Giant Meterwave Radio Telescope

National Center for Radio Astrophysics

Tata Institute of Fundamental Research

Teleset Software

of GMRT Telemetry System

Description and implementation

Author : Laurent Pommier

Project Supervisor : Prof. Pramesh Rao

Date : Xx/xx/xx

Table of Contents

Illustration Index

1. Introduction

The Giant Meterwave Radio Telescope (GMRT) is a radio telescope located in Khodad, at 80 km from Pune (Maharastra, India). It is composed of an array of 30 antennas spread over distances up to 25 km. Each antenna is 45 m in diameter, and has been designed to operate at a range of frequencies from 50 to 1500 Mhz.

Modern radio telescopes are complex assemblies of electronic and electro-mechanical subunits. To allow a successful observation, all these subunits have to be set as per the user requirements. For example, antennas have to track the selected source, front-ends have to be tuned to the chosen frequency band, all the amplifiers along the signal path have to be set up to the appropriate value which would give the optimum signal to noise ratio, local oscillators have to be tuned to select the desired frequency, and the correlator has to be set up to do the appropriate fringe and delay tracking.

In an interferometer like the GMRT, this means that in a coordinated manner, one has to control systems which are several tens of kilometers separated from one another. In addition, systems must be periodically monitored, so that should any of them fail, the affected data can be flagged and remedial action can be taken to fix the faulty unit. Besides since it is not humanly possible to remember all the various safety limits of each subsystem, the telescope control system must also forbid any wrong operation. The Telemetry System is the one in charge of all these control and monitoring tasks in GMRT.

This report introduces Teleset, a Linux PC software which aims at replacing and improving the Online PC of the Telemetry chain. Teleset is a Control Room program for Operators, it communicates with all the telescope antennas. Inside the antennas, ABCcom software is a new PC program, adapted to Teleset and replacing the actual PIU ABC. Rest of the systems remains unchanged. Thus Teleset and ABCcom are forming a new software chain for GMRT Telemetry.

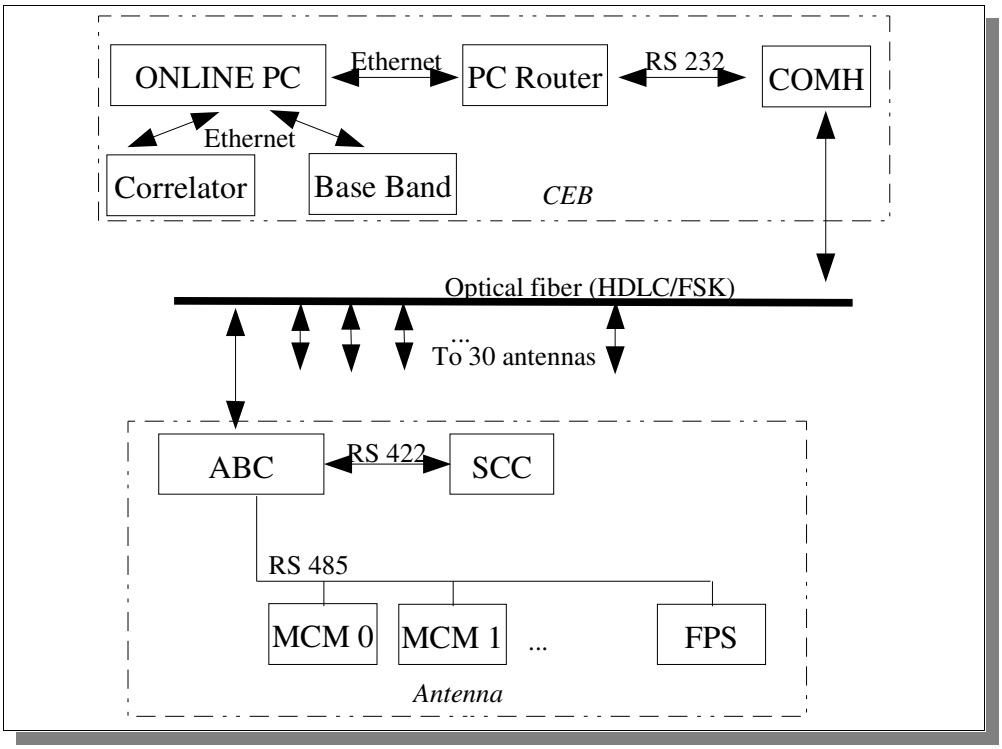
This document first gives an overview of this project context in parts 2 and 3. Then chapter 4 details Teleset important concepts and how they have oriented the software design. Finally parts 5 to 8 describe all the different blocks present in the global diagram of the program, along with their implementation.

2. General presentation of the Telemetry System

The GMRT Telemetry is the system whose goal is to control other systems operations in a coordinated manner and to monitor their current state to check parameter values and detect any dysfunction. Its main tasks are:

- To rotate all the 30 antennas in azimuth and elevation, to track a celestial source
- To bring the required feed in the feed turret to the focus via the Feed Position System (FPS).
- To select the Front End system parameters like observing frequency band, noise calibration...
- To set the Intermediary Frequency (IF) and Local Oscillator (LO) systems including frequencies, IF bandwidths and attenuations, Automatic Level Controller (ALC) operation...
- To set the Base Band and Correlator parameters
- To monitor hundreds of systems parameters at all points along the signal flow path

This system is divided into many components based whether inside the CEB or in antennas shells:



Legend:

. MCM	*Monitor and Control Module*	*. ABC*	*Antenna Based Computer*
. FPS	*Feed Positioning System*	*(=ANTCOM)*	*(=Antenna Computer)*
. COMH	*Communication Handler*	*. SSC*	*Station Servo Computer*
. CEB	*Central Electronic Building*		

Illustration 1 : Components of the Control and Monitoring system

3. Teleset project

3.1. Teleset and ABCcom, a new Telemetry software chain

ABCcom is a software running on a Linux PC to replace the current ABC PIU. It performs the telemetry operations of an antenna, and is documented in reference [6]. Installing ABCcom requires to change as well the Online program, since all the communication protocol between Online and ABC is changed. The new Online program is called Teleset, and is presented in this document.
With this new telemetry software chain, the heavy part of the telemetry intelligence is shifted from the Online PC, to the ABC PC. Indeed in the actual system, all data packets are generated from the Online PC and PIU ABC just transmits them. ABCcom is now generating all the commands, and only high level parameters are sent by Teleset.
For instance, monitoring the LO system implies to generate 30 messages to MCM 2 and 3. Instead of Online sending all those messages, Teleset just transmits a short LO monitor command to ABCcom which composes and sends by itself the 30 packets. The communication between CEB and antennas is reduced and thus the information is transmitted faster.
Besides this, Online is a program that had been implemented and modified during several years in C and Fortran languages. Teleset is now a new C++ program which re-implements the telemetry tasks in an organized and concise manner.

3.2. Teleset specifications and remaining developments

Teleset as presented in this document can perform many of Online operations. More precisely, Teleset has the following list of specifications:

- ✔ it provides a user interface for the Operator commands,
- ✔ it composes commands according to Teleset - ABCcom protocol,
- ✔ it sends messages to several ABCcom connected in parallel, and collects their answer,
- ✔ it decodes ABCcom answers according to the communication protocol,
- ✔ it fills a shared memory and some history files with each of its activity results.
- ✔ it allows a multi user configuration which divides GMRT into several sub arrays of antenna.

Nevertheless some important developments remain to have a full Telemetry software for GMRT:

- ➔ communication with Correlator System
- ➔ communication with Base Band System
- ➔ improvement of Operator interfaces
- ➔ development of a different hardware link (router/comh)

Like Online PC, Correlator and Base Band systems are located inside the CEB. They have specific sets of commands / answers which follow a specific format (message queue, ASCII), independently from Teleset-ABCcom protocol.

4. Teleset configuration

This section describes some of Teleset fundamental specifications and shows how they have influenced the overall design of the software.

4.1.Link between Teleset and ABCcom PCs

GMRT is composed of 30 antennas and a Central Electronic Building (CEB). For this new telemetry chain, a PC inside the CEB is running Teleset program. It is connected in parallel to each of the 30 antennas, with their respective PC running ABCcom program.

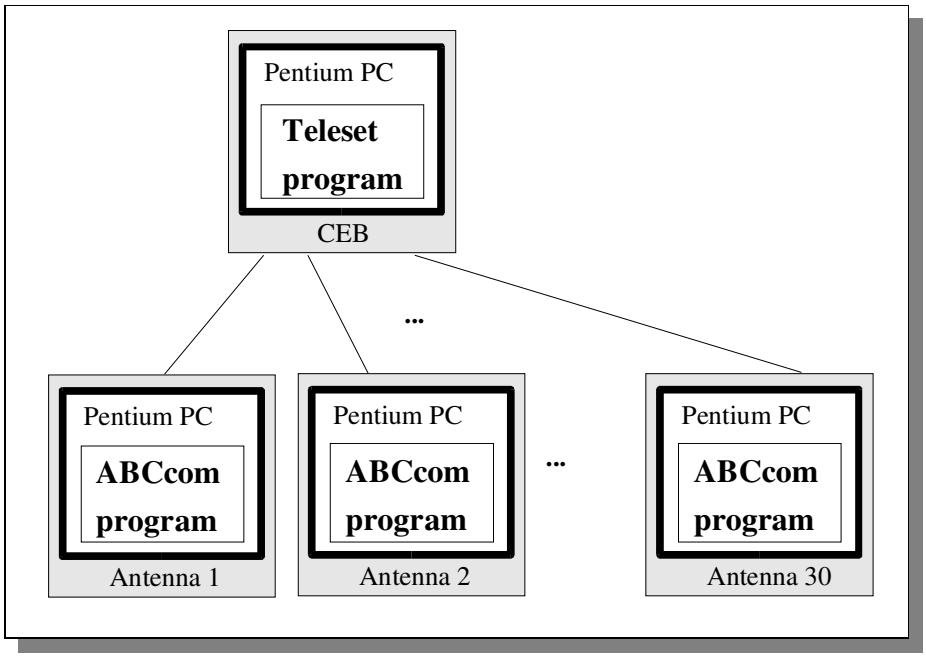


Illustration 2 : Link between CEB and antennas

Until this date, this link has been simulated with RS-232 connections, using PC serial ports. A PC is originally equipped with one or two serial ports, and a multi serial port card was added. The ISI 4608 – PCI card from Multi Tech Systems company offers 8 additional serial ports on a Linux PC.
Teleset reads the PC connections by reading a ***ConfigTSET*** file.

The code to communicate on Linux Serial Port is in C (***comhlink*** files). It provides the open, close, read and write functions for the PC serial ports, and is described in ref. [6], part 12.1.

```
######################################################################
# ConfigTSET
#
# configuration file for Teleset program
#
# file to load FPS feed encoder position
#
# :N A for Serial Port number N and Abc number A
#
#
# Possible ABC Addresses: 0 to 31.
# Possible Serial Port numbers: 0 to 31.
# Pre-requirement: serial port available (multi serial port card)
#
# L.Pommier, 20/12/2005
######################################################################

:5 1
:6 2
:7 3
:8 4
```

Illustration 3 : ConfigTSET file

For the final installation in GMRT, an hardware unit will be used (ex. CEBCOM PIU) to adapt these PCs to the current optical fiber link of GMRT. A PC router like in Illustration 1 may also be needed.

4.2.Automatic State command, LowUser and HighUser classes

Beside sending commands entered by the Operator, Teleset also sends by itself a State command to ABCcom PCs every 3 secs. This is very important since it allows Teleset to get regularly and automatically the state of antennas (parameters values, current operations and eventual errors).
For the design of Teleset software, it means that there must be a process (or a thread) which is doing the automatic State command, while another process (or thread) is waiting for the next Operator command.

- One process is implemented in HighUser class. It gets an Operator command, composes its communication packet and places it in a buffer. It then signals whether the command has been accepted or not.

- The other process is coded in LowUser class. This class transmits HighUser commands on serial ports, receives and decodes corresponding ABCcom answers. When there is no pending HighUser command, every 3 secs LowUser generates an ABC State command to ABCcom PCs and treats their answer.

4.3.Sub-Array concept and shared memory

GMRT is an array of 30 antennas spread over distances up to 25 km. This array can be virtually divided into several sub arrays, each composed of distinct antennas. This allows to use at the same time different sub arrays of antennas for different observations. GMRT is like split into smaller telescopes. Teleset software is designed to allow the multi sub-array feature.

From the beginning of a Teleset program execution, two different configurations can be run:

- a Master configuration.

It is the first Teleset program launched. A Shared Memory is allocated. A HighUser object starts in a thread and corresponds with a LowUser object which runs in the main process. For any additional sub-array index [i], the Master adds a LowUser [i] in a thread.

- a secondary User configuration.

It is when a Teleset program is launched, whereas a Master is already running. After a proper index [i] is entered, a HighUser process gets Operator commands and transmit them to the LowUser [i] via the shared memory. No command is accepted until a valid sub-array of available ABCcom PCs is entered.

Thus one HighUser and one LowUser object are dedicated to each of GMRT sub-arrays. These objects are from the same respective classes, just the Master has extra facilities to add and remove Users.
This configuration choice is implemented in TeleUser class (***mainteleset*** files). When Teleset program starts, the first object called is a TeleUser class object. It first tries to open a ***teleset.dat*** file. If this one does not exist, or if it contains the integer 0, it becomes a Master. If the file exists and contains the integer 1, it becomes a secondary User.

4.4.Teledisp program and shared memory

The GMRT Operator is using Teleset to control and monitor antennas with ABCcom. Teledisp is a separate Linux program that allows to observe on line all Teleset activity and results. Different persons can at the same time log into the Control Room PC where Teleset is running, and launch a Teledisp program.
When Teleset first starts, it creates a shared memory in which is stored all its data. Teledisp program reads this shared memory and displays in a graphical interface all the telemetry information.
Teledisp is developed with QT Designer. The final design being not yet decided, a basic Teledisp was implemented for this project.

4.5.Teleshell mode

When the Master exists, all secondary are forced to exit too and Telemetry tasks are stopped. Then the Master program goes in Teleshell mode. This is a communication implemented between Teleset and ABCcom, to transfer files and to execute Linux shell terminal commands into ABCcom PC remotely from Teleset PC. This communication then concerns only one ABCcom at a time.
A TeleShell object is used for this communication (***teleshell*** files). Like in Telemetry mode, the Operator enters commands in a shell interface. He first selects an ABCcom number. TeleShell sends a packet to ABCcom to switch its main object to an AbcShell (cf. [6], part 7.1). It is then able to transfer a file to this ABCcom PC, to get a file from it, or else to execute a Linux shell command in it. The shell answer is transmitted and displayed in the operator shell interface. Commands are resumed in Appendix B, and the protocol of this communication is described in document [7]. When TeleShell exits, ABCCom returns to Telemetry mode and Teleset program fully ends.

4.6.Teleset configuration examples

The illustration below corresponds to a situation where:

- ✗ One Teleset has been first started, creating a Master (HighUser and LowUser), the Shared Memory and the configuring the ABCcom link.
- ✗ One more Teleset has been started, creating HighUser 1. The Master has created LowUser 1 for it.
- ✗ One Teledisp is running to display these telemetry operations

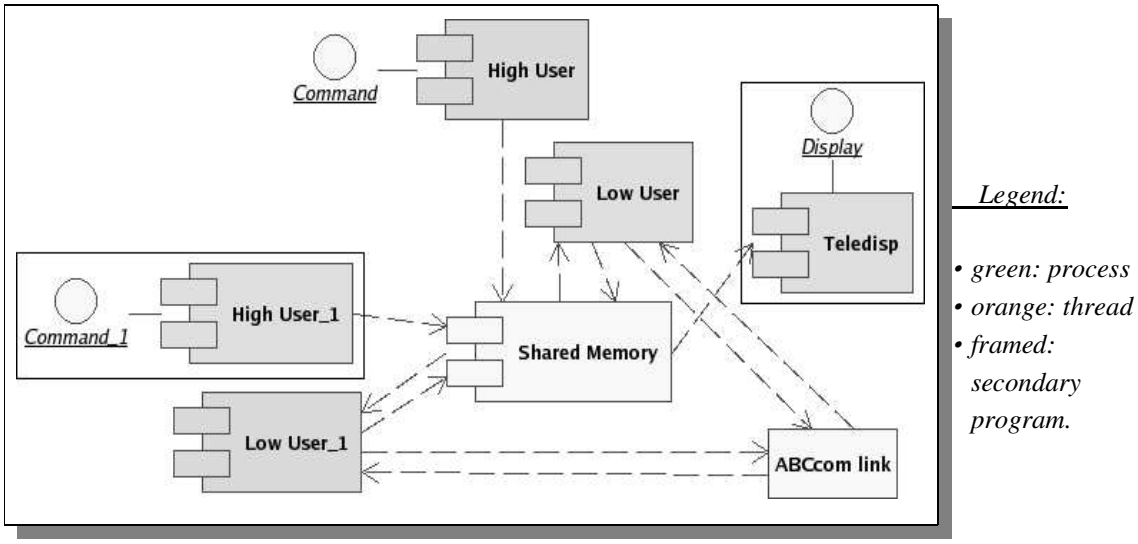


Illustration 4 : UML Component diagram of Teleset

The case diagram below illustrates another situation where two sub-arrays are used with Teleset: 1 sub-array by the Master and 1 by User 1. Both of them plus an external observer are watching the telemetry monitoring with Teledisp.

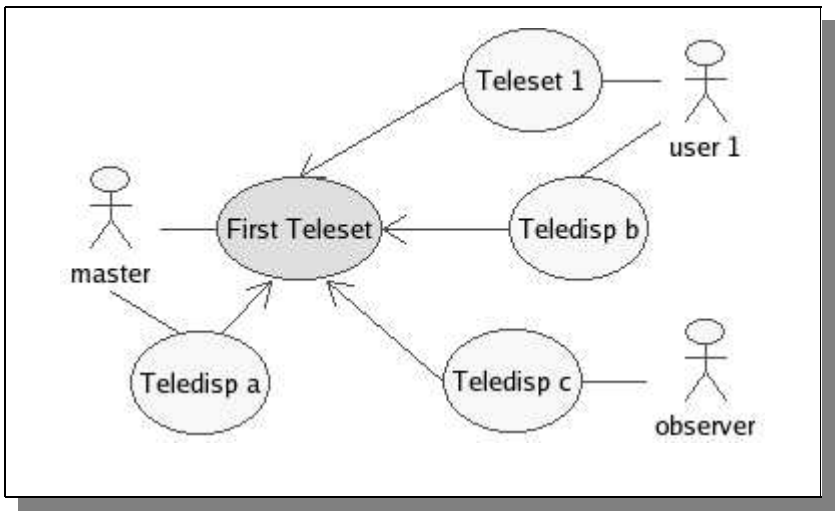


Illustration 5 : UML use case diagram of Teleset

5. HighUser class

HighUser class is implementing the interface between Operator entries and the telemetry system, restricted to one sub-array of GMRT antennas. The Operator must first define a valid sub-array of antennas (not previously used) to be able to send commands to their ABCcom PCs. Each HighUser object is associated through shared memory with a LowUser object which transmits its commands on the ABC link.
A HighUser object is created in the first Teleset program (Master), and it is running within a thread. Any Teleset program launched after (User i) also creates a HighUser object, but this one is run by the program process.
The interface developed in this Teleset version is a Linux Shell window, in which the Operator types command scripts. This script is detailed in Appendix A, "Teleset Manual for Operator". A more modern user window can be developed for the program next version, as well as a file input option with commands and time markers.

5.1. HighUser variables and functions

HighUser class is implemented in ***highuser*** files. Its functions are:

- Start() is called by a process (or thread). It performs the following tasks in a loop:
  - ✔ get the Operator inputs from a Shell interface and compose a command packets according to Teleset-ABCcom protocol (cf. [7]),
  - ✔ place the packet in the shared memory.
  - ✔ wait for the command to be sent on the link (q), and for ABCcom answers to be received (w). When all ABCcom of the sub-array have answered, it signals a success (S). After 15 secs if some ABCcom answers are missing, it signals a failure (F).
- Interface() function is called by Start() to compose a commands packet. It reads the first shell input and accordingly uses a Cmd System object. Then it places the packet in queue inside the shared memory.
- TelGetCmd() implements commands configuring Teleset parameters (sub-array mask...).

HighUser variables are:

- ✗ FileRec is a pointer on the history file recording HighUser activity.
- ✗ UserIndx is the user index (0 for master)
- ✗ There is one Cmd System object per system. All of them are described in the next paragraph.

5.2. Cmd System classes

Cmd System classes are composing command packets according to Teleset-ABCcom protocol (cf. [7]). Their inputs are script words entered by the Operator in the shell interface, and they can refer to some input files stored in a specific directory.

When they are created, these objects get pointers on the shared memory and on HighUser history file. Whenever they get new parameters for their system, they store those values inside the shared memory so that Teledisp can read whatever setting has been lastly done. Their operations and parameters are also written in the history file.

Each Cmd System class is dedicated to one GMRT system. There are CmdSsc, CmdAbc, CmdFps, CmdExp, CmdFe, CmdLo and CmdIf classes respectively for Servo, ABC, FPS, EXP, FE, LO and IF systems. Their implementation follows the inheritance tree below.

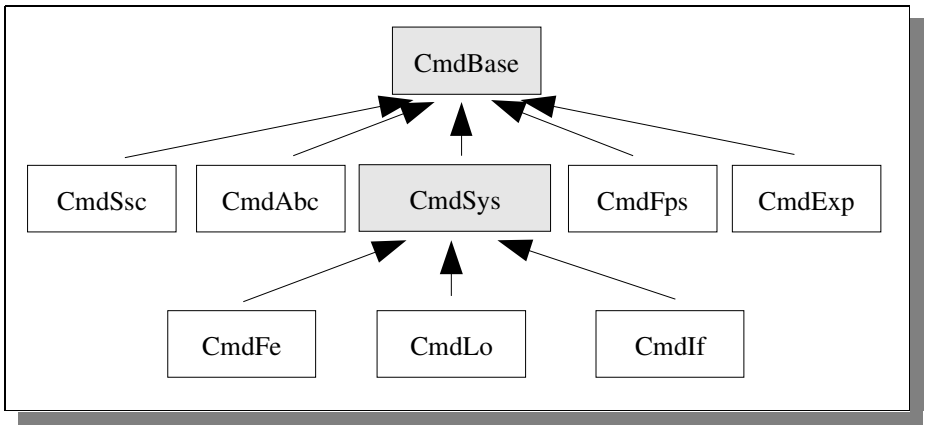


Illustration 6 : Inheritance tree of Cmd System classes

- ✔ CmdBase (***cmdbase*** files)

This abstract base class provides the code to push data into packets to be sent to ABCcom, according to the communication link protocol (cf. [7]). It also provides functions to store parameters in the shared memory, whether they are values relative to all the sub-array (setting parameters) or some relative to a specific antenna (loading parameters).
HighUser::Interface() calls GetCmd() function which calls GetPacket() virtual function and makes the header of the command sub-packet.

- ✔ CmdSys (***cmdsystm*** files)

This abstract base class composes commands common to LO, IF and FE systems in GetPacket() function (do settings, do monitoring, reboot...). ReadSettings() function reads setting parameters from a file (***set1.dat***, cf. Appendix A).

- ✔ CmdSsc (***cmdbase*** files)

CmdSsc class composes commands relative to Servo System. Two different types of Servo commands exist. Some are targeting the Servo Thread of ABCcom program, they mainly concern tracking operations. The other type is for direct commands, which ABCcom faithfully transmits to SSC unit. They are generated by ServoCmd() function. The Teleset-ABCcom format has been designed to be easily compatible with ABCcom-SSC protocol (ref. [1] and [2]).

- ✔ CmdAbc (***cmdbase*** files)

CmdAbc class composes commands relative to ABCcom. They deal with its configuration (local and expert modes, time and date), plus the ABC state command.

- ✔ CmdFps (***cmdbase*** files)

CmdFps class composes commands relative to FPS system. Like for Servo, some commands are directly targeting FPS unit (MCM 14), whereas some are generating operations in the FPS Thread of ABCcom program. Direct commands are coded in FpsDirect() function, according to the FPS communication protocol (cf. [3] and [4]).

- ✔ CmdExp (***cmdbase*** files)

CmdExp class composes commands addressed to a specific MCM. The Operator must first send a 'startexp' command to ABCcom otherwise this last one will ignore these expert commands. According to the MCM communication protocol (cf. [5]), commands are coded in GetPacket() function, .

- ✔ CmdLo (***cmdsystm*** files)

CmdLo class inherits CmdSys class, it then only composes commands specific to LO system.

- ✔ CmdFe (***cmdsystm*** files)

CmdFe class inherits CmdSys class, it then only composes commands specific to FE system.

- ✔ CmdIf (***cmdsystm*** files)

CmdIf class inherits CmdSys class, it then only composes commands specific to IF system. GetIfval() functions reads loading parameters from a file (***ifload_2.dat,*** Appendix A).

6. LowUser classes

LowUser is the class operating between the Teleset shared memory and the ABC link. Each LowUser object is associated with one HighUser object via the shared memory, and therefore it is restricted to a sub-array of antennas. In opposition with HighUser objects, LowUser objects are only elements of the Teleset Master program.
In the Master (first Teleset launched), a LowUser object is run within the program process. To accept another sub-array, the Master creates an additional LowUser object and runs it within a thread. Secondary User programs (next launched Teleset) do not have any LowUser object. Different uses of LowUser class are described in this chapter, and implemented in ***lowuser*** files.

6.1. LowUser variables and functions

LowUser is a base class where are implemented the transmission of HighUser commands to ABC link, the automatic 'State' command to monitor regularly ABCcom PCs (cf. 4.2), and the reception and decoding of ABCcom answers. Its activity is recorded in an history file.

LowUser main functions are:

- UpdateUSubArray() compares LowUser sub-array to the sub-array written in the shared memory for its associated HighUser. Accordingly it opens or closes PC serial ports with AddPorts() and RmPorts () functions. AddPort() function reads Teleset PC connections from ***ConfigTSET*** file (cf. 4.1).
- PollPorts() polls messages from opened serial ports and flags ABCs who answered (FlagAbc()). This allows to signal a command success when all ABCs of the sub-array have answered, or a Time-Out in the opposite case. A Decode object is called to decode ABCcom message (cf. next section). When an automatic State command has been sent, PollPorts() also signals its success or time-out in history file and shared memory.
- WritePorts() gets a HighUser command from the shared memory and sends it successively on each opened port of the sub-array. If no HighUser command is pending, it calls WriteState().
- WriteState() composes a ABC State command, and sends it every 3 secs on each opened port. It rises some synchronization flags (StateFlag, StateAns[]).
- BusyCheck() is used by other functions to exit in case an operation is already in process.

LowUser main variables are:

- pointers on the shared memory and on an history file created for this LowUser object.
- PacketDec is a Decode object used to analyze messages from ABCcom and write results in shared memory.
- UsubArray[4] is the sub-array mask stored in LowUser, used when opening or closing ports.
- StateAns[4] is a sub-array copy used for State automatic command. It is reset to zero when all ABCcom PCs have answered.
- PortArray[] contains serial port structures used for link communication.

6.2.LowUserThread class

LowUserThread class inherits LowUser class. When a User index [i] has to be added, the Master creates a LowUserThread object and its [i] index becomes valid inside the shared memory. The next Teleset has first to give this index before being able to create the associated HighUser object.

At a LowUserThread object creation, a POSIX thread is started with the Run() function. This function performs the following calls in a loop:

- ✔ UpdateUSubArray(); to update any change in the sub-array definition
- ✔ WritePorts(); to write HighUser or automatic State commands to the ABC link
- ✔ PollPorts(); to get and decode ABCcom answers

The loop exists and the thread ends when a flag is raised inside the shared memory. This occurs when the HighUser exists, or when the Master decides to remove the User.

6.3.MasterLUser class

MasterLUser class inherits LowUser class. It is used when the first Teleset program is launched, that is when the Master is started. In the program process, MasterLUser Run() function calls in a loop the following sub-functions:

- ✔ UpdateUSubArray(); to update any change in the sub-array definition
- ✔ WritePorts(); to write HighUser or automatic State commands to the ABC link
- ✔ PollPorts(); to get and decode ABCcom answers
- ✔ UpdateLUser(); to add or remove LowUserThread objects. This function is in MasterLUser class, not in LowUser class.

This loop exits when the master HighUser exits. At MasterLUser deletion, all LowUser objects are deleted and the associate HighUser in secondary Teleset programs are forced to exit too.

☞*Note: HighUser objects have all the same functions, adding and removing a LowUser thread is ignored when they are not associated with the MasterLUser object.*

7. Decode class

Decode class is decoding answer messages from ABCcom according to the Teleset-ABCcom protocol (cf. [7]). Results are written in Teleset shared memory so that Teledisp program can read them. A Decode object is inside each LowUser object, they share the same history file.
When a LowUser object receives a message from an ABCcom, it immediately calls its Decode object to analyze it. This stage offers the main advantage that the decoding work is done once by Teleset, so its output (shared memory and history files) can be straigthly read by others (Teledisp programs...).

7.1. Decode variables and functions

Decode class is implemented in ***lowuser*** files. It has just one function AnalysePack(), which is called when an ABCcom message is received in Teleset LowUser. This function decodes the packet header. A message is further composed of answer sub-packets each dedicated to one GMRT system. Decode class contains a Dec System object per system. Successively each answer sub-packet is analyzed by the corresponding Dec System object.

7.2. Dec System classes

Dec System classes are used to analyze answer sub-packets received from ABCcom. They write their results in the shared memory and in LowUser history file.
Similarly to Cmd System classes, each Dec System class is dedicated to one GMRT system. There are DecSsc, DecAbc, DecFps, DecExp, DecFe, DecLo and DecIf class respectively for Servo, ABC, FPS, EXP, FE, LO and IF systems.
ABCcom answer sub-packets can be of 3 types: a 'State' answer, an 'Askparam' answer, and an Acknowledgment. These answers have a lot of similarities between Systems, and accordingly their implementation is following the inheritance tree below.

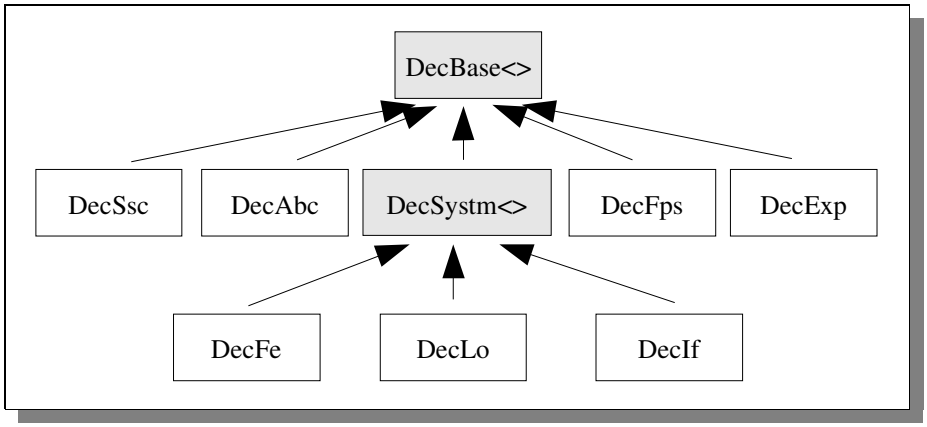


Illustration 7 : Inheritance tree of Dec System classes

- ✔ DecBase<> (***decode*** files)

This abstract base class provides functions to pop data from ABCcom packets, according to the communication link protocol (cf. [7]). Its template is a pointer to a shared memory block in which results have to be written. It also provides the following functions:

  - Analyse() function calls AnalyseState() for State answers, or calls DecParam() for Askparam answers, or finally decodes acknowledgments.
  - DecParam() reads and stores Setting and Loading parameters of the system.
  - ReadMcmTimOut() reads and store data concerning MCM time outs (unused for DecSsc).

- ✔ DecSystm<> (***decode*** files)

This abstract base class decodes in AnalyseState() the state answers for LO, IF or FE systems. It calls DecSys() virtual function for the part specific to the system. Its template is also a pointer to a shared memory block.

- ✔ DecLo (***decode*** files)

DecLo class inherits DecSystm class, it then only analyzes the state answer part specific to LO system.

- ✔ DecFe (***decode*** files)

DecFe class inherits DecSystm class, it then only analyzes the state answer part specific to FE system.

- ✔ DecIf (***decif*** files)

DecIf class inherits DecSystm class, it then only analyzes the state answer part specific to IF system. After the monitoring of IF system, an array of raw data is transmitted to Teleset in state answers. Further decoding of this array can be performed here. Functions for this purpose are written in the source file but not used yet since this system has been recently modified.

- ✔ DecSsc (***decssc*** files)

DecSsc inherits DecBase class. It decodes Servo State answer sub-packets, which can be of various types: time-out and error, SSC events, SSCanswers and display answers, and finally track data. AnalyseState() reads the first bytes and accordingly calls the proper private function. Some parts of those packets are directly coming from SSC unit, they are decoded according to the SSC communication protocol (cf. [1] and [2]).

- ✔ DecAbc (***decode*** files)

DecAbc inherits DecBase class. It decodes State answers sub-packet relative to Abc System.

- ✔ DecFps (***decfps*** files)

DecFps inherits DecBase class. It decodes FPS State answer sub-packets. Some parts of those packets are directly coming from MCM 14, they are decoded according to the FPS communication protocol (cf. [3] & [4]).

- ✔ DecExp (***decexp*** files)

DecFps inherits DecBase class. It decodes FPS State answer sub-packets. Some parts of those packets are directly coming from an MCM, they are decoded according to the MCM communication protocol (cf. [5]).

8. Teleset Shared Memory

A shared memory is an Inter Process Communication. It is required when different programs need to share a common resource data.
In the present case, programs are several Teleset (one Master, some secondary Users) and several Teledisp (cf. Illustration 4.). The first Teleset program (Master) creates the shared memory object by allocating some space in the PC memory. Other programs just attach their pointer to this memory address. Before exiting, the Master marks the shared memory space to be deleted when the last attached program exits.

Under Linux operating system, shared memory is implemented with some C functions. Using them in C++ required to take some precautions. Indeed a shared memory space is allocated at the beginning with a fixed size. The C++ object used for this shared memory must then also have a fixed size (no dynamic structure or virtual function). Its constructor and destructor functions are not called and an Init() function is written instead.
All Teleset Shared Memory is coded in ***sharemem*** files. Any program reading it must include those files in its source code.

The overall block diagram of the share memory is shown in UML class diagram in Appendix A.2. It consists of a main ShareMemory class which is divided into two objects: SetShare and MonShare respectively of SetData and MonData classes. ShareMemory also provides functions to attach, create, detach and delete the memory space for those two objects.

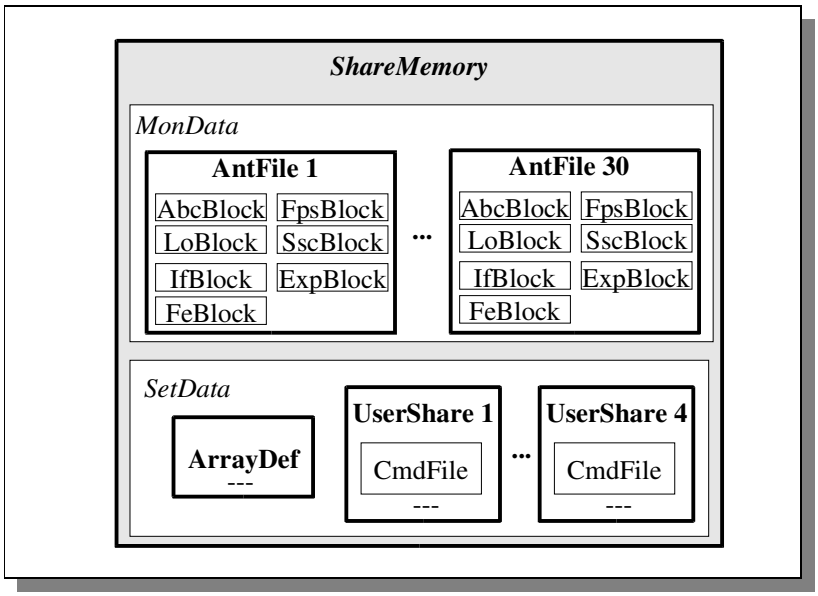


Illustration 8 : Shared Memory structure

8.1.SetData class

SetData contains all the information about Teleset commands. This part of the shared memory is used not only between HighUser and their LowUser objects, but also by Teledisp program to know about Teleset users and their operations.
SetData information is arranged inside the following variables:

- ArrayDef ArrDef; store the sub-arrays definition and provide functions to work with them.
- UserShare UserArray[]; sub-arrays data. UserArray[] has one element for each sub-array (or user / master, cf. below).
- float AntLoad[][][]; Loading parameters, per antenna and per system.
- int NbAbcTimOut[]; number of ABCcom time-outs, per antenna

UserShare class contain information about a sub-array. It stores its sub-array current setting parameters and last commands entered in HighUser interface. Its main variables are:

- BaseFILO CmdFile; FILO list of commands, written by HighUser and read by LowUser, for the transmission between Operator interface and ABC link.
- char SubArray[4]; sub-array definition
- char AbcAns[4]; flags of ABCcom answers, written by LowUser and read by HighUse, to signal eventual success.
- int UserNb, RmNb, AddNb; to add and remove secondary users, used only by MasterLUser.
- float Setparam[][]; current Setting parameters for the sub-array, arranged per system.

8.2.MonData class

MonData class stores all the information about ABCcom answers. It is written by the Decode object of each LowUser, and read by Teledisp programs (or future display programs). ABCcom packets are decoded and stored in the shared memory as ASCII messages and float parameters, so that any program reading the shared memory can just take the information and display it without intermediary processing.

MonData contains an array of AntFile objects. Each Antfile is dedicated to one antenna, and stores the information sent by its ABCcom. It is itself composed of one block per system:

- SysBlock<IfState> IfBlock;
- SysBlock<FeState> FeBlock;
- SysBlock<LoState> LoBlock;
- SysBlock<ExpState> ExpBlock;
- SysBlock<FpsState> FpsBlock;
- ServoBlock SscBlock;
- SysBlock<AbcState> AbcBlock;

- ✔ SysBlock<>

SysBlock<> is a template class used for MCM based systems (all but Servo). It contains the MCM addresses, their total number of time outs, the system Setting and Loading parameters as answered by the ABCcom, and the time at which those parameters were read from ABCcom packet. Finally it has a ListState<> object. The SysBlock template is the ListState template.

- ✔ ListState<>

ListState<> has an 10-size array of template elements. Each element is the decoded data from a State answer (for one system of one ABCcom). Like BaseFILO, this array is used as a circular queue filling elements in FILO manner. Thus it stores the last 10 State answers.
This class also provides functions to access those elements. For instance GetNextRdState() function places a pointer on the next State element the caller wants to read. This is quite tricky and requires time stamps and last pointer used.

- ✔ System state classes

Those classes are used as the template of ListState<> class. They contain the decoded data of their system State answer. Except for Servo, there is one class per system: LoState, IfState, FeState, AbcState, FpsState and ExpState, implemented using SysBaseState and SystmState parent classes.

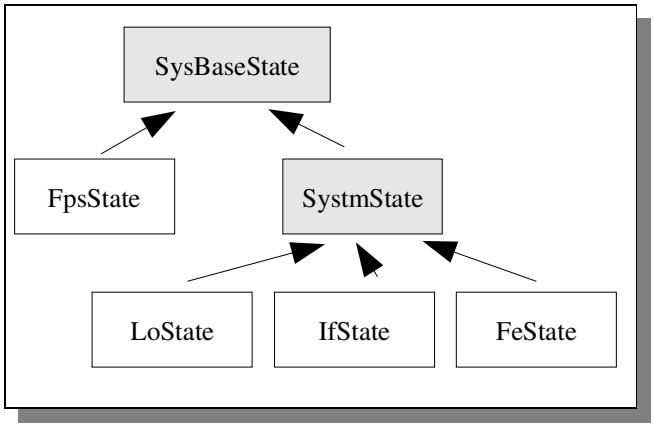


Illustration 9 : Inheritance tree of System State classes

- ✔ ServoBlock

ServoBlock is different from other system blocks since Servo State answers can be of various types, and they contain generally more data that need to be classified. Servo State classes are SscAngle, SscAnlvar, SscDigvar, SscMessage, SscAns. Each of them are recorded in a ListState<> class. ServoBlock then has a ListState<> object for each of its different types of State answers, along with some parameters (acknowledgments, link or command time-out numbers...).

9. Conclusion

Teleset is a Linux PC software for GMRT Telemetry System. It is to be run in the Control Room by the Operator to perform the telemetry operations during observations. It sets all the various parameters of antenna systems, and continuously monitors their current state.

Teleset communicates with all 30 antenna PCs equipped with ABCcom software. This new telemetry software chain provides advanced functionalities compared to the previous system, and reduces the communication time between GMRT control room and antennas.
The multi user configuration allows to virtually divide GMRT array into sub-arrays of antennas, and to run similar Teleset programs for each of them. The first Teleset launched is called the Master and next ones are secondary Users.

The Operator interfaces are Linux Shell terminals, to enter commands in a script format. To display the states of Telemetry system and of antennas, Teleset stores decoded data in a structured shared memory. Any display program like Teledisp can connect to this memory, straigthly print its data. This way several display programs can run at the same time.
All Teleset activities are recorded in a set of history files. These are ASCII text files and therefore they can be directly readable in any text editor.

This software is mainly coded in C++. The object oriented design is very convenient for general organization and future evolutions. The remaining development in Teleset program is the addition of two object blocks in the general diagram. One shall implement the communication with Correlator system and the other with Base Band system. This development duration can be estimated at 6 months depending on the programmer knowledge about those GMRT systems.

To conclude, Teleset software was implemented during 2005 in GMRT laboratory. Using a PC equipped with a multi-serial port card providing 8 connections, the two following simulations were successful:
(i) 1 Teleset master connected to 4 ABCcoms, to test the communication with several ABCcom.
(ii) 1 Teleset master and 3 Teleset users, each connected to 1 ABCcom, to test the multi-user configuration.
During the maintenance period of November 2005, Teleset and ABCcom software chain has been thoroughly and successfully tested in GMRT C9 antenna.

References and label index

References:

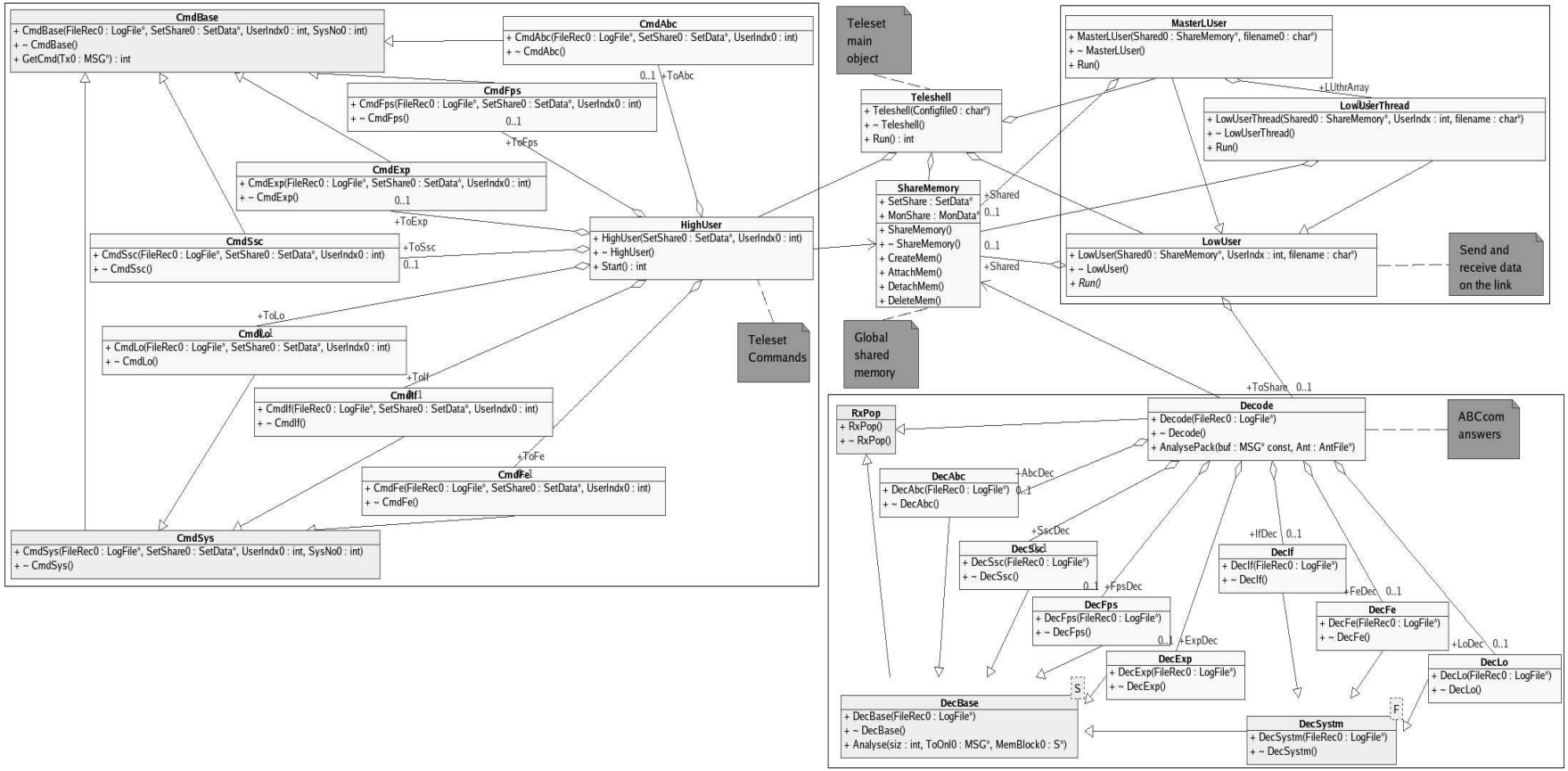
[1] BARC, *Servo software BARC: SSC interface details*, April 1991
[2] BARC, *Newsmu pascal program code*, 1997
[3] Mukund Gadgil, *Feed Positioning System Software*, April 1992, GMRT technical report.
[4] Mukund Gadgil, *FPS DOS program code*, 1992.
[5] Laurent Pommier, *Mcmcom program, a Linux PC / MCM communication*, September 2004, GMRT technical report.
[6] Laurent Pommier, *ABCcom software of GMRT Telemetry System*, January 2006, GMRT technical report.
[7] Laurent Pommier, *Communication protocol between ABCcom and Teleset*, March 2006, GMRT technical report.

Index table:

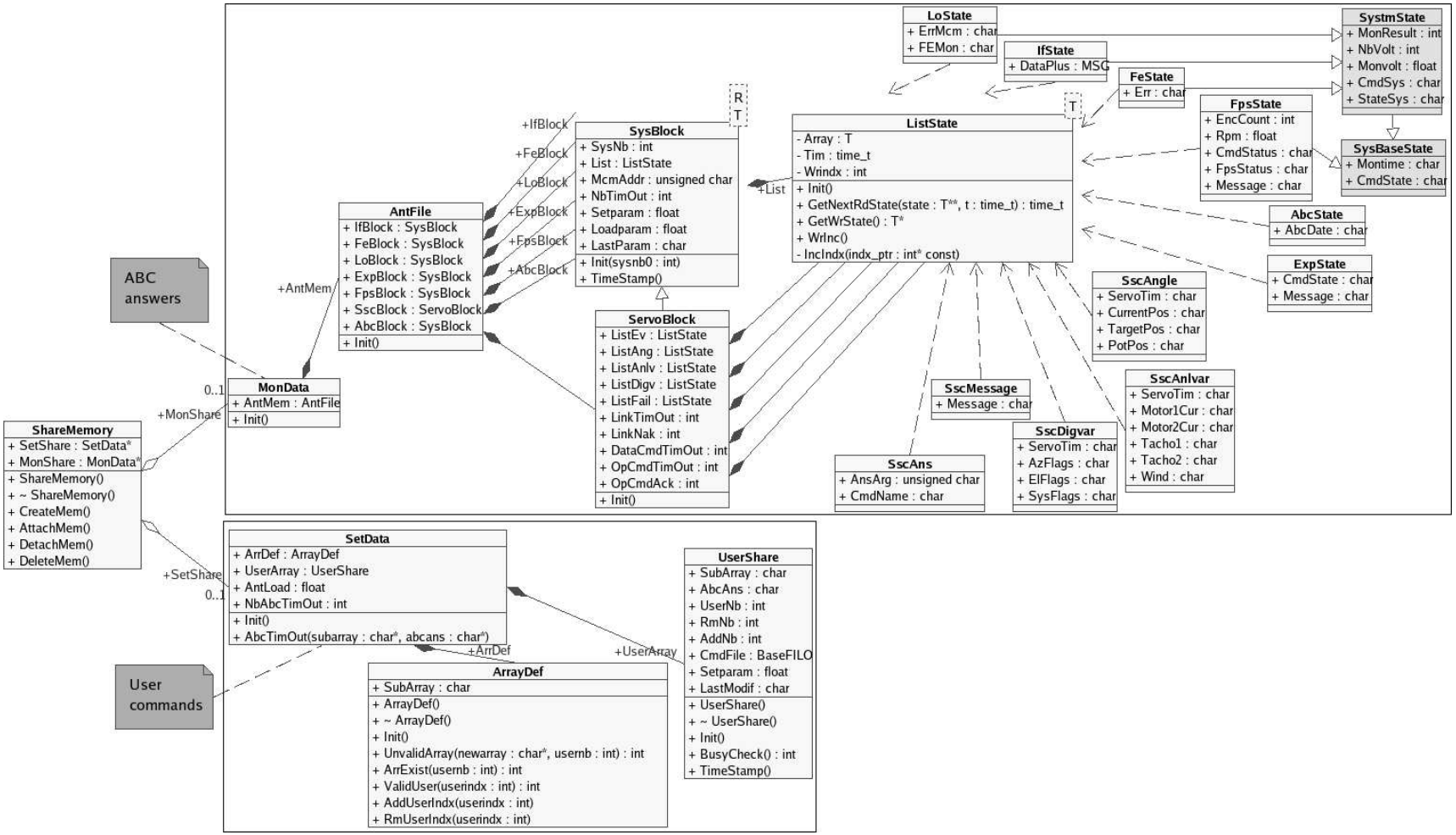
Label	***Full Name***
ABC	Antenna Based Computer
ANTCOM	Antenna Computer
CEB	Central Electronic Building
COMH	Communication Handler
EXP	Expert system
FE	Front End system
FILO	First In Last Out list
FPS	Feed Positioning System
GMRT	Giant Metrewave Radio Telescope
IF	Intermediary Frequency system
LO	Local Oscillator system
MCM	Monitor and Control Module
PIU	Plug In Unit
POSIX	Portable Operating System Interface
RS	Research Standard
SSC	Station Servo Computer

Appendix A: UML diagrams

A.1. General class diagram



A.2. Shared Memory class diagram



Appendix B: Teleset Manual for Operator

This appendix is a short manual for Teleset program user. It lists all user commands implemented at this date for the Teleset – ABCcom new telemetry software chain. The final form of the Operator interface has not yet been decided. Until now, commands are organized by system, and entered as text in a Linux Shell Terminal.

B.1. TeleShell Commands

User entry examples	*Command description*
Shl> put filename	Transfer file from Teleset to ABCcom in source directory
Shl> get filename	Transfer file from ABCcom to Teleset in source directory
Shl> exit	Exit TeleShell, an exit command is sent to ABCcom
Shl> newabc	Exit current ABCcom and restarts communication with new one.
Shl> ls -ltr	Execute Shell command from ABCcom source directory Display ABCcom Shell answer

B.2. Commands to Teleset

User entry examples	*Command description*
Tel> ncmd 3	Next message to ABCcom will have 3 command sub-packets. Not valid for 'tel' commands
Tel> tel defabc 2 3 0 a	Define ABC sub array. 4 bytes, 1 bit / ABC. From left, ABC no = position (ex: ABC 9 is 0 2 0 0)
Tel> tel adduser 2	Add user sub array. (Master command) User number = user index (1...)
Tel> tel rmuser 3	Remove user sub array. (Master command) User number = user index (1...)
Tel> tel showabc	Show current ABC sub array
Tel> tel exit	Exit Teleset Master goes to Teleshell

B.3. Commands to ABC System

User entry examples	*Command description*
Tel> abc state	Ask ABC state
Tel> abc askparam	Ask ABC parameters
Tel> abc settime	Set ABC time and date >Enter time (format: Wed Nov 30 01:35:57 IST 2005)
Tel> abc exit 3	ABCcom number 3 exits
Tel> abc startlocal	Start local Teleset mode in ABCcom
Tel> abc stoplocal	Stop local Teleset mode in ABCcom
Tel> abc startexp 5	Start EXPert thread in ABCcom for MCM no 5
Tel> abc stopexp	Stop EXPert thread in ABCcom

B.4. Commands to LO System

User entry examples	*Command description*
Tel> lo state	Ask LO state
Tel> lo askparam	Ask LO parameters
Tel> lo reboot	Reboot LO
Tel> lo setnew set1.dat	Set LO with set1.dat file parameters (setting parameters stored)
Tel> lo doset	Set LO with last setting parameters
Tel> lo domon	Monitor LO system, decode and prepare error message for state answer System must be already set successfully
Tel> lo autotime 15	Set 15 secs between automatic operation (0 for no operation)
Tel> lo setaage	Do LO setaage
Tel> lo setpiche	Do LO setpiche

B.5. Commands to FE System

User entry examples	*Command description*
Tel> fe state	Ask FE state
Tel> fe askparam	Ask FE parameters
Tel> fe reboot	Reboot FE
Tel> fe setnew set1.dat	Set FE with set1.dat file parameters (setting parameters stored)
Tel> fe doset	Set FE with last setting parameters
Tel> fe domon	Monitor FE system, decode and prepare error message for state answer System must be already set successfully
Tel> fe autotime 20	Set 20 secs between automatic operation (0 for no operation)
Tel> fe seton	Set MCM 5 on
Tel> fe setoff	Set MCM 5 off
Tel> fe setNGon	Set Noise Generation on
Tel> fe setNGoff	Set Noise Generation off
Tel> fe setNG50	Set Noise Generation on at 50%
Tel> fe setNG25	Set Noise Generation on at 25%

B.6. Commands to IF System

User entry examples	*Command description*
Tel> if state	Ask IF state
Tel> if askparam	Ask IF parameters
Tel> if reboot	Reboot IF
Tel> if setnew set1.dat	Set IF with set1.dat file parameters (setting parameters stored)
Tel> if doset	Set IF with last setting parameters
Tel> if domon	Monitor IF system, decode and prepare error message for state answer System must be already set successfully
Tel> if autotime 30	Set 30 secs between automatic operation (0 for no operation)
Tel> if loadGA ifload_2.dat	Enter loading parameters (post gain and pre attenuation) for ABCcom 2, from ifload_2.dat file
Tel> if setGA	>Enter Ch1PA, Ch1PG, Ch2PA, Ch2PG Set IF post gain and pre attenuation ch1 & 2 (setting parameters stored)

B.7. Commands to FPS System

User entry examples	*Command description*		
Tel> fps state	Ask FPS state		
Tel> fps askparam	Ask FPS parameters		
Tel> fps init	Init FPS (read version, set min/max angles, set lower rpm)		
Tel> fps loadcnt fpspos.dat 2	Enter loading parameters (encoder position for 4 feed positions) for ABCcom 2, from fpspos.dat		
Tel> fps mvpos feed	Run to preset at position loaded for feed		
Tel> fps autotime 10	Set 10 secs between automatic operation (0 for no operation)		
Tel> fps cmd	Number	Function	Arguments
	1	Null cmd	
	2	Set Turning point	Enc count (2b)
	3	Set Ramp down cnt	Nb timer ticks (1b)
	4	Set Lower RPM limit	Nb timer ticks (1b)
	5	Set Brake down diff.	Nb enc. pulses (1b)
	6	Set Ramp up cnt	Nb timer ticks (1b)
	7	Set Stop time cnt.	Nb timer ticks (1b)
	8	Set Max PWM cnt	PWM cnt (1b)
	9	Set Max angle	Angle (2b)
	10	Set Min angle	Angle (2b)
	11	Read Turning point	
	12	Read Ramp down cnt	
	13	Read Lower RPM limit	
	14	Read Brake down diff.	
	15	Read Ramp up cnt	
	16	Read Stop time cnt	
	17	Read Max PWM cnt	
	18	Read Max angle	
	19	Read Min angle	
	20	Read Version	
	21	Run to calibrate	
	22	Free run towards	270/-10 deg (1b)
	23	Run to preset	target pos (2b)
	24	Run to fine tune	target pos, PWM cnt (3b)
	25	Password Run	
	26	Reboot	
	27	Stop	

B.8. Commands to Servo System

User entry examples	*Command description*
Tel> sv state	Ask Servo state
Tel> sv askparam	Ask Servo tracking parameters
Tel> sv trkon track.dat	Start tracking mode with track.dat file parameters (setting parameters stored)
Tel> sv trkoff	Stop tracking mode
Tel> sv trkset trkset.dat 2	Enter loading parameters for ABCcom 2, from trkset.dat file
Tel> sv trkaz in	Set Azimuth out or in
Tel> sv trkrate	>Enter RA rate ('/m), DEC rate ('/m), Time ref t0 (hh:mm:ss)
Tel> sv cmd	Code Function Arguments 1. Operational Cmds 40 Coldstart 42 Position ax ang1 [ang2] 44 Track ax time ang1 [ang2] 46 Hold ax 48 Stop ax 4a Close 4c Stow ax 4e Stow release ax 50 Abort 6c Reset HW 2. Display Cmds 30 Read Angles 32 Read Anl Vars 34 Read Dig Vars 36 Read setparameters implemented but 38 Read Antenna status .. 3a Read Version .. #3c Read Error status not in GetData loop ->specific cmd 3. Set Mode Cmds 52 Set time day time date 54 Set stow angles ang1 (E) 56 Set S/W Hi limit ax ang1 [ang2] //1st A, 2nd E 58 Set S/W Lo limit ax ang1 [ang2] 5a Set windvel limit windvel #5c Set current ? not implemented #5e Set speed ? " "

B.9. Commands to EXP System

User entry examples	*Command description*		
Tel> exp state	Ask EXP state		
Tel> exp cmd	Number	Function	Arguments (nb bytes)
	1	Null cmd	
	2	Set Idle Mode	
	3	Set Scan Mode	
	4	Set Mean Mode	1
	5	Set Anl Mask	8
	6	Set Dig Mask 16b	2
	7	Set Dig Mask 32b	4
	8	Set Dig Mask 64b	8
	9	Read Anl Mask	
	10	Read Dig Mask 16b	
	11	Read Dig Mask 32b	
	12	Read Dig Mask 64b	
	13	Read Version	
	14	Read Mode	
	15	Reboot	
	16	Feed Mcm (old)	1
	17	Feed Mcm (new)	2
	18	Fe Box monitor	1
	19	Common Box monitor	

B.10. File References

- **fpspos.dat**

```
#########################################################################
# fpspos.dat
#
# for Teleset program
#
# file to load FPS feed encoder position
#
# columns:
# abc feed610 feed150 feed1420 feed325
#
# L.Pommier, 20/12/2005
#########################################################################

2 1203 6362 11443 16563
9 1313 6433 11553 16673
1 1456 6576 11696 16816
```

- **set1.dat**

```
########################################################################
# set1.dat
#
# for Teleset program
#
# file to send LO, IF and FE setting parameters
#
# L.Pommier, 20/12/2005
########################################################################


# ********************** LO system *********************************
# freq1 freq2
#
255 255

# ********************** FE system *********************************
# NG-cycle Wash-enabl Wash-group
# freqobs1 freqobs2(0 if not dual)
# (50/150/235/325/610/1060/1170/1280/1390/1420 MHz.)
# solar-att1 solar-att2(0 if not dual), -1 FE terminaison
# pol-swap noise-cal(-1 RF off)
#
0 0 0
235 0
-1 14
1 3

# ********************** IF system *********************************
# (ch1) bw (ch2) bw
# (ch1) alc (ch2) alc
#
16 32
1 1
```

- **track.dat**

```
########################################################################
# track.dat
#
# for Teleset program
#
# file to send Servo tracking parameters
#
# source name, Right Ascension, Declinaison
#
# L.Pommier, 20/12/2005
########################################################################

source  13h31m23.19s  +30d28'41.8"
```

- **trkset.dat**

```
#####################################################################
# trkset.dat
#
# for Teleset program
#
# file to load Servo parameters
# ABC 2 (->filename), 8x11 matrix
#
# columns:
# abc, Azimuth offset, Elevation offset, ant latitude, ant longitude
#
# L.Pommier, 20/12/2005
#####################################################################

1 +001:00:00 -001:00:00 19.0927 -74.0536
2 +002:00:00 -002:00:00 19.0927 -74.0536
3 +003:00:00 -003:00:00 19.0927 -74.0536
4 +004:00:00 -004:00:00 19.0927 -74.0536
5 +005:00:00 -005:00:00 19.0927 -74.0536
6 +006:00:00 -006:00:00 19.0927 -74.0536
```

- **ifload_2.dat**

```
#####################################################################
# ifload_2.dat
#
# for Teleset program
#
# file to load IF attenuations and gains
# ABC 2 (->filename), 8x11 matrix
#
# columns:
#       ALC ON          ALC OFF
#    ch1     ch2     ch1     ch2
#  Pa  Pg  Pa  Pg  Pa  Pg  Pa  Pg
#
# lines:
# 150, 235, 325, 610, 1060, 1170, 1280, 1390, 1420, 235-610, 610-235
#
# L.Pommier, 20/12/2005
#####################################################################

Abc 2
14.5 14.5 12 10 16 16 14 12
12 10 14 12 14 12 16 14
14 14 16 16 16 16 18 18
10 08 12 12 12 10 14 14
12 12 12 12 14 14 14 14
12 10 12 10 14 12 14 12
08 08 08 08 10 10 10 10
06 06 06 06 8 8 8 8
08 06 08 06 10 08 10 08
12  14 12  12 14 16 14 14
12  14 12  12 14 16 14 14
```

B.11. Resume table of Teleset commands

To Teleset	***To AbcCom***	***To Servo***			
• Defabc • Showabc • Noans • Ncmd • Exit • Adduser • Rmuser	• State • Askparam • Settime • Exit • Startlocal • Stoplocal • StartExpert • StopExpert	• State • Askparam • Trkon • Trkoff • Trkset • Trkaz • Trkrate	*Direct Commands:*		
		• Coldstart • Position • Track • Hold • Stop	• Close • Stow • Stow release • Abort • Reset HW	• Read Angles • Read Anl Vars • Read Dig Vars • Read setparameters • Read Antenna status	• Read Error status • Set time day • Set stow angles • Set S/W Hi limit • Set S/W Lo limit • Set windvel limit
To LO / IF / FE	***only to LO***	***To FPS***			
• State • Askparam • Reboot • Setnew • Doset • Domon • Montime	• Setaage • Setpiche	• State • Askparam • Init • Loadcounts • Mvpos	*Direct Commands:*		
		• Null cmd • Set turning point • Set ramp down cnt • Set lower rpm limit • Set brake down diff • Set ramp up cnt • Set stop time cnt	• Set max PWM cnt • Set max angle • Set min angle • Read turning point • Read ramp down cnt • Read lower rpm limit • Read brake cnt diff	• Read ramp up cnt • Read stop time cnt • Read max PWM cnt • Read max angle • Read min angle • Run to calibrate	• Free run • Run to preset • Run to fine tune • Password run • Reboot • Stop
only to IF	***only to FE***	***To Expert***			
• LoadGA • SetGa	• Seton • Setoff • SetNGon • SetNGoff • SetNG50 • SetNG25	• State • Askparam	*Direct Commands:*		
		• Set MCM Address • Null cmd • Set Idle Mode • Set Scan Mode • Set Mean Mode	• Set Anl Mask • Set Dig Mask 16b • Set Dig Mask 32b • Set Dig Mask 64b • Read Anl Mask	• Read Dig Mask 16b • Read Dig Mask 32b • Read Dig Mask 64b • Read Version • Read Mode	• Reboot • Feed Mcm (old) • Feed Mcm (new) • Fe Box monitor • Common Box monitor

Appendix C: Tables of Teleset file contents

- C++ files for general classes:

Filename	*Included headers*	*Classes*	*Description*
mainteleset (.cpp, .h)	sharemem.h, highuser.h, lowuser.h	TeleUser	Main Teleset object, Master / Secondary user selection
teleshell (.cpp, .h)	common.h	TeleShell	Mode for Remote Linux shell and files transfer
common (.cpp, .h)	none	PORT, MSG (C struct), LogFile	Basic C structures and classes common to all files
sharemem (.cpp, .h)	common.h	BaseFILO, UserShare, ArrayDef, SetData, SysBaseState, SystmState, LoState, FeState, IfState, FpsState, ExpState, AbcState, SscAngle, SscAnlvar, SscDigvar, SscMessage, SscAns, ListState, SysBlock, ServoBlock, AntFile, MonData, ShareMemory	Shared memory classes and sub-classes

- C++ files for HighUser classes:

Filename	*Included headers*	*Classes*	*Description*
highuser (.cpp, .h)	common.h, sharemem.h, cmdbase.h, cmdsystm.h	HighUser	Interface for operator commands
cmdbase (.cpp, .h)	common.h, sharemem.h	CmdBase, CmdSsc, CmdAbc, CmdExp, CmdFps	Basic command utilities and command classes
cmdsystm (.cpp, .h)	common.h, sharemem.h, cmdbase.h	CmdSys, CmdFe, CmdLo, CmdIf	Parent and command classes for FE / IF / LO

- C++ files for LowUser classes:

Filename	*Included headers*	*Classes*	*Description*
lowuser (.cpp, .h)	sharemem.h, decode.h, decssc.h, decexp.h, decfps.h, decif.h	Decode, LowUser, LowUserThread, MasterLUser	ABCcom Link configuration, transfer of commands and decode answers
decode (.cpp, .h)	sharemem.h	RxPop, DecBase, DecAbc, DecSystm, DecLo, DecFe	Basic utilities to decode, Decode system sub-packets
decssc (.cpp, .h)	decode.h, sharemem.h	DecSsc	Decode SSC answer sub-packets
decexp (.cpp, .h)	decode.h, sharemem.h	CommandPlus, DecodeMCM, DecExp	Decode EXP answer sub-packets
decfps (.cpp, .h)	decode.h, sharemem.h	DecFps	Decode FPS answer sub-packets
decif (.cpp, .h)	decode.h, sharemem.h	DecIf	Decode IF answer sub-packets

- C files for serial port communication:

Filename	*Included headers*	*Description*
comhlink (.c, .h)	none	Serial port communication with Teleset for ABC mode C open/write/read/close functions (RS-232, poll and master/slave)

Appendix D: Teleset history files

- 29Dec14h_HUser0_rec
-

Appendix E: Teleset source code

- ***Makefile*** for Teleset
- Teleset code files:

cmdbase.cpp	***common.cpp***	***decif.cpp***	***highuser.cpp***	***sharemem.cpp***
cmdbase.h	***common.h***	***decif.h***	***highuser.h***	***sharemem.h***
cmdsystm.cpp	***decexp.cpp***	***decode.cpp***	***lowuser.cpp***	***teleshell.cpp***
cmdsystm.h	***decexp.h***	***decode.h***	***lowuser.h***	***teleshell.h***
comhlink.c	***decfps.cpp***	***decssc.cpp***	***mainteleset.cpp***	
comhlink.h	***decfps.h***	***decssc.h***	***mainteleset.h***	