

## Progetto Giano

# Protocollo per la comunicazione tra gbridge e Giano GUI Con estensioni alla comunicazione tra nbridge e Nics GUI

C.Baffa, E.Giani

*Revision : 1.25*

## Sommario

*Nel presente memo descriveremo il dettaglio del protocollo che viene usato nelle comunicazioni tra il software di interfaccia utente Giano GUI e quello di medio livello (middleware) denominato gbridge. Tale protocollo utilizza il formato già descritto per la comunicazione tra il server104 e gbridge ed eredita da esso solo un sottoinsieme dei comandi. Questo protocollo viene usato anche nella comunicazione tra nbridge ed la nuova Nics GUI, e nel seguito daremo il dettaglio delle differenze tra le due situazioni.*

## 1 Introduzione

Il progetto della nuova elettronica di controllo per lo spettrometro Giano comprende un controller intelligente dell' elettronica di acquisizione. Tale controller viene implementato tramite un PC104. Su tale sistema 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 sono svolti sotto il totale controllo del software di basso livello gbridge, che è eseguito sulla workstation di controllo di Giano.

L'applicazione gbridge si interfaccia con il software di alto livello Giano GUI che fornisce all'utente un completo controllo dello strumento e delle misure.

gbridge funziona da intermediario (*applicazione middleware*) tra il software utente e il software di controllo dell'elettronica di basso livello.

Attraverso gbridge passano tutti i comandi, messaggi ed errori da e verso le applicazioni server104 e Giano GUI.

Nel seguito descriveremo il dettaglio del protocollo usato nelle comunicazioni tra Giano GUI e gbridge, che utilizza un formato che rappresenta un'estensione di quello utilizzato nella comunicazione tra il server104 e gbridge.

Questo protocollo è stato adottato anche per la comunicazione tra il nuovo programma utente di Nics, Nics GUI, e l'applicazione nbridge che, in modo analogo a gbridge, funziona da interfaccia "intelligente" tra il software di alto livello e il server *embedded ftest*.

Le differenze tra le due situazioni verranno descritte di volta in volta.

## 2 Aspetti generali del protocollo di comunicazione

La comunicazione tra `gbridge` e `Giano GUI` (ed in modo analogo tra `nbridge` e `Nics GUI`) avviene attraverso un `socket` usando il protocollo descritto nel presente memo.

Il `socket` può essere locale (`socket UNIX`), oppure del dominio Internet. Le applicazioni `Giano GUI` e `gbridge` sono eseguite sulla stessa workstation e ricorrono alla prima soluzione<sup>1</sup>. La tabella seguente indica le porte usate dai vari server

Server	Porta di ascolto
<code>server104</code> dati	8082
<code>server104</code> comandi	8083
<code>gbridge</code> comandi	8085
<code>lillend</code> comandi	8087
<code>lillend</code> web	8089

Questo protocollo è organizzato a pacchetti, e si ispira ai protocolli standard utilizzati in Internet[2]. I pacchetti di comunicazione hanno tutti un formato standard: sono composti da uno **header** di 8 parole di 16 bits e da un'eventuale area dati a seguire, di lunghezza massima 1400 bytes.

La struttura dello *header* è la seguente:

Word	Descrizione	Esempi di valori
1	Maginumber	0xA50F
2	Destinazione	<code>Giano GUI</code> , <code>niosserver</code>
3	Tipo	comando/dato/messaggio/ack ...
4	Comando	DUMMYREAD, RUN, ...
5	Lunghezzadati	da 0 a 1400
6	riservato	
7	Numeropacco	da 1 a 65535
8	Checksum	da 0 a 65535

Diamo di seguito alcuni cenni di spiegazione sul significato dei vari campi insieme a un'analisi più estesa

### 2.1 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.

Il solo valore accettato è 0xA50F.

### 2.2 Destinazione

Indica il processo destinatario del pacchetto.

Il byte alto indica il processore, quello basso il processo.

---

<sup>1</sup>Al momento `gbridge` implementa entrambe le possibilità.

I valori ammessi sono quelli riportati nella tabella sottostante.

Descrizione	Valore
server104	0x1001
gbridge	0x1002
Giano GUI	0x1003
nbridge	0x1004
Privembed	0x1005
couatl	0x1006
lillend	0x1007
lillend web interface	0x1008

## 2.3 Tipo

Indica il tipo di pacchetto: comando, conferma di ricezione (**Ack**), messaggio, segnalazione di errore, informazione o altro.

I valori accettati sono quelli riportati nella tabella.

Descrizione	Valore
COMANDO	0x0010
MESSAGGIO	0x0020
INFO	0x0030
ACK	0x0006
ERRORE	0xFF00

### 2.3.1 Tipo COMANDO

Questo tipo di pacchetto indica l'operazione realmente richiesta dal programma di alto livello, oppure dall'applicazione *middleware* **gbridge** (**nbridge**).

Il comando da eseguire è specificato nel campo 4 dell'intestazione (**Comando**).

### 2.3.2 Tipo MESSAGGIO

Un pacchetto di questo tipo descrive un messaggio informativo destinato all'utente, oppure al sistema di logging di **gbridge** (**nbridge**).

Tale pacchetto include il testo del messaggio, presente nell'**Area dati**, e il livello di importanza per l'utente, specificato nel campo 4 dell'intestazione(**Comando**).

Per maggiori dettagli rimandiamo al § 4.

### 2.3.3 Tipo ERROR

Un pacchetto di questo tipo descrive un messaggio di errore/avvertimento destinato all'utente e al sistema di logging di **gbridge** (**nbridge**).

Tale pacchetto include il testo del messaggio, presente nell'**Area dati**, e il codice dell'errore, specificato nel campo 4 dell'intestazione (**Comando**). Quest'ultimo fornisce informazioni dettagliate sul processo che l'ha generato.

Per maggiori dettagli rimandiamo al § 5.

### 2.3.4 Tipo ACK: acknowledgment del pacchetto

Il protocollo comprende anche la possibilità di una conferma di ricezione (Ack).

Mentre questa non è necessaria per i pacchetti di tipo MESSAGGIO e ERROR, è indispensabile per i pacchetti di tipo COMANDO.

Ogni comando che viene inviato dall'applicazione utente all'applicazione *embedded server104* o *gbridge* (*nbridge*) deve essere confermato dal processo ricevente con un pacchetto di tipo ACK costruito nel seguente modo:

Campo header	Pacchetto inviato	Pacchetto ACK
1	0xA50F	0xA50F
2	0x1001	0x1003
3	0x0010 (COMANDO)	0x0006 (ACK)
4	0x0400 (STATUS)	0x0400 (STATUS)
5	0	0
6		
7	11	11
8	checksum	checksum

Il pacchetto di risposta Ack può eventualmente contenere informazioni aggiuntive nell'area dati.

Una descrizione più precisa è presente nell'analisi dei vari comandi.

### 2.3.5 Tipo INFO

Il pacchetto di tipo INFO è usato durante il processo di presa dati per sincronizzare lo stato dell'acquisizione.

Il pacchetto INFO richiede che sia specificato il campo 4 dell'intestazione (Comando): i valori accettati sono quelli riportati di seguito nella tabella:

Descrizione	Valore
FRAME_STARTED	0x0001
FRAME_MULTI	0x0002
FRAME_READY	0x0003
FRAME_FINISHED	0x0004
FRAME_STOP	0x0005
FRAME_ABORT	0x0006
FRAME_WRITTEN	0x0007
FRAME_INTEG	0x0008
IDLE_ACQ	0x0009
MOTOR_INIT	0x0010
MOTOR_END	0x0011
MOTOR_ERR	0x0012
MOTOR_TIMEOUT	0x0013
SLIT_UNLOCKED	0x0014

Il pacchetto INFO può contenere un'Area dati che specifica una lista di soli numeri interi, terminata dal valore -1, il cui significato cambia a seconda del valore specificato nel campo 4. Il primo valore corrisponde al numero di argomenti specificati nell'Area dati. Di seguito riportiamo i vari casi:

- FRAME\_STARTED: la procedura di acquisizione del frame è iniziata. L'Area dati specifica il numero del frame in acquisizione.
- FRAME\_MULTI: la procedura di multicampionamento non distruttivo è iniziata. L'Area dati contiene diverse informazioni necessarie per etichettare le diverse varie misure.

Gli argomenti dell'Area dati sono tutti numeri interi che indicano:

- il numero di argomenti presente nella *payload*
  - il numero corrente dell'acquisizione
  - il numero complessivo di acquisizioni
  - il tempo di integrazione di ogni singola misura
  - il tempo di integrazione sul rivelatore per ogni singola misura
  - il tempo di ciclo di ogni singola misura: questo valore è utile per calcolare il timeout di ogni singola acquisizione
  - il tipo di operazione, indicata da un carattere. Questo valore specifica se la lettura avviene con lampada di calibrazione accesa oppure no e quale lampada è utilizzata.
- FRAME\_READY: la FIFO implementata sulla scheda buffer contiene un frame completo la cui lettura sta per iniziare.  
L'Area dati specifica il numero del frame.
  - FRAME\_FINISHED: l'acquisizione del numero di gruppi richiesti è terminata.  
L'Area dati contiene il numero dell'ultimo frame acquisito: questo deve coincidere con il numero dei gruppi.

- **FRAME\_STOP**: il processo di acquisizione è stato interrotto dall'utente con il comando STOP. L'**Area dati** specifica il numero dell'ultimo frame letto e trasferito a `gbridge` (`nbridge`). Questo numero è minore o uguale al numero di gruppi programmati.
- **FRAME\_ABORT**: il processo di acquisizione è stato fermato dall'utente con il comando ABORT. L'**Area dati** contiene due numeri:
  - il primo corrisponde al numero del frame interrotto
  - il secondo può essere 0 oppure un valore  $> 0$  per distinguere tra una richiesta esplicita di interruzione da parte dell'utente, e un errore interno del sistema di acquisizione. In questo caso il numero corrisponde al codice dell'errore che ha generato l'interruzione della presa dati.
- **FRAME\_WRITTEN**: il frame acquisito è stato scritto su disco. Il contenuto dell'**Area dati**, al momento, non è significativo, ma pensiamo di specificare o il numero del frame scritto, o il contatore sequenziale di frame, oppure il nome del file fits.
- **FRAME\_INTEG**: è stato inserito con l'obiettivo di mandare informazioni sullo stato del sistema durante l'integrazione.
 

**Al momento non è usato.**
- **IDLE\_ACQ**: il sistema *embedded* invia le informazioni riguardo lo stato delle acquisizioni *idle*. Il primo valore dell'**Area dati** corrisponde allo stato delle acquisizioni *idle*: se vale 0 non sono in esecuzione, se vale 1 sono in *run*.
 

**Al momento non è usato.**
- **MOTOR\_INIT**: indica che il programma `gianoMotors` è stato avviato e ha eseguito correttamente la procedura di *startup* durante la quale i controller dei motori vengono accesi in successione per accedere alle informazioni sulle posizioni dei motori, leggendo i rispettivi potenziometri. Non specifica alcuno argomento nell'**Area dati**.
- **MOTOR\_END**: indica che la procedura di movimento del motore è terminata e il motore ha raggiunto la posizione finale richiesta. L'**Area dati** deve specificare il numero del motore. Questi possono essere mossi anche in contemporanea, visto che ognuno di essi è gestito da un controller separato. È necessario dunque conoscere quale motore ha terminato il movimento.
- **MOTOR\_ERR**: specifica che il movimento in corso è terminato con un errore nel posizionamento finale.
- **MOTOR\_TIMEOUT**: necessario ??
- **SLIT\_UNLOCKED**: necessario ??

## 2.4 Comando

Come già in parte visto al § 2.3.1, l'interpretazione di questo campo cambia a seconda del campo **Tipo** specificato. Di seguito analizziamo le varie possibilità.

### **Campo Tipo = COMANDO**

In questo caso il valore del campo **COMANDO** corrisponde all'operazione realmente richiesta (vedi § 3) I valori attribuiti ai comandi sono stati suddivisi secondo il seguente schema:

- comandi relativi alla generazione di sequenza da 0x0100 a 0x01ff
- comandi relativi ai parametri dell'integrazione da 0x0200 a 0x02ff
- comandi acquisizione da 0x0300 a 0x03ff
- comandi di stato e debug da 0x0400 a 0x04ff

Nel caso del protocollo **Giano GUI – server104** i comandi attualmente definiti sono:

Nome	Valore	Commento
FILLMEM0	0x0101	Azzerà le memorie di sequenza
DUMPMEM	0x0102	Esegue un dump delle memorie di sequenza
READPARGM	0x0104	Legge un parametro dalla memoria
WRITEPARGM	0x0105	Scrive uno o piú parametri in memoria
LOADWAVE	0x0109	Seleziona una forma d'onda
DOUBLE	0x0202	Acquisizione single/double
QUADRANTS	0x0203	Configura i quadranti da acquisire
NOISE	0x0205	Acquisizione di rumore a sensore spento
SYNCHRO	0x0206	acquisizione frame completo di header
SVBTEST	0x0207	acquisizione immagine di test
SVBCHECK	0x0208	abilita check immagine di test
SEQMEM	0x0209	esegue il test della memoria di sequenza
FIFOTST	0x020A	esegue l'autotest delle FIFO
EXPERT	0x020B	configura modalitá laboratorio/operativa
ONDISK	0x0204	Abilita/disabilita la scrittura su disco
STOP	0x0302	Ferma il generatore di sequenze
ABORT	0x0303	Arresta subito il generatore di sequenze
INTEGRA	0x0304	Fa partire l'integrazione
FREERUN	0x0305	Free-run
SOCKDS9	0x0309	Fornisce il nome del socket DS9
REINIT	0x0310	Esegue la reinizializzazione
STATUS	0x0400	Pubblica tutte le variabili di stato
READLOG	0x0410	Stampa i log
VERBOSE	0x0420	Configura la verbositá del server embedded
MSGLEVEL	0x0430	Configura il livello minimo dei messaggi da inviare
DUMMYACQ	0x0444	Restituisce un frame <i>sintetico</i>
KILLTERM	0x0445	Ferma l'esecuzione del server <b>gbridge</b> ( <b>nbridge</b> )
STARTGM	0x0600	Richiede la connessione al programma dei motori
MSTATUS	0x0601	Fornisce lo stato del sistema di controllo dei motori
MOVE	0x0610	Richiede il movimento del motore
MSTOP	0x0611	Ferma il movimento del motore
MEXIT	0x0620	Esegue l'uscita dal programma di controllo dei motori <b>gianoMotors</b>
COUATLEND	0x0621	Termina l'esecuzione del server <b>couatl</b>

Per un'analisi piú dettagliata dei vari comandi, rimandiamo al § 3.

#### **Campo Tipo = INFO**

Rimandiamo al § 2.3.5

#### **Campo Tipo = MESSAGGIO**

Il numero che compare ha valore compreso tra 0 e 3 ed indica la severità del messaggio (vedi § 4).

#### **Campo Tipo = ERROR**

Il valore specificato corrisponde al codice numerico dell'errore (vedi § 5)

## 2.5 Lunghezzadati

Questo campo indica l'eventuale lunghezza dell'area dati che segue lo header. Per ragioni di efficienza di trasporto, non deve superare 1400.

Il campo `Lunghezzadati` esprime la lunghezza in bytes dell'area dati, compreso il carattere terminatore NULL.

## 2.6 riservato

Questa parola è riservata per future espansioni.

## 2.7 Numeropacco

Questo numero identifica, ai fini del processo di origine, il pacchetto.

Viene utilizzato nel protocollo di conferma (`Ack`). Assume tutti i valori permessi ad un numero a 16 bit compresi tra 1 e 65535. Il valore 0 è riservato ai comandi interni `gbridge` (`nbridge`) - *server104*. È un `unsigned short`.

## 2.8 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.

## 2.9 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 richiesti per l'esecuzione di un comando (pacchetti di **Tipo** COMANDO).  
La formattazione di questi è stabilita dal protocollo stesso<sup>2</sup>.
- la stringa con il messaggio per i pacchetti di **Tipo** MESSAGGIO
- la stringa con il messaggio di errore per pacchetti di **Tipo** ERROR

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

---

<sup>2</sup>Per il protocollo di comunicazione `gbridge` - `Giano GUI` la formattazione dei comandi che necessitano parametri aggiuntivi, è descritta al § 3

## 3 Descrizione dei comandi

Diamo qualche cenno sulla struttura dei comandi.  
Salvo dove altrimenti indicato, **il campo dati è in ASCII.**

### 3.1 READPARAM

Questo comando non specifica alcun parametro nell'Area dati.

#### Esempio di Pacchetto READPARAM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0104 (READPARAM)
header[4]	0 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

#### 3.1.1 ACK del pacchetto READPARAM

Il pacchetto ACK relativo, contiene nell'Area dati le informazioni relativi ai parametri delle schede analogiche sotto forma di coppie indirizzo-valore. In particolare il primo numero specifica quante coppie sono contenute nell'area dati. Il parametro letto è individuato dal valore del suo indirizzo, secondo la seguente tabella:

### Indirizzi parametri elettronica analogica

PROG_DIT	0x0001
BOARD_DIT	0x0002
NUM_RESET	0x0003
TRESET	0x0004
NUM_LETTURE	0x0005
PIX_LETTURA	0x0006
LETTURA_PIX	0x0007
PIXEL_WIDTH	0x0008
BASE_SCAN	0x0009
FSYNC_LSYNC	0x000A
LSYNC_VCLK	0x000B
D_LSYNC	0x000C
VCLK_SCAN	0x000D
VCLK_RESET	0x000E
VCLK_CLK1	0x000F
RESET_SCAN	0x0010
SCAN_LSYNC	0x0020
ENDFRAME	0x0030

Al momento tutte e quattro le schede analogiche sono programmate con gli stessi valori dei parametri, per cui il comando REAPARM restituisce 18 coppie di valori, tante quanti sono i parametri presenti nella precedente tabella.

### Esempio di Pacchetto ACK READPARM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1003 (GUICLIENT)
header[2]	0x0006 (ACK)
header[3]	0x0104 (READPARM)
header[4]	90 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	18 1 147 2 99 3 1 4 1 5 1 6 47 7 1 8 48 9 80 10 2 11 2 12 2 13 1 14 1 15 1 16 1 32 1 48 2

Se il comando READPARM viene inviato all'applicazione *embedded* quando quest'ultima è in acquisizione, il pacchetto ACK viene costruito dall'applicazione gbridge a partire dai valori salvati nelle variabili globali: queste descrivono l'ultimo stato aggiornato del sistema.

## 3.2 WRITEPARAM

Questo comando specifica nell'Area dati del pacchetto i valori dei parametri delle schede analogiche, da programmare nella memoria di sequenza di ciascuna scheda.

Nell'area dati viene specificato il numero delle coppie da programmare seguito dall'elenco delle coppie indirizzo-valore.

Ai parametri presenti nella tabella precedente, si aggiungono i seguenti:

### Offset degli indirizzi parametri elettronica analogica

WR_VRESET	0x00080A
WR_VBIAS	0x00080C
WR_OFF12	0x000812
WR_OFF34	0x000814
WR_FILTER1	0x000820
WR_FILTER2	0x000822
WR_SENISON	0x00081C
WR_SENSOFF	0x00081E

A differenza del pacchetto READPARAM, il numero delle coppie non è fisso, ma dipende da quali sono i parametri da programmare.

Ad esempio, nel caso della programmazione dei livelli di offset e bias per tutte e quattro le schede analogiche, avremo il seguente pacchetto:

### Esempio di Pacchetto WRITEPARAM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0105 (WRITEPARAM)
header[4]	87 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	8 2066 1500 2068 1500 6162 1500 6164 1500 10258 1500 10260 1500 14354 1500 14356 1500

## 3.3 LOADWAVE

Il comando deve specificare nell'area dati il canale del sequencer e il nome del file contenente i parametri della forma d'onda.

In previsione della modalità di acquisizione in multicampionamento, la sintassi di questo comando è stata modificata ulteriormente.

Nel caso del multicampionamento, il programma utente invidia all'applicazione `gbridge` il nome del file contenente le meta-istruzioni per la programmazione del sequencer.

Il contenuto del file ASCII verrà spedita dall'applicazione *middleware* come *Area dati* del pacchetto LOADWAVE. Nell'*Area dati* del comando devono essere specificati i seguenti parametri:

- canale: il numero del canale del sequencer. 1, 2, 3 4 o 5 nel caso di tutti e quattro
- nome file: il nome del file

Questo comando non è implementato nel protocollo di comunicazione Nics GUI-nbridge.

### Esempio di Pacchetto LOADWAVE

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0109 (LOADWAVE)
header[4]	36 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	5 /opt/softir/share/gbridge/seqfile.txt

## 3.4 DOUBLE

Questo comando specifica come parametro una flag intera che può assumere i valori 1 o 0. Nel primo caso viene abilitata la doppia acquisizione, nel secondo caso no.

## 3.5 QUADRANTS

Questo comando specifica un valore compreso tra 1 e 5, corrispondente al canale da acquisire.

### 3.5.1 Giano

Specificando 1, 2, 3, 4 viene eseguita l'acquisizione da un solo canale, quello corrispondente al numero presente nel campo dati del pacchetto. Per acquisire un frame intero, cioè tutti e 4 i quadranti, dobbiamo specificare il valore 5.

Nell'elettronica di GIANO, la corrispondenza tra canali (o schede analogiche) e i quadranti del rivelatore NICMOS, è la seguente:

- canale (scheda) 1 → quadrante 3
- canale (scheda) 2 → quadrante 2
- canale (scheda) 3 → quadrante 1
- canale (scheda) 4 → quadrante 4

Da qui l'uso del termine *canale* o *scheda analogica* al posto di *quadrante*, perché la selezione e programmazione del canale 1 (3), comporta la selezione e programmazione del quadrante 3 (1).

In tutta la documentazione, perciò, useremo i termini canali o schede analogiche, per non indurre in confusione.

### 3.5.2 Nics

Per Nics vengono ritenuti validi solo due valori : 1 per l'acquisizione di un singolo quadrante, 4 per l'acquisizione completa dell'array.

Il numero dei quadranti da acquisire non può comunque essere selezionato dall'utente perché dipende dalla modalità di esecuzione del server `ftest`. Quest'ultimo esegue di default una acquisizione completa dell'array (4 quadranti), mentre solo se è stato avviato con l'opzione `-q1` acquisisce un solo quadrante.

Nel caso di Nics, questo comando è locale alla comunicazione `Nics GUI-nbridge`: non viene inviato al task di destinazione `ftest` ma viene gestito internamente, e confermato, da `nbridge`.

L'applicazione `Nics GUI` dovrebbe specificare come campo destinazione del pacchetto il valore `0x1004` (task `nbridge`) e non `0x1001` (task `ftest`).

#### Esempio di Pacchetto QUADRANTS per GIANO

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0203 (QUADRANTS)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	5 (tutti e quattro canali attivi)

#### Esempio di Pacchetto QUADRANTS per NICS

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1004 (NBRIDGE)
header[2]	0x0010 (COMANDO)
header[3]	0x0203 (QUADRANTS)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	4 (tutti e quattro i quadranti attivi)

### 3.5.3 ACK del pacchetto QUADRANTS

Se l'Area dati del pacchetto QUADRANTS risulta malformattata oppure specifica un numero non valido per il numero del canale, l'applicazione *middleware* manda un messaggio di *warning*. Il pacchetto errato non è inviato all'applicazione `server104` ed è la stessa applicazione `gbridge` che si incarica di generare il pacchetto ACK del comando, trascrivendo nell'Area dati del pacchetto di risposta, l'ultimo valore programmato del parametro.

### 3.6 NOISE

Il comando NOISE consente di configurare la flag globale *NoiseMode* che abilita le misure di rumore con il sensore spento.

**Questa operazione è usata principalmente durante la fase di analisi di caratterizzazione dell'elettronica.**

#### Esempio di Pacchetto NOISE

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0205 (NOISE)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.7 SYNCHRO

Questo comando specifica nell'Area dati del pacchetto il valore 1 o 0.

Nel primo caso il comando SYNCHRO avvia l' acquisizione di *frames* completi dell'intestazione (*header di frame*).

Questa è costituita da 4 *word* a 16-bit che specificano oltre una parola di inizio riga (0xFFFF), il numero del frame, il numero di riga e una parola di fine frame (0x0000).

**Questa modalità è usata esclusivamente durante le operazioni di test dell'elettronica, per individuare eventuali errori di lettura e/o trasmissione del frame.**

#### Esempio di Pacchetto SYNCHRO

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0206 (SYNCHRO)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1 (immagine completa dello <i>header di frame</i> )

### 3.8 SVBTEST

Questo comando specifica nell'Area dati del pacchetto il valore 1 o 0.

Nel primo caso, il comando SVBTEST abilita la generazione interna da parte della scheda *buffer* di un'immagine sintetica (immagine di test), con un pattern ben preciso: i *pixels* di ogni riga assumono valori da 1 per il primo *pixel*, fino a 2048 per l'ultimo *pixel*.

**Questo comando viene usato esclusivamente durante la fase di analisi di funzionamento dell'elettronica, per individuare eventuali errori di lettura e/o trasmissione del frame.**

#### Esempio di Pacchetto SVBTEST

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0207 (SVBTEST)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1 (abilita la generazione dell'immagine sintetica da parte della scheda buffer)

### 3.9 SVBCHECK

Questo comando specifica nell'Area dati del pacchetto il valore 1 o 0.

Nel primo caso il comando SVBCHECK consente alla scheda *buffer* di abilitare il controllo dell'immagine di test ricevuta <sup>3</sup>

L'attivazione del comando SVBCHECK comporta anche l'attivazione del comando precedente, perchè quest'ultimo ha senso solo se viene generata l'immagine sintetica.

**Questo comando viene usato esclusivamente durante la fase di analisi di funzionamento dell'elettronica, per individuare eventuali errori di lettura e/o trasmissione del frame.**

#### Esempio di Pacchetto SVBCHECK

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0208 (SVBCHECK)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1 (esegue controllo dei pixels dell' immagine sintetica)

<sup>3</sup>Il controllo consiste nel verificare la correttezza dei dati ricevuti. Se questi non corrispondono ai valore attesi, la scheda *buffer* pone a 1 il bit piú significativo nel valore del pixel.

### 3.10 SEQMEM

Questo comando specifica come parametro il numero di test di memoria di sequenza (al massimo 100) da eseguire sul canale/i attivo/i al momento dell'attivazione della procedura.

**Questo comando viene usato generalmente durante la fase di analisi di funzionamento dell'elettronica, ma può essere eseguito saltuariamente per verificare il corretto funzionamento della memoria di sequenza delle schede analogiche.**

**Esempio di Pacchetto SEQMEM**

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0209 (SEQMEM)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	10 (esegue 10 test sulla memoria di sequenza del canale attivo)

### 3.11 FIFOTST

Questo comando specifica due parametri. Il primo rappresenta il numero della FIFO da cui leggere i dati generati dalla procedura di autotest, il secondo il numero di ripetizione del test (al massimo 15).

**Questo comando viene usato generalmente durante la fase di analisi di funzionamento dell'elettronica, ma può essere eseguito saltuariamente per verificare il corretto funzionamento delle FIFO della scheda *buffer*.**

**Esempio di Pacchetto FIFOTST**

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x020A (FIFOTST)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1 2 (esegue 2 autotest delle FIFO, leggendo i valori prodotti dal canale 1)

### 3.12 EXPERT

Questo comando specifica un solo parametro che può assumere valore 1 o 0. Nel primo caso viene abilitata la modalità di operazione uso laboratorio, con una gestione diversa delle acquisizioni *idle*.

Questo comando viene usato esclusivamente durante la fase di analisi di funzionamento dell'elettronica.

### Esempio di Pacchetto EXPERT

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0206 (EXPERT)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1 (seleziona modalità di laboratorio)

### 3.13 ONDISK

Questo comando specifica come parametro una flag intera che può assumere i valori 1 o 0. Nel primo caso viene abilitata la scrittura dei frame acquisiti su disco, nel secondo caso no.

Quest'ultima opzione può essere usata ad esempio nel caso del freerun.

**Questo comando risulta essere locale alla comunicazione GUI-gbridge (nbridge): non viene inviato al task di destinazione server104 (ftest) ma viene gestito internamente, e confermato, dall'applicazione middleware direttamente.**

**In questo caso il campo destinazione del pacchetto dovrebbe essere posto uguale a 0x1002 (task gbridge) 0x1004 (task nbridge) e non 0x1001 (task embedded server).**

### Esempio di Pacchetto ONDISK

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1004 (NBRIDGE) 0x1002 (GBRIDGE)
header[2]	0x0010 (COMANDO)
header[3]	0x0204 (ONDISK)
header[4]	2 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	1

### 3.14 STOP

Il comando STOP ferma l'operazione di acquisizione, completando il frame in esecuzione al momento della ricezione del comando.

Non specifica alcun argomento.

**Questo comando non è implementato nel protocollo Nics GUI- nbridge.**

### 3.15 ABORT

Il comando ABORT interrompe l'operazione di acquisizione, non completando il frame in esecuzione al momento della ricezione del comando.

Non specifica alcun argomento.

### 3.16 INTEGRA

Il comando INTEGRA deve specificare nell'area dati le seguenti informazioni:

- il nome e la locazione del file FITS
- il nome del file con le **keywords** da inserire nell'intestazione del file FITS
- il tempo di integrazione come numero a virgola mobile, espresso in secondi.

Devono essere fatte le seguenti considerazioni:

- esiste un tempo minimo di integrazione che corrisponde al tempo di scansione dell'array
- il tempo effettivo di integrazione può comunque differire da quello da programmare inserito dall'utente (DIT) per meno dello 0.08%. Questo effetto è analogo a quanto accade con Fasti[3]. I calcoli del tempo effettivo vengono eseguiti dal software **gbridge** ed il suo valore viene registrato nello **header** dei dati che risultano correttamente etichettati.

- il numero di integrazioni.
- il numero di coadds: al momento questo vale 1 per **gbridge** mentre può assumere un valore > 1 per **nbridge**.
- una flag di valore 1 (0) per l'attivazione (disattivazione) del *clipping* dell'immagine (non implementato in **nbridge**).

Questa operazione, se abilitata, viene eseguita direttamente dal demone *server104* ed è usata essenzialmente durante le operazioni di test. Se tale valore non viene espressamente dichiarato, viene letto come valore 0 e il *clipping* non viene abilitato.

#### Esempio di Pacchetto INTEGRA

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0305 (INTEGRA)
header[4]	86 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	/home/softir/data/giano/data0017.fts /opt/softir/share/gbridge/keywords.txt 3.0 5 1 0

### 3.17 FREERUN

Questo comando specifica come parametro il tempo di integrazione come numero a virgola mobile, espresso in secondi.

**NB:**

*Prima di inviare il comando FREERUN è opportuno porre a 0 la flag di scrittura su disco, inviando il comando ONDISK con argomento 0. I frame acquisiti vengono direttamente spediti sul display di ds9.*

#### Esempio di Pacchetto FREERUN

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0305 (FREERUN)
header[4]	4 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	3.0

### 3.18 MULTI

Questo comando specifica nell'Area dati le seguenti informazioni:

- il nome e la locazione del file FITS
- il nome del file con le **keywords** da inserire nello *header* del file FITS
- il nome del file ASCII con la lista di meta comandi per l'operazione di multicampionamento.

#### Esempio di Pacchetto MULTI

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0306 (MULTI)
header[4]	85 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	/home/softir/data/giano/data0017.fts /opt/softir/share/gbridge/keywords.txt multi.txt

### 3.19 SOCKDS9

Questo comando specifica il punto di accesso XPA (X Public Access) dell'applicazione ds9, sotto forma di indirizzo di *socket*.

Per i socket del dominio Internet (default per l'applicazione ds9), l'identificativo è del tipo *ip:port*.

#### Esempio di Pacchetto SOCKDS9

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1003/0x1004 (GBRIDGE/NBRIDGE)
header[2]	0x0010 (COMANDO)
header[3]	0x0309 (SOCKDS9)
header[4]	15 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	c0a81113:41243

### 3.20 Il comando REINIT

Il comando REINIT non specifica alcun argomento. Viene usato per re-inizializzare la catena di acquisizione del sistema *embedded*.

#### Esempio di Pacchetto REINIT

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0310 (REINIT)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.21 STATUS

Il comando STATUS restituisce il valore delle variabili di stato interne del server *embedded* (**gbridge** o **nbridge**).

### Esempio di Pacchetto STATUS

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x01003 (0x1001) (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0444 (STATUS)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

#### 3.21.1 Risposta del comando STATUS

La ricezione del comando STATUS viene confermata come tutti i comandi dal relativo pacchetto ACK.

A seguire viene generato l'output nella seguente forma:

```
status 1: Verbose           =1 (verbosity level <9)
status 2: Menu              =0 (Menu/Daemon flag)
status 3: Logfile           =1 (1= Logfile Active)
status 4: Sendmsg           =1 (Minimum msg lev sent)
status 5: VBuff_id          =EBB2 (buffer board id)
status 6: Acq_type          =0 (single/double read)
status 7: Extra_sub         =0 (Extra pixels subtraction)
status 8: Quadrante         =5 (quadrant to operate)
status 9: Acquired          =17978 (images acquired)
status10: Ncol              =1024 (column number)
status11: Nrow              =1024 (row number)
status12: Tint              =147 (physical integration time)
status13: Dit               =1000 (total integration time)
status14: Base_scan         =80 (pixel scan time)
status15: R_endframe        =2 (end of frame delay)
status16: R_fsync_ksync     =2 (fsync delay)
status17: D_ksync           =2 (ksync duration)
status18: R_ksync_vclk      =2 (ksync delay)
status19: Ngroup            =1 (integrations/group)
status20: Q1.Id             =10 (Board Id)
status21: Q1.Status         =0 (Board status)
status22: Q1.Filtro         =1 (ADC filter used)
status23: Q1.Sensore        =0 (Sensor status)
status24: Q1.Link           =1 (optical link status)
status25: Q1.V_reset        =0 (Value of V_reset)
status26: Q1.V_bias         =35000 (Value of V_bias)
status27: Q1.Offset12      =55500 (offset 1 e 2)
status28: Q1.Offset34      =55500 (offset 3 e 4)
```

status29: Q1.Vcc0n =1 (alimentation status)  
status30: Q1.seqfile =syntetic (sequencer filename)  
status31: Q1.Prog\_resnum =1 (reset performed)  
status32: Q1.Prog\_readclk=48 (duration of pixel)  
status33: Q1.Prog\_readdel=47 (delay of conversion)  
status34: Q1.Prog\_dit =147 (integration time)  
status35: Q1.Prog\_fsync =0 (fsync delays)  
status36: Q1.Prog\_lsync =0 (lsync delays)  
status37: Q1.Time\_cycle =10.0058 (total cycle time in seconds)  
status38: Q2.Id =2 (Board Id)  
status39: Q2.Status =0 (Board status)  
status40: Q2.Filtro =1 (ADC filter used)  
status41: Q2.Sensore =0 (Sensor status)  
status42: Q2.Link =1 (optical link status)  
status43: Q2.V\_reset =0 (Value of V\_reset)  
status44: Q2.V\_bias =65000 (Value of V\_bias)  
status45: Q2.Offset12 =38500 (offset 1 e 2)  
status46: Q2.Offset34 =38500 (offset 3 e 4)  
status47: Q2.Vcc0n =1 (alimentation status)  
status48: Q2.seqfile =syntetic (sequencer filename)  
status49: Q2.Prog\_resnum =1 (reset performed)  
status50: Q2.Prog\_readclk=48 (duration of pixel)  
status51: Q2.Prog\_readdel=47 (delay of conversion)  
status52: Q2.Prog\_dit =147 (integration time)  
status53: Q2.Prog\_fsync =0 (fsync delays)  
status54: Q2.Prog\_lsync =0 (lsync delays)  
status55: Q2.Time\_cycle =10.0058 (total cycle time in seconds)  
status56: Q3.Id =6 (Board Id)  
status57: Q3.Status =0 (Board status)  
status58: Q3.Filtro =1 (ADC filter used)  
status59: Q3.Sensore =0 (Sensor status)  
status60: Q3.Link =1 (optical link status)  
status61: Q3.V\_reset =0 (Value of V\_reset)  
status62: Q3.V\_bias =35000 (Value of V\_bias)  
status63: Q3.Offset12 =35500 (offset 1 e 2)  
status64: Q3.Offset34 =35500 (offset 3 e 4)  
status65: Q3.Vcc0n =1 (alimentation status)  
status66: Q3.seqfile =syntetic (sequencer filename)  
status67: Q3.Prog\_resnum =1 (reset performed)  
status68: Q3.Prog\_readclk=48 (duration of pixel)  
status69: Q3.Prog\_readdel=47 (delay of conversion)  
status70: Q3.Prog\_dit =147 (integration time)  
status71: Q3.Prog\_fsync =0 (fsync delays)  
status72: Q3.Prog\_lsync =0 (lsync delays)  
status73: Q3.Time\_cycle =10.0058 (total cycle time in seconds)

```
status74: Q4.Id          =4 (Board Id)
status75: Q4.Status      =0 (Board status)
status76: Q4.Filtro      =1 (ADC filter used)
status77: Q4.Sensore     =0 (Sensor status)
status78: Q4.Link        =1 (optical link status)
status79: Q4.V_reset     =0 (Value of V_reset)
status80: Q4.V_bias      =35000 (Value of V_bias)
status81: Q4.Offset12    =40000 (offset 1 e 2)
status82: Q4.Offset34    =40000 (offset 3 e 4)
status83: Q4.VccOn       =1 (alimentation status)
status84: Q4.seqfile     =syntetic (sequencer filename)
status85: Q4.Prog_resnum =1 (reset performed)
status86: Q4.Prog_readclk=48 (duration of pixel)
status87: Q4.Prog_readdel=47 (delay of conversion)
status88: Q4.Prog_dit    =147 (integration time)
status89: Q4.Prog_fsync  =0 (fsync delays)
status90: Q4.Prog_ksync  =0 (ksync delays)
status91: Q4.Time_cycle  =10.0058 (total cycle time in seconds)
```

### 3.22 READLOG

Questo comando stampa il log del contenuto delle FIFO.  
Non richiede la specifica di alcun parametro.

**Questo comando non è implementato nel protocollo di comunicazione Nics GUI-nbridge.**

### 3.23 VERBOSE

Questo comando specifica come parametro un valore compreso tra 0 e 9 che descrive il livello di verbosità dei messaggi ricevuti dal `server104`.

**Questo comando non è implementato nel protocollo di comunicazione Nics GUI-nbridge.**

### 3.24 MSGLEVEL

Questo comando specifica come argomento un valore intero compreso tra 0 e 3, corrispondente al livello minimo della severità del messaggio. Tutti i messaggi con severità maggiore o uguale a quella configurata, vengono inviati all' applicazione utente, gli altri vengono solo scritti sul file di log dell'applicazione `gbridge (nbridge)`.

### Esempio di Pacchetto MSGLEVEL

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1003/0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0430 (MSGLEVEL)
header[4]	0
header[5]	2
header[6]	numero pacchetto
header[7]	checksum
area dati	3

### 3.25 DUMMYACQ

Questo comando manda in esecuzione una singola acquisizione dummy e specifica come parametro aggiuntivo il *path* completo del file fits in cui salvare (eventualmente) l'immagine acquisita.

Il numero di gruppi da acquisire viene posto di default a 1.

#### Esempio di Pacchetto DUMMYACQ

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0444 (DUMMYDATA)
header[4]	36 (lunghezza area dati in bytes comprensiva del carattere NULL)
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	/home/softir/data/fasti/data0017.fits keywords.txt

### 3.26 KILLTERM

Questo comando consente di terminare l'applicazione `gbridge` (`nbridge`). Non specifica alcun argomento.

#### Esempio di Pacchetto KILLTERM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1003/0x1004 (GBRIDGE/NBRIDGE)
header[2]	0x0010 (COMANDO)
header[3]	0x0445 (KILLTERM)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.27 STARTGM

Questo comando richiede la connessione al server `couatl` che controlla le funzionalità del programma `gianoMotors`. Quest'ultimo è il software che gestisce direttamente i controller dei motori.

Non specifica alcun argomento.

#### Esempio di Pacchetto STARTGM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0600 (STARTGM)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.28 MSTATUS

Il comando richiede informazioni sullo stato del sistema di controllo dei motori e le loro posizioni.

Non specifica alcun argomento.

NB: la risposta a questo tipo di comando deve essere decisa!!

#### Esempio di Pacchetto MSTATUS

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0601 (MSTATUS)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.29 MOVE

Questo comando specifica come argomento il numero del motore che deve essere mosso e la configurazione finale delle ottiche. Quest'ultima sarà descritta da un codice corrispondente a una precisa combinazione di fenditura, filtro e posizione del reticolo.

### Esempio di Pacchetto MOVE

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0610 (MOVE)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	numero motore codice

### 3.30 MSTOP

Questo comando specifica come argomento il numero del motore che deve essere fermato. Questa procedura deve essere chiamata solo in caso di reale necessità. In condizioni normali i motori raggiungono senza problemi la posizione finale richiesta.

#### Esempio di Pacchetto MSTOP

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0611 (MSTOP)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	numero motore

### 3.31 MEXIT

Questo comando non specifica alcun argomento nell'Area dati e richiede l'uscita dal programma gianoMotors.

#### Esempio di Pacchetto MEXIT

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0620 (MEXIT)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

### 3.32 COUATLEND

Questo comando non specifica alcun argomento nell'Area dati e determina la fine dell'applicazione *server couatl*.

#### Esempio di Pacchetto COUATLEND

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0621 (COUATLEND)
header[4]	0
header[5]	0 (campo vuoto)
header[6]	numero pacchetto
header[7]	checksum
area dati	nessuna

Riassumendo:

Comando	Param 1	Param 2	Param 3	Param 4	Param 5	Param 6	Param 7
READPARM							
WRITEPARM	num coppie	addr1 val1	addr2 val2	..	..	..	..
LOADWAVE	channel	section	file size	filename	file content		
DOUBLE	flag						
QUADRANTS	quad						
NOISE							
SYNCRO	flag						
SVBTEST	flag						
SVBCHECK	flag						
SEQMEM	num. test						
FIFOTST	num. fifo	num. test					
EXPERT	flag						
ONDISK	flag						
STOP							
ABORT							
INTEGRA	path fits file	keyword file	DIT (sec.)	ngroup	coadds	clipping	
FREEERUN	DIT (sec.)						
SOCKDS9	sockname						
REINIT							
READLOG							
VERBOSE	level						
MSGLEVEL	msglevel						
DUMMYACQ	path fits file						
KILLTERM							
STARTGM							
MSTATUS							
MOVE	num motore	combinazione					
MSTOP	num motore						
MEXIT							
COUATLEND							

**I parametri specificati nell' Area dati DEVONO essere separati da uno spazio.**

## 4 Livello dei messaggi

Come accennato in precedenza, ogni messaggio può specificare nel campo **Comando** dell'header un livello di severità. Questo valore può assumere valori compresi tra 0 e 3, dove ciascun livello corrisponde ad una azione diversa che il software utente **Giano GUI** deve intraprendere:

- livello 0: il messaggio ricevuto è di tipo informativo. Non viene inviato ma può essere registrato nel log file del server *embedded*. Il senso di questi messaggi risiede nel loro uso durante il debug. Nella versione finale, utilizzata al telescopio, in genere non vengono più inviati.
- livello 1: da mostrare all'utente.
- livello 2: da mostrare all'utente e da loggare
- livello 3: solo da loggare

Tutti i messaggi inviati dal server *embedded* vengono trascritti nel file di log dell'applicazione *middleware gbridge (nbridge)*.

Solo quelli con severità maggiore o uguale a quella impostata dal comando **MSGLEVEL** saranno inviati all'applicazione di destinazione **Giano GUI (Nics GUI)**.

Ad esempio, se **MSGLEVEL** specifica un valore uguale a 3, l'applicazione utente **Giano GUI (Nics GUI)** non riceverà alcun messaggio proveniente direttamente dal server *embedded*, mentre tali messaggi saranno ricevuti sempre da *gbridge (nbridge)*.

## 5 Messaggi di errore

Il sistema di messaggistica degli errori è leggermente cambiato.

Adesso facciamo uso del campo 4 dell'intestazione per specificare un codice di errore che comprende al suo interno maggiori informazioni codificate come gruppi di bit.

- bit 15: se vale 0 si tratta di uno *warning*, se vale 1 è un errore
- bit[14-11]: indica il task operativo che ha generato l'errore
- bit[10- 0]: codice dell'errore

L'area dati contiene il messaggio di errore inviato da `gbridge`. Il software di alto livello può usare il codice di errore per generare un proprio messaggio, senza la necessità di pubblicare (se non in un eventuale file di log) il messaggio inviato dal *software gbridge*.

### 5.1 Codice identificativo dei TASK

INITDEV_TASK	0x1000	Processo di initializzazione
INTERNAL_TASK	0x2000	Processo interno (generico)
PROGRAM_TASK	0x3000	Processo programmazione elettronica
ACQ_TASK	0x4000	Processo di acquisizione
SOCKETIO_TASK	0x5000	Processo gestione network
PROTOCOL_TASK	0x6000	Processo gestione coerenza protocollo
TEST_TASK	0x7000	Processo di test dell'elettronica

### 5.2 Codice identificativo degli errori

	<b>Errori Accesso Memoria</b>	
GB_EMMEMIO	0x300	error in I/O access to mmaped memory
GB_BADADDR	0x301	memory bad address
GB_NOMMAPMEM	0x302	no memory mmaped
GB_EMEMALLOC	0x310	memory allocation error
GB_EINVALMEM	0x311	invalid memory pointer

	<b>Errori Programmazione</b>	
GB_EBADARG	0x320	invalid argument
GB_ENVALBOARD	0X321	invalid board (quadrant) number

	<b>Errori Program. Sequencer</b>	
GB_ESEQPROG	0X330	error in programming sequencer
GB_ESPROGFORMAT	0X331	wrong format in sequencer program
GB_ESPROGLONG	0x332	sequencer program too long
GB_ESPROGSHORT	0x333	sequencer program too short
GB_ESEQVERIF	0x334	sequencer memory verify error

	<b>Errori link ottico</b>	
GB_ELINK	0x340	errors in optical link
GB_ENOLINK	0x341	no optical link active
GB_ELINKVB	0x342	errors in optical link on buffer board
GB_ETRLINKVB	0x343	errors in transfer on VB board link

	<b>Errori scheda buffer</b>	
GB_EBUFFVERIF	0x344	buffer board verify error
GB_EBUFFID	0x345	invalid buffer board ID

	<b>Errori schede analogiche</b>	
GB_ESYSNOTOP	0x346	analog board not operative
GB_EIDLERUN	0x347	idle acquisitions running
GB_PROGERR	0x348	error in acquisition parameters programming
GB_HAMEG	0x349	error in Hameg power supply

	<b>Errori Lettura del Frame</b>	
GB_EIMGSNC	0x350	image desynchronization
GB_EPIXLESS	0x351	image desynchronization: less pixels
GB_EPIXMORE	0x352	image desynchronization: too pixels
GB_EHEADDESNC	0x353	image header desynchronization

	<b>Errori nelle FIFO</b>	
GB_EFOVERRUN	0x354	fifo or frame data overrun
GB_ETIMEOUT	0x355	timeout n I/O operations
GB_EFHEADER	0x356	error in image header
GB_EFDATA	0x357	error in image data autotest

	<b>Errori Protocollo Dati</b>	
GB_RANGE_ROW	0x360	error in interval row numbers
GB_MATCH_ROW	0x361	error in matching row number
GB_ACQ_PROT_ERR	0x362	error in image protocol
GB_ACQ_TIMEOUT	0x367	global acq timeout elapsed

	<b>Errori Interni</b>	
GB_ENVALOPT	0x380	invalid case option
GB_EFILEOPEN	0x381	error in opening file
GB_EINTERNAL	0x382	internal error
GB_GBRIDGE_KILL	0x383	gbridge received SIGTERM signal
GB_NSPACE_INVAL	0x384	
GB_CFTISIO_ERR	0x385	cfitsio error routine
GB_FITS_KE_EIO	0x386	can't access keyword file
GB_DS9_NOXPA	0x387	
GB_ESLIST	0x388	internal gbridge error
GB_ACQ_SAVE_ERR	0x389	error in saving data on disk
GB_ESYSBUSY	0x38A	warning, system is busy in acquisition

	<b>Errori Protocollo Comandi</b>	
GB_PROT_ERR	0x401	error in trasmission protocol
GB_CMD_NOTACK	0x402	command not confirmed by embed
GB_CHKSUM_ERR	0x403	protocol checksum error
GB_PROT_EFORMAT	0x404	not conformed format
GB_EINVALMASK	0x405	not conformed format
GB_FITGB_EPCK0	0x406	not conformed format
GB_FITGB_EPCK1	0x407	
GB_FITGB_EPCK2	0x408	

	<b>Errori I/O Socket</b>	
GB_IO_EBADFD	0x420	bad file number
GB_IO_EOF	0x421	end of file
GB_IO_TIME_READ	0x422	timeout in reading
GB_IO_TIME_WRITE	0x423	timeout in writing
GB_IO_TIME_CONNECT	0x424	timeout during socket connection
GB_IO_CONNECT_ERR	0x425	socket connection error
GB_IO_NONBLOCK_ERR	0x426	error in setting non-blocking
GB_ECOMMEMBED	0x427	embedded server not responding
GB_IO_POLLERR	0x428	error in poll
GB_IO_POLLHUP	0x429	POLLHUP condition
GB_IO_POLLNVAL	0x42A	POLLNVAL condition
GB_IO_CLOSED	0x42B	connection is closed

## 6 Capitolo con specifiche per Nics GUI- nbridge

### 6.1 Stato dell'acquisizione

Il programma Nics GUI come pure nbridge, necessita di conoscere lo stato dell'acquisizione durante le diverse operazioni.

Il sistema può trovarsi in uno dei seguenti stati:

- **Idle**: il sistema è pronto ad eseguire un qualunque comando inviato dall'applicazione Nics GUI
- **Busy**: il sistema ha ricevuto un comando di integrazione ed è in attesa della conferma da parte di ftest.
- **Running**: il sistema è in acquisizione
- **Abort**: la procedura di integrazione è stata interrotta

Lo stato di acquisizione del sistema può cambiare per i seguenti motivi:

1. è stata avviata un'acquisizione
2. l'acquisizione è stata fermata dall'utente
3. l'acquisizione è stata fermata da nbridge. Le possibili cause possono essere : un errore nel numero di riga, timeout sul socket dei dati, errore di allocazione di memoria, etc.
4. l'acquisizione è fermata da ftest. Le possibili cause possono essere ricercate tra: un errore del SVB, un errore generato dalla scheda di I/O, un errore nella conferma della riga, etc.

#### Caso 1

Nel primo caso, lo stato dell'acquisizione registrato da nbridge passa in un primo momento da da Idle a Busy. In questo stato, ogni comando inviato dall'applicazione Nics GUI diverso da ABORT o STATUS, non viene accettato da nbridge.

Lo stato dell'acquisizione passa infine da Busy a Running quando nbridge riceve da ftest il messaggio `'Frame acquisition started'`.

#### NB:

Ci teniamo a sottolineare che la ricezione del pacchetto ACK del comando INTEGRA (oppure FREERUN), significa **solo** che il comando è stato accettato da ftest, non che l'acquisizione sia effettivamente iniziata.

La ricezione della stringa `'Frame acquisition started'` significa, invece, che SVB è stato programmato correttamente e il sistema *embedded* sta procedendo all'integrazione.

Ne segue che lo stato dell'acquisizione deve essere posto a Running solo quando nbridge riceve il messaggio precedente.

## 6.2 Errori del sistema di acquisizione

Durante il processo complessivo di acquisizione, divisibile nelle fasi di integrazione, lettura e trasferimento, si possono verificare alcuni errori che ne causano l'interruzione.

In questi casi `nbridge` passa lo stato dell'acquisizione da `Running` a `Abort`, ed infine a `Idle`.

In questo paragrafo forniamo l'elenco dei messaggi di errori che vengono segnalati qualora l'operazione di acquisizione non vada a buon fine.

L'interruzione di una acquisizione può avvenire per una degli eventi riportati ai punti 2, 3 e 4 del § 6.1.

### Caso 2

Il programma `Nics GUI` invia il comando `ABORT` al task `ftest` per interrompere l'acquisizione.

Quando `nbridge` legge la richiesta proveniente da `Nics GUI`, pone lo stato dell'acquisizione a `Abort` e instrada il comando al server `ftest`. Una volta ricevuta la conferma del comando, `nbridge` intraprende la seguente serie di operazioni:

- svuota il buffer del `socket` dati per assicurarsi che non rimangano dati appartenenti all'integrazione interrotta
- esegue il reset delle variabili usate nel controllo del trasferimento delle immagini
- pone lo stato dell'acquisizione a `Idle`

### Caso 3 e 4

In questi due casi l'interruzione di una delle tre fasi del processo di acquisizione (integrazione, lettura, trasferimento), avviene in modo asincrono ed è segnalata da un messaggio di errore il cui task origine può essere sia `ftest`, sia `nbridge`. Il messaggio di errore ricevuto dall'applicazione `Nics GUI`, esplicativo del problema verificatosi, è sempre preceduto da una intestazione del tipo `'Fatal Error: ''`.

In questo caso l'applicazione `Nics GUI` deve considerare l'operazione di integrazione conclusa.

Di seguito riportiamo i messaggi di errore che il programma `Nics GUI` può ricevere, e forniamo una piccola spiegazione della causa che li ha generati, indicando anche il processo origine.

- **Fatal Error: Memory allocation error**

*Errore generato da nbridge*

In corrispondenza del comando `QUADRANTS` e al momento della ricezione del comando di acquisizione, viene effettuato un controllo sulla memoria allocata per lo stoccaggio delle immagini. Se questa non è allocata, oppure è allocata ma non è della dimensione giusta, `nbridge` procede a assegnare una regione di memoria delle dimensioni corrette. Se l'operazione fallisce, viene generato il messaggio di errore precedente e il processo di acquisizione è interrotto.

- **Fatal Error: command timeout. Command not confirmed by embedded system**

*Errore generato da nbridge*

Questo messaggio viene generato quando il comando di acquisizione è stato accettato da `nbridge` ma ancora non confermato da `ftest`, e quindi lo stato dell'acquisizione è `Busy`.

Se il processo permane nello stato `Busy` per un tempo superiore a circa 10 secondi senza aver ricevuto il pacchetto `ACK` del comando, allora `nbridge` genera questo messaggio di errore e pone lo stato dell'acquisizione a `Idle`.

- **Fatal Error: acquisition timeout**

*Errore generato da nbridge*

All'inizio di ogni misura<sup>4</sup> l'applicazione `nbridge` configura un timeout il cui valore è dato da:

$$timeout = (\text{tempo di integrazione della singola misura}) + 10 \text{ secondi}$$

Il valore precedente è calcolato per essere maggiore di quello necessario a portare a termine le operazioni di integrazione, lettura e trasferimento del frame<sup>5</sup> In tal modo se l'operazione di acquisizione non termina entro l'intervallo prestabilito, `nbridge` genera l'errore precedente e interrompe la procedura di acquisizione inviando il comando ABORT a `ftest`.

Questa situazione si può verificare a causa di un blocco del SVB oppure se risulta spenta una parte del sistema.

- **Fatal Error: Row value is outside valid range**

*Errore generato da nbridge*

L'applicazione `nbridge` ha ricevuto una riga il cui numero non appartiene all'intervallo di valori validi [0, 1023].

`nbridge` pone lo stato dell'acquisizione uguale a `Abort`, invia il comando ABORT a `ftest` e spedisce al programma `Nics GUI` il messaggio di errore descritto.

- **Fatal Error: Protocol error in data transfer**

*Errore generato da nbridge*

Questo errore si può verificare durante il trasferimento del frame. In particolare si è ripetuto per più di 50 volte consecutive la richiesta di ripetizione della medesima riga. Questo viene interpretato come un errore nel protocollo di comunicazione.

`nbridge` ferma il trasferimento del frame, pone lo stato dell'acquisizione uguale a `Abort`, invia il comando ABORT a `ftest` e spedisce a `Nics GUI` l'errore precedente per segnalare l'interruzione del processo di acquisizione.

- **Fatal Error: Error during saving of FITS file**

*Errore generato da nbridge*

Si è verificato un errore durante la scrittura del file FITS su disco.

`nbridge` pone lo stato dell'acquisizione uguale a `Abort`, invia il comando ABORT a `ftest` e spedisce a `Nics GUI` l'errore precedente per segnalare l'interruzione del processo di acquisizione.

- **Fatal Error: Acquisition aborted. Invalid command during image transfer**

*Errore generato da ftest*

---

<sup>4</sup>L'inizio di ogni integrazione è segnalata da `ftest` con l'invio del messaggio "Frame acquisition started". Tale messaggio viene però inviato solo in corrispondenza della prima integrazione. Nel caso di acquisizioni multiple, viene allora usato il messaggio "IntegrationFinished" inviato a `nbridge` al termine della lettura dalla scheda di I/O del frame acquisito.

<sup>5</sup>La velocità di lettura dalla scheda di I/O è in media di circa 5 MB/sec, mentre quella di trasferimento si aggira intorno a 3.5 MB/s. Il tempo impiegato per queste due operazioni è al massimo pari a 3 sec., ma si ottengono anche tempi morti di 5 sec.

Questo errore può manifestarsi quando `ftest` invia sul `socket` dei dati il frame acquisito. L'operazione di trasferimento del frame avviene secondo il seguente protocollo: il server `ftest` invia a `nbridge` l'immagine letta una riga per volta. Ogni riga è preceduta dal numero della riga a cui i dati si riferiscono (0, 1, 2, ... 1023).

`nbridge` deve confermare la corretta ricezione della riga con il messaggio 'FrameRowOK'. In caso contrario, `nbridge` chiede a `ftest` che la riga sia spedita nuovamente, inviando il messaggio "FrameRowRepeat *n*", dove *n* indica il numero della riga in questione.

Nell'operazione di trasferimento `ftest` è abilitato a ricevere sul `socket` comandi, oltre al comando ABORT, solo i due messaggi precedenti: un qualunque messaggio diverso da questi, causa l'interruzione del trasferimento e `nbridge` genera l'errore descritto.

- **Fatal Error: Acquisition Aborted. Timeout during image transfer**

*Errore generato da ftest*

Si è verificata una condizione di timeout durante il trasferimento dei dati. In particolare non è stata ricevuta la conferma (FrameRowOK) o la richiesta di ripetizione (FrameRowRepeat) della riga.

Se entro un intervallo di tempo uguale a *90 sec.*, `ftest` non riceve la conferma o la richiesta di ripetizione della riga, il trasferimento dell'immagine viene interrotta e `nbridge` invia al processo cliente Nics GUI il messaggio di errore riportato.

- **Fatal Error: Acquisition Aborted. Error during data transfer**

*Errore generato da ftest*

Durante il trasferimento dei dati si è manifestato uno degli errori descritti precedentemente oppure è stato ricevuto il comando ABORT.

`ftest` interrompe il trasferimento del *frame* verso `nbridge`.

- **Fatal Error: Acquisition Aborted. SVB inactive**

*Errore generato da ftest*

Questo messaggio viene generato quando SVB risulta non attivo. In questo caso `ftest` interrompe, oppure non avvia, la procedura di integrazione.

`nbridge` considera l'operazione di acquisizione interrotta, resetta lo stato di acquisizione da Running (o Busy) a Idle e invia al programma Nics GUI questo errore.

- **Fatal Error: Acquisition Aborted. SVB not correctly programmed**

*Errore generato da ftest*

Questa situazione è analoga alla precedente.

- **Fatal Error: Acquisition Aborted. SVB not responding**

*Errore generato da ftest*

Si sono verificati dei problemi nella comunicazione con SVB.

`ftest` interrompe la procedura di acquisizione e `nbridge` invia il messaggio di errore precedente al programma cliente Nics GUI.

- **Fatal Error: Acquisition Aborted. No clocking device!!**

*Errore generato da ftest*

Si sono verificati dei problemi con SVB.

`ftest` interrompe la procedura di acquisizione e `nbridge` invia il messaggio di errore precedente al programma cliente `Nics GUI`.

- **Fatal Error: Acquisition Aborted. Illegal integration time or group number**

*Errore generato da ftest*

Questa situazione non dovrebbe verificarsi, perchè il valore del tempo di integrazione e del numero dei gruppi viene controllato in precedenza. Comunque corrisponde ad un errore nella programmazione dei parametri dell'integrazione.

`ftest` interrompe la procedura di acquisizione e `nbridge` invia il messaggio di errore precedente al programma cliente `Nics GUI`.

- **Fatal Error: Acquisition Aborted. Ftest allocation memory error**

*Errore generato da ftest*

Si è verificato un errore interno a `ftest` relativo all'allocazione di memoria per immagazzinare il frame.

`ftest` interrompe la procedura di acquisizione.

- **Fatal Error: Acquisition Aborted. Error in I/O board reading**

*Errore generato da ftest*

Durante l'acquisizione dalla scheda di I/O si è verificato un errore di lettura.

`ftest` interrompe la lettura dei dati e il processo di acquisizione è fermato.

- **Fatal Error: Acquisition Aborted. Error in I/O board reading (buffer overflow)**

*Errore generato da ftest*

Si è verificata una condizione di overflow del buffer. Questo significa che i dati vengono letti troppo lentamente dalla scheda di I/O <sup>6</sup>, rispetto alla velocità di acquisizione dell'elettronica: il buffer di memoria in cui vengono immagazzinati temporaneamente i dati è pieno.

`ftest` interrompe la lettura dei dati dalla scheda di I/O e la procedura di acquisizione è fermata.

- **Fatal Error: Acquisition Aborted. Timeout during I/O board reading**

*Errore generato da ftest*

Si è verificato un timeout durante la lettura dalla scheda di I/O. `ftest` interrompe la lettura dei dati e il processo di acquisizione è fermato.

---

<sup>6</sup>I ritardi della lettura dalla scheda di I/O possono essere causati anche da una bassa velocità di trasferimento dei dati verso `nbridge`

- **Fatal Error: Acquisition Aborted. Ioctl error in I/O board**

*Errore generato da ftest*

Questo errore viene generato in seguito a un errore di programmazione della scheda di I/O. `ftest` interrompe l'operazione di lettura dalla scheda di I/O e l'acquisizione viene interrotta.

- **Fatal Error: Acquisition Aborted. I/O board not initialized**

*Errore generato da ftest*

La scheda di I/O non è stata configurata correttamente. `ftest` interrompe l'operazione di lettura dalla scheda di I/O e l'acquisizione viene interrotta.

## 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.
- [3] “Manuale d’uso del Software - III, Il programma di Controllo `Ftest`”, C. Baffa, E. Giani, Arcetri Technical Report, 4-2005