzato per l'SCM e per l'LCM. Contiene all'interno alcune schede elettroniche che permettono la regolazione e il

comando dell'alimentazione fornita verso tutti i settori della stringa e verso l'SCM, inoltre contiene delle schede che permettono la comunicazione con l'SCM che a sua volta fornisce i comandi provenienti dalla shore station circa le azioni da intraprendere per il controllo delle alimentazioni di tutta la stringa.

All'accensione di una stringa il modulo SCM è alimentato per primo perché esso comanda le azioni da intraprendere su tutta la stringa e dunque anche nell'SPM. Poiché l'alimentazione che proviene dalla JB ha una tensione di 1000 V AC essa non può essere indirizzata direttamente verso l'SCM pertanto deve essere prima ridotta e raddrizzata. Questo lavoro è svolto dall'SPM senza alcun monitoraggio. Tale alimentazione (48 V DC) può quindi alimentare l'SCM consentendo l'attivazione della scheda di controllo SCM_DAQ/SC. In questo modo la shore station può controllare tutta la stringa permettendo l'alimentazione e l'attivazione dei singoli settori. La comunicazione SCM – SPM avviene con stesse modalità e protocollo utilizzato per il Laser Beacon.

5.4.3 Descrizione del dispositivo UNIV1 e del protocollo supportato MODBUS

Prima di procedere con la descrizione della realizzazione dei simulatori dei moduli Laser Beacon e SPM ritengo sia utile illustrare brevemente il funzionamento del protocollo MODBUS della compagnia MODICON che è supportato dal processore PIC17C756 contenuto nella scheda UNIV1. Tale protocollo, come è stato detto nei capitoli precedenti, è utilizzato per le comunicazioni tra i moduli interessati e l'SCM e nella comunicazione seriale tra le schede elettroniche all'interno dell'SCM stesso.

L'UNIV1 contiene il circuito integrato MAX485 della compagnia Maxim (vedi figura 5-11) che permette l'interfaccia tra i pin del processore ed un bus standard RS485. Il twisted pair del segnale RS485 poi è connesso ai pin 19 e 20 del connettore J4 (vedi figura 5-11).

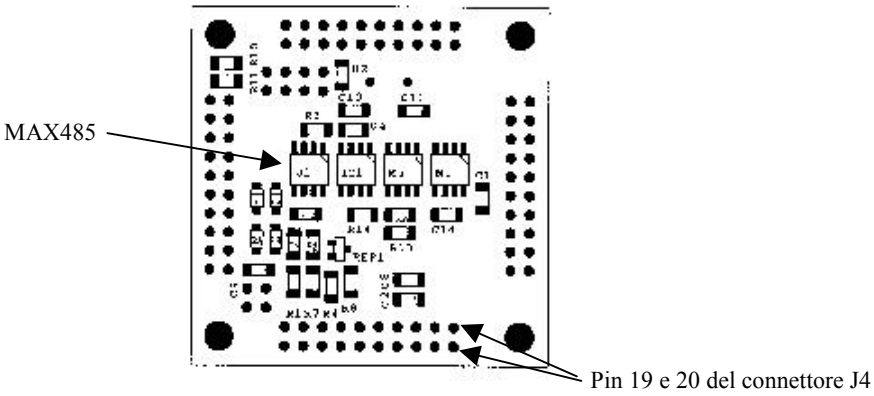

Figura 5-11. Disegno di implementazione della scheda UNIV1 (faccia inferiore).

L'UNIV1 così configurato tuttavia non permette la comunicazione con il PC per questo motivo è stato realizzata una interfaccia che converte lo standard RS485 nello standard RS232. Questa interfaccia di nome UNIVAPP (figura 5-12) contiene un UNIV1 (UNIV1 + interfaccia = UNIVAPP) e permette di comunicare attraverso la porta RS232 di un personal computer.

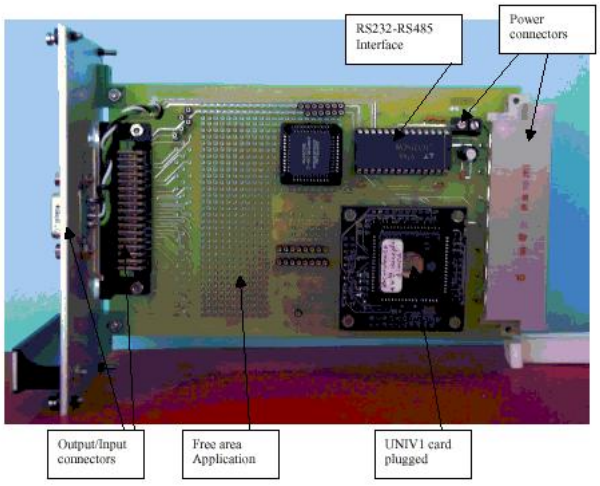

Figura 5-12. Fotografia del dispositivo UNIVAPP, è possibile notare la scheda UNIV1 montata in esso.

La conversione dello standard RS485 nello standard RS232 e viceversa è realizzata dall'interfaccia RS485 - RS232 segnalata in figura 5-12. Si tratta dell'IC LTC1334 della compagnia Linear Technology Corporation. Infatti mentre l'RS232 è un segnale asincrono che varia tra –12 V e + 12 V, l'RS485 è un segnale sincrono differenziale la cui tensione differenziale è minore di 5 V. In figura 5-13 è possibile osservare lo schema delle interconnessioni adottate.

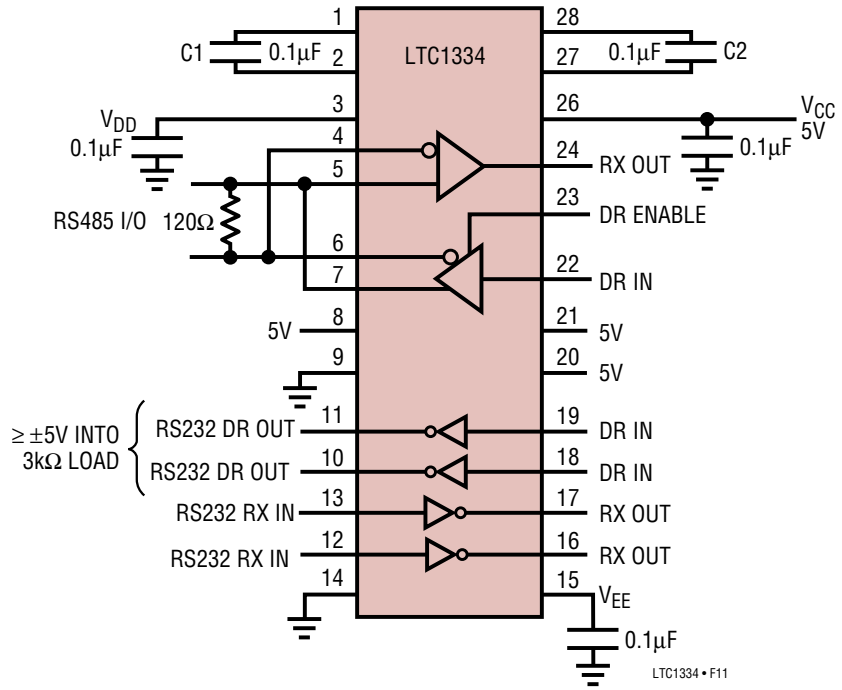

Figura 5-13. Schema delle interconnessioni del IC LTC1334.

Dunque il collegamento tra la scheda UNIV1 e il PC attraverso l'UNIVAPP è puntuale cioè il PC può essere connesso ad un solo UNIV1 (oppure al massimo a due utilizzando entrambe le porte COM1 e COM2). Tuttavia ci può essere la necessità di utilizzare molte schede UNIV1 che controllano diversi processi. Per far fronte a questa esigenza è stato realizzato un strumento, chiamato CRATE POWER (figura 5-14), che permette di contenere fino a 5 tra UNIVAPP

o BIDIDEV. Queste schede comunicano tutte attraverso un unico bus seriale con standard RS485 connesso attraverso una porta GPIB al personal computer.

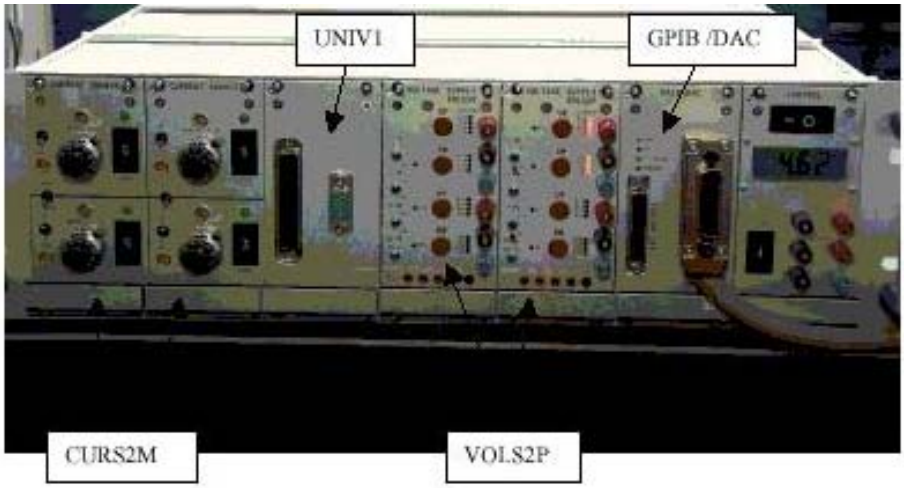

Figura 5-14. Fotografia del Crate Power.

In questa rete di processori il personal computer assume il ruolo di master e gli UNIV1 sono gli slave. In particolare in questa configurazione è possibile arrivare fino a 16 UNIV1 connessi tutti attraverso lo stesso RS485. Ciascuno slave ha un indirizzo che lo contraddistingue dagli altri dispositivi. Questo indirizzo è fissato dalla opportuna configurazione di 4 jumpers ($2^4 = 16$), indicati in figura 5-15.

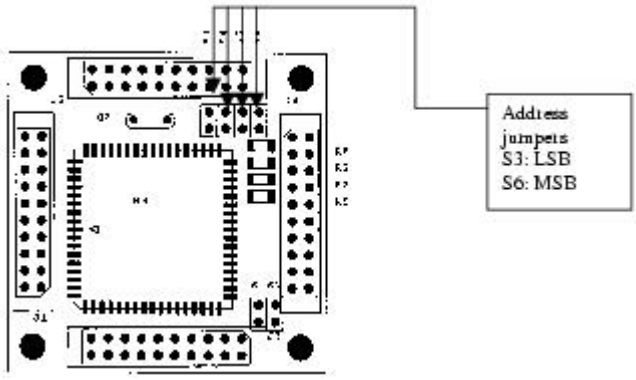

Figura 5-15. Disegno di implementazione della scheda UNIV1 (faccia superiore).

La trasmissione dei dati con il protocollo MODBUS avviene con un meccanismo di domanda (fatta dal master) e risposta (dello slave) come schematizzato in figura 5-16. In particolare il messaggio di trasmissione (sia del master che dello slave) è composto da una stringa che contiene i seguenti campi [28]:

- il primo campo contiene l'indirizzo dello slave (un byte) ;
- il secondo campo contiene il codice della funzione desiderata (un byte);
- il campo successivo contiene eventuali dati o codici di indirizzo a seconda della funzione considerata (numero di bytes variabile);
- l'ultimo campo è utilizzato per controllo degli errori (uno o due bytes come specificato in seguito).

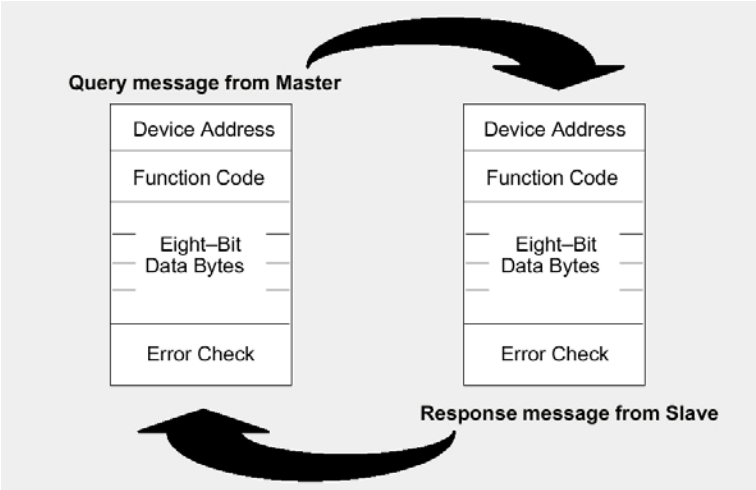

Figura 5-16. Ciclo di domanda-risposta del protocollo MODBUS.

Il protocollo MODBUS può utilizzare due modalità di trasmissione: ASCII o RTU.

Nel modo ASCII ogni byte del protocollo è inviato utilizzando due

caratteri ASCII. Ogni carattere interpreta una cifra esadecimale del byte. Ogni fame di dati inviati contiene 1 bit di start, 7 bit di dati, 1 bit di parità (se richiesto) e 1 bit di stop. Nel modo ASCII il controllo degli errori viene effettuato con un byte terminale che contiene LRC (Longitudinal Redundancy Check). Il byte LRC è dato dal complemento a due della somma di tutti i byte del messaggio. Quando il processore riceve la sequenza trasmessa ricalcola l'LRC e lo confronta con quello arrivato: una eventuale differenza segnala un errore di trasmissione.

Nella modalità RTU ogni byte del protocollo è inviato con un byte. Ogni frame di dati inviati contiene 1 bit di start, 8 bit di dati, 1 bit di parità (se richiesto) e 1 bit di stop. Il controllo degli errori avviene tramite 2 bytes contenenti il CRC.

Come detto, il secondo byte contiene la funzione specifica richiesta dal master. L'elenco dei codici di funzione utilizzati dall'UNIV1 è riportato in tabella 5-1.

Funzione	Codice
Read Boolean Output	01
Read Boolean Input	02
Read Output Register	03
Read Holding Register	04
Write Single Boolean Output	05
Write Single Holding Register	06
Write Multiple Boolean Output	0F

Tabella 5-1. Funzioni MODBUS implementate sull'UNIV1.

Nelle tabelle seguenti vengono descritti i registi di memoria relativi ai comandi elencati in tabella 5-1 presenti sull'UNIV1.

Boolean Output [0]	Port C
Boolean Output [1]	Port D
Boolean Output [2]	Port F
Boolean Output [3]	Port G
Boolean Output [4]	DDRC
Boolean Output [5]	DDRD
Boolean Output [6]	DDRF
Boolean Output [7]	DDRG
Boolean Output [8]	ADCON1
Boolean Output [9]	I2C Address
Boolean Output [10]	I2C Command
Boolean Output [11]	Boolean out I2C
Boolean Output [12]	I2C enable
Boolean Output [13]	Boolean out SPI
Boolean Output [14]	TCON1

Tabella 5-2. Boolean Output.

Boolean Input [0]	Error bit comunicaton
Boolean Input [1]	Port C
Boolean Input [2]	Port D
Boolean Input [3]	Port F
Boolean Input [4]	Port G
Boolean Input [5]	Boolean input I2C

Tabella 5-3. Boolean Input.

Holding Register [0..1]	DAC1
Holding Register [2..3]	DAC2
Holding Register [4..5]	Non usato
Holding Register [6..7]	Baud Rate
Holding Register [8..9]	Write HR I2C
Holding Register [10..11]	Timer Period
Holding Register [12..13]	Cycle ratio of Timer
Holding Register [14..15]	Cycle number of Timer

Tabella 5-4. Holding Register.

Input Register [0..1]	ADC0
Input Register [2..3]	ADC1
Input Register [4..5]	ADC2
Input Register [6..7]	ADC3
Input Register [8..9]	ADC4
Input Register [10..11]	ADC5
Input Register [12..13]	ADC6
Input Register [14..15]	ADC7
Input Register [16..17]	ADC8
Input Register [18..19]	ADC9
Input Register [20..21]	ADC10
Input Register [22..23]	ADC11
Input Register [24..25]	Read input I2C

Tabella 5-5. Input Register.

La velocità del flusso di dati deve essere la stessa per tutti i dispositivi che comunicano con il protocollo e può essere impostata scrivendo nel quarto registro degli Holding Registers (come mostra la tabella 5-4) il valore 0 (valore di defaut) per 9600 bit/s, 1 per 19200 bit/s o 2 per 38400 bit/s.

Prima di descrivere la realizzazione dei moduli simulatori dei moduli Laser Beacon e SPM è conveniente mostrare due esempi di comunicazione nel protocollo MODBUS – RTU. Questi sono utilizzati per manifestare le comunicazioni implementate sui moduli realizzati. La prima comunicazione riguarda la scrittura del dato ‘A5’ sul registro C contenuto negli Output Registers. La tabella 5-6 mostra il comando inviato dal master mentre la tabella 5-7 la risposta dello slave.

Significato	Valore
Indirizzo del dispositivo	09
Funzione	0F
Campo alto dell'indirizzo	00
Campo basso dell'indirizzo	00
Campo alto del n.ro di bit	00
Campo basso del n.ro di bit	08
N.ro di byte	01
Dato	A5
Campo basso del CRC	3F
Campo basso del CRC	48

Tabella 5-6. Comando inviato dal master nel primo esempio.

Significato	Valore
Indirizzo del dispositivo	09
Funzione	0F
Campo alto dell'indirizzo	00
Campo basso dell'indirizzo	00
Campo alto del n.ro di bit	00
Campo basso del n.ro di bit	08
Campo basso del CRC	55
Campo basso del CRC	45

Tabella 5-7. Risposta dello slave nel primo esempio.

Il secondo esempio descrive la lettura del registro scritto nella comunicazione precedente.

Significato	Valore
Indirizzo del dispositivo	09
Funzione	01
Campo alto dell'indirizzo	00
Campo basso dell'indirizzo	00
Campo alto del n.ro di bit	00
Campo basso del n.ro di bit	08
Campo basso del CRC	3C
Campo basso del CRC	84

Tabella 5-8. Comando inviato dal master nel secondo esempio.

Significato	Valore
Indirizzo del dispositivo	09
Funzione	01
N.ro di byte	01
Dato	A5
Campo basso del CRC	93
Campo basso del CRC	93

Tabella 5-9. Risposta dello slave nel secondo esempio.

5.4.4 Realizzazione dei simulatori dei moduli Laser Beacon e SPM

La comunicazione tra questi due moduli e l'SCM avviene tramite il protocollo MODBUS – RTU, secondo il protocollo master – slave descritto. In questa rete la funzione del master è svolta dall'SCM. Dunque per effettuare il test di questa funzionalità occorre simulare la presenza dei due moduli rispondendo correttamente ai comandi inviati dal master. Per simulare la presenza di un

dispositivo si può scegliere tra due alternative: la prima è quella di disporre di un hardware capace di supportare lo standard comunicativo, la seconda è quella di emulare la presenza dell'hardware attraverso un software inviando segnali direttamente dal PC. Mi è sembrato conveniente scegliere la seconda alternativa. Dunque la difficoltà principale è stata ricondotta nella realizzazione di un software che permetta di inviare stringhe corrette di bit conformemente al protocollo MODBUS – RTU che è decisamente più articolato del protocollo di comunicazione del Pressure Sensor e del Sound Velocimeter.

In analogia alla procedura di comunicazione realizzata per l'emulazione del Pressure Sensor e del Sound Velocimeter anche in questo caso è stata necessaria l'implementazione di due software VI: uno che assuma il comportamento dell'SCM e l'altro che assuma il comportamento dello slave (Laser Beacon, SPM o altro). Tuttavia, data la rigidità del protocollo questi due software non sono stati realizzati contemporaneamente. Infatti è stato necessario fare esperienza con il protocollo. Questo è stato ottenuto realizzando un VI che permetta di comunicare con un dispositivo slave che supporta già tale protocollo. Il dispositivo utilizzato a tale scopo è stato l'UNIVAPP di cui è dotato il Gruppo Test Bench (figura 5-17).

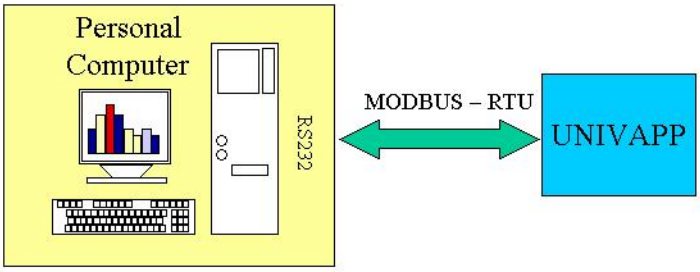

Figura 5-17. Comunicazione PC – UNIVAPP su protocollo MODBUS – RTU.

Ho poi verificato che il VI realizzato per assumere il comportamento del master, simulando la presenza dell'SCM, dialoghi correttamente con lo slave hardware UNIVAPP. Successivamente ho poi realizzato il secondo VI. Analogamente alla procedura di comunicazione realizzata per l'emulazione del Pressure Sensor e del Sound Velocimeter, il secondo VI è stato caricato sullo stesso PC dialogando attraverso le porte seriali (figura 5-18).

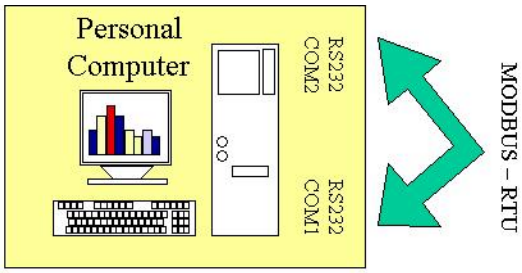

Figura 5-18. Comunicazione sullo stesso PC protocollo MODBUS – RTU.

Anche in questo caso è stato realizzato un terzo programma VI utilizzato per la gestione di entrambi i VI indicati. Un ulteriore VI permette di richiamare i VI relativi ai vari moduli.

Nelle figure seguenti è possibile osservare i pannelli frontali del software prodotto, mentre per i loro diagrammi a blocchi rimando all'appendice.

La figura 5-19 mostra uno schema a forma di albero nel quale è riassunta la dipendenza tra i vari VI a partire dalla radice rappresentata dal 'Local Interfaces'. Si possono notare due grandi rami: quello di destra è relativo al software di simulazione dei moduli che trasmettono attraverso caratteri ASCII (Sound Velocimeter e Pressure Sensor) mentre quello di sinistra è relativo ai moduli che trasmettono attraverso il protocollo MODBUS – RTU (Laser Beacon e SPM).

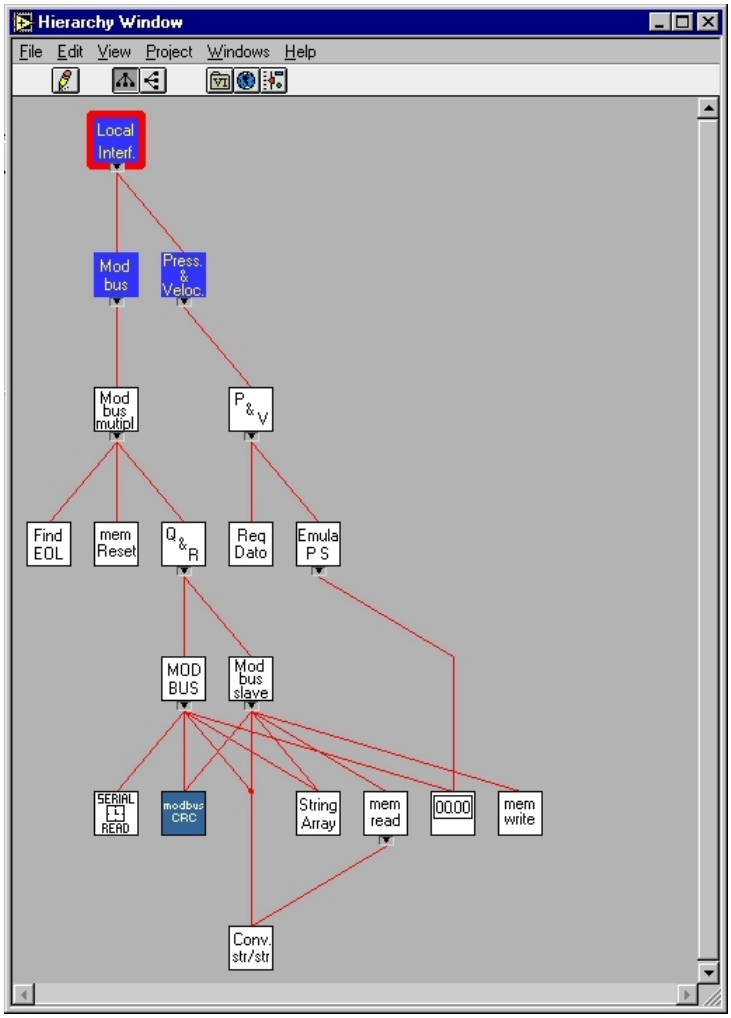

Figura 5-19. Gerarchia dei VI contenuti nel 'Local Interfaces'.

Nella figura 5-20 è illustrato il pannello frontale del VI 'Local Interfaces'.

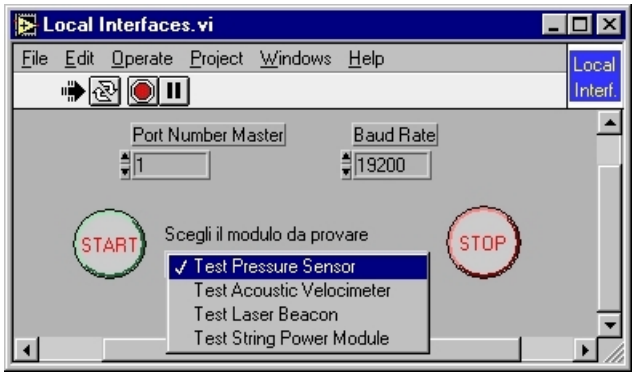

Figura 5-20. Pannello frontale dello strumento virtuale che riassume il software prodotto per la simulazione dei moduli locali.

Nella figura è possibile osservare un menu a tendina dove è possibile selezionare il modulo che si vuole simulare. Effettuata la scelta del modulo, della porta seriale del master (1 sta per COM2) e della velocità di trasmissione, l'esecuzione del programma avviene in seguito alla pressione del tasto 'START'. Se si vuole terminare l'esecuzione e uscire dal programma occorre premere il tasto 'STOP'.

Si supponga di aver selezionata la voce 'Test Pressure Sensor'. In seguito alla pressione del tasto START compare la finestra rappresentata in figura 5-21 corrispondente al VI 'Press & Veloc'.

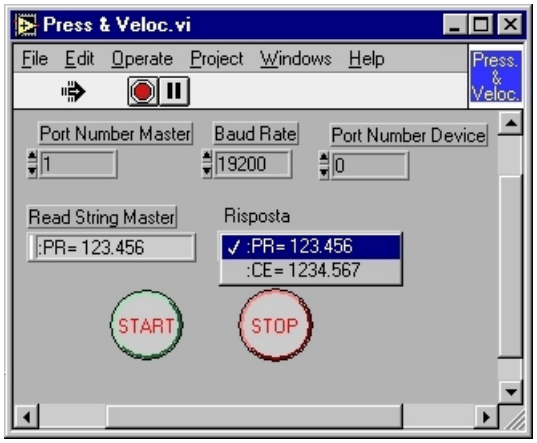

Figura 5-21. Pannello frontale e diagramma a blocchi del VI 'Press & Veloc'.

All'apertura del pannello frontale del 'Press & Veloc' viene subito inviato il comando *CR* da parte del simulatore dell'SCM che sarà illustrato più avanti. Nella figura 5-21 è possibile osservare i parametri: 'Port Numer Master' e 'Baud Rate' che vengono passati dal 'Local Interfaces'. In aggiunta è impostato il numero della porta seriale del simulatore del Pressure Sensor (0 sta per COM1) e la risposta da inviare in seguito al comando *CR* dal menu a tendina etichettato con 'Risposta'. Infatti il VI che permette la simulazione del Pressure Sensor e del Sound

Velocimeter è lo stesso mentre varia solamente la stringa di risposta. La stringa di caratteri sotto 'Read String Master' mostra i dati ricevuti dal simulatore dell'SCM.

L'attivazione del VI 'Press & Veloc' consente l'attivazione del VI 'Pressure & Velocimeter' in esso contenuto (figura 5-22). Tuttavia questo pannello frontale non viene mostrato in fase di esecuzione.

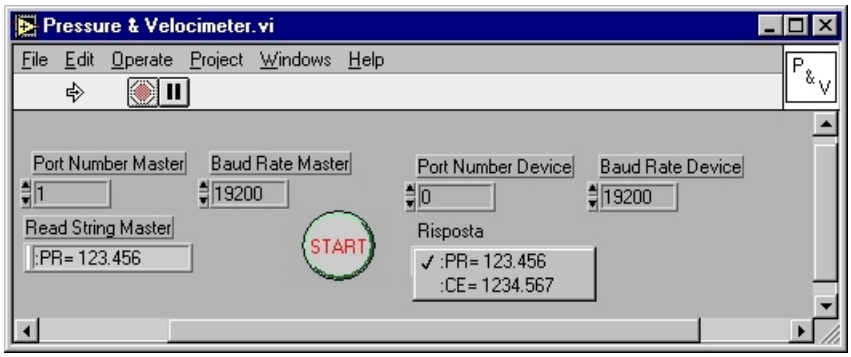

Figura 5-22. Pannello frontale del VI 'Pressure & Velocimeter'.

Il VI 'Pressure & Velocimeter' gestisce il VI 'Emulatore Pressure Sensor1' e il VI 'Richiesta1' mostrati in figura 5-23 e 5-24. Il primo è il simulatore del modulo Pressure Sensor o Sound Velocimeter, il secondo è il simulatore dell'SCM. Entrambi non sono visibili in fase di esecuzione.

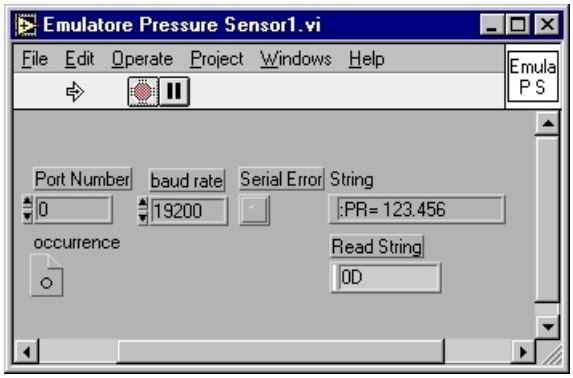

Figura 5-23. Pannello frontale del VI 'Emulatore Pressure Sensor1'.

Il VI 'Emulatore Pressure Sensor1' contiene i parametri Port Number

(del modulo) e baud rate già visti. Inoltre contiene l'indicazione della stringa ricevuta ('Read String') che assume il valore '0D' espresso in esadecimale (corrispondente al codice ASCII *CR*) e della stringa da porre in uscita ('String'). Gli ultimi due oggetti sono l'indicazione di un risultato binario corrispondente alla situazione di errore nella lettura dalla porta seriale e il controllo di un segnale si sincronizzazione ('occurrence') che è passata dal VI 'Pressure & Velocimeter' ai fini di sincronizzare la lettura e la trasmissione sulle porte seriali.

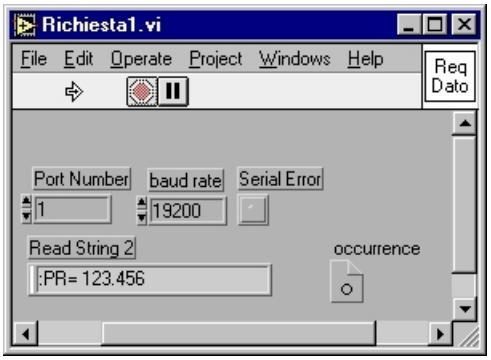

Figura 5-24. Pannello frontale del VI 'Richiesta1'.

Il pannello frontale del VI 'Richiesta1' è analogo all'ultimo descritto.

Dopo aver illustrato il software presente nel ramo destro è possibile illustrare quello sinistro.

Se nel pannello frontale del VI principale si seleziona la voce 'Test Laser Beacon' o 'Test String Power Module' viene mostrata la finestra di figura 5-25 che già è in modalità 'run'.

Il pannello frontale di questo VI contiene gli ovvi parametri 'Port Number Master', 'Port Number Slave' e 'Baud Rate Device'. Il parametro 'Device' invece identifica l'indirizzo del dispositivo UNIV1 con cui il simulatore dell'SCM deve dialogare (cioè inoltre il valore del primo byte di tutte le domande e le risposte).

Figura 5-25. Pannello frontale del VI 'all Modbus'.

I tasti 'START' e 'STOP' hanno la stessa funzione già descritta prima. Gli ultimi due oggetti contenuti nel pannello frontale 'all Modbus' contengono delle stringhe: in quello etichettato con 'String' è possibile inserire le domande fatte dal simulatore dell'SCM mentre in quello etichettato con 'Read String' sono mostrate le risposte del simulatore del modulo Laser Beacon o SPM. Questi due oggetti possono contenere molte domande e risposte. Infatti ho ritenuto che fosse più comodo scrivere più domande insieme invece che porle una alla volta. Tuttavia il protocollo non prevede la possibilità di inviare più comandi insieme in quanto prima di porre un'altra domanda occorre aspettare la risposta alla domanda precedente, cioè occorre inviare una domanda per volta, raccogliere le risposte e mostrarle. Questo è il compito del VI 'Modbus multi_instructions' il cui pannello frontale è mostrato in figura 5-26. Questo VI è richiamato dal VI 'all Modbus', ma il suo pannello frontale non viene aperto nell'esecuzione. È possibile notare che le due domande rappresentate sul pannello sono le stesse rispettivamente del primo e del secondo esempio mostrati nelle tabelle 5-6 e 5-8 (e le risposte sono ovviamente le stesse delle tabelle 5-7 e 5-9).

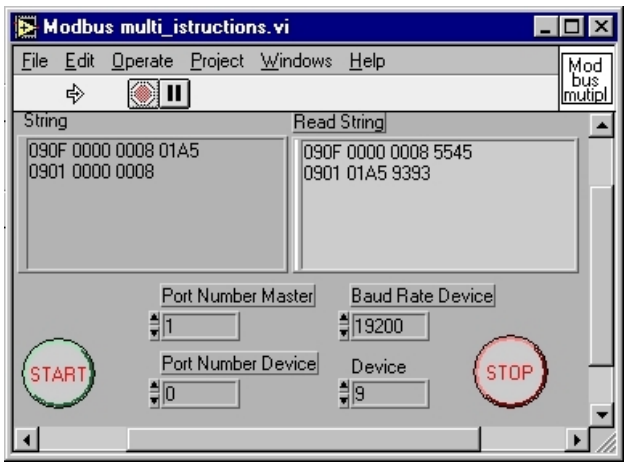

Figura 5-26. Pannello frontale del VI 'Modbus multi_istructions'.

Il VI 'Modbus multi_instructions' oltre a richiamare il VI che permette di inviare e ricevere una domanda per volta, che sarà descritto più avanti, utilizza anche il VI 'Find EOL'. Quest'ultimo permette di contare il numero di domande da inviare contando il numero di *CR* introdotti. Il pannello frontale di questo VI è mostrato in figura 5-27.

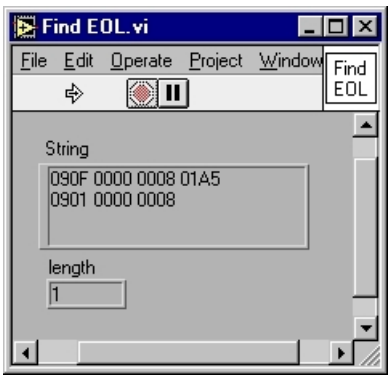

Figura 5-27. Pannello frontale e diagramma a blocchi del VI 'Find_EOL'.

Il compito di inviare una domanda e raccogliere la risposta è assolto dal VI 'Question & response' (figura 5-28).

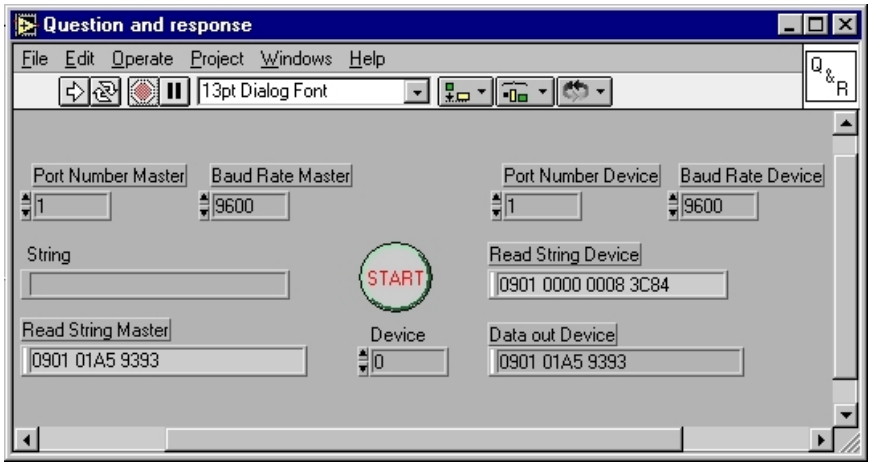

Figura 5-28. Pannello frontale del VI 'Question and response'.

Questo VI richiama i due VI indicati in precedenza: 'Modbus _slave1' che simula la presenza di un UNIV1 con indirizzo 09 (figura 5-29) e 'one instruction Modbus' che simula l'SCM (figura 5-30).

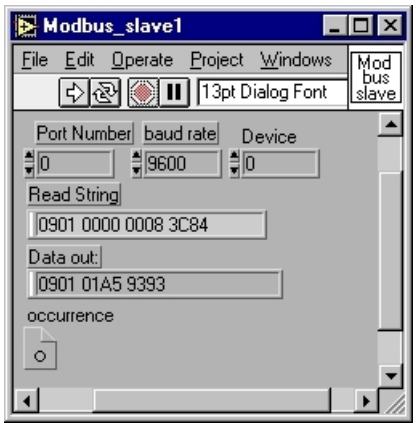

Figura 5-29. Pannello frontale del VI 'Modbus_slave1'.

Il pannello frontale del VI 'Modbus_slave1' contiene oltre agli abituali controlli ('Port Number', 'Baud Rate', 'Device') la stringa ricevuta ('Read String') e la stringa inviata ('Data out:') e il controllo dell'occorrenza proveniente dal VI 'Question & response'.

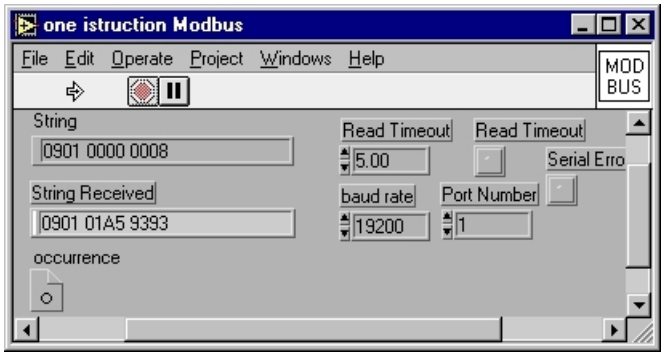

Figura 5-30. Pannello frontale del VI ‘one instruction Modbus’.

Il pannello frontale del VI ‘one instruction Modbus’ è simile a quello descritto in precedenza. In più rispetto al primo contiene l’indicazione del ‘Read Timeout’ che viene accesa quando il VI non riceve dati per un tempo superiore a quello fissato dal controllo ‘Read Timeout’. Confrontando la figura 5-30 ma anche le figure 5-25, 5-26 e 5-27 con le tabelle 5-6 e 5-8 è possibile notare che le domande non contengono i due byte finali CRC per il controllo degli errori. Infatti questo calcolo è realizzato dal VI ‘MB_crc’ contenuto nel VI ‘one instruction Modbus’ (figura 5-31): il compito di questo VI è di calcolare il CRC della domanda espressa in forma di vettore di byte, di integrare la domanda con il calcolo del CRC e trasformarla in formato stringa espressa in esadecimale.

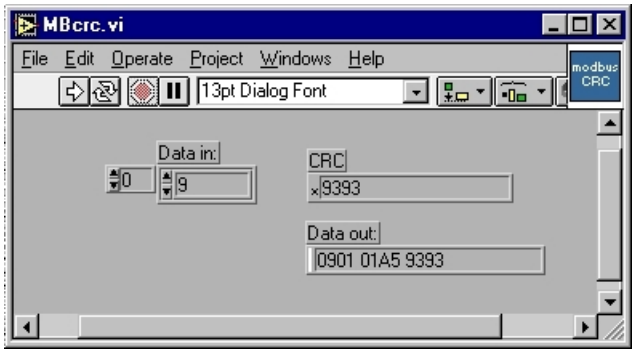

Figura 5-31. Pannello frontale del VI ‘MB_crc’.

Il compito di trasformare la stringa passata dal VI ‘Question & response’ in forma vettoriale è assolto dal VI ‘Str2array’ il cui pannello frontale è mostrato in figura 5-32.

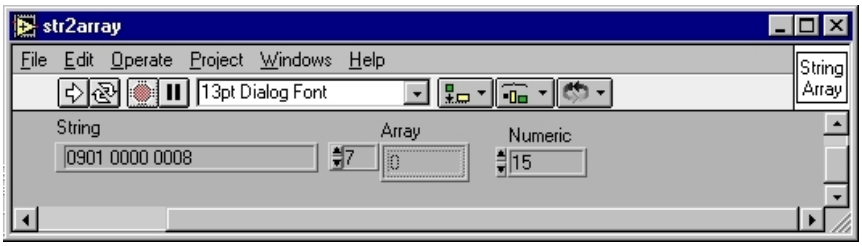

Figura 5-32. Pannello frontale del VI ‘str2array’.

Per poter rappresentare più istruzioni nell’oggetto ‘Read String’ rappresentato in figura 5-26 è stato necessario implementare il VI ‘convert_strHex2str’ (figura 5-33) che effettua la conversione di stringhe espresse in formato esadecimale in stringhe espresse in formato ASCII.

Figura 5-33. Pannello frontale del VI ‘convert_strHex2str’.

Gli ultimi 3 VI contenuti nel ‘Local Interfaces’ si riferiscono a operazioni con la memoria. Infatti ogni volta che viene acceso e attivato il VI ‘Modbus multi_istructions’, il ‘memory_reset’ (figura 5-34) in esso richiamato scrive tutti zeri nei 4 file (rappresentanti i registri: Boolean Output, Boolean Input, Holding Register e Input Register) di dimensione opportuna. In particolare il valore del

controllo ‘Register’ contenuto nel pannello frontale del VI ‘memory_reset’ assume il seguente significato:

0 per Boolean Output;

1 per Boolean Input;

2 per Holding Register;

3 per Input Register.

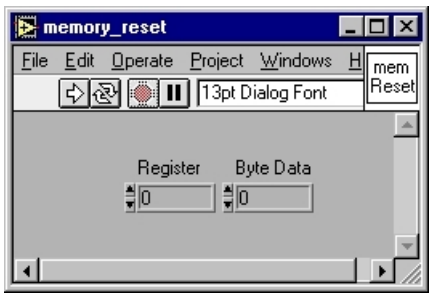

Figura 5-34. Pannello frontale del VI ‘memory_reset’.

I restanti 2 VI sono contenuti nel VI ‘Modbus_slave1’ e permettono la scrittura (VI ‘memory_write’, figura 5-35) e il recupero dei dati (VI ‘memory_read’, figura 5-36).

Figura 5-35. Pannello frontale del VI ‘memory_write’.

Figura 5-36. Pannello frontale del VI 'memory_read'.

5.4.5 Collegamenti SCM – Local Interfaces

Un'altra complicazione che occorre superare riguarda il numero di connessioni che occorre garantire nei collegamenti con l'SCM. Come è stato specificato nel secondo paragrafo del seguente capitolo, il Test Bench deve disporre di almeno 5 connessioni seriali indipendenti di cui 2 RS232 e 3 RS485 affinché la fase di test possa svolgersi in completa autonomia. Le due porte seriali in dotazione standard del PC perciò non sono sufficienti e occorre dotare il PC di ulteriori connessioni. Una breve ricerca che ho svolto ha assicurato l'esistenza di schede commerciali multiporta. Queste schede vengono inserite all'interno dell'unità centrale dove è contenuta la scheda madre del PC e possono gestire 4, 8 o 16 porte seriali supplementari RS232 o con interfaccia RS485. Un esempio di questi dispositivi multiporta è la scheda Meteor 4 della Aurora Technologies. Il numero delle porte (sempre uguale alle potenze di 2) è grosso modo proporzionale al costo della scheda.

Ai fini dell'acquisto della scheda multiporta adatta ad essere utilizzata nel banco di prova occorre considerare la necessità di ulteriori 3 porte seriali che tuttavia non fanno parte del modulo Local Interfaces. Queste porte seriali sono

relative ai dispositivi BIDIDEV di cui si è già discusso nel precedente capitolo. Infatti questi dispositivi hanno la possibilità di essere monitorati attraverso il protocollo MODBUS. I BIDIDEV in questione sono:

- 1 BIDIDEV per trasmettere il segnale di clock.
- 1 BIDIDEV per ricevere il segnale di clock;
- 1 BIDIDEV per emulare il trasmettitore dei dati dalla shore station.

Poiché questi dispositivi hanno la possibilità di utilizzare lo standard di comunicazione RS232 e RS485 vi è la possibilità di limitare il numero di porte totali da aggiungere a 4. Infatti 3 connessioni RS485 sono necessarie per le connessioni all'SCM mentre la quarta può essere utilizzata pure con standard RS485 ma utilizzata come bus seriale per tutti e tre i BIDIDEV che dovranno essere indirizzati opportunamente.

5.5 Modulo di studio per segnali periodici

Nel paragrafo 4.4.3 è stata manifestata l'esigenza della implementazione di un modulo che permetta lo studio e la caratterizzazione di segnali periodici come ad esempio il segnale di clock. L'obiettivo nella fase di test è quello di garantire che il segnale sotto studio soddisfi alcune caratteristiche fissate. Infatti il segnale interessato può assumere determinate caratteristiche se il test è effettuato su un'unica scheda elettronica mentre può assumere caratteristiche differenti se il test è effettuato su un oggetto che contiene varie schede che lavorano contemporaneamente.

5.5.1 Descrizione dell'hardware utilizzato

Grazie alla dotazione del Laboratorio del gruppo II dell'Istituto Nazionale di Fisica Nucleare (Sezione di Bari) è stato possibile implementare un software che permetta lo studio e la caratterizzazione di un segnale periodico come quello utilizzato per la trasmissione del clock in ANTARES. In particolare sono stati utilizzati un generatore di impulsi in qualità di apparato generatore del segnale oggetto di test e un oscilloscopio digitale utilizzato come apparato rivelatore.

I dispositivi utilizzati sono: il generatore di impulsi HP 81110A e l'oscilloscopio digitale TDS 754C.

Il primo dispositivo, prodotto dalla compagnia Hewlett Packard, genera, sulle sue due uscite, impulsi digitali con vari standard (TTL, LVDS, ECL, CMOS e altro) fino alla frequenza di 330 MHz. La sequenza degli impulsi può essere impostata direttamente sullo strumento o può essere scelta una sequenza pseudo-random. Tutte le impostazioni possono essere impostate sul pannello mostrato in figura 5-37 o attraverso un PC connesso con una porta GPIB.

Figura 5-37. Pannello frontale del generatore di impulsi HP 81110A.

Il secondo dispositivo è un oscilloscopio digitale prodotto dalla compagnia Tektronix (figura 5-38). La frequenza di campionamento in tempo reale può arrivare fino a 1 gigacampioni al secondo (GS/s). L'oscilloscopio è dotato di memoria interna fino a 4 MB ed ha una larghezza di banda massima di 500 MHz. Tra le funzioni sono incluse anche quelle statistiche per facilitare l'analisi dell'andamento dei segnali di comunicazione. Anche questo strumento può essere completamente controllato dal PC attraverso la porta GPIB.

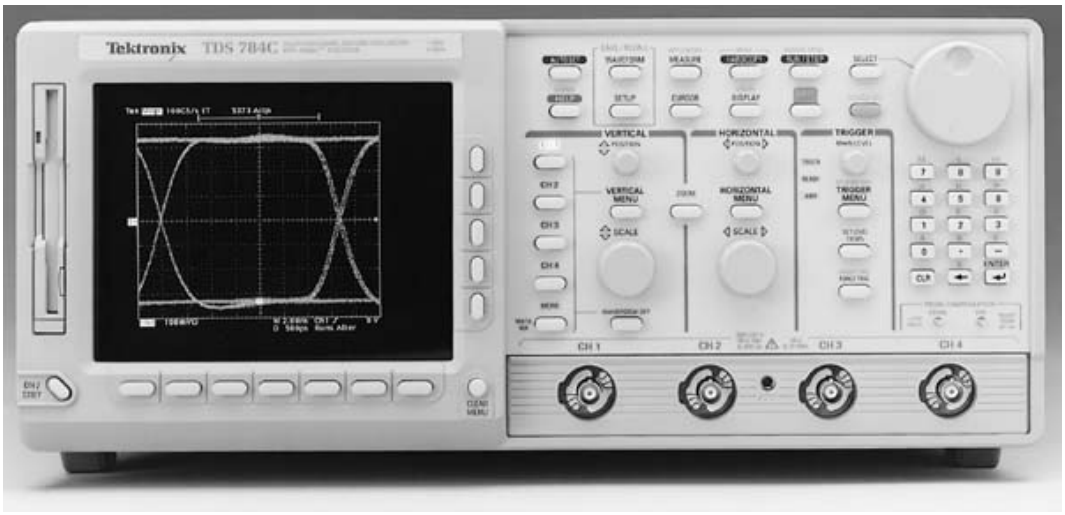

Figura 5-38. Pannello frontale dell'oscilloscopio digitale TDS 754C.

5.5.2 Realizzazione del modulo di studio per segnali periodici

La figura 5-39 mostra lo schema della gerarchia dello strumento virtuale 'Acquisizione segnale'. Nelle figure seguenti è possibile osservare i pannelli frontali del software per il modulo di studio per segnali periodici, mentre i loro diagrammi a blocchi sono illustrati in appendice.

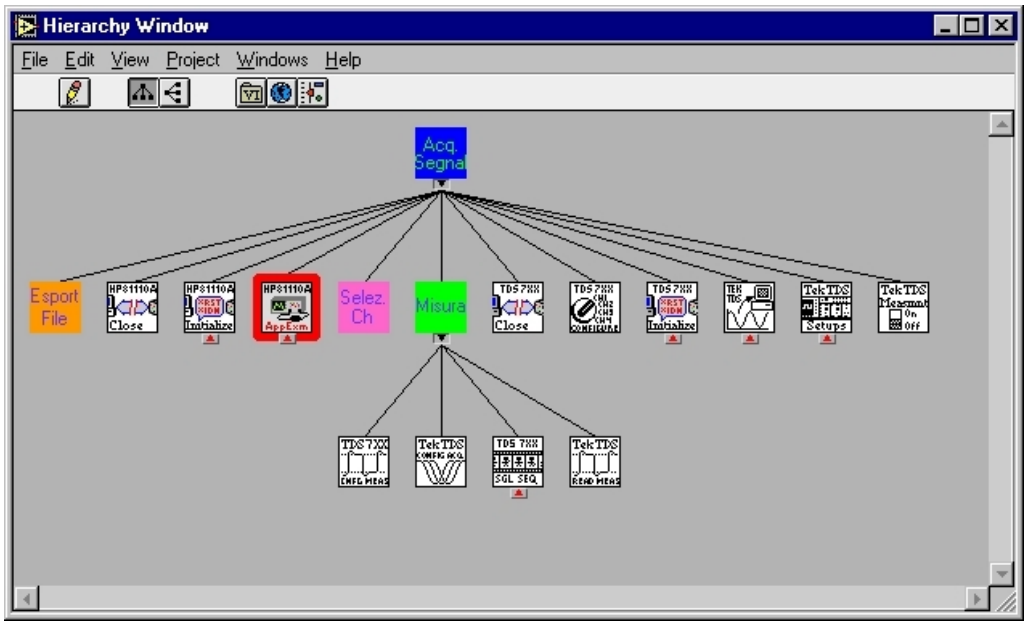

Figura 5-39. Gerarchia dei VI contenuti nel 'Acquisizione segnale'.

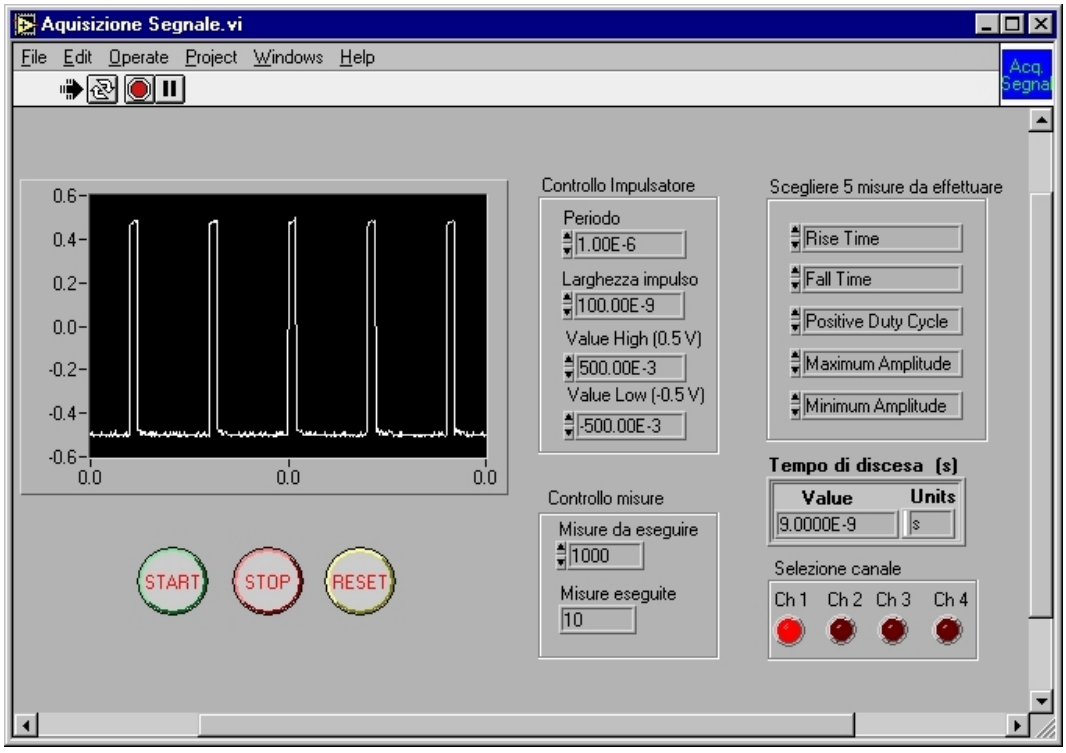

Figura 5-40. Pannello frontale del VI 'Acquisizione segnale'.

Il pannello frontale in figura 5-40 mostra numerosi oggetti raggruppati. Accanto al grafico che mostra l'andamento della forma d'onda si può notare un

gruppo di 4 oggetti indicato con ‘Controllo Impulsatore’ che permettono il controllo di alcune caratteristiche del segnale proveniente dal generatore di impulsi. Accanto a questi vi è un altro raggruppamento di 5 oggetti da dove è possibile selezionare 5 parametri sui quali basare lo studio e la caratterizzazione del segnale da misurare. I parametri scelti sono: il tempo di salita, il tempo di discesa, il duty cycle, l’ampiezza massima e l’ampiezza minima.

Un altro raggruppamento è indicato con ‘Controllo misure’ e comprende il numero di acquisizioni da effettuare (ciascuno comprendente le 5 misure indicate), che può essere impostato dall’utente, e il numero di acquisizioni effettuate. Accanto ci sono altri 2 raggruppamenti: uno mostra il valore e l’unità della misura appena effettuata e l’altro indicato ‘Selezione canale’ che contiene la selezione dei canali utilizzati (in figura è scelto il primo canale). L’ultimo raggruppamento è dato da 3 pulsanti. Il pulsante di ‘START’ avvia l’acquisizione, il pulsante di ‘STOP’ permette di terminare l’acquisizione anche se incompleta e il pulsante di ‘RESET’ reimposta i parametri di esecuzione ai valori di default.

La gerarchia di figura 5-39 mostra un gran numero di VI utilizzati per l’implementazione del VI radice. La maggior parte di essi fanno parte delle librerie degli strumenti interessati, mentre i VI ‘Esporta’, ‘selez-canale’ e ‘Misura’ sono stati da me implementanti.

In figura 5-41 viene mostrato il pannello frontale del VI ‘Esporta’ che ha il compito di inizializzare i 5 file dove verranno memorizzate le misure.

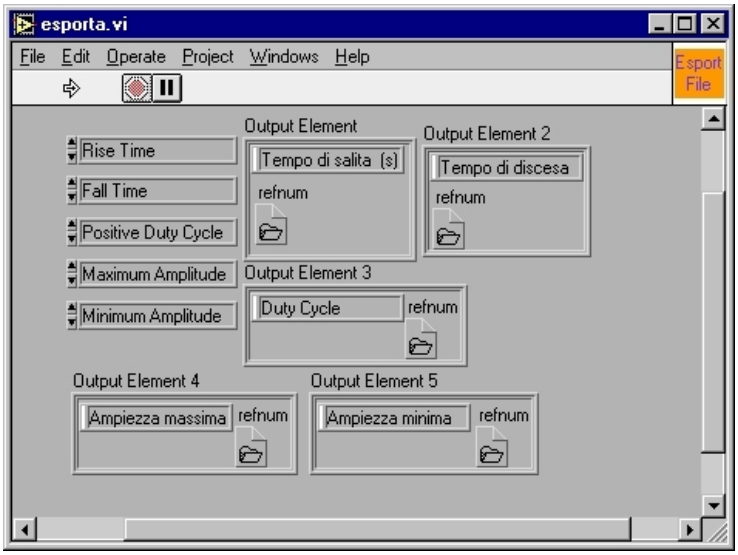

Figura 5-41. Pannello frontale del VI 'Esporta'.

Il VI 'selez-canale' (figura 5-42) ha il compito di verificare che l'utente abbia selezionato un canale e in alternativa di fornire un messaggio di errore.

Figura 5-42. Diagramma a blocchi del VI 'selez-canale'.

Il VI 'Misura' (figura 5-43) racchiude diverse librerie e fornisce la misura richiesta.

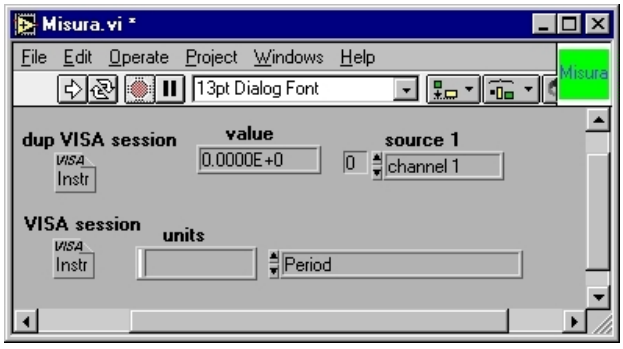

Figura 5-43. Pannello frontale del VI 'Misura'.

L'esecuzione del software ha, dunque, permesso la memorizzazione in 5 file di testo delle 1000 acquisizioni effettuate, ciascuno contenete una misura differente. Di conseguenza è stato possibile caratterizzare il segnale prodotto dal generatore di impulsi in base ai parametri selezionati. In particolare è stato possibile costruire diagrammi statistici in forma di istogrammi che forniscono il numero punti aventi un dato valore al variare del valore considerato dove la popolazione totale dei punti è costituita dalle 1000 acquisizioni. Ogni diagramma si riferisce ad un parametro misurato. Questi grafici rappresentati nelle figure 5-44, 5-45, 5-46, 5-47 e 5-48 forniscono una ricca informazione e possono fornire la base per successive considerazioni.

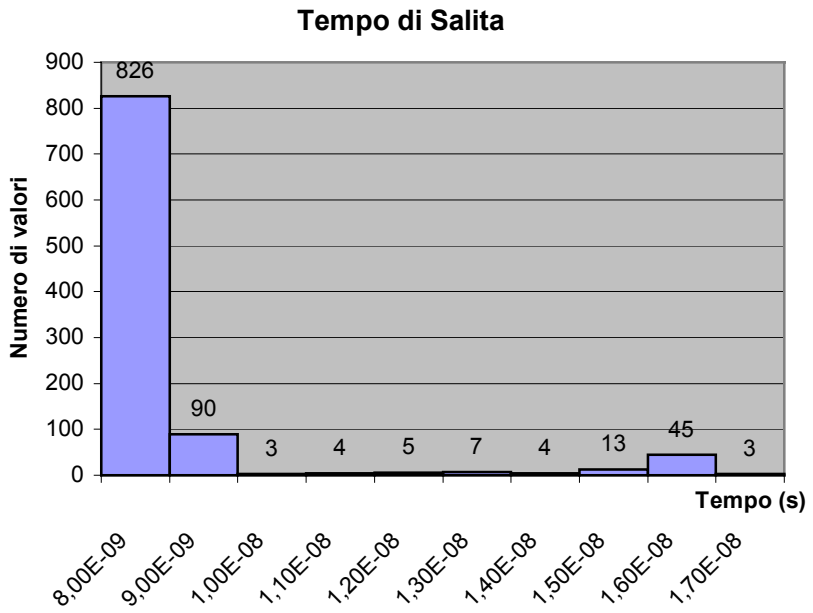

Figura 5-44. Istogramma del Tempo di salita relativo a 1000 acquisizioni.

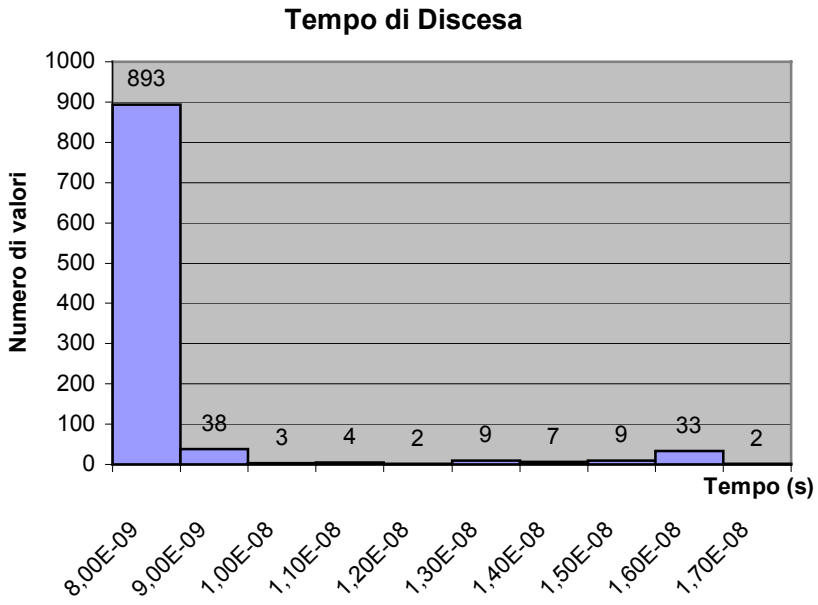

Figura 5-45. Istogramma del Tempo di discesa relativo a 1000 acquisizioni.

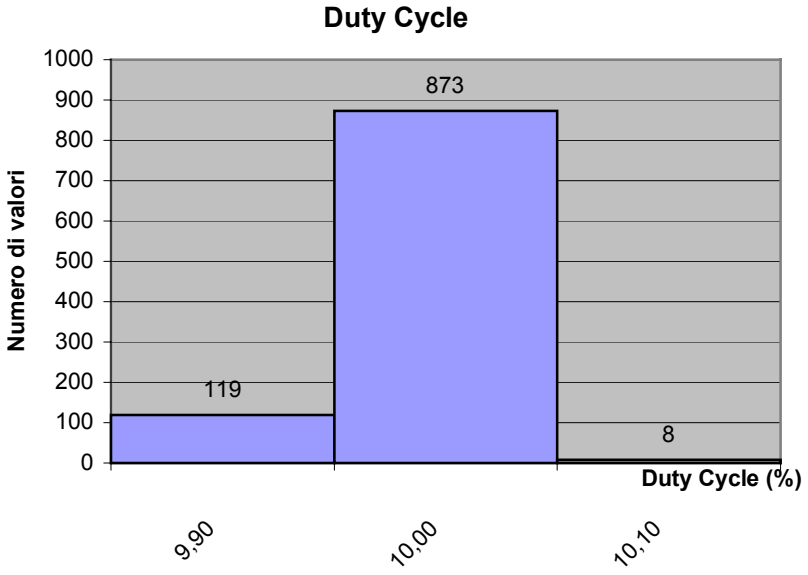

Figura 5-46. Istogramma del Duty cycle relativo a 1000 acquisizioni.

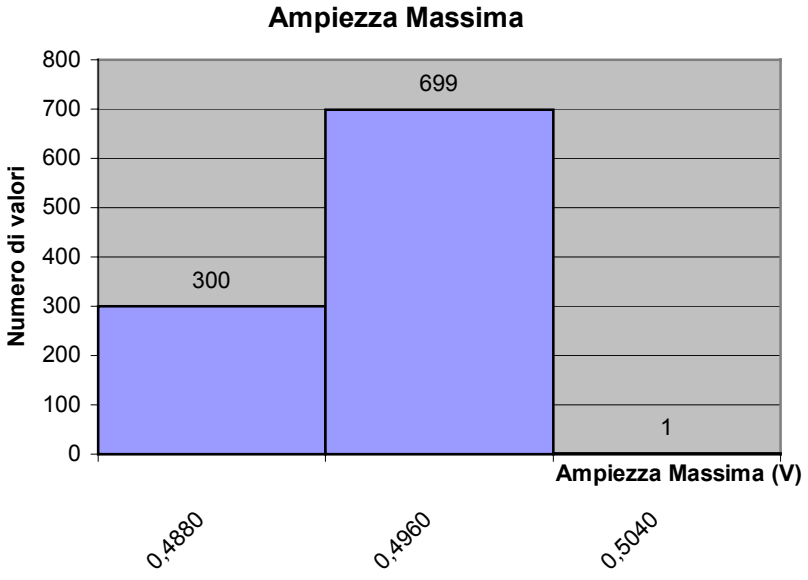

Figura 5-47. Istogramma dell'Ampiezza massima relativa a 1000 acquisizioni.

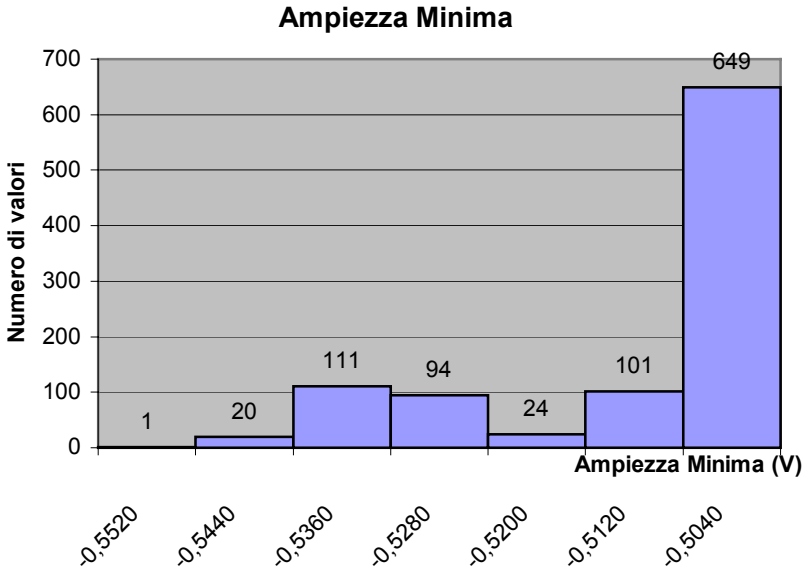

Figura 5-48. Istogramma dell'Ampiezza minima relativa a 1000 acquisizioni.