



Atacama Large Millimeter Array

ALMA-SW-NNNN
Revision:1
2002-02-21
*Software-Interface
Description*
Dirk Muders

Software Interfaces for Submillimeter Telescope Instrumentation

Software-Interface Description

D. Muders, J. Hatchell, R. Lemke, M. Olberg, H. Hafok

Keywords: Submm Instruments, Control Commands, APEX	
Author Signature: Dirk Muders	Date: 2002-02-21
Approved by:	Signature:
Institute:	Date:
Released by:	Signature:
Institute:	Date:

Contents

1	Introduction	3
2	Frontends	4
2.1	High-level Control Commands	5
2.1.1	Startup, initialisation and shutdown	5
2.1.2	Configuration	5
2.1.3	Timing and synchronisation	6
2.1.4	Observing	6
2.2	Low-level Control Commands	6
2.3	Error interrupts	8
3	Backends	8
3.1	High-level Control Commands	8
3.1.1	Startup and shutdown	8
3.1.2	Configuration	8
3.1.3	Timing and synchronisation	9
3.1.4	Observing	9
3.2	Low-level Control Commands	10
3.3	Error interrupts	10
4	Intermediate frequencies	10
4.1	High-level Control Commands	10
4.2	Low-level Control Commands	11
4.3	Error interrupts	11
5	Timing, synchronisation, and real-time updates	11
5.1	High level commands	12
5.1.1	Configuration	12
5.2	Error interrupts	12
5.3	Real time information	12
A	Existing Implementations	14

1 Introduction

As part of the APEX telescope development, there is an immediate need to implement APEX specific frontends and backends in the telescope control software. For APEX we will use the ALMA Test Interferometer Control Software (TICS) and the ALMA Common Software (ACS). New devices will be implemented using the "ticsDevice" class which extends the ACS Distributed Object (DO). This is the standard technique used in TICS and allows to access the devices throughout the whole APEX control system, especially by TICS, but also for monitoring purposes and maintenance.

We draw on experience with existing radio and mm/submm telescopes, where there are many frontends and backends in use, to compile a list of generic control commands. For APEX, these generic commands will provide a common interface to the various existing instruments. The generic commands will be translated into the specific format required by individual instruments in software invisible to the user.

We intend to make the list generic so that all or parts of it can be used by other observatories. For the design of the control interface of further instrumentation, the command formats listed here can be implemented directly. In that case one would also need to specify a concrete hardware interface, eg. the CAN Bus and an implementation. This could be done in a future revision of this document.

In compiling these commands, we have looked in detail at the control commands of instruments at the Effelsberg, IRAM, HHT, (SEST), & KOSMA telescopes, as well as aspects of the proposed systems for ALMA and JCMT, as follows (the detailed implementations of these instruments can be found in Appendix A):

MPIfR Instruments:

- CHAMP (460/490 GHz 16 element focal plane heterodyne array, currently at the CSO)
- MACS 32 × 1 GHz autocorrelator (CHAMP's backend, currently at the CSO)
- MAMBO / SIMBA (bolometer arrays at the IRAM 30m and the SEST; (almost ?) no software interface necessary; future bolometers using SQUIDs will need some frontend control software)
- Bogle (continuum backend; one at the IRAM 30m, one at the SEST)
- ABBA (continuum backend; at the IRAM 30m)

SEST Instruments: (no details available yet, but expect few changes)

- SEST 1mm/3mm heterodyne receiver
- SEST autocorrelator

KOSMA Instruments:

- Array receiver
- KOSMA array AOS

Logically, we have separated the hardware into four sections. We begin with control of frontends (FE), ie. detectors / receivers (Sect. 2) and backends (BE), ie. spectrometers and continuum backends (Sect. 3).

We have separated out the intermediate frequency (IF) chain ie. the stages required to match the IF from the first, cooled mixer to the input of the spectrometer. Physically, this may lie partly in the frontend or backend system or it may be a completely separated system. Control commands for the IF are in Sect. 4.

Finally, we consider timing & synchronisation (TS) to be generated in a central device in order to allow for multiple FE or BE during one observation. This is described in Sect. 5.

This logical partitioning leads to 4 abstract base classes which can then be used to derive actual implementations for the various instruments. This encapsulates the existing interfaces and provides common hardware access methods.

Throughout, we have separated commands into the high-level (user) commands necessary to set up and perform observations and low-level (technical) commands for internal procedures like the details of tuning and for maintenance. For the high-level commands, we have listed as many commands as possible, even though not every command is needed by every instrument.

The low-level commands access hardware devices directly. The high-level commands can be implemented via the TICS composite device concept which groups several low-level devices together to make a new device. An example would be a receiver where the observer is mainly interested in tuning the receiver to a given frequency. This, however, involves many settings for low-level devices but the intelligence (which sequence of actions, feedback, etc.) needs to be coded on an intermediate level.

The technical commands tend to be more instrument-specific and here we have included commands that apply to more than one instrument, rather than a complete list. Many current instruments do not make their low level commands available beyond a terminal directly attached to the instrument. The need to operate APEX remotely as much as possible makes it important that as many procedures as possible can be accessed without physically being present.

For each operation we list the command, a brief description, and the response that would be expected back from the instrument.

Commands can also be divided into composite commands (likely referring to composite devices), made up of other commands, and atomic commands, which have no reference to other commands. Composite commands are marked (composite).

Many commands are of the "set/get" type which set something in the hardware (eg. a voltage) or read back the current value. This is implemented in "ticsDevice" using ACS properties. There is a read-only (RO) property that is used to get the current value and a read-write (RW) property to set a new value. Reading back the latter returns the desired value while monitoring the RO property allows checking the progress (for long-running commands).

All commands can return with 'fail' plus an error diagnostic, if they were unsuccessful, even where this is not listed explicitly. Also in addition to the listed responses, the return can be 'command does not exist' or 'set input manually'. The 'command does not exist' response is for commands that are not implemented on that particular instrument. 'set input manually' is necessary for occasions where adjustments have to be made directly in hardware eg. the tuning of some receivers.

The command feedback will be implemented either via ACS callbacks (non-blocking commands with asynchronous return) or via blocking (synchronous) ACS method calls. Both methods are automatically available for "ticsDevices" or "ACS DOs". A decision on which to use will be made individually according to the needs.

We also include unsolicited error interrupts - messages received from the instruments by the control software without being prompted by a command. These interrupts will be implemented via the ACS alarms that can be attached to any ACS DO property. An alarm is triggered and passed up to the application (TICS) once the property is outside a predefined value range.

2 Frontends

Frontends can be categorised as continuum/spectral line and single pixel/array receivers, but all have some kind of frequency selectivity (whether local oscillators for spectral line receivers, or filters for bolometers), and calibration elements. Many control commands are common to all types, though spectral line receivers and array receivers have more commands due to their added complexity.

2.1 High-level Control Commands

2.1.1 Startup, initialisation and shutdown

turn on Turn on any parts of the instrument for which power is under software control, and initialise by setting defaults. (Composite).

Return: success / fail

turn off Turn off any parts of the instrument for which power is under software control. (Composite).

Return: success / fail

reset hardware reset (restart system) and set defaults. (Composite).

Return: success / fail

2.1.2 Configuration

set/get defaults set: run through some of the following configuration commands and set to default values using a database which stores default values for each instrument. get: report defaults from the database. (Composite).

Return: success / fail

report configuration Run through the following commands in 'get' mode to generate a list of the current configuration of the receiver. (Composite).

Return: current configuration

set/get frequency Set the observing frequency in preparation for tuning.

Return: frequency

set/get sideband Observing frequency in upper or lower sideband (spectral line only).

Return: sideband

set/get single/dual sideband Sideband suppression (spectral line only).

Return: single/dual sideband, suppression / dB

set/get polarisations Set the polarisation(s) to use (where available).

Return: list of polarisations in use

set/get elements Which elements or pixels of an array receiver to use (array receivers only).

Return: list of elements in use

set/get image/dewar rotation Orientation of array beam pattern on the sky (array receivers only).

Return: rotation angle / degrees. There should also be an online display of the dewar rotation in astronomical coordinates provided at a higher level (the receiver software has no knowledge of astronomical frames).

set/get image/dewar tracking (array receivers only).

Return: dewar tracking on/off

tune receiver The implementation of this command will vary from receiver to receiver. In some cases it will consist of a series of low level commands (setting the mixer bias, backshort position etc.) and be a composite command. In others, the command will be dealt with directly by the receiver. In some receivers, tuning may have to be carried out by hand. (can be Composite).

Return: progress messages, success / fail and error diagnostic.

2.1.3 Timing and synchronisation

We are assuming for now that synchronisation and blanking signals are being supplied by a central timing generator (see Section 5).

frequency switch on/off Tell receiver to frequency switch using synchronisation/blanking signal from central timing generator.

Return: success / fail

2.1.4 Observing

start observation Synchronise with central timing generator and start observing routine.

Return: success / fail, error diagnostic

status Status of observation / instrument - what it is currently doing.

Return: status information

get receiver temperature Report the hot and cold load temperatures and calculate the receiver temperature. (Composite).

Return: hot load, cold load and receiver temperature / K

2.2 Low-level Control Commands

Many of these low-level receiver commands are involved in tuning the receiver. The receiver may handle this automatically, or the high-level 'tune receiver' command may in fact send several of these low-level commands. Where commands are marked 'tune', they invoke multistep procedures.

version get hardware/software version

Return: version information

set/get chopper mode Is the receiver looking at hot load, cold load, or sky?

Return: description of chopper position

get dewar temperature Report the dewar temperature.

Return: dewar temperature / K

get dewar pressure

Return: dewar pressure / Pa

set/get FE LO synthesiser frequencies frequency for LO phase locking (typically 10 GHz range).

Return: frequencies / MHz

set/get Gunn frequency

Return: Gunn frequency

set/get Gunn bias

Return: Gunn bias / V

set/get multiplier bias

Return: multiplier bias / V

tune/set/get mixer bias

Return: mixer bias / V

set/get backshort position

Return: backshort position

tune/set/get magnetic field Tuning the magnetic field is a composite procedure requiring one or more magnetic field scans followed by calculation and setting of the optimum magnetic field. (Composite).

Return: magnetic field setting / volts

measure magnetic field scan Can scan voltage upwards or downwards to account for hysteresis.

Return: array of (V/volts, power/watts)

set/get PLL synthesiser frequency frequency for PLL (typically 100 MHz range). May be used for LSR correction.

Return: frequencies / MHz

lock PLL / get PLL lock status

Return: locked/not locked, PLL lock frequency, error diagnostic

set/get PLL harmonic

Return: harmonic number

set/get bias voltage

Return: bias voltage / mV

set/get bias current

Return: bias current / μ A

set/get magnetic field current

Return: magnetic field current / mA

measure IV curve

Return: array of (I/ μ A, V/mV)

measure conversion / total power curve

Return: array of (V/mV, TP/mV (?))

set/get first IF amplifier on/off

Return: on/off

set/get first IF amplifier current

Return: IF amplifier current / mA

set/get first IF amplifier bias

Return: IF amplifier bias / mV

set/get AC/DC mode (bolometers)

Return: AC or DC

set/get first IF attenuator

Return: IF attenuation / dB

set/get IF level

Return: IF level / W

2.3 Error interrupts

PLL out of lock

Dewar tracking failed

Dewar temperature too high

Device failure With an explanatory message.

3 Backends

Backends can be mainly categorised as digital autocorrelation spectrometers (DAS / correlators), acousto-optical spectrometers (AOS) and filterbanks. There are also more exotic devices such as Chirp Transform Spectrometers (CTS), which would however have very similar control commands to the ones compiled here.

3.1 High-level Control Commands

3.1.1 Startup and shutdown

turn on Turn on any parts of the instrument under software control, and initialise. (Composite).

Return: success / fail

turn off Turn off any parts of the instrument under software control. (Composite).

Return: success / fail

reset Hardware reset (restart system) (Composite).

Return: success / fail

3.1.2 Configuration

set/get defaults Run through some of the following configuration commands and set to default values, using a database of defaults stored for each instrument. (Composite).

Return: success / fail

report configuration Implemented in the control software. Run through the following commands in 'get' mode to generate a list of the current configuration of the receiver. (Composite).

Return: current configuration

set/get number of spectral channels Set the number of frequency channels in use.

Return: number of channels

set/get gain factors/attenuator

Return: gain factors / attenuation / dB

set/get file code / project number / scan number Information needed to generate filename for data recording. This should be generated automatically in software according to some system, but may require some input from the user (eg. project no.).

Return: filename

set/get data format description Set is required for spectrometers that can output in more than one format.

Return: A brief description of how the data is being stored

set/get frequency resolution (for autocorrelators eg. MACS).

Return: frequency resolution / kHz

3.1.3 Timing and synchronisation

We are assuming for now synchronisation and blanking signals are being supplied by a central timing generator (see Section 5).

set/get phase time Tell the backend what phase time to expect from the central timing generator, if it can't deduce this from the blanking signal.

Return: phase time expected by backend

set/get blanking time Tell the backend what blanking time to expect from the central timing generator (can be variable), if it can't deduce this from the blanking signal.

Return: blanking time expected by backend

set/get no. of phases Tell the backend how many phases per cycle (per synchronisation) to expect from the central timing generator, if it can't deduce this from the synchronisation and blanking signals.

Return: no. of phases

set/get timestamp interval How often to add a timestamp to the output files. This is needed for on-the-fly/scan mapping, where integrations are dumped at short intervals as the telescope tracks across the source.

Return: timestamp interval / ms

3.1.4 Observing

start Synchronise with signal from central timing generator, and start taking data.

Return: success / fail, progress report, error diagnostic

stop Stop taking data. If possible, finish the observing cycle (this will depend on synchronisation with the telescope steering). Alternatively, set number of cycles before starting (below).

Return: success / fail

abort Abort taking data, even if partway through a scan.

Return: success / fail

set/get no. of cycles Alternative to stop. Define observing time in terms of synchronisation cycles. Observing then stops automatically.

Return: no. of cycles

transfer Write data to file. Should be automatic.

Return: filename

status Status of observation / instrument - what it is currently doing eg. taking data? How many cycles to go?

Return: status information

3.2 Low-level Control Commands

version get hardware/software version.

Return: version information

get dark current / zero offset

Return: dark current / counts

comb Generate a comb spectrum for frequency calibration (AOS).

Return: success / fail, type of comb (line frequencies?)

get noise diode Measure noise generator for test purposes.

Return: success / fail

stability test Start a test using the noise diode.

Return: success / fail, expected completion time, Allan variance data

get CCDs How many CCDs are attached ? (KOSMA)

Return: no. of CCDs

get sig/ref Is the sig/ref line high or low, ie. which memory bank is active ? (KOSMA)

Return: sig or ref

clear messages Clear the message buffer (BOGLE).

Return: succeed/fail

control graphics/plotter Needed where there is a configurable display or plotter output to the backend. This command breaks down into the lower level commands needed to control the graphical device.

Return: display/plotter configuration

3.3 Error interrupts

Device failure With descriptive message.

4 Intermediate frequencies

Following the initial conversion from (sub)mm to radio intermediate frequencies, further downmixing with further local oscillator signals may be required to reach the final frequencies and bandwidth required by the correlator for the bandwidth, resolution etc. required by the observer. This may take place in the receiver or in the correlator system (or hypothetically, separate to both). Therefore, control of this stage is listed separately.

4.1 High-level Control Commands

None

4.2 Low-level Control Commands

set/get IF synthesiser frequencies frequencies for IF band generation via further mixers / downconverters.

Return: frequencies / MHz

set/get IF filter(s) filters to determine IFs.

Return: IF filters in use

set/get IF bandwidth(s) IF bandwidths to use.

Return: IF bandwidths

set/get IF power levels Adjust power levels of the IF signal through the chain.

Return: IF power levels

set/get IF attenuation Adjust power levels of the IF signal through the chain.

Return: IF attenuation / dB

4.3 Error interrupts

Device failure With descriptive message.

5 Timing, synchronisation, and real-time updates

In addition to the commands received from the control system, timing and synchronisation signals are required in real time by frontends and backends. In current systems, these tend to consist of two signals: a synchronisation signal and a blanking signal. The blanking signal repeats periodically and triggers a new phase of the observations. The length of the blanking signal is adjusted to allow all the hardware to reach the desired observing configuration (eg. the chopping secondary to move, or frequency adjustments in the frontend) during the blanking time before data is recorded by the backend. The synchronisation signal repeats every n phases where n is the number of phases in an observing cycle.

In current systems, the sync. and blanking signals are sometimes generated by the FE and sometimes the BE, and sometimes by a separate device, and distributed via direct digital lines between instruments. We consider for the time being the timing signals to be generated by a separate device - the timing generator. This allows more than one receiver, or more than one backend, to be used synchronously for the same observations. In some configurations, the timing generator may physically lie in the backend or frontend.

Also listed at the end of this section is information required in real time by various subsystems, particularly the information necessary for the receivers to perform a LSR correction, and the current time, which in this context is primarily needed for timestamping the data.

Examples of operations which need to be coordinated between frontends/backends/antenna are:

Frequency switching

Wobbler switching / chopping

Polarisation switching eg. with rotating filters

Jiggle mapping When the secondary mirror traces a pattern, moving in azimuth and/or elevation, between integrations, to fully sample with undersampled arrays (eg. SCUBA at JCMT). Need to coordinate telescope moves with data taking.

5.1 High level commands

5.1.1 Configuration

set/get phase period The length of each observation phase, eg. one frequency of a frequency switched observation, the time spent on source or on sky in a wobbler switched observation.

Return: phase period / ms

set/get no. of phases number of phases in each cycle.

Return: no. of phases

get synchronisation period period of synchronisation signal, equal to phase length times no. of phases.

Return: synchronisation period

set/get blanking time Length of blanking signal. In some cases, eg. frequency switching, this could be variable in which case this input is not required but the following one is.

Return: blanking time

set/get variable blanking Use variable blanking signal based on input from another instrument. (NB: not all backends will be able to cope with variable blanking).

Return: yes if using variable blanking, no if not

5.2 Error interrupts

Device failure With descriptive message.

5.3 Real time information

Other real time information that must be distributed (via the CAN bus) to the instruments:

Time For timestamping, tracking etc.

Radial velocity For adjustments to sky frequency to compensate for the observer's velocity and observing reference frame.

Telescope position eg. for dewar rotation, if this is not implemented in hardware.

Error status

CHAMP IF control

For non-cooled IF conversion within the receiver.

Generic command	Translation to instrument format
set/get synthesiser frequencies	set IF synthesiser (channel no., frequency)
set/get IF filters	set IF filter (channel no., bandwidth)
set/get IF bandwidths	–
set/get IF power levels	–
set/get IF attenuation	–

KOSMA

Generic command	Translation to instrument format
Startup/shutdown turn on turn off reset	switch Gunn on switch Gunn off reset receiver
Configuration defaults report configuration set/get frequency set/get sideband set/get single/dual sideband set/get polarisations set/get elements set/get image/dewar rotation set/get image/dewar tracking tune receiver	– get config. info. set/get LO frequency set/get synthesiser freq. set/get PLL sideband – – – set/get image rotation – tune receiver
Timing and synchronisation frequency switch on/off	–
Observing start observation status get receiver temperature	– get status info. get temperature_1,2,3
Low-level control commands version set/get chopper mode get dewar temperature get dewar pressure set/get Gunn frequency set/get Gunn bias set/get multiplier bias tune/set/get mixer bias set/get backshort position tune/reset/get magnetic field lock PLL/get PLL lock status set/get PLL harmonic set/get bias voltage set/get bias current set/get magnetic field current measure IV curve measure conversion/total power curve measure magnetic field scan get first IF amplifier I/V set/get IF attenuator get IF level set/get first IF amplifier on/off	– – – get dewar pressure – set/get Gunn bias – tune/reset mixer bias – tune/reset/get magnetic field lock PLL/get PLL lock status set/get PLL harmonic set/get bias voltage set/get bias current set/get magnetic field current measure IV curve measure conversion curve measure magnetic field scan – set/measure IF power level – –

Effelsberg IF

ULO settings (details??)

Generic command	Translation to instrument format
set/get synthesiser frequencies	–
set/get IF filters	–
set/get IF bandwidths	–
set/get IF power levels	–
set/get IF attenuation	–

Effelsberg timing

Generic command	Translation to instrument format
set/get phase period	phase time in ms (Takt)
set/get no. of phases	number of phases
get synchronisation period	–
set/get blanking time	blanking time /micros*10
set/get variable blanking	–

Backends

ABBA

ABBA configuration parameters are mostly issued as an argument to the wb / fs / tp commands (wobbler switch, fast scanning, total power (skydip) modes) - where so, these are marked wb/fs/tp as appropriate.

Generic command	Translation to instrument format
Startup and shutdown	
turn on	–
turn off	–
reset	–
Configuration	
defaults	–
report configuration	(run get commands in software)
set number of channels	–
set gain factors / attenuator	gain factor (wb/fs/tp)
set file code / project number/ scan/subscan no. (wb/fs/tp)	file code
Timing and synchronisation	
set/get phase time	wobbler period = $2 \times$ phase time (wb/tp)
	–
set/get blanking time	blanking time / ms (fs/wb/tp)
	–
set/get no. of phases	2 for fs or wb, 1 for tp
set/get timestamp interval	(determined by wb/tp or fs)
	–
set/get data format description	(check mode wb/tp or fs)
set/get frequency resolution	–
Observing	
start	wb, fs or tp command
stop	stop
abort	–
set/get no. of cycles	–
transfer	(by ftp)
status	–
Low-level control commands	
control graphics/plotter	–
get dark current / zero offset	–
comb	–
get noise diode	–
stability test	–
get CCDs	–
get sig/ref	–
clear messages	–

ALMA

Software Interfaces for Submillimeter Telescope Instrumentation

KOSMA

Based on commands from the AOS library (direct control) rather than one of the overlaid software implementations.

Generic command	Translation to instrument format
Startup and shutdown turn on turn off reset	aos_hardware_configuration – aos_reset
Configuration defaults report configuration set number of channels set gain factors / attenuator set file code / project number / scan / subscan no.	aos_init – aos_set_channels previous setting – – –
Timing and synchronisation set/get phase time set/get blanking time set/get no. of phases set/get timestamp interval set/get data format description set/get frequency resolution	totp_scans_per_switchp in switchp mode blank_scans in aos_start_switchp 1 for totp, 2 for switchp – – –
Observing start stop abort set/get no. of cycles transfer status	aos_start_total_power aos_start_switchp – aos_abort scans in aos_start_total_power switchp_cycles in aos_start_switchp aos_get_bank aos_define_range aos_get_range aos_status aos_switchp_status aos_totalp_status continuous_loops_done aos_busy
Low-level control commands & version control graphics/plotter get dark current / zero offset comb get noise diode stability test get CCDs get sig/ref clear messages	aos_get_mon_version aos_get_lib_version – – – – – get_CCDs aos_get_sr_line –

Many returns are done via messages (with adjustable verbose level) - see aos_verbose_handler, aos_verbose_level

Other KOSMA AOS library features: error handling (via error number and error message).

BOGLE

BOGLE is the continuum backend at the 30m/SEST.

Generic command	Translation to instrument format
Startup and shutdown turn on turn off reset	– – reset
Configuration defaults report configuration set number of channels set gain factors / attenuator set file code / project number/ scan /subscan number set/get data format description set/get frequency resolution	– – configure:nchans – – Configure:Tmode ascii/binary header on, longform on header?, longform? –
Timing and synchronisation set/get phase time set/get blanking time set/get no. of phases set/get timestamp interval	– configure:nphases –
Observing start stop abort set/get no. of cycles transfer status report	measure start – – configure:nblocks transfer status?
Low-level control commands version control graphics/plotter get dark current / zero offset comb get noise diode stability test get CCDs get sig/ref clear messages	revision? configure:analog:F1-F4 configure:analog:PEN(1)–PEN(4) – – – – – – clrmessage

MACS / MACSE

CSO autocorrelator and Effelsberg version

Generic command	Translation to instrument format
Startup and shutdown turn on turn off reset	start stop –
Configuration defaults report configuration set/get number of spectral channels set/get gain factors / attenuator set/get file code / project number/ scan /subscan number set/get data format description set/get frequency resolution	sdef – mode x – – – mode x
Timing and synchronisation set/get phase time set/get blanking time set/get no. of phases set/get timestamp interval	sphase sblank setnop –
Observing start stop abort set/get no. of cycles transfer status	stopbl 0 stopbl 1 – – – –
Low-level control commands version control graphics/plotter get dark current / zero offset comb get noise diode stability test get CCDs get sig/ref clear messages	– – – – – – – – –
simulate on-target switch sampler control store total power offsets	otsim x ssc x sttpos

MACS timing

Generic command	Translation to instrument format
set/get phase period	sphase
set/get no. of phases	setnop
get synchronisation period	–
set/get blanking time	setblank
set/get variable blanking	swabl x