

# INTERNAL REPORT

## **Integrazione del Digital Base Band Converter nel software di controllo del Sardinia Radio Telescope**

A. Melis, G. Comoretto

Report N. 17, released: 29/12/2011

Reviewer: C. Migoni



Osservatorio  
Astronomico  
di Cagliari

## **Abstract**

*Il Digital Base Band Converter (DBBC) è la piattaforma hardware scelta dalla comunità EVN come backend standard per le osservazioni VLBI.*

*Le FPGA di cui dispone permettono di riconfigurare la funzionalità del dispositivo per sfruttarne le potenzialità anche in modalità Single Dish, rendendolo a tutti gli effetti un backend per il continuo e per spettroscopia utilizzabile per osservazioni astronomiche o per monitoraggio RFI.*

*In questo documento descriviamo l'integrazione del DBBC in NURAGHE, il software di controllo del Sardinia Radio Telescope basato sul framework Alma Common Software (ACS).*

# 1 Introduzione

Il Digital Base Band Converter (DBBC) è una piattaforma hardware, sviluppata all'IRA di Noto, che sta pian piano sostituendo i vecchi sistemi analogici dei radiotelescopi presenti nella rete EVN (European VLBI Network) con una tecnologia digitale basata su logiche programmabili.

Lo scopo principale del DBBC è quello di fungere da terminale per le osservazioni VLBI per astronomia e geodesia, ma poichè questo tipo di utilizzo avviene mediamente tre/quattro mesi all'anno, si è sfruttata la possibilità di riconfigurare la piattaforma grazie alle FPGA di cui dispone. Questa possibilità, unita alle potenzialità del DBBC in termini di banda istantanea, hanno portato allo sviluppo di differenti applicazioni che richiedono una completa integrazione nel software di controllo del radiotelescopio nel quale viene utilizzato.

In particolare, è stata creata una configurazione che permette di fare un monitoraggio costante di tutta la banda del ricevitore (per ora solo il ricevitore L-P [5], successivamente anche per il banda C e per il ricevitore multi-feed a 22 GHz) in uso in quel momento, qualunque sia la sorgente o il tipo di osservazione in atto (continuo, spettroscopia, polarimetria, pulsar ecc.); questo ha condotto alla realizzazione di un component più semplificato ma che richiede le medesime procedure seppur meno articolate.

Il Sardinia Radio Telescope ha un software di controllo basato sul framework Alma Common Software (ACS) e in questo documento descriviamo l'integrazione ACS del DBBC per il controllo sia in modalità VLBI che in modalità single dish.

Anche il radiotelescopio di Medicina disporrà presto di un DBBC nella stessa configurazione, quindi il lavoro svolto verrà sfruttato completamente anche nel sistema di controllo dell'antenna VLBI di Medicina.

## 2 Descrizione del framework ACS

L'Atacama Large Millimeter Array (ALMA) è un progetto che coinvolge diversi istituti europei e nord americani. Si tratta di un interferometro costituito da almeno 64 antenne da 12 metri di diametro operanti nel range del millimetrico e del sub-millimetrico, con baselines fino a 14 Km. Vista la complessità del progetto, è stato sviluppato un framework orientato agli oggetti chiamato Alma Common Software (ACS) per la creazione di un sistema distribuito per la gestione delle antenne.

Nonostante l'assoluta diversità rispetto ad SRT, la versatilità di ACS ci permette di utilizzare tale framework per l'implementazione di NURAGHE. In figura 1 vediamo un diagramma a blocchi generale del sistema:

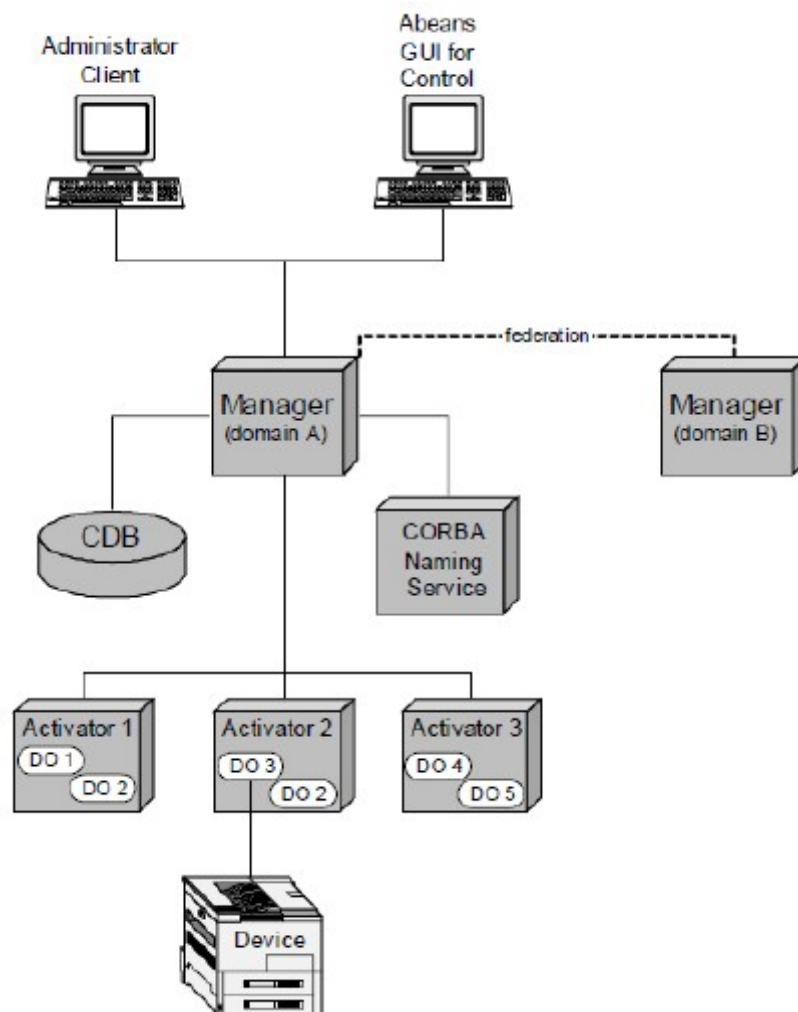


Figura 1: Schema a blocchi del framework ACS

A livello più alto c'è il Manager che dispone di un database centralizzato (CDB) nel quale sono registrati tutti i container e i component presenti ai livelli inferiori. I file di configurazione dei container e dei component elencano le caratteristiche e le proprietà che li identificano e sono scritti in linguaggio XML.

I container (indicati con Activator in figura) rappresentano semplicemente dei contenitori di component e rappresentano il mezzo attraverso il quale i component vengono distribuiti nel sistema; i component o Distributed Objects (DO) sono gli elementi finali, sono cioè gli oggetti che si interfacciano con l'hardware e con i dispositivi che si vogliono controllare: sono quindi le entità che effettivamente interagiscono con gli oggetti fisici.

Quello che viene realizzato è quindi un sistema distribuito, cioè un sistema nel quale ogni component possa girare anche su più macchine e sistemi operativi: questa interoperabilità è garantita da ACS che utilizza il middleware CORBA.

Ogni component espone la sua interfaccia nel sistema utilizzando l'Interface Definition

Language (IDL); questa viene mappata in uno dei tre linguaggi supportati da ACS (C++, Python e JAVA).

### 3 Breve descrizione del Digital Base Band Converter

Il Digital Base Band Converter [1] è la piattaforma che la comunità VLBI europea ha scelto come backend di nuova generazione per sostituire gli obsoleti terminali analogici.

E' un sistema completo, con un PC di controllo montato al proprio interno per la configurazione del backend e per il suo controllo remoto.

Dispone di quattro schede di conversione analogico-digitali con una banda istantanea fino a 1 Ghz ciascuna e di schede di elaborazione basate su FPGA Xilinx Virtex 5.

Il sistema dispone inoltre di quattro moduli di condizionamento per controllare il livello dei segnali e per il loro prefiltraggio prima della conversione in formato digitale.

E' inoltre presente una scheda di conversione da digitale ad analogico, un sintetizzatore che a partire dal riferimento esterno di 10 Mhz fornisce il clock di campionamento per i campionatori e il clock per le schede di elaborazione, e infine una scheda con un'interfaccia 10 Gbit per l'invio di dati ad alta velocità.

### 4 Descrizione del Component DBBC

In questo capitolo descriveremo con maggior dettaglio il component che gestirà il DBBC in NURAGHE che è stato denominato semplicemente *DBBC*.

#### 4.1 *Interfaccia IDL*

Il component DBBC eredita tutti i metodi necessari per la gestione di un backend dall'interfaccia *GenericBackend* che è stata implementata in modo tale da bypassare la dipendenza dell'hardware specifico di un certo backend rispetto ad un altro.

L'interfaccia IDL specifica del DBBC contiene la dichiarazione dei metodi generici previsti, che in questo caso sono fondamentalmente due: *LoadScan* e *LoadVLBI*. Il primo configura il sistema in modalità "spettrometro a scansione" [2], mentre il secondo lo configura in modalità VLBI; sono in fase di sviluppo nuove configurazioni per il DBBC, e ognuna di esse avrà il corrispondente metodo da aggiungere nel file *DBBC.idl*.

Il file IDL deve essere compilato da un compilatore IDL richiamato dall'*acsMakefile*. I file che si ottengono sono un mapping in C++, e nel nostro caso sono due: *DBBCS* e *DBBCC*, dove S sta per stubs e C sta per component. Questi due file dovranno essere inclusi rispettivamente

nell'header del file di implementazione del DBBC il primo, e nei file header di eventuali component che dovessero includere il DBBC il secondo.

## **4.2 Implementazione C++ del component**

L'implementazione del component, scritta in linguaggio C++, è composta dal file *DBBCImpl*, nelle estensioni .h e .cpp.

In *DBBCImpl.h* vengono incluse tutte le librerie di ACS necessarie e vengono dichiarati tutti i metodi da implementare, sia quelli scritti da noi ex novo (*LoadScan* e *LoadVLBI*) sia quelli ereditati dal *GenericBackend*.

Oltre a tutte le properties ereditate dal *GenericBackend*, in questo file dovremo includere tutti gli header dei file di comunicazione con l'hardware e la dichiarazione dei puntatori agli oggetti che vengono istanziati nel file .cpp, questo dopo aver constatato che il compilatore non accetta che vengano dichiarati in quest'ultimo; in particolare è stato definito un puntatore all'oggetto *DBBCIntfClient* che funge da client per la comunicazione con la server socket, e per la modalità spettrometro a scansione che vedremo più avanti viene definito un puntatore alla classe *ScanSpectrometer* che contiene tutti i metodi per la programmazione dei registri dell'FPGA.

In *DBBCImpl.cpp* vanno inclusi, oltre a *DBBCImpl.h*, gli headers delle librerie di sistema utilizzate.

Va inoltre inclusa la classe *SimpleParser* che implementa il sistema di parsing dei comandi NURAGHE, la macro *IRA\_LOGFILTER\_DECLARE* che serve a filtrare i messaggi di log, la macro *\_PROPERTY\_REFERENCE\_CPP* che serve per creare le properties ereditate dal *GenericBackend*, e una macro per aggiungere le funzioni di supporto MACI DLL.

Nei paragrafi seguenti descriviamo tutti i metodi implementati nel component.

### **4.2.1 Costruttore e Distruttore**

Il costruttore inizializza gli smartpointer definiti nel file header.

Il corpo del distruttore non viene solitamente implementato in ACS.

### **4.2.2 Metodo initialize**

Con questo metodo vengono inizializzati i parametri utilizzati per la comunicazione con

l'hardware.

Il DBBC si interfaccia con il mondo esterno mediante una socket che viene creata in un programma server che gira sul PC di controllo; il funzionamento è descritto in [3] anche se sono state integrate altre funzionalità. Il server è sempre attivo e rimane in attesa che qualche client esterno faccia richiesta di connessione; la prima cosa da fare è quindi quella di recuperare l'indirizzo IP e il numero di porta sulla quale il server attende connessioni. Queste informazioni sono contenute nel CDB, e in particolare nel file *DBBC.xsd*.

Per recuperare i parametri dal CDB si utilizza la classe *Configuration*. In particolare con i metodi *getAddress()* e *getPort()* recuperiamo rispettivamente l'IP e il numero di porta.

Si crea l'oggetto software *intf* di tipo *DBBCIntfClient* che si interfaccia al PC di controllo del DBBC passandogli come parametri IP e Port.

La chiamata al metodo *IsOpen()* di *intf* apre la connessione con il server DBBC.

### **4.2.3 Metodo LoadVLBI**

Questo metodo serve per configurare il DBBC in modalità VLBI. Il sistema è controllato dal *Field System*, cioè il sistema di controllo software e hardware standard per tutte le stazioni che afferiscono alla rete VLBI, e quindi esula da ACS; l'unica cosa che viene fatta è quella di caricare il corretto firmware nelle FPGA.

Viene creato un vettore di char al quale assegniamo una stringa che lato server attiva uno script che programma le FPGA con i relativi firmware presenti sul PC del DBBC. Questa stringa viene inviata al server attraverso il metodo *PutLine()* di *intf*.

### **4.2.4 Metodo LoadScan**

Questo metodo configura il DBBC in modalità spettrometro a scansione.

La descrizione dello spettrometro si trova in [2]; viene implementata una configurazione con una FFT polifase [4] per analizzare tutta la banda disponibile di 512 Mhz senza buchi spettrali, facendo una scansione automatica dei primi 256 Mhz e poi dei successivi, ottenendo alla fine 4096 canali spettrali.

Nel metodo *LoadScan* viene innanzitutto configurata una delle schede FPGA con il corretto firmware; la modalità è la stessa vista per il metodo *LoadVLBI*, ovviamente con una stringa differente che attiva uno script differente.

Si possono implementare fino a 4 spettrometri in contemporanea, ma per il momento ne è stato integrato solamente uno, nella prima scheda nello stack del DBBC.

Il primo passo è quello di impostare il tempo di integrazione e la fase del dll (delay locked loop). Come detto in precedenza, vengono analizzate due bande separatamente, e ognuna produce 2048 punti spettrali, quindi viene creato un puntatore chiamato *data* per gestire un

vettore di 2048 locazioni di tipo unsigned long int. A questo punto viene allocata la memoria necessaria per l'oggetto *ScanSpectrometer* che deve gestire la comunicazione con l'hardware, viene quindi scritto sull' FPGA il valore del tempo di integrazione e della fase del dll stabiliti in precedenza.

Un loop avvia le operazioni per l'acquisizione dello spettro, e va avanti finchè non si decide di interrompere l'esecuzione del metodo.

Nel corpo del loop si avvia la scansione della prima metà della banda, vengono caricati in *data* i 2048 canali quando il metodo *DataReady()* di *ScanSpectrometer* ritorna true e si procede con la seconda metà; alla fine avremo 4096 valori che vengono salvati e plottati, e si riparte con la scansione.

#### **4.2.5 Metodi *cleanUp* e *aboutToAbort***

Questi due metodi vengono richiamati in caso di terminazione. La terminazione di un component può avvenire in diversi modi, e quindi questi due metodi vanno fondamentalmente a coprire ogni possibile "morte".

Nel caso del *DBBCComponent*, viene richiamato il metodo *Close()* di *intf*; il metodo chiude la connessione e il server DBBC rimarrà in attesa di nuove connessioni.

## **5 Conclusioni e lavori futuri**

In questo report abbiamo descritto l'integrazione del Digital Base Band Converter nel framework ACS su cui si basa il software di controllo del Sardinia Radio Telescope.

Il component risulta abbastanza differente rispetto a quelli disponibili o in fase di sviluppo per gli altri backend, questo sia perchè è stata creata una configurazione che non richiede che si utilizzino tutte le procedure di un backend per osservazioni astronomiche, sia perchè i backend disponibili al momento sono stati progettati con un'unica configurazione mentre per il DBBC sono in fase di realizzazione differenti applicazioni; tra queste, oltre ad implementare quattro degli spettrometri visti per ognuna delle schede CORE2 del DBBC, verrà sviluppato uno spettrometro a scansione con 32768 canali per l'intera banda, uno spettropolarimetro a scansione e uno spettrometro capace di zoomare la sottobanda di interesse. Inoltre, mentre di solito ogni ricevitore ha il proprio backend dedicato, il DBBC verrà utilizzato per tutti i ricevitori e questo introduce delle specificità nella gestione dello strumento.

Per l'immediato futuro, il DBBC potrà quindi essere utilizzato come piattaforma osservativa anche per ricevitori che non dispongono di un backend come il ricevitore 100 Ghz disponibile



per SRT, che essendo monofeed con banda istantanea di 500 Mhz rientra perfettamente nelle specifiche. Il component descritto in questo rapporto verrà quindi arricchito di tutte le funzionalità richieste per una tipica osservazione astronomica, e questo anche nella prospettiva dei nuovi backend CASPER su cui svilupperemo dei firmware il cui controllo erediterà parte del lavoro fatto per il DBBC.

## Glossario

**ACS: *Alma Common Software*.** Framework software per il controllo di grandi radiotelescopi, basato su un'architettura distribuita

**CORBA: *Common Object Request Broker Architecture*.** Middleware commerciale per l'implementazione di un sistema software ad oggetti distribuiti che interagiscono tra di loro

**DO: *Distributed Object*.** Elemento software indipendente, visibile ad altri DO nel sistema grazie al framework CORBA

**IDL: *Interface Definition Language*.** Linguaggio per la descrizione di oggetti distribuiti nel sistema CORBA

**DBBC: *Digital Base Band Converter*.** Piattaforma digitale, sviluppata all'IRA di Noto, scelta dalla comunità EVN come backend standard per le osservazioni VLBI

**CASPER: *Collaboration for Astronomy Signal Processing and Electronics Research*.** E' un consorzio internazionale il cui obiettivo è quello di semplificare il flusso di progetto di strumentazione radioastronomica (<https://casper.berkeley.edu/>)

**FPGA: *Field Programmable Gate Array*.** Sono circuiti integrati digitali riconfigurabili dall'utente

**MACI: *Management and Access Control Interface*.** Servizio che gestisce le informazioni e l'interconnessione di tutti i DO e il loro ciclo di vita.

**CDB: *Configuration Database*.** Database centralizzato che si occupa della definizione, accesso e mantenimento della configurazione di un sistema ACS

**CORE2:** Scheda di elaborazione digitale contenente una FPGA Xilinx Virtex5 LX220

## Bibliografia

[1] G. Comoretto, G. Tuccari: *Reference design for the Digital BBC Architecture*, Arcetri Technical Report 2-2008

[2] Giovanni Comoretto, Andrea Melis, Gino Tuccari : *A wideband multirate FFT spectrometer with highly uniform response* , Experimental Astronomy, 2011, Volume 31, Number 1, Pages 59-68

[3] A. Melis, G. Comoretto : *Software di controllo per il Digital Base Band Converter: Software di comunicazione con la FPGA*, Arcetri Technical Report 5-2010

[4] A. Melis, G. Comoretto : *A 512 Mhz Polyphase Filterbank with overlapping bands*, Arcetri Technical Report 1-2011

[5] G. Valente, T. Pisanu, P. Bolli, S. Mariotti, P. Marongiu, A. Navarrini, R. Nesti, A. Orfei, J. Roda : *The dual-band LP feed system for the Sardinia Radio Telescope prime focus*, Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy V, edited by Wayne S. Holland, Jonas Zmuidzinas, Proc. Of SPIE vol. 7741, 774126 - 2010

## Indice generale

<b>1</b>	<b>Introduzione.....</b>	<b>1</b>
<b>2</b>	<b>Descrizione del framework ACS.....</b>	<b>1</b>
<b>3</b>	<b>Breve descrizione del Digital Base Band Converter.....</b>	<b>3</b>
<b>4</b>	<b>Descrizione del component DBBC.....</b>	<b>3</b>
<b>4.1</b>	<b>Interfaccia IDL.....</b>	<b>3</b>
<b>4.2</b>	<b>Implementazione C++ del component.....</b>	<b>4</b>
<b>4.2.1</b>	<b>Costruttore e Distruttore.....</b>	<b>4</b>
<b>4.2.2</b>	<b>Metodo initialize.....</b>	<b>4</b>
<b>4.2.3</b>	<b>Metodo LoadVLBI.....</b>	<b>5</b>
<b>4.2.4</b>	<b>Metodo LoadScan.....</b>	<b>5</b>
<b>4.2.5</b>	<b>Metodi cleanUp e aboutToAbort.....</b>	<b>6</b>
<b>5</b>	<b>Conclusioni e lavori futuri.....</b>	<b>6</b>
	<b>Glossario.....</b>	<b>7</b>
	<b>Bibliografia.....</b>	<b>8</b>