Osservatorio Astrofisico di Arcetri
C.N.R. – C.A.I.S.M.I

Nics – La camera-spettrometro del TNG

Il protocollo Transnix-Xnics

C. Baffa

Rapporto Interno di Arcetri N° **2/2000**
Firenze, Luglio 2000

Sommario.

Il presente rapporto interno contiene la descrizione del protocollo di comunicazione (API) con **Transnix** *il programma di controllo della sezione transputer di* **Nics** *(Near Infrared Camera Spectrometer). La versione descritta è la* 1.15 *del 21 Giugno 2000.*

La interfaccia qui descritta è utilizzata dal programma di interfaccia utente **Xnics** *per l'inizializzazione ed il controllo di* **Nics**.

Chapter 1

Introduzione

Il gruppo infrarosso di Arcetri ha sviluppato, per il **TNG**, la camera-spettrometro infrarossa **Nics**. Questo strumento, dotato di rivelatore ternario 1024×1024 ha un'elettronica complessa basata su una rete di transputer interfacciata con un *TRAM Carrier* per bus *PCI* e controllata da un PC Linux, tramite l'apposito programma **Xnics**.

Informazioni generali su **Nics** possono essere trovate in [3], notizie sulla struttura hardware si trovano in [1, 4], sulla struttura software di basso livello in [2, 5], per quella di livello intermedio in [6].

Chapter 2

Inizializzazione della rete transputer

La rete transputer, all'accensione non contiene alcun programma se non un *boot loader* che permette di caricare il programma di inizializzazione.

Per prima cosa il programma **Xnics** deve eseguire un comando `ispy` e confrontare il risultato con il contenuto di un apposito file. Questo per verificare la presenza di tutti i nodi necessari. È opportuno dare al programma che analizza il file la capacità di capire se il risultato ottenuto sia un *super-set* di quello desiderato, in modo da poter aggiungere dei pezzi *sperimentali*, senza che questi ostacolino l'esecuzione dei programmi.

Al presente il risultato del comando ispy è:

```
Using /dev/link0 ispy 3.23
   # Part rate Link# [  Link0  Link1  Link2  Link3 ] by Andy!
   0 T805g-25 1.4M 0 [   HOST    1:1    3:0    ... ]
   1 T16  -20 1.8M 1 [    ...    0:1    ...   C004 ]
   3 T805d-30 1.5M 0 [    0:2    4:1    ...    5:1 ]
   4 T16  -20 1.8M 1 [    ...    3:1    ...    ... ]
   5 T805d-30 1.8M 1 [    ...    3:3    6:1    ... ]
   6 T805d-30 1.8M 1 [    ...    5:2    7:1    ... ]
   7 T2   -20 1.8M 1 [    ...    6:2    ...    8:3 ]
   8 T2   -20 1.4M 3 [    ...    ...    ...    7:3 ]
```

A questo punto il programma **Xnics** fa eseguire il boot del programma **Transnix**. Il protocollo di questo boot consiste semplicemente nella trasmissione byte a byte del file che contiene il programma transputer, solitamente *transnix.btl*.

Le comunicazioni avvengono da ora in poi tramite pacchetti in formato fisso formati da uno header (8 INT16) e un campo dati (1024 INT16). Sono in binario. Sono gli stessi pacchetti che i vari processi si scambiano all'interno della rete di transputer.

Quando un nodo è pronto, manda un pacchetto allo HOST con il comando

message[5] := (INT16 BLN.status) [0x45].

Occorre ora fare il boot del **DSP**. Il formato di tale boot è descritto più avanti. Il nodo **DSP** è pronto quando invia un pacchetto 'status'

Da questo momento in poi **Transnix** manda una serie di pacchetti informativi, warning e segnalazione di errori che devono essere pubblicati.

A questo punto occorre verificare che il sistema, incluso il **DSP**, funzioni. Per questo si esegue un *led toggle* e si legge l'identificativo della scheda del BHN (il sequencer sviluppato dal CWG). Tale valore va pubblicato e deve essere immagazzinato nello header *FITS* assieme ai dati. Potrebbe essere opportuno che ci sia una tabella dei valori accettabili e degli identificativi (Per esempio **Nics**, **Nics**-riserva-8, ecc.)

Ora vanno caricate le forme d'onda. Questa parte converte in binario dei files in formato ASCII con notazione simbolica. Il formato dettagliato è descritto in seguito.

Occorre ora inizializzare i motori, procedura descritta nel seguito.

Vengono ora programmate alcune variabili: prima di tutto una serie di bias (comando setlc) penso da files, perché i valori relativi sono legati al rivelatore, e non cambiano spesso. Potrebbero essere legati allo strumento. Il comando è A543 0040 0047 0004 0020 4710 0000 0002 <indirizzo> <valore> , dove indirizzo e valore sono codificati a 32 bit, da cui message[3]=4.

Dò una tabella che andava bene per il multiplexer.

Indirizzo	Valore	Significato
16	1700	Bias, in millivolt, del quadrante 1
17	1700	Bias, in millivolt, del quadrante 2
18	1700	Bias, in millivolt, del quadrante 3
19	1700	Bias, in millivolt, del quadrante 4
20	3500	Tensione di polarizzazione 1, in millivolt
21	4000	Tensione di polarizzazione 2, in millivolt

Occorre ora settare le dimensioni della parte attiva del rivelatore con i comandi setarea e setbox. Questi valori vengono programmati internamente, in quanto normalmente si esplora tutto un un rivelatore Hawaii. I comandi sono dsp.i.setarea e dsp.i.setbox.

Prima di lanciare la prima integrazione, va mandato un pacchetto che specifica se si vuole un doppio campionamento (default) o un singolo campionamento. I comandi sono BLN.two.samp e BLN.one.samp. Ovviamente quando l'utente cambia il modo di acquisizione tale comando va rieseguito.

A questo punto andrebbero fatte partire le integrazioni idle ed il sistema è pronto.

Chapter 3

Comunicazioni Xnics-Transnix

Descriviamo qui la seconda versione del protocollo: utilizziamo pacchetti trasimesisi attraverso il link.

Le comunicazioni sono sostanzialmente controllate da nicsgate, la maggior parte delle comunicazione (esclusi eventuali allarmi) sono iniziate dallo host.

Le comunicazioni avvengono tramite pacchetti in formato fisso formati da uno header (8 INT16) e un campo dati (1024 INT16). Sono in binario. Sono gli stessi pacchetti che i vari processi si scambiano all'interno della rete di transputer.

Entrambe le parti controllano la lunghezza ed il magic number: se non corrispondono si mandano un NAK (stesso header, ma con campo dati NAK, 0x15). In tale caso va generato un warning.

3.1 Il Protocollo elementare Xnics-Transnix

Quando **Xnics**è attivo il protocollo di sincronizzazione dalla parte PC funziona con una serie di chiamate a ioctrl.

Il protocollo di basso livello si gestisce con il check delle funzioni ioctrl:

LINKWRITABLE
LINKREADABLE

quando la funzione restituisce OK si scrive o si legge. Per dettagli si veda il rapporto in [5].

Se non è possibile scrivere per un tempo prestabilito - ALARM
Se non è possibile leggere per un tempo prestabilito - ALARM
Il protocollo di scambio pacchetti è il seguente:

- Lo host (**Transnix**) manda un pacchetto con il formato descritto di seguito, e lo immagazzina, marcandolo come "ackn_pending"
- Il Base Low Node (BLN) dopo averlo letto, se il formato è accettato, restituisce lo stesso pacchetto sostituendo alla area dei dati o la sigla "OK", oppure un messaggio diagnostico (per ora quasi nulla).

- Se entro un tempo stabilito (TIMEOUT_ACKN_PENDING, direi 2 sec) il pacchetto viene riinviato per un numero fissato di volte (3) poi si ha un allarme.
- altrimenti

  Il pacchetto viene etichettato come "exec_pending" per un tempo fissato (TIMEOUT_EXEC_PENDING, direi 2 sec) altrimenti si va al punto 1, per un numero fissato di volte (3), altrimenti allarme.
- Quando il comando è stato eseguito, viene inviato un secondo pacchetto di 'telemetria', contenete il risultato dell'esecuzione. Ad esempio il nome del file letto o scritto, o il valore di una porta letta, od altro. Se il comando non dà luogo a telemetria, il pacchetto ha un campo dati vuoto. Puo' accadere che, a seconda del comando, questi pacchetti di telemetria siano piu' di uno.

Nel caso di comandi di integrazione, viene restituita un'intera immagine (2048 pacchetti). Questo comando va gestito in modo particolare: ad esempio il time-out deve comprendere il tempo di integrazione.

Transnix in questo schema di protocollo non ha alcuna uscita sull'esterno, per cui quando genera messaggi informativi, warnings od errori manda dei pacchetti allo host che deve provvedere a mostrarli all'utente, secondo le opportune modalità (formato message di **Transnix**).

3.2 Formato dei pacchetti dei comandi.

Il formato dettagliato è

```
  +----------------------------+
0 | magic number (A543)        |
  +----------------------------+       Example
1 | Target transputer          |     A543    magic number
  +----------------------------+     0040    target node (BLM)
2 | Target process             |     0047    target daemon (executor)
  +----------------------------+     0000    0 data
3 | lenght of data or 0        |     0020    Source of paket (host)
  +----------------------------+     4710    Command (exec \dsp command)
4 | Source of packet           |     0000    Param #0
  +----------------------------+     000B    Param #1 (toggle led)
5 | Command                    |     0000     Data
  +----------------------------+     0000     .
6 | Parameter 0                |     0000     .
  +----------------------------+     0000     .
7 | Parameter 1                |     0000
  +----------------------------+     ...
  |                            |     0000     512 items
  | Data ............          |
  |                            |
  +----------------------------+
```

Il formato è ridondante rispetto alle necessità di comunicazione **Transnix-Xnics**, ma permette in questo modo di avere accesso a **tutte** le funzionalità interne senza dover scrivere un complesso codificatore-decodificatore.

Tutti i pacchetti cominciano con una maschera ([0]) che serve per verificare che il contenuto non sia stato sporcato. Per il momento usiamo sempre un valore di 0xA543 .

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa.

3.3 Destinazione dei comandi

I comandi usualmente utilizzati sono pochi e cercheremo di riassumerli, organizzandoli in maniera gerarchica. In ogni caso, l'indirizzo di partenza (source of packet [4]) è:

Nome	Valore	descrizione
HOST.A	0x0020	Host processor

I 'target transputer' (argomento [1]) normalmente usati sono:

Nome	Valore	descrizione
BLN.A	0x0040	Base Low Node
BHN.A	0x0100	Base High Node
DSP.A	0x0110	Dsp sequencer module
SC1.A	0x1000	Serial controller 1
SC2.A	0x2000	Serial controller 2

Gli altri sono:

Nome	Valore	descrizione
ADC1.A	0x0900	ADC first board first module
ADC2.A	0x0A40	ADC first board second module
ADC3.A	0x0C00	ADC second board first module
ADC4.A	0x0E40	ADC second board second module

I processi destinatari possibili (target processes [2]) sono almeno questi:

Nome	Valore	descrizione
BLN.executor	0x0047	Base Low Node executor
BLN.telemetry	0x0048	Base Low Node telemetry daemon
BHN.buff.data	0x0106	Base Hi Node data buffer and sorting
ADC1.datahand	0x0903	ADC1 Node data handler
DSP.booter	0x0113	DSP Node dsb boot
DSP.executor	0x0114	DSP Node executor
SER1.ser.hand	0x1002	SER1 serial handler

3.4 I comandi al BLN executor

Il demone executor è il destinatario naturale per l'esecuzione di comandi sia di livello intermedio che di livello basso. Il demone telemetry contiene il database dei quantità importanti che possono essere consultate dall'esterno. Alcune anzi devono essere riportati nello header fits dei dati.

Normalmente i comandi ([5]) rivolti ai vari demoni hanno come primo byte il valore dell'identificativo del processo target.

I comandi più comuni rivolti al processo executor sono:

Nome	Valore	descrizione
BLN.exec.spry	0x4707	Spry of packets to get errors
BLN.cmd.dsp	0x4710	Execute **DSP** command in [7]
BLN.exec.dummy	0x4784	Get a frame of dummy data
BLN.exec.shut	0x47FF	Execute a general shutdown
BLN.two.samp	0x4786	Double samples integrations
BLN.one.samp	0x4785	Single samples integrations
BLN.offdata	0x4792	Switch off data trasmission
BLN.ondata	0x4793	Switch on data trasmission

Il comando BLN.exec.spry (A543 0040 0047 0000 0020 4707 0000 0000 ...) produce una serie di pacchetti 'errati' rivolti a tutti i nodi, per verificare l'efficienza della rete è usato in fase di test.

La struttura dello header è:

```
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.executor
msgb[3] := 0          -- numero dei parametri
msgb[4] := HOST.A
msgb[5] := BLN.exec.spry
msgb[6] := (INT16 0)
msgb[7] := (INT16 0)
```

I comandi BLN.init.wave e BLN.load.wave non sono più disponibili.

Il comando BLN.exec.dummy (A543 0040 0047 0000 0020 4784 0000 0000 ...) fa produrre al nodo ADC1 un file completo di dati artificiali per la fase di test. Come telemetria rende il nome di un file fittizio (/tmp/temp.fts).

La struttura dello header è:

```
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.executor
msgb[3] := 0          -- numero dei parametri
msgb[4] := HOST.A
msgb[5] := BLN.exec.dummy
msgb[6] := (INT16 0)
msgb[7] := (INT16 0)
```

Il comando BLN.cmd.dsp (A543 0040 0047 00NN 0020 4710 0000 HHH <parametri>...) permette l'esecuzione di uno dei comandi disponibili sul **DSP**, consentendo l'esecuzione di operazioni di livello elementare. Puo' avere alcuni argomenti, in formato INT16. Come telemetria rende quella restituita dal **DSP**. Piu' avanti descriveremo alcuni dei comandi in modo dettagliato.

La struttura dello header è:

```
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.executor
msgb[3] := NN         -- numero dei parametri
msgb[4] := HOST.A
msgb[5] := DSP.cmd.dsp
msgb[6] := (INT16 0)
msgb[7] := HHH        -- comando dsp da eseguire.
```

3.5 Il Boot del DSP

Per eseguire il boot del **DSP** occorre inviare il file contenete il codice. Normalmente tale file è il kernel56.560. Tale file viene letto ed inviato in pacchetti lunghi al massimo 1024 bytes al processo DSP.executor. L'elemento *[3]* dello header contiene il numero dei bytes (len) diviso per due contenuto nel pacchetto. Un valore minore di 512 segnala l'ultimo pacchetto Il formato dello header è:

```
msgb[0] := GENERAL.MW
msgb[1] := DSP.A
msgb[2] := DSP.executor
msgb[3] := (INT16 (len/2)) -- importante!
msgb[4] := HOST.A
msgb[5] := DSP.boot.dsp
msgb[6] := (INT16 0)    -- 0 => seguono altri pacchetti, 1 indica EOF
msgb[7] := (INT16 <packet number>)  -- numero del pacchetto 0...N
```

3.5.1 I comandi relativi al DSP.

Il comando BLN.cmd.dsp (A543 0040 0047 00NN 0020 4710 0000 HHHH <parametri>...) permette l'esecuzione di uno dei comandi disponibili sul **DSP** ed ha numerosi sub-comandi (HHHH in [7]):

Nome	Valore	descrizione
dsp.i.setsa	0	set CCDSQ port A at status <data>
dsp.i.setsb	1	set CCDSQ port B at status <data>
dsp.i.setlc	2	set CCDSQ location <addr> to value <data>
dsp.i.xmdump	5	do memory dump from DSP from X bus <addr>
dsp.i.ymdump	6	do memory dump from DSP from Y bus <addr>
dsp.i.getid	10	get Sequencer ID + SYNC from DSP sequencer
dsp.i.tled	11	toggle sequencer DSP-LED
dsp.i.setca	13	set C11A to value (0 - 255)
dsp.i.setcb	14	set C11B to value (0 - 255)
dsp.i.setarea	19	set detector size (x,y)
dsp.i.setbox	20	set detector active area
dsp.i.go	22	start a CCD readout sequence

La struttura dello header è la seguente:

```
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.executor
msgb[3] := NN          -- numero dei parametri
msgb[4] := HOST.A
msgb[5] := BLN.cmd.dsp
msgb[6] := (INT16 0)
msgb[7] := HHH         -- comando \dsp da eseguire.
```

I comandi dsp.i.setsa e dsp.i.setsb programmano le porte **A** (16 bit) e **B** (12 bit) del **DSP** (A543 0040 0047 0004 0020 4710 0000 00XX <valore> ...). La porta A contiene l'uscita dei clock verso il rivelatore (vedi Comoretto et al., 1996). La porta **B** è spare, per ora. Hanno entrambi un solo parametro (INT32 mappato

su 2 INT16). Restituiscono il valore programmato. Il comando dsp.i.setsa vale 0x0, valore pericoloso, ne abbiamo un alias con T.SET.A = 0x52.

Il comando dsp.i.setlc (setlocation) programma una locazione nello spazio di indirizzamento del **DSP** (A543 0040 0047 0001 0020 4710 0000 0002 <indirizzo> <valore> ...). Ad esempio permette di programmare la Xilinx di decodifica dei clock. Ha due parametri (2 INT32 mappati su 4 INT16). Restituisce l'indirizzo ed il valore programmato.

Il comando dsp.i.getid legge l'identificazione della scheda Controller CCD e il valore del registro di sincronizzazione (non usato) (A543 0040 0047 0000 0020 4710 0000 000A ...). Non ha parametri. Restituisce l'identificativo ed il contenuto del registro (INT16).

I comandi dsp.i.xmdump e dsp.i.ymdump leggono alcune longwords dai due spazi di indirizzo del **DSP**, lo **X** e lo **Y**, (A543 0040 0047 0001 0020 4710 0000 000X <valore> ...). Hanno solo parametro (INT mappato su 2 INT16). Restituiscono alcuni valori fino a riempimento del pacchetto (INT).

Il comando dsp.i.tled commuta lo stato del led sulla scheda Controller CCD. (A543 0040 0047 0000 0020 4710 0000 000B ...). Non ha parametri. Non ha telemetria.

Il comando dsp.i.go fa partire delle integrazioni (A543 0040 0047 0002 0020 4710 0000 0016 <tempo> <iterazioni>...). Ha due parametri (2 INT mappati su 4 INT16). Come telemetria rende un pacchetto per ogni integrazione eseguita.

3.6 Programmazione delle forme d'onda

Le forme d'onda si programmano con un singolo pacchetto. Il formato è un tipo specializzato dei pacchetti di comando al **DSP**. I valori vengono riempiti dal programma **Xnics**. Il vecchio formato dei files wav è obsoleto ed è stato sostituito da un formato piu' leggibile, in cui ci i vari segnali siano specificati in modo simbolico.

Una waveform è definita da un array di valori da impostare in sequenza su di una porta arbitraria del **DSP**. Di solito viene impiegata la porta decodificata A (al primo indirizzo dello spazio di indirizzamento del CCDSEQ). Ogni valore contiene una parte (16 bit meno significativi) che viene scritta sulla porta, e una (8 bit piu' significativi) che pilota la logica di ritardo del controller.

La logica di ritardo è descritta in dettaglio in una serie di rapporti interni ([4],[1], ...). Il valore di ritardo specificato determina quanti wait states vengono inseriti nel successivo accesso allo spazio di indirizzamento del controller, e quindi il tempo minimo che la configurazione di bit resta valida sul bus prima di venir sovrascritta. In particolare, determina il tempo esatto che le configurazioni di una waveform, eccetto l'ultima, restano valide.

Un valore di ritardo di zero implica che non vengono inseriti wait states. La parola resta scritta per 100 ns. Un valore N diverso da zero inserisce 256-N wait states, per un tempo totale di (257-N)*100 ns. Un valore di 255 non è valido, e una durata di 200ns viene ottenuta ripetendo due volte la stessa configurazione senza wait states.

Ogni waveform ha un formato composto da un codice identificativo, l'indirizzo della porta su cui questa viene scritta, nello spazio del bus CCDSEQ, il numero di parole di cui è composta, e la sequenza di valori.

Il software di integrazione identifica le waveform dal codice identificativo. Se due sequenza hanno lo stesso codice, la seconda sovrascrive la prima, ma non rilascia la memoria utilizzata dalla prima. La memoria totale disponibile per le sequenze è di 256 words.

Il nuovo formato delle tabelle di forme d'onda è il seguente:

I file sono in ascii. Il carattere '#' indica che tutto quanto segue è un commento. Le righe vuote vengono ignorate. Il primo valore (numerico) indica il numero di righe valide, cioe' il numero di elementi della forma d'onda. Ogni riga valida comprene due valori: il valore da emettere, lungo 16 bit, descritto come la somma di una serie di valori simbolici elencati piu' avanti, e una durata, con valore compreso tra 0 e 255, con esclusione dell'1.

I segnali in logica negativa vanno indicati come gli altri, e alla fine il programma esegue un XOR con una maschera opportuna (DUMMY), per invertirli.

I valori simbolici sono:

Nome	Valore	descrizione
ZERO	0x0000	tutto a zero
PIXEL	0x0001	un ciclo (0/1/0) avanza di un pixel
XSYNC	0x0002	normalmente a 1, a 0 inizializza lo SR di riga
LINE	0x0004	un ciclo (0/1/0) avanza di una riga
YSYNC	0x0008	normalmente a 1, a 0 inizializza lo SR di colonna
READ	0x0010	a 1 alimenta i buffer di uscita
RST0	0x0100	reset quadrante 1
RST1	0x0200	reset quadrante 2
RST2	0x0400	reset quadrante 3
RST3	0x0800	reset quadrante 4
RSTX	0x0F00	reset quadranti 1-4
SOC0	0x1000	inizio conversione quadrante 1
SOC1	0x2000	inizio conversione quadrante 2
SOC2	0x4000	inizio conversione quadrante 3
SOC3	0x8000	inizio conversione quadrante 4
SOCX	0xF000	inizio conversione quadranti 1-4
DUMMY	0xFF0A	SOCX + XSINC + YSYNC + RSTX

Esempio di una forma d'onda:

```
#
#  TABLES\WF20.WF
#  20          $<$- waveform label
#  7-4-95
#  fast Y shift (400 ns/line)
#
#   61454 , 0
#   61450 , 0
#
20                              # waveform label
#
2                               # number of values
#
LINE + ZERO   , 0               # Cshift of Y (line) register
ZERO          , 0               # dummy, no-op value
```

E questa forma d'onda viene codificata come segue:

```
-- struttura dello header
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.executor
msgb[3] := 50            -- numero dei parametri (fisso)
msgb[4] := HOST.A
msgb[5] := dsp.cmd.dsp
msgb[6] := (INT16 0)
msgb[7] := BLN.load.wave -- comando \dsp da eseguire.
```

L'area dati comprende 50 word, ed è strutturata come segue:

```
msg[8-13] : 00DC  0000  0001  0000  0003  0000            -- Fisso
msg[14-15]:                                      0004  0000 -- Numero di
                                                           -- valori + 2
msg[16-17]: 0014  0000                                     -- Label della
                                                           -- forma
msg[18-19]:             0000  0000                         -- fisso
msg[20-87]:                         FF0E  0000  FF0A  0000 -- dati
            0000  0000  0000  0000  0000  0000  0000  <...>
```

I dati sono composti da due INT16: il primo indica la maschera da scrivere in uscita, il secondo la durata-1 in unità di 100ns.

3.7 I comandi di start integrazione

Il comando DSP.i.go fa partire delle integrazioni (A543 0040 0047 0002 0020 4710 0000 0016 <tempo> <numero itegrazioni> <Num_reset> <read_dummy> <campionamenti>...). Ha **cinque** parametri (INT mappati su 10 INT16). Come telemetria rende un pacchetto per ogni integrazione eseguita. Per ora funziona solo la versione che accetta 1 come numero di integrazioni (con **Xnics**).

Il significato dei parametri è:

0: Tempo di integrazione (in millisecondi, **escluso** il tempo di lettura elementare, che attualmente è di 928ms)

1: Numero di integrazioni complete

2: Numero di reset (il default è 16 o quanto precedentemente specificato, zero significa usa il default).

3: Numero di pre-letture dummy (valore logico, !=0 significa 1)

4: Numero di campionamenti per frame, ciascuno spaziato il tempo di integrazione elementare (default 2)

Integrazioni in multicampionamento: una volta abilitato, è ora possibile fare integrazioni in multicampionamento. A questo fine occorre programmare il tempo di integrazione con il valore dell'intervallo tra i vari campionamenti, programmare il numero delle integrazioni (parametro 1) a 1, e porre il numero totale di letture nel parametro 4 (deve essere maggiore di 2). Ogni lettura viene inviata separatamente, ed, al presente finisce in un file fits separato.

Prima di lanciare la prima integrazione, va mandato un pacchetto che specifica se si vuole un doppio campionamento (default) o un singolo campionamento. I comandi sono BLN.two.samp (0x4786) e BLN.one.samp (0x4785) e quindi con pacchetti del tipo A543 0040 0047 0000 0020 4785 0000 0000 ... Ovviamente quando l'utente cambia il modo di acquisizione tale comando va rieseguito.

Per abilitare o disabilitare il multicampionamento, occorre impostare le integrazioni con il modo *singolo campionamento* (BLN.one.samp) e poi abilitare il modo *doppio campionamento* (BLN.multisamp, 0x4787). Per ripristinare il doppio campionamento è sufficiente un pacchetto di *doppio campionamento*.

Prima di **ogni** integrazione, va mandato un pacchetto che specifica se si vogliono avere i dati (default) o le integrazioni sono idle. I comandi sono BLN.offdata (0x4792) e BLN.ondata (0x4793) e quindi con pacchetti del tipo A543 0040 0047 0000 0020 4792 0000 0000 ... Alla fine delle integrazioni, è bene eseguire un comando BLN.offdata.

3.8 Comunicazioni con il processo di telemetria

3.8.1 I comandi al processo di telemetria

Il demone telemetry contiene il database dei quantità importanti che possono essere consultate dall'esterno. Alcune anzi devono essere riportati nello header fits dei dati. Tale demone puo' essere interrogato tramite il comando *BLN.tlm.get* (*0x4860*):

```
-- struttura dello header
msgb[0] := GENERAL.MW
msgb[1] := BLN.A
msgb[2] := BLN.telemetry
msgb[3] := (INT16 0)  -- numero dei parametri
msgb[4] := HOST.A
msgb[5] := BLN.tlm.get
msgb[6] := (INT16 0)
msgb[7] := HHHH       -- telemetria richiesta
```

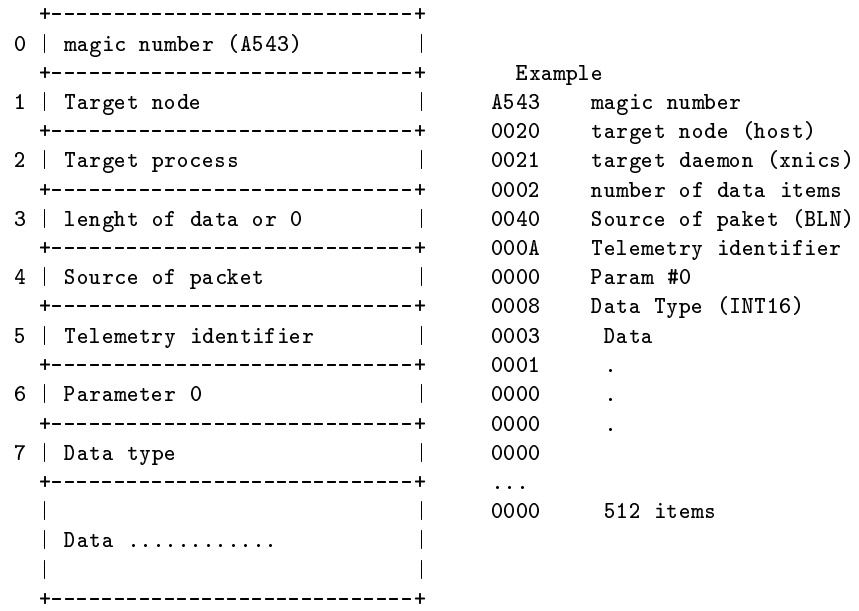
Il comando BLN.tlm.get (A543 0040 0048 0000 0020 4860 0000 HHHH ...) richiede il valore della grandezza codificata in [7] secondo la seguente tabella:

Nome	Valore	descrizione
T.LASTCOMM	0x5050	last host command received
T.LAST.TEL	0x5053	last telemetry send
T.B.HOSTID	DSP.i.getid	CCD Sequencer Host ID
T.FILE.STAT	BLN.file.stat	status of out file (1=open)
T.FILE.NAME	0x5051	last data file written
T.W.TABLE	BLN.load.wave	last wavetable loaded
T.XMDUMP	DSP.i.xmdump	Value of **DSP**X BUS (ADDR VALUE)
T.YMDUMP	DSP.i.ymdump	Value of **DSP**Y BUS (ADDR VALUE)
T.SET.A	DSP.i.setsa	CCDSQ port A last value
T.SET.B	DSP.i.setsb	CCDSQ port B last value
T.SETAREA	DSP.i.setarea	set detector size (x,y)
T.SETBOX	DSP.i.setbox	set detector active area
T.SET.TLC	DSP.i.setlc	CCDSEQ location set, addr and value
T.DOSCAN	DSP.i.doscan	do CCDSQ test sequence ## times
T.SHOWT	DSP.i.showt	shows conditioning values
T.TLED	DSP.i.tled	toggle led status
T.SER1.A	SER1.output.a	output on first serial on SER1
T.SER1.BAL	SER1.pres.bal	Read of the pressure nn Balzers
T.SER2.A	SER2.output.b	output on first serial on SER2
T.SER2.208	SER2.temp.208	Read of a temperature on LS208

Notiamo che, quando possibile, l'identificativo coincide con il comando, o il sub-comando che lo genera. In particolare il valore della porta A puo' essere richiesto sia con il valore T.SET.A, sia con il piu' pericoloso valore DSP.i.setsb (0x0 **sconsigliato**). Il formato dei pacchetti è descritto piu' avanti.

3.8.2 Formato dei pacchetti di telemetria

Il formato dettagliato dei pocchetti che restituiscono il valore della telemetria richiesta è



Il formato è unico per la comunicazione **Transnix-Xnics**, e quindi ammette modifiche in modo relativamente indolore. I dati vengono codificati nella zona data. In particolare le stringhe sono in ascii, un caratter per ogni INT16, mentre gli INT sono codificati in due successivi INT16. Il programma di interfaccia si preoccupa di riggiustare lo swap dei byte. La zona dati è breve per le sringhe, vorrei commenti sulla necessità di estenderli.

Il campo Data_type puo' avere i seguenti valori:

Nome	Valore	descrizione
T.TY.bool	4	telemetry item is boolean
T.TY.ascii	6	telemetry item is ascii
T.TY.int16	8	telemetry item is short (int16)
T.TY.int32	10	telemetry item is int (int32)
T.TY.real	14	telemetry item is real (real32)

Il significato dei vari valori è descritto di seguito:

T.LASTCOMM Due int16, contiene l'ultimo comando/sottocomando ricevuto.

T.LAST.TEL Un int16, Contiene l'indice dell'ultima telemetria inviata.

T.SETAREA Tre int, contiene le dimensioni del quadrante e il numero di quadranti

T.SETBOX Quattro int, contiene la dimensione (X e Y) e lo offset. (X e Y) dell'area attiva

T.FILE.NAME Inattivo, ascii, ultimo file scritto

T.W.TABLE Inattivo, ascii, ultimo wave file letto

T.SET.A Un int16, ultimo valore scritto sulla porta A

T.SET.B Un int16, ultimo valore scritto sulla porta B

T.XMDUMP Molti INT32, ultimo dump richiesto del bus X del **DSP**

T.YMDUMP Molti INT32, ultimo dump richiesto del bus X del **DSP**

T.SET.TLC Due int, Indirizzo e valore dell'ultima scrittura sul bus CCD del **DSP**

T.DOSCAN <dummy>, per segnalare la fine di uno scan

T.SHOWT Mostra i valori di condizionamento delle linee

T.FILE.STAT Inattivo. Stato del file dati.

T.B.HOSTID Cinque Int, Numero della scheda CCD e stato della sincronizzazione. ID scheda clock. ID primo ADC (bit 0:3) + stato (bit 4:7). ID + stato secondo ADC.

T.TLED <dummy>, per segnalare il toggle del led.

3.9 I messaggi inviati da Transnix.

3.9.1 Il formato message di Transnix.

Transnix non ha alcuna uscita sull'esterno, per cui quando genera messaggi informativi, warnings od errori manda dei pacchetti allo host che deve provvedere a mostrarli all'utente, secondo le opportune modalità.

```
-- struttura dello header
msgb[0] := GENERAL.MW
msgb[1] := HOST.A
msgb[2] := HOST.trans
msgb[3] := (INT16 NN)  -- lunghezza (in bytes del messaggio)
msgb[4] := BLN.A
msgb[5] := HHHH        -- comando
msgb[6] := (INT16 0)
msgb[7] := (INT16 0)
```

Il campo comandi può contenere:

Nome	Valore	descrizione
BLN.msg.sio	0x4344	Output di un messaggio informativo
BLN.err.sio	0x4341	output di un errore
BLN.war.sio	0x4341	output di un warning

Segue il messaggio, codificato in ASCII e immagazzinato come un carattere per ogni INT16 del campo dati. La lunghezza è contenuta in msgb[3].

Il formato di output deve essere:

(ERR—WAR—MSG) : <nodo origine> : <messaggio>

Il nodo di origine si trova in msgb[4], e vale:

Nome	Valore	descrizione
BLN.A	0x0040	Base Low Node
BHN.A	0x0100	Base High Node
DSP.A	0x0110	Dsp sequencer module
SC1.A	0x1000	Serial controller 1
SC2.A	0x2000	Serial controller 2
ADC1.A	0x0900	ADC first board first module
ADC2.A	0x0A40	ADC first board second module
ADC3.A	0x0C00	ADC second board first module
ADC4.A	0x0E40	ADC second board second module

3.9.2 Formato dei dati.

Il formato dettagliato è

```
  +-----------------------------+
0 | magic number (A543)         |
  +-----------------------------+       Example
1 | Target node                 |     A543    magic number
  +-----------------------------+     0020    target node (host)
2 | Target process              |     0021    target daemon (Host.\xnics)
  +-----------------------------+     0400    number of data items (1024)
3 | lenght of data or 0         |     0100    Source of paket (BHN)
  +-----------------------------+     4601    BLN.data.send
4 | Source of packet            |     0000    0
  +-----------------------------+     0NNN    row number
5 | Telemetry identifier        |     XXXX     Data
  +-----------------------------+     XXXX     .
6 | Parameter 0                 |     XXXX     .
  +-----------------------------+     XXXX     .
7 | Data type                   |     XXXX
  +-----------------------------+     ...
  |                             |     XXXX     512 items
  | Data ............           |
  |                             |
  +-----------------------------+
```

Il formato è simile agli altri pacchetti per la comunicazione **Transnix-Xnics**. I dati vengono codificati nella zona data come INT16. La numerazione delle righe va da 0 a 2048, da sinistra a destra e dall'alto in basso. **Transnix** si preoccupa della gestione dei quadranti. Salvo errori di trasmissione i pacchetti dovrebbero essere in ordine crescente, ma io farei o un check o userei l'informazione codificata nello header per ordinarli.

3.10 Il controllo dei vari sottosistemi

3.10.1 Controllo dei motori

I motori vengono controllati tramite una linea seriale. Il sistema dei transputer si occupa solo del trasporto dei messaggi da e per i controllers dei motori e non fa alcun check. I comandi non vanno terminati con un CR od un LF: se ne occupa il driver della seriale. Ogni output dei controllers viene impacchettato

come telemetria e spedito indietro allo host. I controller sono collegati in cascata tramite la porta seriale. il formato generale dei comandi è:

[controller]XXX[argomento]

Se il valore [controller] è presente, il comando va ad uno specifico controller. Altrimenti il comando è **eseguito da tutti** i controllers. La presenza del valore [argomento] dipende dal comando. Per molti comandi, se l'argomento non è presente ed il comando è indirizzato ad un controller specifico, vale come richiesta del valore corrente. Se il comando è una richiesta di informazioni, è accettato, ma non eseguito se non è diretto ad un controller specifico.

Se il comando non viene riconosciuto viene ignorato, non c'e' diagnostica, tuttavia, se il comando viene eseguito ne viene fatta un eco, altrimenti no. Come già detto, l'ouput deicontrollers viene considerato telemetria ed instradato verso la tabella della telemetria (in **Transtst**) e **Xnics**(in **Transnix**).

Il formato dettagliato è:

```
  +----------------------------+
0 | magic number (A543)        |
  +----------------------------+       Example
1 | Target transputer          |     A543    magic number
  +----------------------------+     1000    target node (SC1.A)
2 | Target process             |     0047    target daemon (ser.hand)
  +----------------------------+     0002    2 data
3 | lenght of data or 0        |     0020    Source of paket (host)
  +----------------------------+     1004    Command (serial 1 out)
4 | Source of packet           |     0000    Param #0
  +----------------------------+     0000    Param #1
5 | Command                    |     0031     Data  ("1A")
  +----------------------------+     0041     .
6 | Parameter 0                |     0000     .
  +----------------------------+     0000     .
7 | Parameter 1                |     0000
  +----------------------------+     ...
  |                            |     0000     512 items
  | Data ............          |
  |                            |
  +----------------------------+
```

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa.

Alcuni comandi di uso comune:

Nome	Descrizione
[nodo]A[valore]	fissa l'accelerazione in giri/sec.
[nodo]V[valore]	fissa la velocità in giri/sec.
[nodo]H[valore]	fissa la direzione di movimento (+/-)
[nodo]D[valore]	fissa il numero di passi da eseguire.
[nodo]G	esegue un movimento.
[nodo]GH[valore]	esegue un reset della ruota in una direzione (+/-).
[nodo]OSC[valore]	fissa lo stato attivo dello switch di home (0/1).
[nodo]RA	Riporta lo stato dei limit switch.
[nodo]RC	Riporta il successo dellla procedura di reset.
[nodo]MN	obbliga i movimenti ad essere assoluti.

3.10.2 Controllo e lettura delle temperature

I due diversi dispositivi per le temperature sono controllati tramite linea seriale. Il sistema dei transputer si occupa solo del trasporto dei messaggi da e per i dispositivi e non fa alcun check. I comandi NON VANNO terminati con CR o LF: se ne occupa il driver della seriale. Ogni output viene impacchettato come telemetria e spedito indietro allo host. I controller sono collegati sulle due diverse porte il lettore di temperatura, Lake Shore 208, è sulla porta A, mentre il controllore a due canali Lake Shore 330 è sulla porta B.

3.10.3 Lake Shore 208

Il 208 è un puro lettore di temperatura, permette di leggere fino a 8 canali, ma ne sono collegati solo 3. Ha un insieme di comandi limitato e permette la scansione automatizzata dei canali, che noi, per ora non usiamo. Per leggere il canale X basta mandare la stringa:

YCX<CR><LF>WS

In realtà questi sono due comandi, separati dal carriag return e dal line feed, ma il primo NON DA' ECO, quindi, per uniformità di handshake con xnir, conviene usare il comando composto o meglio ancora il comando SER2.temp.208. In entrambi i casi la risposta è del tipo:

[eventuale errore]<canale>[segno]<lettura><unita'(C—K—F)><status><cr><LF>

La risposta viene inviata indietro come telemetria. Il comando per scrivere sulla porta è:

SER2.output.a	0x2004	Data to be output on serial A

Ho costruito un comando per leggere un particolare canale, ed è il comando da usarsi nelle operazioni ordinarie:

SER2.temp.208	0x2008	Read a temperature on LS208

Questo comando è equivalente a quello indicato sopra, se si pone il canale da leggere nella parola 7 (Parametro 1).

Il formato dettagliato è:

```
  +-----------------------------+
0 | magic number (A543)         |
  +-----------------------------+       Example
1 | Target transputer           |     A543    magic number
  +-----------------------------+     2000    target node (SC2.A)
2 | Target process              |     2002    target daemon (ser2.hand)
  +-----------------------------+     0000    0 data
3 | lenght of data or 0         |     0020    Source of paket (host)
  +-----------------------------+     2008    Command (Temp. reading)
4 | Source of packet            |     0000    Param #0
  +-----------------------------+     0000    Param #1
5 | Command                     |     0000     Data
  +-----------------------------+     0000     .
6 | Parameter 0                 |     0000     .
  +-----------------------------+     0000     .
7 | Channel to read             |     0000
  +-----------------------------+     ...
  |                             |     0000     512 items
  | Data ............           |
  |                             |
  +-----------------------------+
```

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa.

Alcuni comandi di uso comune:

YAN	selezione del canale N
WS	lettura del canale selezionato.
F0<K—C—F>	selezione dell'unità di misura (Kelvin, ecc.)
WY	Scan Status, per avere una risposta.
YND	seleziona il tempo di lettura (D) del canale N.

3.10.4 Lake Shore 330

Il 330 è uno strumento per il controllo attivo delle temperature, oltre che per la lettura delle stesse. Ha un canale di controllo ed un totale di due canali di lettura.

Come per il 208 molti comandi non danno echo, per cui conviene SEMPRE montare insieme, questa volta con un carattere ";" comandi di set e di interrogazione, per conservare lo schema di doppio handshake. Daro' alcuni esempi.

La risposta viene inviata indietro come telemetria. Il comando per scrivere sulla porta è:

Nome	Valore	descrizione
SER2.output.b	0x2005	Data to be output on serial B

Il formato dettagliato è:

```
  +-----------------------------+
0 | magic number (A543)         |
  +-----------------------------+       Example
1 | Target transputer           |     A543    magic number
  +-----------------------------+     2000    target node (SC2.A)
2 | Target process              |     2002    target daemon (ser2.hand)
  +-----------------------------+     0000    2 data
3 | lenght of data or 0         |     0020    Source of paket (host)
  +-----------------------------+     2005    Command (Out on ser B)
4 | Source of packet            |     0000    Param #0
  +-----------------------------+     0000    Param #1
5 | Command                     |     0041     Data "AB"
  +-----------------------------+     0042     .
6 | Parameter 0                 |     0000     .
  +-----------------------------+     0000     .
7 | Channel to read             |     0000
  +-----------------------------+     ...
  |                             |     0000     512 items
  | Data ............           |
  |                             |
  +-----------------------------+
```

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa.

Alcuni comandi di uso comune:

Nome	Descrizione
CUNI<K—C—F>;CUNI?	selezione dell'unità di misura (Kelvin, ecc), per il canale di lettura
SUNI<K—C—F>;SUNI?	selezione dell'unità di misura (Kelvin, ecc), per il canale di controllo
SCHN<A—B>;SCHN?	seleziona come lettura uno dei due canali (A o B).
CCHN<A—B>;CCHN?	seleziona come controllo uno dei due canali (A o B).
SDAT?	riporta il valore del canale di lettura
CDAT?	riporta il valore del canale di controllo
RANG<0-3>;RANGE?	programma la potenza del riscaldatore, da 0=off a 3=high
HEAT?	legge la percentuale applicata della potenza massima.
SETP<valore>;SETP?	programma la temperatura da mantenere nel canale di controllo.

All'inizializzazione, io darei i seguenti comandi:

CUNIK;CUNI?	(letture in kelvin)
SUNIK;SUNI?	(programmazione in kelvin)
RANG0;RANGE?	Spegne il riscaldatore
SCHNB;SCHN?	Il canale B è di sola lettura (e' inattivo)
CCHNA;CCHN?	Il canale A controlla la temperatura
SETP77;SETP?	pone l'obiettivo in temperatura a 77K

3.10.5 Lettura della pressione

La misura della pressione interna di Nics avvienet tramite due sensori collegati al TPG252 della Balzers. I due dispositivi coporono intervalli diversi di pressione e non sono quasi mai conteporaneamente utili. Il primo sensore legge pressioni tra 10^3 e 5.10^-3 mbar, mentre il secondo non puo' lavorare al di sopra di 10^-2 mbar. Noi leggiamo sempre i valori di entrambi, e si puo' pubblicare solo quello significativo. Nel range di sovrapposizione le letture possono differire fino al 10-20%.

Il comando per leggere i due valori è: iPRX. La risposta è dela forma:

<codice>,X.XXXEsX,<codice>,Y.YYYEsY

Il valore di <codice> indica lo stato del sensore:

Valore	Significato
0	successo
1	underflow
2	overflow
3	errore nel sensore
4	sensore spento
5	sensore assente
6	errore interno

Per leggere i valori, basta mandare un pacchetto con 0 dati ad SC1 con il comando:

Nome	Valore	descrizione
SER1.pres.bal	0x1008	Read the pressure from Balzers

Il formato dettagliato è

```
  +-----------------------------+
0 | magic number (A543)         |
  +-----------------------------+       Example
1 | Target transputer           |     A543    magic number
  +-----------------------------+     1000    target node (SC1.A)
2 | Target process              |     1002    target daemon (ser1.hand)
  +-----------------------------+     0000    0 data
3 | lenght of data or 0         |     0020    Source of paket (host)
  +-----------------------------+     1008    Command (Press reading)
4 | Source of packet            |     0000    Param #0
  +-----------------------------+     0000    Param #1
5 | Command                     |     0000     Data
  +-----------------------------+     0000     .
6 | Parameter 0                 |     0000     .
  +-----------------------------+     0000     .
7 | Channel to read             |     0000
  +-----------------------------+     ...
  |                             |     0000     512 items
  | Data ............           |
  |                             |
  +-----------------------------+
```

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa. Se [3] vale zero, viene eseguita una lettura con il comando PCX, altrimenti il comando presente nella zona dati viene inviato (senza alcun controllo o modifica) al Balzers.

Alcuni comandi utili:

Nome	Descrizione
SEN	stato del sensore
TID	identificazione dei sensori.
ERR	stato di errore.

3.10.6 Accesso al display LCD

Il controller LCD viene programmato tramite una linea seriale. Il sistema dei transputer si occupa del trasporto dei messaggi verso il sistema LCD (non vi è mai eco) e posiziona lo output in base alla provenienza. Eventuali messaggi provenienti dallo host finiscono sulla prima riga, sovrapposti a quelli del BLM. Lo spazio è di 16 caratteri. La seconda riga riporta messaggi provenienti dai nodi BHM e **DSP**, mentre la terza e la quarta riga riportano messaggi provenienti dai due controller ADC.

Il formato dettagliato è:

```
  +-----------------------------+
0 | magic number (A543)         |
  +-----------------------------+       Example
1 | Target transputer           |     A543    magic number
  +-----------------------------+     1000    target node (SC1.A)
2 | Target process              |     1002    target daemon (ser1.hand)
  +-----------------------------+     0002    2 data items
3 | lenght of data or 0         |     0020    Source of paket (host)
  +-----------------------------+     1005    Command (serial 2 out)
4 | Source of packet            |     0000    Param #0
  +-----------------------------+     0000    Param #1
5 | Command                     |     0041     Data  ("AA")
  +-----------------------------+     0041     .
6 | Parameter 0                 |     0000     .
  +-----------------------------+     0000     .
7 | Parameter 1                 |     0000
  +-----------------------------+     ...
  |                             |     0000     512 items
  | Data ............           |
  |                             |
  +-----------------------------+
```

La lunghezza del campo dati validi è indicata in [3]. La lunghezza totale del pacchetto è fissa.

Contents

Bibliography

[1] Comoretto, G., Baffa, C., Gavrioussev, V., Lisi, F., Sozzi, M., 1999, "The Data Acquisition System for Nics - Hardware Solutions", International Meeting on Astronomical Techologies, S.Agata, Memorie della Società Astronomica Italiana, In press.

[2] Baffa, C., Comoretto, G., Gavrioussev, V., Giani, E., Lisi, F., 1999, "The Data Acquisition System for Nics - Software Solutions", International Meeting on Astronomical Techologies, S.Agata, Memorie della Società Astronomica Italiana, In press.

[3] Lisi, F., Baffa, C., Gennari, S., Oliva, E., 1999, "**Nics**, the near IR imager/spectrograph of the TNG", International Meeting on Astronomical Techologies, S.Agata, Memorie della Società Astronomica Italiana, In press.

[4] Comoretto, G., Baffa, C., 1998, "Nics - Programma di controllo per il DSP sequencer", Rapporto interno dell'Osservatorio Astrofisico di Arcetri, N°4/1998.

[5] Giani, E., Baffa, C., Checcucci, A., 1999, "Driver Linux per carrier TRAM", Rapporto interno dell'Osservatorio Astrofisico di Arcetri, N°4/1999.

[6] Baffa, C., Comoretto, G., "Il software TransNix per la Camera Infrarosso Nics", Rapporto interno dell'Osservatorio Astrofisico di Arcetri, N°3/1996.