Istituto Nazionale di Fisica Nucleare

LABORATORI NAZIONALI DI FRASCATI

STAGE INVERNALE PRESSO I LNF

REALIZZAZIONE DI UNA LIBRERIA SOFTWARE IN AMBIENTE LABVIEW PER IL CONTROLLER CAMAC "*JENET* "

Tutor

Ing. Baccarelli Gianfranco
Ing. Masciarelli Mario

Studenti

Lucantonio Alessandro
Silvestri Marco
Tiseo Carlo
Villa Massimo

Scopo dello stage

Lo stage effettuato presso i laborati di Frascati dell'INFN si propone di sviluppare in ambiente LabView un sistema di acquisizione con dispositivi CAMAC.

Il nostro "banco di lavoro" è la sala controllo BTF (*Beam Test Facility*). In questa sede sono presenti strumenti per controllare il fascio di elettroni generato nel Linac, sistemi di acquisizione dati, console Solaris e apparati industriali in bus VME e CAMAC. La sala dove avviene l'esperimento vero e proprio è schermata dal resto delle strutture con uno spesso muro in cemento armato. In essa arriva un fascio di elettroni perfettamente calibrato in energia e quantità di particelle. La grande mole di dati che compone l'esperimento viene acquisita, catalogata e immagazzinata con dispositivi industriali. Per facilitare e in alcuni casi rendere possibile l'analisi dei risultati dell'esperienza, vengono elaborati programmi specifici per lo più in ambiente LabView.

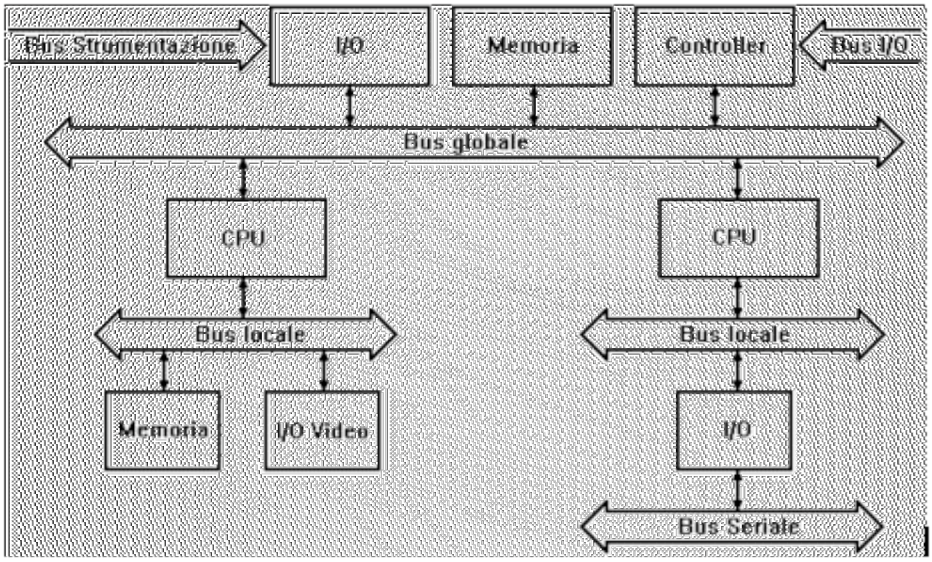
Il nostro obiettivo era dar vita ad un programma in LabView che si mettesse in comunicazione attraverso il protocollo TCP/IP con il controller Jenet su bus CAMAC, situato appunto qui nella BTF, e che svolgesse le funzioni standard di quest'ultimo nel modo più semplice ed immediato.

È necessario, prima di illustrare il programma stesso, avere informazioni più dettagliate circa gli strumenti che stiamo utilizzando e il modo con cui questi comunicano tra loro.

1– Il Bus

Come abbiamo accennato in precedenza, i dati risultati dall'esperimento vengono acquisiti da calcolatori di tipo industriale basati su bus VME e CAMAC.

In un calcolatore il *bus* è la struttura fisica che consente il trasferimento di dati dalla CPU alla memoria e alle periferiche (dispositivi I/O) del sistema, come esemplificato in figura:



In ambito industriale si rivela indispensabile l'uso di bus quando la comunicazione deve avvenire tra parti provenienti da case costruttrici diverse, ed è perciò necessario che esso sia standardizzato.

In base al tipo di trasmissione utilizzato i bus si classificano in:

- *bus a trasmissione parallela*: corti, veloci e in grado di inviare più informazioni contemporaneamente, come VME, Compact PCI, PCI, CAMAC;
- *bus a trasmissione seriale*: economici e in grado di coprire distanze apprezzabili, ma capaci di trasmettere solo un bit alla volta, come EtherNet e RS422.

Anche riguardo la modalità di trasmissione esistono due varianti:

- *Sistemi single-master* in cui e' previsto un solo *master* (invia i comandi e svolge la funzione di CPU) e piu' *slave* (ricevono i comandi e rispondono quando interpellati);
- *Sistemi multi-master* in cui un *controller* si occupa di gestire l'arbitraggio tra i vari dispositivi master e slave presenti quando essi richiedono l'accesso al bus;

Infine anche la sincronizzazione del traferimento dei dati diversifica i bus in :

- *bus sincroni* in cui esite un segnale di riferimento temporale, il *clock*, a scandire la successione degli eventi. I segnali agiscono in corrispondenza dei fronti di salita e/o discesa del clock e devono avere tutti la stessa velocità.
- *bus asincroni* in cui la successione degli eventi è scandita dai sottosistemi che trasferiscono i dati. L'unico vincolo imposto alla trasmissione è il tempo di timeout entro cui i vari dispositivi devono rispondere.

Nei bus paralleli single-master come il CAMAC il trasferimento dei dati (ricordiamo che i bus operano a livello fisico) avviene tramite un insieme di linee elettriche che collegano il master e gli slave. Oltre alle linee per l'alimentazione dei dispositivi nel *backplane* sono previste le linee di:

- *Data*: sono le linee che trasmettono le informazioni. Possono essere unidirezionali (linee separate per *Write* e *Read*) o bidirezionali;
- *Address*: sono unidirezionali (dal master allo slave) e specificano l'indirizzo di memoria in cui scrivere e leggere i dati
- *Interrupt*: linea unidirezionale con cui lo slave avvisa il master di richieste o errori attraverso il segnale L (*Look-at-me*);
- *Arbitraggio*: decidono chi ha l'accesso al bus in caso di più master che ne chiedono simultaneamente l'utilizzo, evitando che questi ultimi vadano in conflitto. Naturalmente i bus single-master ne sono privi.

Alcuni bus sono multiplexati: in poche parole più segnali sono riservati ad un'unica linea, ed un'unica linea trasmette alternativamente segnali il cui significato varia nel tempo (ad esempio si possono multiplexare tra loro i segnali di Address e Data). Il *multiplexing* è vantaggioso perchè riduce il numero delle linee, le dimensioni ed il costo del bus.

1.1– Il Bus CAMAC

Negli anni '60 la diffusione dei calcolatori elettronici ha reso possibile la computerizzazione di piccoli e grandi esperimenti per i quali erano presenti sistemi di interfacciamento non standard.

Per quanto riguardava la fisica nucleare e delle particelle, a causa della grande quantità di dati da acquisire ed elaborare, servivano sistemi di interfacciamento più potenti quali il *CAMAC* e il *FASTBUS*.

Quello più usato era il CAMAC, dato che l'hardware e il software erano più consolidati mentre il FASTBUS era in via di sviluppo.

Oggi il sistema più usato per l'interfacciamento è il VME, ma data la grande diffusione del CAMAC negli impianti di ricerca, sono stati sviluppati dei

controller che rendono utilizzabile il CAMAC con le nuove tecnologie, riducendo di molto le spese economiche degli impianti.

Infatti il nostro stage è nato per sviluppare un software in grado di interagire con un nuovo cotroller CAMAC con interfaccia ethernet per modificarne le impostazioni e richiederne i dati tramite la porta TCP.

1.1.1 - Struttura del CAMAC

Il CAMAC è un apparato modulare le cui componenti essenziali vengono dette *crate*, ognuna delle quali è costituita da 25 slot. Eccetto gli slot 24-25 (riservati al *Crate Controller*, cioè il master), nei restanti è possibile inserire dei moduli o *plug-in* di elaborazione (che costituiscono gli slave), collegati tra loro e con il Crate Controller attraverso il *dataway*, ovvero l'insieme delle linee su cui i dati e i comandi transitano. Il dataway è parte del *backplane*, che comprende anche le linee di alimentazione.

Esistono diversi tipi di linee:

- *Bussed Signal Lines*:connettono i moduli tra loro e con il CC.
  - *data* (*write*): si tratta di 24 linee unidirezionali dal master allo slave, su cui possono transitare 24 bit contemporaneamente;
  - *data* (*read*): 24 linee unidirezionali dallo slave al master per la lettura di dati;
  - *timing*: linee per la sincronizzazione tra master e slave;
  - *command*: linee unidirezionali dal master allo slave per l'invio di comandi;
  - *responses*: linee dallo slave al master che restituiscono l'esito dell'operazione richiesta;
- *Point to Point Lines*: linee separate che collegano ogni modulo al CC. Sono di due tipi:
  - *N*: attraverso questa linea il CC seleziona il modulo;
  - *L* (*Look-at-me*): inviando un segnale su questa linea, il modulo richiede l'attenzione del CC.

Il CC mette in comunicazione i moduli con un computer e si occupa della gestione del flusso di informazioni sul dataway. I comandi e i dati vengono inviati dal computer al CC mediante il protocollo di rete TCP/IP, di cui parleremo avanti.

1.1.2 – Il CC Jenet

Il CC Jenet occupa gli slot 24 e 25 del crate, in figura ne e' mostrata una immagine:

Immagine del controller CAMAC *JENET*

Il CC possiede un proprio indirizzo IP (vedi 2-Il protocollo TCP/IP), a differenza dei moduli, accessibili solo attraverso comandi Jenet. Su di esso è costantemente in esecuzione un web server su sistema operativo Linux che risponde alle richieste inviabili attraverso un browser (mediante connessione alla porta 80 di Jenet). È anche possibile inviare i comandi riconosciuti da Jenet sulla porta 2000, attraverso una sessione telnet (shell remota, vedi 3-Unix per la shell).

Principali comandi riconosciuti da Jenet

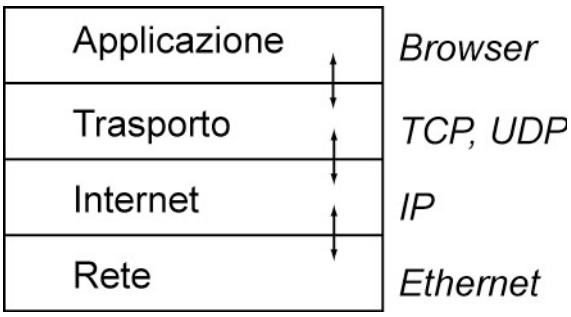
Comando	**Descrizione**
CCCZ	Inizializza il dataway
CCCC	Reinizializza il Crate
CFSA	Esegue un comando CAMAC a 24 bit; restituisce lo status del dataway e il valore di ritorno della funzione
CSSA	Esegue un comando CAMAC a 16 bit; restituisce Q (booleano che indica lo status del dataway) e il valore di ritorno della funzione (ad es. il risultato di una lettura)
CTLM	Esegue il test della funzione Look-At-Me sullo slot specificato
CLMR	Ritorna il valore del registro LAM (verifica se lo slot ha chiesto l'attenzione del CC)
CTSTAT	Ritorna gli ultimi valori di Q e X (booleano che indica se la funzione è stata eseguita)

2 – Il protocollo TCP/IP

La comunicazione con il Controller Jenet avviene attraverso il *protocollo TCP/IP*. Le informazioni viaggiano su un rete locale di tipo Ethernet. Il termine TCP/IP indica un modello che descrive il processo di trasmissione dei dati in 4 *layer* (livelli):

- *Application* (*Applicazione*): si occupa di fornire alle applicazioni (browser come Netscape, client di posta elettronica, ecc.) i servizi di rete, compresi la rappresentazione dei dati ed il mantenimento della comunicazione fra due computer (detti *host*).
- *Transport* (*Trasporto*): si occupa del trasporto attraverso i protocolli TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*). Il primo dei due è di tipo connection-oriented, ossia è necessario instaurare una connessione fra gli host prima di iniziare a scambiare i dati. L'UDP è invece connectionless: i dati possono essere inviati e ricevuti direttamente.
- *Internet* (*Internet*): questo livello si occupa di indirizzare, suddividere e instradare i pacchetti sulla rete. Il *protocollo IP* lavora in questo layer.
- *Network* (*Rete*): lavora a stretto contatto con l'hardware. Si occupa di prelevare ed immettere i frame dati, diversi a seconda della tipologia di rete, sul cavo Ethernet e di controllarne la correttezza.

Un modello di questo tipo presenta diversi vantaggi, come la possibilità di implementare nuovi protocolli ad uno qualsiasi dei layer dovendosi preoccupare solo della compatibilità con i livelli adiacenti, come mostrato in figura:



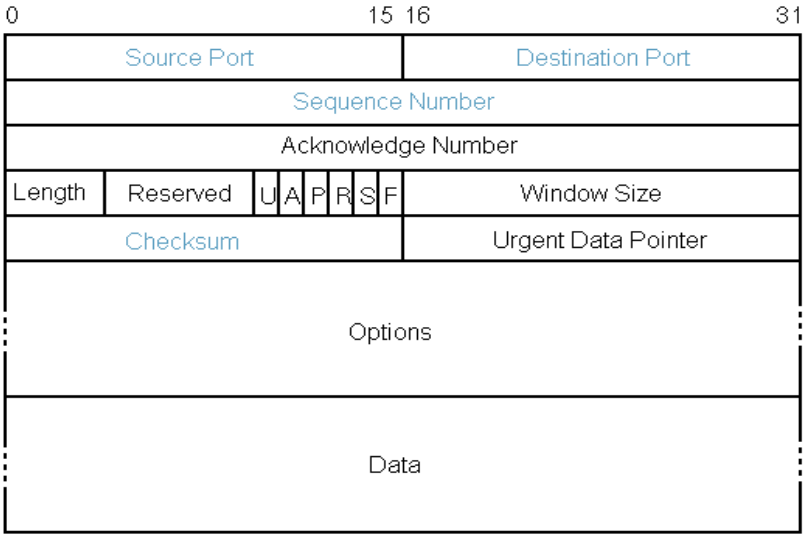
Le informazioni viaggiano nella rete sotto forma di *pacchetti* quindi vengono frammentate prima di essere immesse in rete. Il nome di queste unità di dati dipende dal punto in cui si trovano nella "pila" del modello TCP/IP: si chiamano *messaggi di applicazione* al livello applicazione, *pacchetti IP* o datagrammi UDP a livello trasporto, *frame Ethernet* a livello rete.

Proviamo ora a seguire i dati nel loro cammino verso il basso nella pila di protocolli e vedere come giungono a destinazione. In una applicazione che usa

TCP (come FTP, per il trasferimento dei file) i dati passano al modulo TCP che li frammenta e aggiunge a ciascun pacchetto un'intestazione (*header TCP*), costituita da vari campi tra i quali:

- *Source port* e *destination port*: le porte identificano su ciascuna macchina i processi locali che comunicano con quelli remoti ed i servizi disponibili su un host di rete. La combinazione tra indirizzo IP (vedi più avanti), protocollo di trasporto e numero di porta prende il nome di *socket*. Per instaurare una connessione TCP c'è bisogno che un host (*server*) si metta in ascolto su una porta e un altro host (*client*) si connetta a tale porta. Alcune porte sono riservate: sono le cosiddette *well-known ports* (ad esempio il servizio FTP è attivo sulla porta 21). I campi source port e destination port hanno un range di valori compreso fra 0 e 65535.
- *Sequence number*: non è detto che i pacchetti arrivino al destinatario nello stesso ordine in cui sono stati inviati. Questo campo serve proprio all'host ricevente per ricostruire l'esatta sequenza delle informazioni.

Header TCP



A ciascun pacchetto viene successivamente aggiunta l'intestazione IP, che comprende l'indirizzo IP del mittente, quello del destinatario ed il *campo checksum* (presente anche nell'header TCP) per la verifica dell'integrità dei dati.

Infine l'intestazione Ethernet, che comprende importanti campi che vedremo in seguito, viene anteposta a ciascun pacchetto che ora si definisce frame. Le informazioni vengono quindi trasmesse sul cavo di rete. Al momento della ricezione dei pacchetti da parte dell'host destinatario, quest'ultimo provvede a riassemblarli in base al loro sequence number.

Ma come si distinguono tra loro i computer sulla rete? Grazie agli **indirizzi IP**, costituiti da 4 byte (32 bit); uno dei modi per rapprensentare gli indirizzi IP è la notazione decimale puntata (ad es. 192.168.52.14). Ogni ottetto (gruppo di 8 bit, pari a 1 byte) ha un range di valori (in decimale) compreso tra 0 e 255. In particolare lo 0 nell'ultimo ottetto identifica la rete (Network Address) mentre il 255 in tutti gli ottetti è l'indirizzo di broadcast, grazie al quale il pacchetto può essere inviato a tutti gli host presenti sulla rete. Inoltre quando tutti gli ottetti hanno valore 0 (0.0.0.0), identificano l'host corrente.

Gli indirizzi IP possono essere *statici*: se vengono impostati su ciascuna macchina tramite il pannello delle opzioni TCP/IP, in modo che ogni computer mantenga sempre lo stesso indirizzo; *dinamici* se vengono assegnati da un server DHCP (*Dynamic Host Configuration Protocol*) nel momento in cui gli host si collegano alla rete. Il server DHCP provvede ad assegnare temporaneamente (*leasing*) il primo IP disponibile alla macchina appena connessa e lo rende di nuovo assegnabile nel momento in cui viene disconnessa.

Tuttavia bisogna precisare che gli indirizzi IP sono un metodo di identificazione software per le macchine connesse in rete, mentre le schede di rete (*hardware*) sono le vere responsabili della "cattura" dei pacchetti. Quando il layer IP passa il pacchetto al driver Ethernet, esso provvede a "tradurre" gli indirizzi IP in indirizzi fisici delle interfacce di rete (*MAC address* a 6 byte), che identificano univocamente ogni dispositivo di rete e non possono essere facilmente modificati. Per sapere quale è il MAC address del destinatario, la prima volta che viene instaurata una comunicazione, il mittente invia un pacchetto **ARP** (*Address Resolution Protocol*) a tutti gli host sulla rete (in broadcast), contenente l'indirizzo IP del destinatario, il proprio indirizzo IP ed il proprio MAC address. L'host che possiede l'indirizzo IP risponde al mittente con il proprio MAC address, il quale viene memorizzato in una tabella che abbina gli indirizzi IP ai relativi MAC address. Il pacchetto con i dati viene quindi immesso sulla rete; tutti gli host connessi alla rete lo "leggono" e l'host che ha il proprio MAC address corrispondente a quello nel campo destinatario dell'header Ethernet, accetta il pacchetto. Abbiamo così descritto il sistema TCP/IP per la trasmissione delle informazioni.

Torniamo a parlare di indirizzi IP più in generale. Per permettere una migliore organizzazione delle reti, gli indirizzi IP disponibili sono stati suddivisi in classi in base alle dimensioni del network da gestire. Le autorità di gestione di Internet assegnano alle società che vogliono connettere i propri computer alla Rete intervalli di IP address appartenenti alla classe più adatta.

- Indirizzi di *classe A*: il valore del primo ottetto è compreso tra 1 e 126 e identifica la rete (il valore rimane fisso), i restanti identificano gli host. Permette di ottenere 126 reti formate da 16.774.214 host ciascuna.
- Indirizzi di *classe B*: il valore del primo ottetto è compreso tra 128 e 191 ed insieme al secondo identifica la rete (entrambi gli ottetti sono fissi). È possibile ottenere 16.384 reti formate da 65.534 host ciascuna.
- Indirizzi di *classe C*: il valore del primo ottetto è compreso tra 192 e 223. 24 bit identificano la rete (i primi 3 byte sono fissi), 8 bit identificano gli host. È possibile ottenere 2.097.152 reti con 254 host ciascuna.
- Indirizzi di *classe D* ed *E*: coprono rispettivamente i range 224-239 e 240-255. Sono indirizzi riservati.

Ricordiamo che nei valori sopra riportati riguardanti gli host sono stati sottratti l'indirizzo di *broadcast* ed il *network address*. In totale si possono avere 4.294.967.296 possibili indirizzi IP. Sembrerebbe una grande cifra, ma se pensiamo ad aziende con migliaia di computer, ci rendiamo conto che non lo è.

Per far sì che più computer di una rete possano accedere ad Internet senza che sia necessario riservare un indirizzo IP per ognuno di essi, è possibile installare un server **NAT** (*Network Address Translation*) o un *gateway* e riservare un IP solo per tale computer. Quindi anche le grandi aziende possono richiedere indirizzi IP per i loro server NAT o gateway, in numero di certo inferiore agli host della LAN (rete locale) che essi stessi "nascondono". Vi sono delle classi di indirizzi IP privati, da utilizzare per gli host delle LAN:

Da 10.0.0.0 a 10.255.255.255
Da 172.16.0.0 a 172.31.255.255
Da 192.168.0.0 a 192.168.255.255

Ovviamente assegnando indirizzi IP privati agli host della LAN non si ha la certezza che tali indirizzi non esistano in Internet (due computer connessi ad Internet non possono avere lo stesso IP). Il server NAT ha il compito di sostituire il proprio indirizzo IP a quello privato presente nel campo sorgente dei pacchetti inviati dall'host della LAN su Internet. Ovviamente il PC destinatario non potrà sapere da quale computer della LAN proviene il pacchetto e quindi invierà il pacchetto di risposta al server NAT, che a sua volta lo smisterà a quel preciso host. Questo significa che, escludendo i pacchetti di risposta a connessioni già stabilite, gli indirizzi IP privati sono raggiungibili solo dall'interno della rete LAN.

Per il corretto funzionamento di una rete, ogni host deve poter distinguere quale parte dell'indirizzo identifica l'host e quale la rete. Questo avviene grazie all'ausilio della *subnet mask* (maschera di sottorete), impostabile manualmente dal

pannello delle opzioni TCP/IP.

- Classe A: Rete.Host.Host.Host ha come subnet 255.0.0.0
- Classe B: Rete.Rete.Host.Host ha come subnet 255.255.0.0
- Classe C: Rete.Rete.Rete.Host ha come subnet 255.255.255.0

Supponiamo ora che una società abbia bisogno di 2 sottoreti da 126 host ciascuna. Una rete di classe C non è sufficiente, ma comprarne due sarebbe sprecato. Si ricorre al subnetting, che consiste nell'utilizzare alcuni bit "presi in prestito" (*borrowed*) dalla parte host dell'indirizzo IP per identificare la rete. Nell'esempio precedente converrebbe acquistare una rete di classe C e suddividerla in 2 sottoreti da 126 host ciascuna. Bisogna tenere presente che il numero di subnet che si possono creare è dato da 2^x dove x rappresenta i bit presi in prestito dalla parte host dell'indirizzo. Sempre per quanto riguarda l'esempio precedente, prenderemo in prestito un bit (2 sottoreti) in modo da avere 126 host per ogni sottorete, risultato dato dalla formula 2^y-2, dove y è il numero di bit rimasti nel campo host (7). Avremo una subnet mask pari a 255.255.255.128 (128 equivale in binario a 10000000, il primo bit è fisso perché è il bit borrowed, gli altri identificano l'host). Le subnet che si vanno a creare diverranno x.x.x.127 e x.x.x.255, con base data da 255-128=127. Gli indirizzi IP assegnabili saranno: da x.x.x.1 a x.x.x.126 e da x.x.x.129 a x.x.x.254.

Come stabilire, a questo punto se il destinatario dei propri pacchetti si trova sulla stessa sottorete del mittente? Si utilizza il processo di *messa in AND* (ANDing process). Tale processo consiste nel confrontare il risultato dell'operazione AND bit a bit tra il proprio indirizzo e la propria subnet mask con quello tra l'indirizzo del destinatario e la propria subnet mask. Se i risultati sono identici i due PC possono comunicare direttamente.

Host A: 192.168.0.5
11000000.10101000.00000000.00000101: IP Address
11111111.11111111.11111111.00000000: Subnet Mask Host A
11000000.10101000.00000000.00000000: Risultato AND bit a bit

Host B: 192.168.0.25
11000000.10101000.00000000.00011001: IP Address
11111111.11111111.11111111.00000000: Subnet Mask Host B
11000000.10101000.00000000.00000000: Risultato AND bit a bit

Nel caso i due PC comunicanti non siano sulla stessa rete è necessario un **router** per lo smistamento dei pacchetti. Un router può essere un semplice computer che lavora a livello IP con un minimo di 2 schede di rete (una per

ognuna delle reti da connettere). Esso mantiene la cosiddetta tabella di routing con gli indirizzi IP delle reti associate a ciascuna scheda. Supponiamo che il router C colleghi la rete A con la rete B. Quando un computer della rete A invia un pacchetto ad un indirizzo IP della rete B gli indirizzi IP sorgente e destinazione contenuti nell'header IP saranno quelli, rispettivamente, dell'host in A e di quello in B, dunque gli indirizzi reali. Nel passare dal layer IP al layer fisico (Ethernet) verrà inserito nel campo destinazione del frame Ethernet il MAC address di C (router) in quanto, dopo il processo di messa in AND, i due host comunicanti sono risultati appartenere a due reti differenti. Il router, che è in grado di comunicare in modo diretto con gli host della rete B, modifica solo l'header Ethernet, mettendo nel campo sorgente il proprio MAC address e nel campo destinazione il MAC address del computer in B, lasciando intatta l'header IP.

Indirizzi in un frame Ethernet per un pacchetto da A a B (prima di arrivare a C)

	Source	Destination
Header IP	A	B
Header Ethernet	A	C

Indirizzi in un frame Ethernet per un pacchetto da A a B (superato C)

	Source	Destination
Header IP	A	B
Header Ethernet	C	B

Fino ad adesso si è fatto riferimento agli host solo attraverso gli indirizzi IP ed i MAC address. In realtà, navigando in Internet, gli utenti digitano nei loro browser indirizzi web come *www.yahoo.com*, invece degli indirizzi IP. Attraverso il protocollo **DNS** (*Domain Name Service*) gli indirizzi web vengono tradotti in indirizzi IP in maniera trasparente all'applicazione e memorizzati in una cache temporanea, in modo da non dover essere riconvertiti quando si digita un indirizzo al quale si è già acceduto (nella stessa sessione di navigazione). I server DNS sono organizzati in maniera gerarchica: ad esempio per trovare l'indirizzo IP corrispondente a download.yahoo.com la richiesta DNS giungerà prima al server del dominio .com, il quale la inoltrerà al DNS server di yahoo che cercherà nella propria tabella il record "download" ed il corrispondente indirizzo IP.

3 - Il Sistema Operativo Unix

Un Sistema Operativo alla stregua di Unix è un particolare programma che funge da ambiente per permettere la comunicazione tra gli applicativi e l'hardware; proprio per questo le specifiche applicazioni non devono preoccuparsi della compatibilità con l'hardware della macchina su cui girano e viceversa, in quanto tale funzione è controllata dal S.O. stesso.
Le componenti più importanti del S.O. sono:

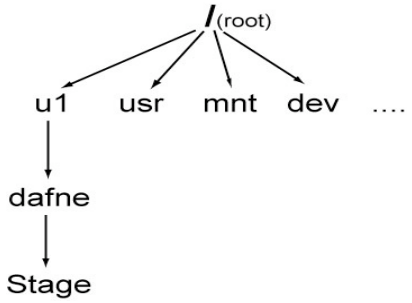
- il *File System*, che gestisce le memorie di massa (hard disk);
- la *Shell*, che permette all'utente di inviare comandi al S.O.

Esistono due tipi di *shell*: grafiche (GUI, *Graphic User Interface*) e non grafiche (CLI, *Command Line Interface*).

Shell grafiche sono per esempio KDE e Gnome, diffuse in ambiente *Linux* (variante di Unix) e CDE, diffusa in ambiente *Solaris* (altra variante). Tra le shell non grafiche troviamo Bash (Linux) e Tcsh, da noi utilizzata in ambiente Solaris 9.

In quanto priva dell'interfaccia grafica propria di Windows, per comunicare con Tcsh è necessario utilizzare alcuni comandi da tastiera. Il File System di Unix è di tipo gerarchico, quindi per passare in una sotto-directory bisogna tener conto della directory corrente. Tutte le directory discendono da un'unica radice (*root*), per cui è possibile indicare il percorso completo a partire dalla root stessa (/).

Per esempio, volendo raggiungere la sotto-directory dafne contenuta in /u1 si può digitare cd /u1/dafne indipendentemente dalla directory corrente (*path assoluto*), oppure spostarsi prima in u1 e poi entrare in dafne col comando cd dafne (*path relativo*).



Struttura gerarchica del File System Unix

I principali comandi da tastiera per muoversi all'interno della gerarchia Unix sono:

ls _ fornisce l'elenco dei file contenuti nella directory corrente;

ls-la _ permette di vedere anche i file nascosti e di vedere più dettagli;

cd <directory>	_ permette di muoversi nella directory indicata;
cd ~	_ *home*: ovunque ci si trovi riporta alla directory home (nel nostro caso dafne);
cd .	_ indica la directory corrente;
cd ..	_ permette di spostarsi nella directory immediatamente precedente;
cd /	_ permette di tornare alla *root*. Indispensabile per eseguire un *path assoluto*;
ps	_ fornisce l'elenco dei processi in esecuzione;
who	_ fornisce l'elenco degli utenti connessi al sistema;
man	_ apre la pagina informazioni del comando digitato (per es. man who);
&	_ digitato dopo un programma da lanciare permette l'esecuzione in background, evitando che si blocchi la *shell*;
kill -9 <n° processo>	_ blocca il processo in esecuzione;
bg	_ riprende il processo bloccato;
-display <nº macchina>	_ digitato dopo un programma da lanciare permette di visualizzarlo sul monitor della macchina indicata;

Per lanciare un programma basta digitarlo e premere invio.

I principali comandi per operare con file e directory sono:

mk <file>	_ crea (*make*) un file con il nome indicato;
rm <file>	_ cancella (*remove*) il file con il nome indicato;
cp <file sorgente> <destinazione>	_ copia (*copy*) e incolla il file nella destinazione indicata;
mv <file sorgente> <destinazione>	_ taglia (*move*) e incolla il file nella destinazione indicata;
mkdir <directory>	_ crea una directory con il nome indicato;
rmdir <directory>	_ cancella la directory con il nome indicato;
cpdir <directory sorgente> <destin.>	_ copia e incolla la directory nella destinazione indicata;
mvdir <directory sorgente> <destin.>	_ taglia e incolla la directory nella destinazione indicata;

Attenzione a specificare se si vuole operare con file o directory, soprattutto in caso di omonimia. Inoltre nel copiare e/o spostare file o directory meglio usare

sempre il *path assoluto* per localizzare sorgente e destinazione con chiarezza e univocità.

L'esecuzione dei programmi è materialmente affidata alla *CPU* (il microprocessore).

I software sono costituiti da una serie di istruzioni elementari che la CPU è in grado di comprendere: si tratta di funzioni aritmetiche o spostamenti di informazioni in memoria codificati nel cosiddetto *linguaggio macchina*, nel quale il flusso di dati è rappresentato in sistema *binario*.

Nei sistemi *multitasking* come Unix, in cui più applicazioni possono essere eseguite in parallelo, il S.O. dedica un determinato intervallo di tempo a ciascuna applicazione (nonché alle proprie funzioni di gestione). Le istruzioni di tali applicazioni vengono inviate al microprocessore, il quale le analizza (*unità di controllo*) e le esegue (ALU, *Arithmetic Logic Unit*).

I dati verranno poi trasferiti alle periferiche grazie ai noti bus.

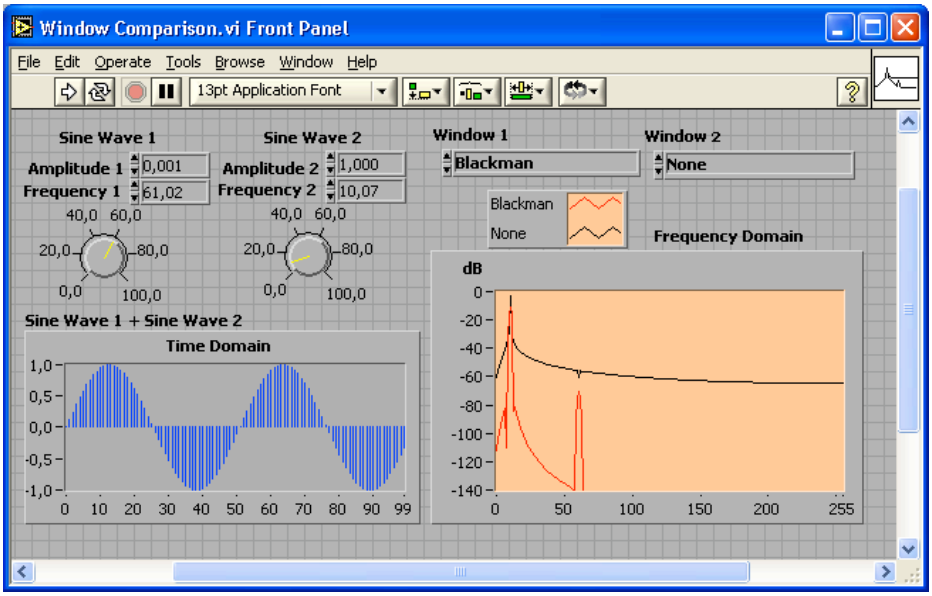
4 - LabView

LabView è un ambiente di sviluppo software a livello grafico che conserva la flessibilità di un linguaggio di programmazione semplificando notevolmente l'implementazione di applicazioni scientifiche.

LabView è un linguaggio di programmazione *data flow*, in cui cioè è il flusso di dati e non il tempo a scandire l'esecuzione delle operazioni. Ciò consente lo svolgimento di molteplici operazioni in parallelo.

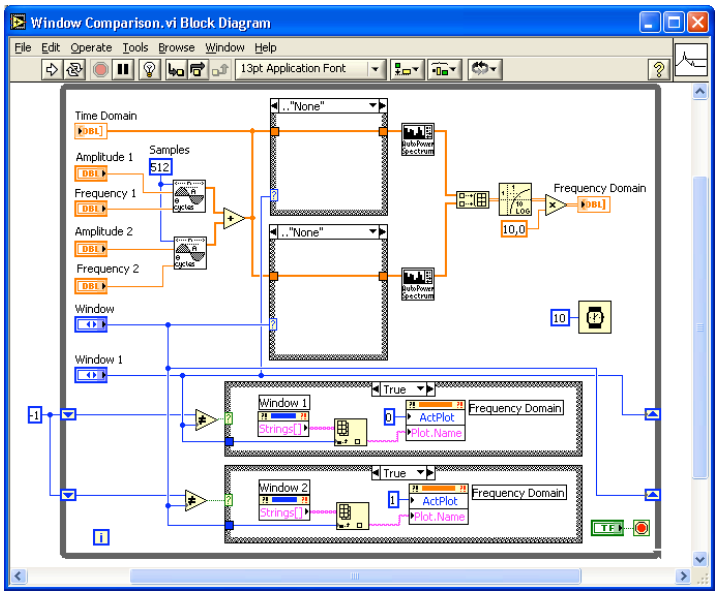
I programmi eseguibili con LabView sono detti *Virtual Instruments* (**VIs**) e includono un *Front Panel* e un *Block Diagram*:

- Il *Front Panel* (*Pannello Frontale*) rappresenta l'interfaccia utente del programma. Utilizzando gli strumenti predefiniti contenuti nella *Control palette* atti a misurare e automatizzare un segnale, si può creare e gestire un'interfaccia flessibile per numerose applicazioni. In figura e' riportato un front panel di esempio:



Esempio di *Front Panel*

- Il *Block Diagram* (*Diagramma a Blocchi*) contiene il source code (codice sorgente) grafico, ovvero una rappresentazione schematica e intuitiva della VI. Di seguito viene mostrato un esempio di block diagram:



Esempio di *Diagram*

Accedendo alla ricca libreria di funzioni contenuta nella *Function palette* è possibile acquisire segnali (input); scrivere i dati su file, rappresentare i segnali sottoforma di funzioni grafiche, display o indicatori che li analizzino e li misurino (output); creare dispositivi di controllo; eseguire operazioni aritmetiche, booleane, comparative; convertire dati in stringhe, operare con stringhe; scambiare dati tra applicazioni di 2 host via TCP o UDP.

LabView rende più intuitivo e veloce lo sviluppo del software, poichè il programmatore non deve riscrivere il codice per le funzioni che vuole implementare, in quanto molte di esse sono già presenti nelle librerie del programma.

I tipi di dati gestiti dai programmi si differenziano visivamente in base al colore dei cavi che collegano le VI o le funzioni di libreria, ad esempio nel modo seguente:

- Percorsi (dati non standard, per es. indirizzi di memoria);
- Stringhe;
- Booleani (Vero o Falso);
- Linee di errore;
- Intero a 32 bit.

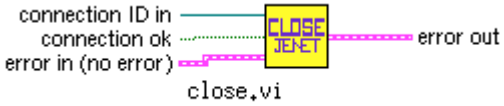
Per facilitare l'utilizzo delle funzioni di libreria, LabView possiede un efficace sistema di aiuto contestuale (*Context Help*): per avere informazioni su un qualsiasi componente (anche VI) del programma è sufficiente posizionare il cursore del mouse su di esso.

5 – Struttura del software sviluppato

Il software da noi sviluppato è costituito da 3 categorie di VI:

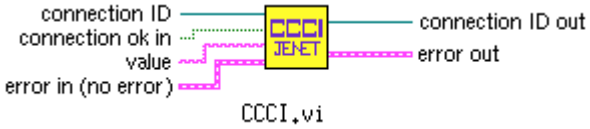
- VI per la gestione della *connessione TCP* al Crate Controller;
- VI per la gestione dei *comandi* riconosciuti dal Crate Controller;
- VI per la gestione degli *errori* (*Error Handler*).

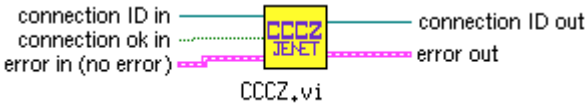
Per quanto riguarda la prima categoria, abbiamo sviluppato 2 VI: una per l'apertura della connessione ed una per la chiusura. Nella prima fase il programma provvede ad instaurare un canale di comunicazione TCP/IP sulla porta 2000 del Crate Controller (indirizzo IP prefissato). Eventuali errori vengono gestiti dalla VI Error Handler. In figura sono riportate le icone relative a questi due VI:

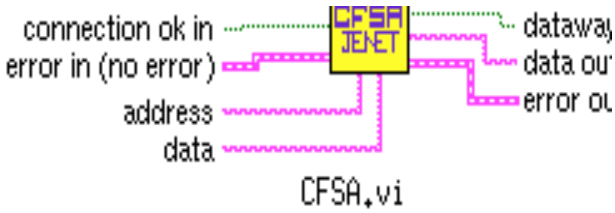


Vengono passati come parametri alle VI per la gestione dei comandi: il *Connection ID* (identificatore univoco della connessione); il *Connection OK*, ossia un valore booleano (TRUE o FALSE) che informa dell'avvenuta connessione; il flusso di errori (*Error Out*). Dopo l'esecuzione dei comandi (descritti in seguito) occorre chiudere il canale di comunicazione attraverso l'apposita VI, passandole come parametri il Connection ID, lo status della connessione ed il flusso di errori.

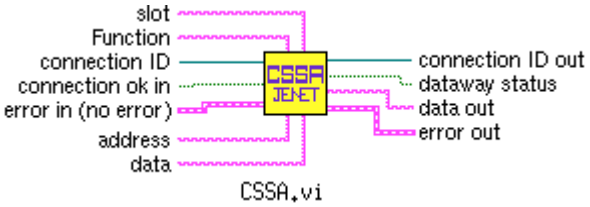
Le VI per la gestione dei comandi sono 9, una per ciascun comando. Sono dotate di tre ingressi fondamentali: Connection ID, Connection OK (status della connessione) ed Error In. Alcune VI possiedono degli input per i parametri specifici del comando. In uscita vi sono almeno due connettori: il Connection ID, da utilizzare per la chiusura della connessione o per l'invio di altri comandi; l'Error Out, che contiene gli errori avvenuti fino a questo punto, da connettere agli Error In di altre VI. Le VI dei comandi che restituiscono risultati di elaborazioni o letture hanno due output aggiuntivi chiamati *Data Out* (risultato) e *Dataway Status* (booleano che indica lo status del dataway CAMAC).
In figura sono mostrate le icone relative ai comandi descritti in precedenza:


connection ID
connection ok in
value
error in (no error)
CCCI
JENET
connection ID out
error out
CCCI.vi


connection ID in
connection ok in
error in (no error)
CCCZ
JENET
connection ID out
error out
CCCZ.vi


connection ok in
error in (no error)
address
data
CFSA
JENET
dataway
data ou
error ou
CFSA.vi

CLMR.vi


slot
Function
connection ID
connection ok in
error in (no error)
address
data
CSSA
JENET
connection ID out
dataway status
data out
error out
CSSA.vi


connection ID
connecton ok in
error in (no error)
CTCI
JENET
connection ID out
inhibit
error out
CTCI.vi

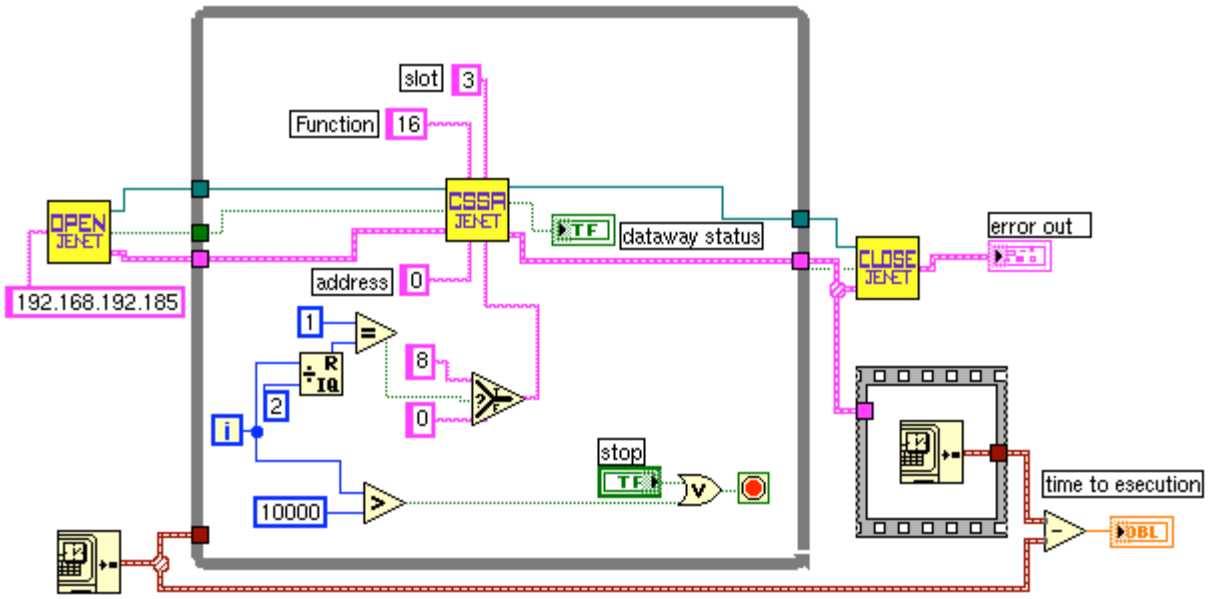

connection ID in
connection ok in
slot
error in
CTLM
JENET
connection ID out
LAM status
error out
CTLM.vi


connection ID in
connection ok in
error in (no error)
CTSTAT
JENET
connection ID out
dataway status
X signal
error out
CTSTAT.vi

Gli errori vengono gestiti all'interno delle VI di comando attraverso l'**Error Handler**. Quest'ultimo è in grado di riconoscere gli errori TCP più comuni e quelli del Crate Controller, fornirne descrizioni in base ai codici di errore e concatenare errori in sequenza. In questo modo si avrà alla fine del programma un unico testo di errore, contenente le descrizioni di tutti gli errori avvenuti durante l'esecuzione. Il testo è visualizzabile, insieme ad una checkbox che indica la corretta o errata esecuzione, collegando un indicatore all'Error Out della VI di chiusura (o di qualsiasi VI intermedia). In figura e' mostrata l'icona di tale VI:

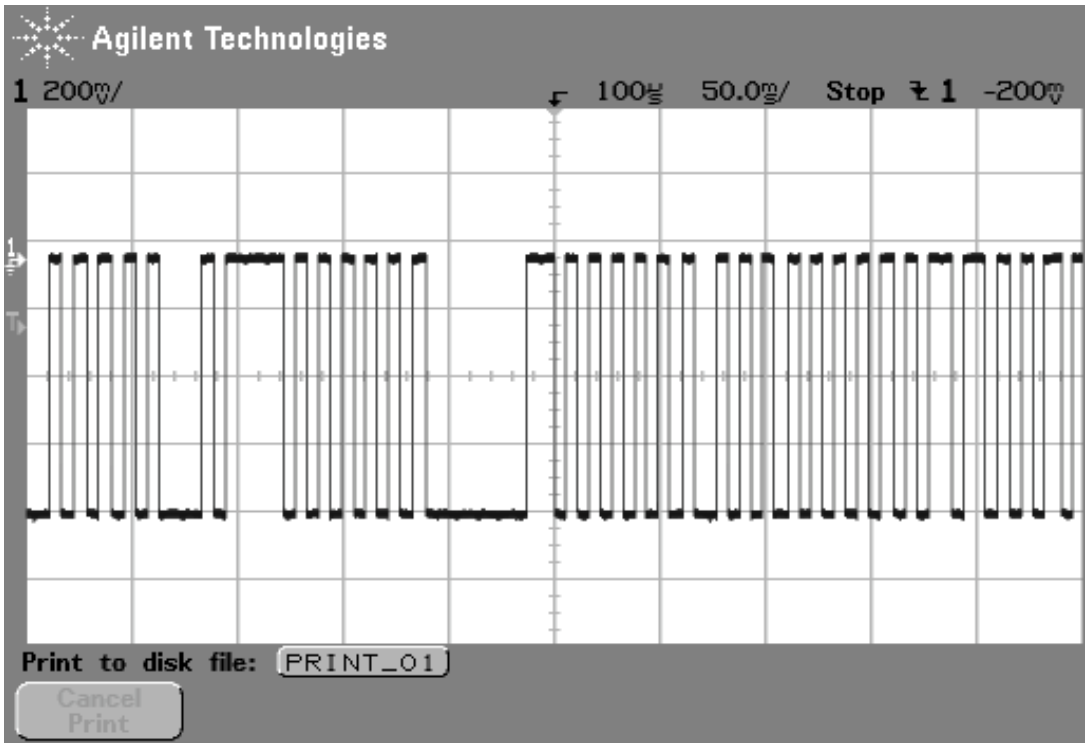
6 – Test e conclusioni

Per esemplificare come verranno utilizzate le librerie da noi create abbiamo realizzato un semplice programma di test il cui scopo e' generare un'onda quadra sull'uscita di un modulo di I/O digitale CAMAC. In figura e' mostrato il block diagram del programma di test:



Block Diagram del programma di test

Il programma esegue 10.000 cicli di comando (ad ogni ciclo, tramite il comando CSSA, viene imposta in modo alternato un'uscita a livello Hi o Lo) e calcola il tempo totale di esecuzione di questi comandi. Da varie prove eseguite risulta che mediamente occorrono circa 7 secondi per eseguire i 10.000 comandi. Nella figura seguente e' riportato un plot della misura eseguita con un oscilloscopio direttamente sul segnale di uscita digitale. Analizzando tale plot si nota facilmente che gli intervalli temporali in cui l'uscita e' Hi o Lo non sono uniformi (non si riesce cioe' ad ottenere un'onda rettangolare perfetta), cio' e' dovuto all'indeterminismo temporale comunicazione TCP/IP.



Plot temporale dell'uscita digitale utilizzata nel test finale

Dal block diagram del programma di test si nota che avere a disposizione delle librerie per il JENET semplifica molto la scrittura del software applicativo in quanto l'utilizzatore finale del controller deve solo preoccuparsi dell'applicazione che deve realizzare, essendo gia' disponibili in libreria i VI per gestire la connesione via TCP/IP e per l'esecuzione dei comandi. Posssiamo concludere quindi che l'obiettivo e' di realizzare dei drivers in LabView per il JENET stato raggiunto con successo ed in modo efficace.