INAF - Osservatorio Astrofisico di Arcetri

Progetto Giano

Protocollo per la comunicazione tra Balor/Gbridge e GUI-tecnica

Con estensioni alla comunicazione tra Nbridge e Nics GUI

E.Giani, C.Baffa

Rapporto Interno di Arcetri N° **2-2012**
Revision : 1.38 **Firenze, 3/2012**

Sommario

Nel presente memo descriveremo il dettaglio del protocollo che viene usato nelle comunicazioni tra il software di interfaccia utente GUI-tecnica *e quello di medio livello (*middleware*) denominato* Balor/Gbridge*. Tale protocollo utilizza il formato già descritto per la comunicazione tra* Iron-Golem/Server104 *e* Balor/Gbridge *ed eredita da esso solo un sottoinsieme dei comandi. Questo protocollo viene usato anche nella comunicazione tra* Nbridge *e 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 (Iron-Golem/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 Balor/Gbridge, che è eseguito sulla workstation di controllo di Giano.
L'applicazione Balor/Gbridge si interfaccia con il software di alto livello, la GUI, che fornisce all'utente un completo controllo dello strumento e delle misure.
Balor/Gbridge funziona da intermediario (*applicazione middleware*) tra la GUI e il software di controllo dell'elettronica di basso livello. Attraverso Balor/Gbridge passano tutti i comandi, messaggi ed errori da e verso le applicazioni Iron-Golem/Server104 e la GUI.
Balor/Gbridge lavora in due modalità distinte: osservativa, quella di *default* e tecnica.
La prima sarà quella usata al telescopio, la seconda in laboratorio e in fase di manutenzione dello strumento.
La GUI usata nell'uno o l'altro caso è diversa: la GUI osservativa Giano-IDL è un'applicazione orientata all'osservatore, con un'avanzata gestione in tempo reale delle immagini acquisite e con accesso diretto al database del telescopio; la GUI-tecnica titan, invece, dedica particolare attenzione all'analisi di funzionamento e rilevazione degli errori dei vari sotto-sistemi dello strumento Giano.
Quando il sistema lavora in modalità osservativa la comunicazione tra Giano-IDL e Balor/Gbridge avviene esclusivamente in modo `sincrono`: Balor/Gbridge risponde solo a esplicite richieste generate dalla GUI le segnalazioni asincrone costituite da messaggi, errori o pacchetti informativi provenienti da Balor/Gbridge non sono gestite dalla GUI.
Per mantenere compatibilità tra i due diversi modi di lavorare di Balor/Gbridge, abbiamo dovuto apportare alcune modifiche al protocollo di comunicazione tra Balor/Gbridge e Giano-IDL, riducendo contestualmente anche il set dei comandi in uso. Le specifiche sono riportate nel documento [6].
Di seguito descriviamo il dettaglio del protocollo *standard*, la versione originale, usato nelle comunicazioni tra GUI-tecnica e Balor/Gbridge. Il protocollo *standard* rappresenta un'estensione di quello utilizzato nella comunicazione tra il Iron-Golem/Server104 e Balor/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 Balor/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 Balor/Gbridge e GUI-tecnica (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. La GUI e Balor/Gbridge sono eseguite sulla stessa workstation e ricorrono alla prima soluzione[1]. La Tab.1 indica le porte usate dai vari server.

[1] Al momento Balor/Gbridge implementa entrambe le possibilità.

Server	Porta di ascolto
Iron-Golem/Server104 dati	8082
Iron-Golem/Server104 comandi	8083
Balor/Gbridge comandi	8085
Lillend comandi	8087
Lillend web	8089

Tabella 1: Porte usate dai diversi server

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 riportata in Tab.2.

Word	Descrizione	Esempi di valori
1	Magicnumber	`0xA50F`
2	Destinazione	`Giano GUI,niosserver`
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

Tabella 2: Struttura degli *header*

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 valido è `0xA50F`.

2.2 Destinazione

Indica il processo destinatario del pacchetto. Il byte alto indica il processore, quello basso il processo. I valori ammessi sono quelli riportati in Tab.3[2].

[2]Il processo di destinazione `privembed` viene specificato nell'intestazione di un pacchetto di protocollo **SOLO** dall'applicazione Balor/Gbridge quando comunica direttamente con il processo embedded.

Descrizione	Valore
Iron-Golem/Server104	0x1001
Balor/Gbridge	0x1002
GUI-tecnica	0x1003
Nbridge	0x1004
Privembed	0x1005
Couatl	0x1006
Lillend	0x1007
Lillend web interface	0x1008
Xill	0x1009

Tabella 3: Possibili destinazioni dei messaggi

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

Tabella 4: Tipo di pacchetto

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 int Tab.4.

2.3.1 Tipo COMANDO

Questo tipo di pacchetto indica l'operazione realmente richiesta dal programma di alto livello, oppure dall'applicazione *middleware* Balor/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 Balor/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 Balor/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 Balor/Gbridge (Nbridge) deve essere confermato dal processo ricevente con un pacchetto di tipo ACK costruito nel modo descritto in Tab.5.

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

Tabella 5: Struttura di pacchetto ACK

Il pacchetto di risposta `Ack` *può eventualmente contenere informazioni aggiuntive nell'area dati. Una descrizione più dettagliata è presente nell'analisi dei vari comandi.*

2.3.5 Tipo INFO

Il pacchetto di tipo INFO è usato dai diversi processi per comunicare in modo asincrono a Balor/Gbridge l'aggiornamento dello stato di esecuzione del sotto-sistema supervisionato. Il pacchetto INFO richiede che sia specificato il campo **4** dell'intestazione (`Comando`) che consente di individuare a quale sotto-sistema si riferisce l'informazione allegata al pacchetto.
I valori accettati sono quelli riportati in Tab.6.

Il pacchetto INFO può contenere un'`Area dati` binaria che rappresenta il contenuto di una struttura, il cui tipo è diverso a seconda del sotto-sistema che origina il pacchetto.

Descrizione	Valore	Descrizione	Valore
_IFRAME_STARTED	0x0001	_IMOTOR_INIT	0X0010
_IFRAME_MULTI	0x0002	_IMOTOR_END	0X0011
_IFRAME_READY	0x0003	_IMOTOR_STOP	0X0012
_IFRAME_FINISHED	0X0004	_IMOTOR_ERR	0X0013
_IFRAME_STOP	0X0005	_IMOTOR_TIMEOUT	0X0014
_IFRAME_ABORT	0X0006	_IMOTOR_INFO	0x0015
_IFRAME_WRITTEN	0X0007	_IMOTOR_UPDATE	0X0016
_IFRAME_INTEG	0X0008	_ISLIT_UNLOCKED	0X0017
_IIDLE_ACQ	0X0009	_IXILL_INFO	0x0030

Tabella 6: Valori possibili del campo 4 per pacchetti INFO

Le strutture che possono costituire l'`Area dati` di un pacchetto INFO, sono le seguenti:

```
typedef struct _FRAME_Info_t {
    int index;         /**< index inside allocated mempool */
    int current;            /**< current frame number */
    int meas_x_group;       /**< number of repeated measures per group */
    int op;                 /**< multi-sampling acquisition operation */
    int errcode;            /**< acquisition error code (if any, else 0) */
    double dit;             /**< integration time of the single measure  */
    double prog_tint;       /**< integ time on detector for a single measure */
    double time_cycle;      /**< time cycle for a single measure */
    int prev_integ_row;     /**< file row number of the previous integration */
    int curr_integ_row;     /**< file row number of the current integration  */
    time_t token;           /**< the structure token (equal to the start int time) */
} FRAME_Info;



typedef struct sensorblock LILLEND_Info;
struct sensorblock {       /**< structure containing a single block of
                                sensor definitions */
    char datafile[32];      /**< filename for logging block data */
    char rrdfile[32];       /**< filename for rrdtools database */
    char basedir[64];       /**< base directory for logging block data */
    FILE *DatFptr;          /**< Pointer to data log file */
    int Sensdelay;          /**< Interval, in seconds, between sensors reads */
    time_t timestamp;       /**< log time of data.If -1 block not initialized*/
    double All_meas[MAXMEASURES];       /**< array of measures read */
    short All_meas_flag[MAXMEASURES];   /**< output format flag for array
                                           of measures read */
    struct device_s Sensor0;/**< first device (static) of the chain */
};
```

```
typedef struct _PRESLIT_Info_t {
    unsigned short status;  /**< Xill synthetic status */
    unsigned short th_status;   /**< Thorlabs status       */
    unsigned short pw_status;   /**< Power board status    */
    char thName[NAMELEN];   /**< thorlabs position name */
    long thPosition;        /**< Thorlabs device position*/
    unsigned short dimmer;  /**< dimmer value */
    int acc_x;              /**< board accelerometer x value */
    int acc_y;              /**< power board accelerometer y value */
    int acc_z;              /**< power board accelerometer z value */
    double temp;            /**< pre-slit sensor temperature */
    double alpha;           /**< power board derotator angle */
} PRESLIT_Info;
```

```
 typedef struct _MOTOR_Info_t {
    unsigned short status;  /**< Couatl synthetic status  */
    int num;                /**< motor number from 1 to 9 */
    int active;             /**< motor active bit status  */
    int enabled;            /**< flag to signal if motor movement is enabled */
    float ohm;              /**< motor position read from potentiometer (in ohm) */
    char namepos[32];       /**< motor position name        */
} MOTOR_Info;
```

Descriviamo il significato dei differenti valori del campo **4**.

- _IFRAME_STARTED: la procedura di acquisizione del frame è iniziata. Nella struttura FRAME_Info viene specificato il numero del frame in acquisizione e il tempo di inizio esposizione.

  Se l'acquisizione corrisponde a un multicampionamento, la struttura definisce anche i campi corrispondenti al numero complessivo di misure, al tempo di integrazione di ogni misura, al tipo di operazione e alla riga di esecuzione del programma di presa dati.

- _IFRAME_READY: la FIFO implementata sulla scheda buffer contiene un frame completo la cui lettura sta per iniziare. Nella struttura FRAME_Info viene specificato il numero del frame in acquisizione.

- _IFRAME_FINISHED: l'acquisizione del numero di gruppi richiesti è terminata. Nella struttura FRAME_Info viene specificato il numero del frame acquisito per ultimo.

- _IFRAME_STOP: il processo di acquisizione è stato interrotto dall'utente con il comando STOP. La struttura FRAME_Info specifica il solo elemento *num* corrispondente all'ultimo frame letto e trasferito a Balor/Gbridge (Nbridge). Questo numero è minore o uguale al numero di gruppi programmati.

- _IFRAME_ABORT: il processo di acquisizione è stato fermato dall'utente con il comando ABORT. La struttura `FRAME_Info` contiene:
  - l'elemento *num* che corrisponde al numero del frame interrotto
  - il codice di errore *errcode*. Quest'ultimo può essere uguale a 0 oppure a 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.

- _IFRAME_WRITTEN: il frame acquisito è stato scritto su disco. Il solo elemento valido della struttura è il primo, corrispondente al numero del frame.

- _IFRAME_INTEG: è stato inserito con l'obiettivo di mandare informazioni sullo stato del sistema durante l'integrazione.

  Al momento non è usato.

- _IIDLE_ACQ: il sistema *embedded* invia le informazioni riguardo lo stato delle acquisizioni *idle*.

  Al momento non è usato.

- _IMOTOR_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 informazione sulle posizioni e leggere i valori dei rispettivi potenziometri.

  L'`Area dati` è vuota.

- _IMOTOR_INFO: invia all'applicazione remota le informazioni su tutti i motori.

  Nell'`Area dati` viene eseguita la copia della struttura `MOTOR_Info` i cui campi sono descritti di seguito.

  - **status**: lo stato sintetico del sistema, descritto in [4]
  - **num** : il numero del motore che ha terminato il movimento
  - **ohm**: la posizione del motore espressa in ohmi, letta dal potenziometro,
  - **namepos**: il nome della posizione associata al valore in ohm, oppure NONE se esiste alcuna corrispondenza. La corrispondenza nome-posizione è definita in un file di sistema (vedi [8]).

- _IMOTOR_END: indica che la procedura di movimento del motore è terminata e il motore ha raggiunto la posizione finale richiesta. L'`Area dati` contiene la copia della struttura `MOTOR_Info` con i dati aggiornati.

- _IMOTOR_ERR: specifica che il movimento in corso è terminato con un errore nel posizionamento finale. L'`Area dati` contiene la copia della struttura `MOTOR_Info` con i dati aggiornati.

- _IMOTOR_STOP: specifica che il movimento in corso è terminato con un errore nel posizionamento finale. L'`Area dati` contiene la copia della struttura `MOTOR_Info` con i dati aggiornati.

- _IMOTOR_UPDATE: consente di aggiornare la posizione in ohm dell'ottica in movimento. L'`Area dati` contiene la copia della struttura `MOTOR_Info` con i dati aggiornati.
- _ISLIT_UNLOCKED: **Al momento non è usato**.
- _IXILL_INFO: fornisce la configurazione e lo stato del sistema di pre-slit.

  L'`Area dati` contiene la copia della struttura `PRESLIT_Info` i cui campi sono:

  - **status**: lo stato sintetico del sistema di pre-slit, come descritto in [4]
  - **th_status**: lo stato del sistema Thorlabs
  - **pw_status**: lo stato del sotto-sistema della PowerBoard.
  - **thPosition**: posizione in micro-step del device Thorlabs
  - **thName**: nome logico della posizione in step
  - **dimmer**: percentuale d'intensità della lampada di calibrazione (10% o 100%)
  - **temp**: temperatura del sistema di pre-slit in Celsius
  - **acc_x**: componente x dell'accelerometro
  - **acc_y**: componente y dell'accelerometro
  - **acc_z**: componente z dell'accelerometro
  - **alpha**: angolo del derotatore

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.

2.4.1 Campo Tipo = COMANDO

In questo caso il valore del campo Comando corrisponde all'operazione richiesta (vedi § 3) dall GUI. 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
- comandi destinati al programma di controllo del sistema dei motori (Couatl) da 0x0600 a 0x06ff
- comandi destinati al programma di telemetria (Lillend) da 0x0700 a 0x07ff
- comandi destinati al programma di pre-slit (Xill) da 0x0900 a 0x09ff

I comandi attualmente definiti sono elencati in Tab.7.

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

2.4.2 Campo Tipo = INFO

Rimandiamo al § 2.3.5

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
LOADWAVE	0x0109	Seleziona una forma d'onda
GROUP	0x0201	Numero di acquisizioni con stesso programma nel sequencer
DOUBLE	0x0202	Acquisizione single/double
QUADRANTS	0x0203	Configura i quadranti da acquisire
ONDISK	0x0204	Abilita/disabilita la scrittura su disco
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 modalità laboratorio/operativa
DUMMYFILE	0x020C	Abilita/disabilita il caricamento di un file custom per le idle acquisition
GETIMAGEFILENAME	0x020D	Richiede il nome del file fits
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
MULTI	0x0306	Multi sampling acquisition
SOCKDS9	0x0309	Fornisce il nome del socket DS9
REINIT	0x0310	Esegue la reinizializzazione
STATUS	0x0400	Pubblica tutte le variabili di stato
ASTATUS	0x0401	Pubblica lo stato sintetico del sistema di acquisizione
READLOG	0x0410	Stampa i log
VERBOSE	0x0420	Configura la verbosità del server embedded
MSGLEVEL	0x0430	Configura il livello minimo dei messaggi da inviare
DUMMYACQ	0x0444	Restituisce un frame *sintetico*
KILLTERM	0x0445	Ferma l'esecuzione del server Balor/Gbridge (Nbridge)
NOGUISS	0x0446	disabilita la modalità osservativa
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
MINVERT	0x0611	reverse grating movement direction
MSTOP	0x0612	Ferma il movimento del motore
MEXIT	0x0620	Esegue l'uscita dal programma di controllo dei motori gianoMotors
COUATLEND	0x0621	Termina l'esecuzione del server couatl
XSTATUS	0x0900	Fornisce lo stato del sistema di pre-slit
SWITCH	0x0910	accende/spenge un sotto-sitema della Power Board
WHEEL	0x0912	muove la ruota Thorlabs
WHEEL_STOP	0x0921	ferma la ruota Thorlabs
XILLCONF	0x0922	spedisce il file delle posizioni della giostra Thorlabs

Tabella 7: Comandi del il protocollo GUI-tecnica – Iron-Golem/Server104

2.4.3 Campo Tipo = MESSAGGIO

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

2.4.4 Campo Tipo = ERROR

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

2.5 Lunghezzadati

Questo campo indica l'eventuale lunghezza in bytes 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 *privati* tra Balor/Gbridge (Nbridge) - *server embedded*.
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 informazioni sia in formato ASCII sia binario. Solo i pacchetti di protocollo di tipo INFO hanno un'`Area dati` binaria.

2.9.1 Area dati in formato ASCII

La maggior parte dei pacchetti con una payload ha un'area dati di tipo Ascii. Elenchiamo varie possibilità.

- il set di parametri richiesti per l'esecuzione di un comando (pacchetti di **Tipo** COMANDO). La formattazione di questi è stabilita dal protocollo stesso[3].
- la stringa con il messaggio per i pacchetti di **Tipo** MESSAGGIO
- la stringa con il messaggio di errore per pacchetti di **Tipo** ERROR

2.9.2 Area dati in formato binario

L'applicazione Balor/Gbridge presenta una nuova struttura rispetto al progetto originale in cui gestiva la comunicazione solo con il processo di acquisizione.
Balor/Gbridge raprresenta uno snodo centrale per altri processi: il programma di controllo dei motori di Giano e del sistema di pre-slit. Questi processi iniviano all'applicazione *middleware* molte informazioni simultanemaente. L'estensione del programma a nuove potenzialità ha perció reso necessario una revisione anche del protocollo di comunicazione per introdurre la possibilità di gestire una *payload* di tipo binario: le informazioni sono impacchettare in una struttura che viene poi copiata nell'`Area dati` del pacchetto.
Per l'`Area dati` in formato binario eseguiamo il padding a zero dell'area di memoria in cui viene copiata la struttura. In questo modo abbiamo una situazione coerente a quella dell'area dati ascii che provvede sempre a terminare con un carattere nullo l'area di memoria.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0104 (READPARM)
header[4]	0
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 8: Esempio di Pacchetto READPARM

3 Descrizione dei comandi

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

[3]Per il protocollo di comunicazione Balor/Gbridge - GUI-tecnica la formattazione dei comandi che necessitano parametri aggiuntivi, è descritta al § 3

3.1 READPARM

Questo comando non specifica alcun parametro nell'`Area dati`. Un esempio di tale pacchetto è dato in Tab.8.

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

Tabella 9: Indirizzi dei parametri dell'elettronica analogica

3.1.1 ACK del pacchetto READPARM

Il pacchetto ACK contiene nell'`Area dati` le informazioni relative ai parametri delle schede analogiche sotto forma di coppie indirizzo-valore. Il primo numero presente nell'`Area dati` specifica quante sono in totale le coppie da leggere. Ogni parametro associato ad una scheda analogica è individuato dal proprio indirizzo, secondo la corrispondenza riportata in Tab.9.
Tutte e quattro le schede analogiche sono programmate con gli stessi valori dei parametri. Il comando READPARM restituisce 18 coppie di valori, tante quanti sono i parametri presenti nella tabella citata.
NOTA: Se il comando READPARM viene inviato all'applicazione *embedded* **quando quest'ultima è in acquisizione, il pacchetto ACK viene costruito da Balor/Gbridge a partire dai valori salvati nelle variabili globali che descrivono lo stato dell'elettronica relativo all'ultimo aggiornamento.**

3.2 WRITEPARM

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 in Tab.8, si aggiungono quelli elencati in Tab.11.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1003 (GUICLIENT)
header[2]	0x0006 (ACK)
header[3]	0x0104 (READPARM)
header[4]	lunghezza area dati in bytes
header[5]	0
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

Tabella 10: Esempio di Pacchetto ACK READPARM

A differenza del pacchetto READPARM, 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 pacchetto mostrato in Tab.12.

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

Tabella 11: Offset degli indirizzi parametri elettronica analogica

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. Un esempio di pacchetto è dato in Tab.13.
Per la procedura di multicampionamento, la GUI invia all'applicazione Balor/Gbridge il nome del file contenente le meta-istruzioni per la programmazione del sequencer.
Il contenuto del file ASCII viene spedito 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 con le meta-istruzioni

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

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0105 (WRITEPARM)
header[4]	lunghezza area dati in bytes
header[5]	0
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

Tabella 12: Esempio di Pacchetto WRITEPARM

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0109 (LOADWAVE)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	5 /opt/giano/share/gbridge/seqfile.txt

Tabella 13: Esempio di Pacchetto LOADWAVE

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 GROUP

Questo comando specifica nell'`Area dati` il numero di acquisizioni consecutive eseguite con lo stesso programma nella memoria del sequencer. Questo comando può essere inviato, se necessario, prima della richiesta di acquisizioni di multi campionamento, per programmare il numero di gruppi di ripetizione della misura.

3.6 QUADRANTS

Questo comando programma il canale del rivelatore da cui acquisire l'immagine. Nell'`Area dati` specifica un valore compreso tra 1 e 5.

3.6.1 Comando QUADRANTS per Giano

Valori uguali a 1, 2, 3, 4 programmano l'acquisizione da un solo canale, quello corrispondente al numero specifcato come argomento.
Per acquisire un frame intero (4 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 doumentazione, perció, useremo i termini canali o schede analogiche, per non indurre in confusione.

3.6.2 Comando QUADRANTS per 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 deve specificare come campo destinazione del pacchetto il valore **0x1004** (task Nbridge) e non **0x1001** (task ftest).

3.6.3 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 Iron-Golem/Server104 ed è la stessa applicazione Balor/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.

3.7 ONDISK

Questo comando specifica come parametro una flag intera che può assumere 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-Balor/Gbridge *(*Nbridge*): non viene inviato al task di destinazione* Iron-Golem/Server104 *(*ftest*) ma viene gestito internamente, e confermato dall'applicazione middleware direttamente. In questo caso il campo destinazione del pacchetto*

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

Tabella 14: Esempio di Pacchetto QUADRANTS per GIANO

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

Tabella 15: Esempio di Pacchetto QUADRANTS per NICS

deve essere posto uguale a **0x1002** *(task* Balor/Gbridge*)* **0x1004** *(task* Nbridge*) e non* **0x1001** *(task embedded server).*

3.8 NOISE

Il comando NOISE configura la flag *NoiseMode* che abilita le misure di rumore a sensore spento.
Questa operazione è usata durante la fase di analisi di caratterizzazione dell'elettronica.

3.9 SYNCHRO

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). Questo comando specifica nell'`Area dati` del pacchetto il valore 1 o 0, per abilitare oppure no questa *feature*.
Questa modalità è usata esclusivamente durante le operazioni di test dell'elettronica, per individuare eventuali errori di lettura e/o trasmissione del frame.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1004 (NBRIDGE) **oppure** 0x1002 (GBRIDGE)
header[2]	0x0010 (COMANDO)
header[3]	0x0204 (ONDISK)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	1

Tabella 16: Esempio di Pacchetto ONDISK

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

Tabella 17: Esempio di Pacchetto NOISE

3.10 SVBTEST

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 specifica nell'`Area dati` il valore 1 o 0 per abilitare oppure no questa *feature*.
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.

3.11 SVBCHECK

Questo comando abilita nella scheda *buffer* il controllo dell'immagine di test ricevuta. Questo 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.
L'attivazione del comando SVBCHECK comporta necessariamente il comando precedente, per la generazione dell'immagine sintetica. Specifica nell'`Area dati` l valore 1 o 0 per abilitare oppure no questa *feature*.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0206 (SYNCHRO)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	1 (immagine completa dello *header di frame*)

Tabella 18: Esempio di Pacchetto SYNCHRO

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

Tabella 19: Esempio di Pacchetto SVBTEST

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.

3.12 SEQMEM

Questo comando avvia uno o piú di test della memoria di sequenza da eseguire sul canale/i attivo/i. Specifica nell'`Area dati` il numero di test da eseguire per un massimo di 100.
Questo comando viene usato generalmente durante la fase di analisi di funzionamento dell'elettronica. Puó essere eseguito saltuarialmente per verificare il corretto funzionamento della memoria di sequenza delle schede analogiche.

3.13 FIFOTST

Questo comando avvia una procedura di autotest delle FIFO. Specifica nell'`Area dati` due parametri: il numero della FIFO da cui leggere i dati generati e il numero di ripetizioni del test, per un massimo di 15.

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

Tabella 20: Esempio di Pacchetto SVBCHECK

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

Tabella 21: Esempio di Pacchetto SEQMEM

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

3.14 EXPERT

Il comando EXPERT abilitata la modalità di operazione uso laboratorio,con una diversa gestione delle acquisizioni *idle*. Specifica nell'`Area dati` un solo numero intero: `1` o `0` per abilitare o disabilitare questa *feature*.
Questo comando viene usato esclusivamente durante la fase di analisi di funzionamento dell'elettronica.

3.15 DUMMYFILE

Il comando DUMMYFILE abilita la modalità di caricamento da un file del programma di *idle acquisition* . Specifica nell'`Area dati` un massimo di due parametri. La presenza del secondo parametro dipende dal valore del primo.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x020A (FIFOTST)
header[4]	lunghezza area dati in bytes
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)

Tabella 22: Esempio di Pacchetto FIFOTST

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

Tabella 23: Esempio di Pacchetto EXPERT

Il primo è una flag che puó valere 1 o 0. Se questa vale 1 la *feature* è attivata e il secondo parametro è il nome del file con le istruzioni per le meta-istruzioni per eseguire le *idle*.
Se la flag è 0, le acquisizioni *idle* sono quelle pre-caricate di default all'inizializzazione dal sistema di acquisizione.
Questo comando è usato durante la fase di caratterizzazione del rivelatore scientifico.

3.16 GETIMAGEFILENAME

Questo comando richiede il nome del file FITS in scrittura. Non specifica alcun argomento nell'`Area dati` e viene usato esclusivamente da Giano-IDL quando il sistema lavora in modalità osservativa.

3.16.1 ACK del pacchetto GETIMAGEFILENMAE

Nel pacchetto ACK del comando GETIMAGEFILENAME viene restituito il nome, completo del path, del successivo file fits da acquisire.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x020C (DUMMYFILE)
header[4]	lunghezza area dati in bytes comprensiva
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	1 /home/giano/data/resources/dummyobj.txt

Tabella 24: Esempio di Pacchetto DUMMYFILE

Campo	Contenuto
header[0]	0xA50F
header[1]	0x1001
header[2]	0x0010
header[3]	0x020D
header[4]	0
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 25: Esempio di pacchetto GETIMAGEFILENAME

3.17 STOP

Il comando STOP ferma l'operazione di acquisizione, completando il frame in esecuzione al momento della ricezione del comando. Non specifica alcun argomento.
Nelle operazioni di multicampionamento con un numero di gruppi programmato > 1, il termine dell'acquisizione avviene alla fine della sequenza di comandi, quindi l'operazione di multicampionamento in corso viene completata e non viene iniziata quella relativa al gruppo successivo.
Questo comando non è implementato nel protocollo Nics GUI-Nbridge.

3.18 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.

3.19 INTEGRA

Il comando INTEGRA avvia una o piú acquisizioni i cui parametri devono essere specificati nell'`Area dati` come elencato di seguito.

ACK GETIMAGEFILENAME

Campo	Contenuto
header[0]	0xA50F
header[1]	0x1003 (Giano-IDL)
header[2]	0x0006
header[3]	0x020D
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	/home/giano/data/giano/20120308/data0011.fts

Tabella 26: Esempio di pacchetto ACK GETIMAGEFILENAME

- il nome e la locazione del file FITS (solo per Nbridge)
- 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 eseguti dal software Balor/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 è fissato a 1 per Balor/Gbridge mentre puó assumere 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 Iron-Golem/Server104 ed è usata essenzialmente durante le operazioni di test. Se questo parametro non viene passato come argomento al comando, Balor/Gbridge lo pone uguale a 0 e il *clipping* non viene abilitato.

3.20 FREERUN

Questo comando avvia una serie continua di acquisizioni, per un massimo di 10000 frames. Specifica come parametro il tempo di integrazione espresso in secondi.
NB: *Prima di inviare il comando* FREERUN *è opportuno inviare il comando* ONDISK *con argomento* 0 *per disabilitare la scrittura dei frames su disco. I frame acquisiti vengono automaticamente mostrati sul display di ds9, se attivo.*

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 SERVER104
header[2]	0x0010 (COMANDO)
header[3]	0x0305 (INTEGRA)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	/home/fasti/data/fasti/data0002.fts 3.0 5 1 0 (Nbridge)
`area dati`	3.0 5 1 0 (Balor/Gbridge)

Tabella 27: Esempio di Pacchetto INTEGRA

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

Tabella 28: Esempio di Pacchetto FREERUN

3.21 MULTI

Il comando avvia un'acquisizione di multi-campionamento. Le operazioni del campionamento multiplo sono definite in un file ASCII come una lista di meta-comandi. Il file con le meta-istruzioni è processato da Iron-Golem/Server104 che lo traduce in un file oggetto per programmare il *sequencer*. L'`Area dati` deve specificare le seguenti informazioni:

- il numero del canale attivo: 1, 2, 3, 4 oppure 5 per tutti e quattro i canali attivi. 5 è il default.
- il nome del file ASCII con la lista di meta comandi per l'operazione di multicampionamento.

3.22 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'identificato è del tipo *ip:port*.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104)
header[2]	0x0010 (COMANDO)
header[3]	0x0306 (MULTI)
header[4]	lunghezza area dati in bytes comprensiva
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	5 /home/giano/data/resources/multi.txt

Tabella 29: Esempio di Pacchetto MULTI

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001) (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0400 (STATUS)
header[4]	0
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 30: Esempio di Pacchetto STATUS

3.23 REINIT

Il comando REINIT re-inizializza la catena di acquisizione del sistema *embedded*.
Non specifica alcun argomento.

3.24 STATUS

Il comando STATUS restituisce lo stato delle variabili del sistema di acquisizione.

3.25 Risposta del comando STATUS

La ricezione del comando STATUS viene confermata come tutti i comandi dal relativo pacchetto ACK. A seguire viene mostrato l'output generato dal demone Iron-Golem/Server104:

```
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)
```

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

Tabella 31: Esempio di Pacchetto SOCKDS9

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
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 32: Esempio di Pacchetto REINIT

```
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_lsync  =2 (fsync delay)
status17: D_lsync        =2 (lsync duration)
status18: R_lsync_vclk   =2 (lsync 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.VccOn       =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.VccOn       =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.VccOn       =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_lsync  =0 (lsync delays)
status91: Q4.Time_cycle  =10.0058 (total cycle time in seconds)
```

3.26 ASTATUS

Questo comando richiede lo stato sintetico del sistema di acquisizione.
Non richiede la specifica di alcun parametro.

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1002 (GBSERVER)
header[2]	0x0010 (COMANDO)
header[3]	0x0401 (ASTATUS)
header[4]	0
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 33: Esempio di Pacchetto ASTATUS

3.27 ACK ASTATUS

La payload del pacchetto di risposta al comando ASTATUS descrive lo stato complessivo del sistema di acquisizione in forma sintetica. Il dettaglio del contenuto dell'area dati è illustrato nei due documenti [4] e [5]. Il primo è relativo al funzionamento del sistema in modalità osservativa, l'altro in modalità tecnica.

3.28 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.

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

Tabella 34: Esempio di Pacchetto MSGLEVEL

3.29 VERBOSE

Il comando configura il livello di verbosità dei messaggi ricevuti dal Iron-Golem/Server104. Specifica come argomento un numero intero compreso tra 0 e 9, con 0 corrispondente al valore minimo.
Questo comando non è implementato nel protocollo di comunicazione Nics GUI-Nbridge.

3.30 MSGLEVEL

Questo comando configura il livello minimo della severità dei messaggi. Specifica come argomento un intero compreso tra 0 e 3.
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 Balor/Gbridge (Nbridge).

3.31 DUMMYACQ

Questo comando manda in esecuzione una singola acquisizione sintetica (*dummy*). Per Nbridge specifica come parametro aggiuntivo il *path* completo del file fits in cui salvare (eventualmente) l'immagine acquisita, mentre per Balor/Gbridge l'area dati è vuota.
Il numero di gruppi da acquisire viene posto uguale a 1.

3.32 KILLTERM

Questo comando termina l'applicazione Balor/Gbridge (Nbridge). Non specifica alcun argomento.

3.33 NOGUISS

Questo comando disabilita in Balor/Gbridge la flag globale SSFlag. Questa risulta attiva di *default* all'avvio di Balor/Gbridge per garantire un corretto funzionamento dell'applicazione Giano-IDL quando il sistema opera in modalità osservativa.
Quando la flag è abilitata (TRUE), tutti i messaggi, gli errori, i pacchetti INFO e i pacchetti ACK dei comandi diversi da ASTATUS, MSTATUS, XSTATUS e GETIMAGEFILENAME sono rediretti sul file

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1001 (SERVER104/FTEST)
header[2]	0x0010 (COMANDO)
header[3]	0x0444 (DUMMYDATA)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	/home/fasti/data/fasti/data0017.fts (Nbridge)
`area dati`	0 (Balor/Gbridge)

Tabella 35: Esempio di Pacchetto DUMMYACQ

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
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 36: Esempio di Pacchetto KILLTERM

`GUISS.out` invece di essere inviati a Giano-IDL. In questo modo Balor/Gbridge e Giano-IDL comunicano in modo sincrono: Balor/Gbridge risponde SOLO su richiesta esplicita della GUI.
Quando è attiva la GUI tecnica titan, la modalità osservativa abilitata dalla flag SSFlag deve essere modificata: la GUI-tecnica invia il comando NOGUISS al momento della connessione a Balor/Gbridge che pone la flag SSFlag uguale a FALSE.
Il valore della flag viene nuovamente portato al valore TRUE quando la GUI si disconnette, in modo da lasciare attiva di *default* la modalità osservativa.

3.34 STARTGM

Questo comando avvia la connessione al server Couatl che controlla le funzionalità del programma gianoMotors, il software che gestisce i controller dei motori delle ottiche di Giano (vedi [9]). Non specifica alcun argomento.

3.35 MSTATUS

Il comando richiede lo stato sintetico del sistema di controllo dei motori e le loro posizioni. Non specifica alcun argomento.

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

Tabella 37: Esempio di Pacchetto STARTGM

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

Tabella 38: Esempio di Pacchetto MSTATUS

3.36 ACK MSTATUS

La payload del pacchetto di risposta al comando MSTATUS descrive lo stato del sistema dei motori in forma sintetica. Il dettaglio del contenuto dell'area dati è illustrato nei due documenti [4] e [5]. Il primo è relativo al funzionamento del sistema in modalità osservativa, l'altro in modalità tecnica.

3.37 MOVE

Questo comando muove l'ottica selezionata dalla GUI. Specifica come argomento il numero del motore e il nome della posizione. La conversione tra *nome della posizione* e *posizione in ohm* viene eseguita dal programma `gianoMotors`.
I motori da gestire sono in totale otto ma quelli da muovere sono solamente tre.
La corrispondenza tra il numero del motore ed il motore è la seguente:

- reticolo: 1
- filtro: 6
- fenditure: 8

I files con le posizioni dei diversi elementi ottici (estensione .pos) si trovano nella directory di sistema: `/opt/softir/bin/share/gianoMotors/IS2/config`

Campo	Contenuto
header[0]	0xA50F (MAGICKMASK)
header[1]	0x1006 (COUATL)
header[2]	0x0010 (COMANDO)
header[3]	0x0610 (MOVE)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	1 HR

Tabella 39: Esempio di Pacchetto MOVE

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

Tabella 40: Esempio di Pacchetto MEXIT

3.38 MINVERT

Questa procedura è chiamata per invertire la direzione di movimento del reticolo qualora la posizione scelta non sia quella desiderata e si voglia evitare di aspettare troppo a lungo prima di tornare alla posizione originale. Il comando non specifica alcun argomento ed è previsto solo per la modalità tecnica di Balor/Gbridge.

3.39 MEXIT

Questo comando richiede l'uscita dal programma gianoMotors senza la terminazione del *server* Couatl. Non specifica alcun argomento nell'`Area dati`.

3.40 COUATLEND

Questo comando determina la fine dell'applicazione *server* Couatl. Non specifica alcun argomento nell'`Area dati`.

3.41 XSTATUS

Il comando XSTATUS richiede lo stato complessivo del sistema di pre-slit. Non specifica alcun argomento nell'`Area dati`.

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

Tabella 41: Esempio di Pacchetto COUATLEND

Campo	Contenuto
header[0]	0xA50F
header[1]	0x1009 (XILL)
header[2]	0x0010 (COMANDO)
header[3]	0x0900 (XSTATUS)
header[4]	0
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	

Tabella 42: Esempio di Pacchetto XSTATUS

3.41.1 Risposta del comando XSTATUS

La payload del pacchetto di risposta al comando XSTATUS descrive lo stato complessivo del sistema di pre-slit in forma sintetica, compreso quello dei sottosistemi della `Power Board` e del sistema di posizionamento della `Thorlabs`.
Il dettaglio del contenuto dell'area dati è illustrato nei due documenti [4] e [5]. Il primo è relativo al funzionamento del sistema in modalità osservativa, l'altro in modalità tecnica.

3.42 SWITCH

La `Power Board` comprende piú device:

- una lampada di calibrazione che può essere accesa/spenta e di intensità variabile
- un dimmer per regolare le diverse intensità della lampada di calibrazione
- una lampada alogena che puó essere accesa/spenta e inserita/disinserita
- due filtri grigi che vengono inseriti/disenseriti
- un sensore di temperatura
- un accelerometro che restituisce il valore lungo i tre assi e, una volta calibrato, l'angolo di posizione del derotatore
- uno shutter da aprire/chiudere
- un relay per accendere/spengere il device `Thorlabs`

I comandi destinati alla PowerBoard sono del tipo:

SWITCH <operazione> <valore>
dove <`operazione`> è un numero con il seguente significato:

- 0 alimentazione lampada di calibrazione
- 1 alimentazione lampada alogena
- 2 posizionamento lampada alogena
- 3 posizionamento filtro 1
- 4 posizionamento filtro 2
- 5 posizionamento shutter
- 6 alimentazione del controller `Thorlabs`
- 7 selezione intensità lampada di calibrazione
- 8 richiesta della temperatura della PWB
- 9 richiesta dei valori dell'accelerometro
- 10 richiesta dello stato della PWB
- 11 reset della CPU della PWB

I comandi [0-6] devono specificare il campo <`valore`> che puó assumere solo due valori:

- **0** (Off/Offline)
- **1** (On/Inline)

Per il comando 7 il valore 0 corrisponde all'intensità della lampada di calibrazione al 10%, mentre 1 al 100%.

Campo	Contenuto
header[0]	0xA50F
header[1]	0x1009 (XILL)
header[2]	0x0010 (COMANDO)
header[3]	0x0910 (SWITCH)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	4 1

Tabella 43: Esempio di Pacchetto SWITCH: inserimento Filtro 2

I comandi [8-11] sono comandi in sola lettura e quindi non sarebbe necessario specificare il campo <valore>. Per mantenere però uniformità nella costruzione e lettura del comando, specifichiamo un valore uguale a 0, che comunque viene ignorato.

3.43 WHEEL

Il comando destinato al device `Thorlabs` è del tipo:

WHEEL <nome posizione>

`nome posizione` è una stringa ASCII corrispondente all'elemento ottico da posizionare. La corrispondenza tra <nome posizione> e la posizione in micro-step è codificata in un file che viene letto dal programma di controllo della pre-slit (vedi [7]).
I nomi delle posizioni sono:

- HOLE
- BLACK
- SCIENCE
- CALIB

Campo	Contenuto
header[0]	0xA50F
header[1]	0x1009 (XILL)
header[2]	0x0010 (COMANDO)
header[3]	0x0912 (WHEEL)
header[4]	lunghezza area dati in bytes
header[5]	0
header[6]	numero pacchetto
header[7]	checksum
`area dati`	CALIB

Tabella 44: Esempio di Pacchetto WHEEL: calibrazione

3.44 WHEEL_STOP

Questo comanda ferma il movimento della giostra `Thorlabs`. È stato implementato nel codice di Balor/Gbridge e di Xill ma non è ancora stato provato.
Non specifica alcun argomento nell'`Area dati`.
È previsto solo per la modalità tecnica di Balor/Gbridge.

3.45 XILLCONF

Questo comando è di recente implementazione e viene inviato da Balor/Gbridge al processo Xill subito prima del comando di movimento WHEEL.
Il comando specifica nell'`Area dati` il contenuto del file `thorlabs.pos` con le posizioni in micro-step dei quattro elementi ottici montati sulla giostra `Thorlabs`.
Per maggiori dettagli rimandiamo al documento [8].

Riassumiamo i valori dei comandi e dei loro parametri in Tab.45.

Comando	Param 1	Param 2	Param 3	Param 4	Param 5
READPARM					
WRITEPARM	num coppie	addr1 val1	addr2 val2	..	..
LOADWAVE	channel	section	file size	filename	file content
GROUP	ngroup				
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	DIT (sec.)	ngroup	coadds	clipping
FREEERUN	DIT (sec.)				
MULTI	path fits file	path multi file			
SOCKDS9	sockname				
REINIT					
STATUS					
ASTATUS					
READLOG					
VERBOSE	level				
MSGLEVEL	msglevel				
DUMMYACQ	path fits file				
KILLTERM					
NOGUISS					
STARTGM					
MSTATUS					
MOVE	motor num	logical position			
MEXIT					
COUATLEND					
XSTATUS					
SWITCH	`Power Board` device number	status			
WHEEL	`Thorlabs` logical position				
WHEEL_STOP					
XILLINFO	file con posizioni micro-step				

Tabella 45: Valori dei comandi e dei loro parametri

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

4 Livello dei messaggi

Ogni messaggio può specificare nel campo `Comando` dell'header un livello di severità. Questo valore assume valori compresi tra 0 e 3. Ciascun livello corrisponde ad un'azione diversa che la GUI-tecnica 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 piú inviati.
- livello 1: da mostrare all'utente.
- livello 2: da mostrare all'utente e da registrare nel file di log
- livello 3: solo da registrare nel file di log

Tutti i messaggi inviati dal *server embedded* sono riportati nel file di log dell'applicazione *middleware* Balor/Gbridge (Nbridge), mentre solo quelli con severità maggiore o uguale a quella impostata dal comando MSGLEVEL sono inviati all'applicazione GUI-tecnica (Nics GUI).
Se MSGLEVEL specifica un valore uguale a 3, l'applicazione GUI-tecnica (Nics GUI) non riceverà alcun messaggio proveniente direttamente dal *server embedded*, ma saranno ricevuti comunque da Balor/Gbridge (Nbridge).

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

Tabella 46: Codice identificativo dei TASK

5 Messaggi di errore

Il sistema di messaggistica degli errori è leggermente cambiato.
La versione attuale fa uso del **campo 4** dell'intestazione per specificare un codice di errore che descrive una serie di informazioni piú dettagliate assemblate 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 Balor/Gbridge come stringa ASCII, terminata dal carattere NULL.
La GUI puó 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 middleware*.

5.1 Codice identificativo dei TASK

Il codice di identificazione dei task che generano messaggi di errore è dato nella tabella 46.

5.2 Codice identificativo degli errori

Il significato dei codici di erore che vengono generati dai vari tasks è dato nelle seguenti tabelle:

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

	Errori Programmazione	
GB_EBADARG	0x320	invalid argument
GB_ENVALBOARD	0x321	invalid board (quadrant) number

	Errori Program. Sequencer	
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

	Errori link ottico	
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

	Errori scheda buffer	
GB_EBUFFVERIF	0x344	buffer board verify error
GB_EBUFFID	0x345	invalid buffer board ID

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

	Errori Lettura del Frame	
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

	Errori nelle FIFO	
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

	Errori Protocollo Dati	
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

	Errori Interni	
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

	Errori Protocollo Comandi	
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	

	Errori I/O Socket	
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 Varianti 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.

Il primo caso è relativo al normale funzionamento e verrà descritto qui di seguito, mentre i casi successivi saranno descritti nel § 6.2.

6.1.1 Caso 1 - è stata avviata un'acquisizione

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: Sottolineiamo 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 delle cause citate ai punti 2, 3 e 4 del $ 6.1.

6.2.1 Caso 2 - l'acquisizione è stata fermata dall'utente

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 apprtenenti all'integrazione interrotta
- esegue il reset delle variabili usate nel controllo del trasferimento delle immagini
- pone lo stato dell'acquisizione a `Idle`

6.2.2 Casi 3 e 4 - l'acquisizione è stata fermata da uno dei due demoni

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 dimensione corretta. 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[4] l'applicazione Nbridge configura un timeout il cui valore è dato da:

  timeout = (tempo di integrazione della singola misura) + 10 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[5] 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

  Questo errore può manifestarsi quando ftest invia sul `socket` dei dati il frame acquisito.

[4] 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.

[5] 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 questo due operazioni è al massimo pari a 3 sec., ma si ottengono tempi morti fino a 5 sec.

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 [6], 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.

- **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.

[6] 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. 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

[4] "Comandi sintetici per la GUI IDL di Giano", E. Giani, C.Baffa, Memo Marzo 2012

[5] "Comandi sintetici di stato per le applicazioni server di Giano", E.Giani, C.Baffa, Memo, Febbraio 2012

[6] "Protocollo per la comunicazione tra Balor/Gbridge e Giano-IDL", E.Giani, C.Baffa, Memo, Marzo 2012

[7] "Xill: il software di controllo del sistema della pre-slit di Giano", E.Giani, C.Baffa, Arcetri Technical Report,8-2011

[8] "I file ausiliari di Giano", E.Giani, C.Baffa, Memo, Novembre 2011

[9] "Interfacciamento con il programma gianoMotors: il server couatl", E.Giani, C.Baffa, Arcetri Technical Report, 6-2010

Indice