

Progetto Giano

Protocollo per i comandi dell'elettronica di Giano

C.Baffa, E.Giani

Versione 1.09, Firenze 5 Maggio 2006

Sommario

Nel presente memo descriveremo il dettaglio del protocollo che viene usato nelle comunicazioni tra i due software, `server104` e `gbridge`, che controllano l'elettronica di acquisizione di Giano. Tale protocollo utilizza un formato estendibile alle comunicazioni tra gli altri moduli del progetto.

1 Introduzione

Il progetto della nuova elettronica di controllo per lo spettrometro Giano comprende un controller intelligente della elettronica di acquisizione. Tale controller viene implementato tramite un chip programmabile (una FPGA Stratix) che simula un microprocessore RISC. Su tale processore gira un programma di controllo (`server104`) che si occupa della programmazione e della gestione del sistema di acquisizione e di interfaccia con il rivelatore. Tali compiti vengono eseguiti sotto il totale controllo del software di basso livello che gira sulla workstation di controllo di Giano, `gbridge`. Quest'ultimo comunica anche con l'interfaccia di controllo utente, che per i test di laboratorio sarà l'applicazione GTK+ `GuiLab`.

Nel seguito descriveremo il dettaglio del protocollo che viene usato nelle comunicazioni tra `server104` e `gbridge`, che utilizza un formato estendibile alle comunicazioni tra gli altri moduli del progetto.

2 Aspetti generali del protocollo di comunicazione

La comunicazione tra `gbridge` e `server104` avviene tramite ethernet, con una socket unix-like, usando il protocollo descritto nel presente memo. Questo protocollo è organizzato a pacchetti, ispirandosi ai protocolli standard utilizzati in Internet[2].

I pacchetti di comunicazione hanno tutti un formato standard: sono composti di uno **header** di 8 parole di 16 bits e da un'eventuale area dati, a seguire.

La struttura dello header è la seguente:

Word	Descrizione	Esempi di valori
1	Magicnumber	0xA50F
2	Destinazione	Guilab, gbridge, server104
3	Tipo	comando/dato/messaggio/ack ...
4	Comando	DUMMYREAD, RUN ...
5	Lunghezzadati	da 0 a 1500
6	riservato	
7	Numeropacco	da 0 a 65535
8	Checksum	da 0 a 65535

Diamo alcuni cenni di spiegazione sui campi:

1. **Magicnumber.** Questa parola contiene una maschera che identifica l'inizio del pacchetto. È composta da un pattern di bit caratteristico (10100101 00001111). Agevola la sincronizzazione del protocollo.
2. **Destinazione.** Indica il processo destinatario del pacchetto. Raccogliamo che il byte alto indichi il processore, e quello basso il processo.
3. **Tipo.** Indica il tipo di pacchetto, se è un comando, una conferma di ricezione (**Ack**), un messaggio, una segnalazione di errore, o altro.
4. **Comando.** Indica l'operazione realmente richiesta, ad esempio in caso di pacchetto con Tipo uguale a Comando, il comando effettivo, o se il pacchetto è di tipo messaggio, la priorità (errore, avviso, ecc...).
5. **Lunghezzadati.** Indica l'eventuale lunghezza dell'area dati che segue lo header. Per ragioni di efficienza di trasporto, non deve superare 1400.
6. **riservato.** Questa parola è riservata per future espansioni.
7. **Numeropacco.** Identifica, ai fini del processo di origine, il pacchetto. Viene utilizzato nel protocollo di conferma (**Ack**).
8. **Checksum.** Maschera di controllo per la verifica dell'integrità del pacchetto. È la somma, troncata ai 16 bit bassi, delle parole 1-7 dello header.

Il protocollo comprende anche la possibilità di una conferma di ricezione (**Ack**). Questo avviene, ad esempio, nel controllo del processore Nios (elettronica di piano focale). Infatti, non essendoci un sistema operativo reale sul processore Risc, occorre che la workstation di controllo attenda una risposta prima di inviare il comando successivo.

Per inviare la conferma, il processo ricevente, invia indietro il solo header ricevuto, dopo aver sostituito al campo **Tipo** il valore di **Ack** (0x06, ricevuto, è il valore ASCII per ACK). Il pacchetto di risposta **Ack** può eventualmente contenere informazioni aggiuntive nell'area dati, che andranno comunque definite dal protocollo.

3 Area dati

Dopo l'header iniziale, il pacchetto può contenere un'area dati la cui dimensione in bytes è specificata dal campo **Lunghezza dati** dell'header. L'area dati può contenere il set di parametri necessari per l'esecuzione di un comando come pure la stringa con il messaggio o l'errore da comunicare al processo destinatario.

Nel caso in cui l'area dati contenga parametri relativi ad un comando, la formattazione di questi è stabilita dal protocollo stesso.

Le informazioni contenute nell'area dati sono codificate in formato ascii.

4 I valori dei campi per gbridge-server104

Diamo nelle seguenti tabelle i valori effettivamente usati per i vari campi dello header.

1. **Magicnumber.** Il solo valore accettato è 0xA50F.
2. **Destinazione.** Indica il processo destinatario del pacchetto.

Descrizione	Valore
GuiLab	0x1003
gbridge	0x1002
server104	0x1001

3. **Tipo.** Indica il tipo di pacchetto.

Descrizione	Valore
COMANDO	0x0010
MESSAGGIO	0x0020
ACK	0x0006
ERRORE	0xFF00

4. **Comando.** Indica l'operazione realmente richiesta. Se il pacchetto è di tipo messaggio, indica la severità (0-3). Nel caso del protocollo gbridge – server104 alcuni dei valori sono:

Nome	Valore	Commento
FILLMEM0	0x0101	Azzera le memorie di sequenza
DUMPMEM	0x0102	Esegue un dump delle memorie di sequenza
READPARM	0x0104	Legge un parametro dalla memoria
WRITEPARM	0x0105	Scrive uno o più parametri in memoria
LOADOBJ	0x0108	Carica un programma nelle memorie di sequenza
LOADWAVE	0x0109	Carica una forma d'onda da un file
DIT	0x0200	Tempo di integrazione
GROUP	0x0201	Numero di acquisizioni
DOUBLE	0x0202	Configura l'acquisizione in double/single mode
QUADRANTS	0x0203	Configura i quadranti da acquisire
RUN	0x0300	Fa partire il generatore di sequenze
START	0x0301	Run ma specificando DIT e GROUP
STOP	0x0302	Ferma il generatore di sequenze
ABORT	0x0303	Arresta subito il generatore di sequenze
DRYRUN	0x0307	Fa partire il generatore di sequenze
FLIPSVB	0x0308	Carica un programma che genera un'onda quadra
REINIT	0x0310	Reinizializza il sistema
STATUS	0x0400	Pubblica tutte le variabili di stato
READLOG	0x0410	Stampa i log
VERBOSE	0x0420	Configura il livello dei messaggi del server104
DUMMYACQ	0x0444	Restituisce un frame <i>sintetico</i>
NOP	0x04FF	Non esegue alcuna operazione. Usato come keep-alive

5. **Lunghezze dati.** Indica l'eventuale lunghezza dell'area dati. Per ragioni di efficienza di trasporto, non deve superare 1400.
6. **riservato.** Questa parola è riservata per future espansioni.
7. **Numeropacchetto.** Identifica, ai fini del processo di origine, può assumere tutti i valori permessi ad un numero a 16 bit. È un `unsigned short`.
8. **Checksum.** Maschera di controllo per la verifica dell'integrità del pacchetto. È la somma, troncata ai 16 bit bassi, delle parole 1-7 dello header. Può assumere tutti i valori permessi ad un numero a 16 bit. È un `unsigned short`.

Riferimenti bibliografici

- [1] "User Guide for the HAWAII-2 2048x2048 Pixel Focal Plane Array", A. Haas, Rockwell Scientific Company, LLC, 2002.
- [2] "Internet Core Protocols: The Definitive Guide" E. Hall, Sebastopol, CA (USA), 2000.