

The GXSM-3.0 manual



Percy Zahl, Andreas Klust, Stefan Schröder, Thorsten Wagner
and more <http://gxsm.sf.net>

May 22, 2022

Preface to the GXSM-3.0 manual

The GXSM-3.0 history and very early predecessors are dating back to 1995. It came a long way even lived on different operating systems with initial DSP code fragments in C, Turbo-Pascal and ideas I have to give credit to Gerhard Meyer to get me started with the very very early stages of my experiences with STM control. It evolved and diverted quickly into it's very own system from then on. Other branches related to work of Ullrich Köhler aka PM-STM had some influences and eventually a PM-STM operating on OS/2 was a live. With that operating system phasing out a move to Linux was undertaken and some software named "Xxsm" based on the X-Forms GUI was evolving and turned pretty mature by end of 2000 with a big move over to the early Gtk toolkit and Gxsm got established. From there a long and steady journey was up on it. With more and more multi-core/threading machine little related pit-falls evolved and were hard to find and even harder to fix – if not even unfixable without major reworking. But now in 2017 a huge move and undertaking whopped that grown up Gxsm2 over into what now is the all new Gxsm3 with a rewritten GUI at all levels and logics behind. Lasted big hit is the reimplementation of the OpenGL based 3D/Volume data view to inspect and visualize up to 5D data sets in real-time and dynamically unleashing the full power of a gaming grade GPU. GL-4.0 essential.

For historical reasons the first section of the following preface will remain in German.

Das Jahr 2000 ist vorraussichtlich mein letztes am Institut für Festkörperphysik der Universität Hannover. In diesem Vorwort möchte ich die Historie der ehemals ‘nur’ STM-Software festhalten. Ich habe die letzten fünf Jahre, die Zeit vor meiner Diplomarbeit eingeschlossen, dort verbracht und in der Arbeitsgruppe von Michael Horn-von-Hoegen begonnen, mich mit der Tunnelmikroskopie und den zugehörigen Techniken zu beschäftigen. Im Sommer 1995 zeigte mir A.Meier ein kleines, noch unberührtes, unter einer Schutzhülle verborgenes Gerät – ein Micro STM. Wir besaßen nur dieses Gerät zusammen mit einem Tunnelstromverstärker, einer Signalprozessor gesteuerte Meßkarte (PC31) – und – dem Wissen, daß damit ein gewisser G.Meyer in Berlin erfolgreich ein Gerät gleichen Prinzips bedient. Ich machte es mir zur Aufgabe, der ganzen Sache Leben einzuhauen ...

Von G.Meyer hatte ich unterdessen einige Software zusammengesammelt und wir versuchten, die diversen DOS/Pascal Programme auf unserem hochmodernen P90 zum Laufen zu bringen.

Währenddessen vertiefte ich mich in die Geheimnisse der DSP-Programmierung und der Kommunikation zwischen Host-PC und DSP.

Des Fortschritts wegen beschloß ich, ein bereits vorhandenes OS/2-Programm namens PMSTM weiter zu entwickeln. PMSTM basierte auf einer schon erheblich älteren OS/2 Version von L.Anderson und wurde von H.Bethge zum Messen eingesetzt.

Gleichzeitig produzierte R.Kumpe einige Assemblerroutinen zur Ansteuerung seiner Data-translation-Karte unter OS/2 – welche ebenfalls einen spezial STM-Eigenbau, jedoch noch mit analogem Regler, bedienen sollte.

Parallel wurden mit H.Pietsch die notwendigen Zusätze für AFM implementiert.

Es verging einige Zeit, aber dann war es soweit: Die neue OS/2 Software konnte das Gerät steuern und Daten aufnehmen und anzeigen. An Luft wurden erste Versuche mit Goldproben und abgeknipster Wolframspitze erfolgreich abgeschlossen. Weitere Verbesserungen in nahezu jeder Hinsicht konnte ich im Verlauf meiner Diplomarbeit einbringen – das Gerät war unterdessen im UHV der Maschine "Quantum" im Einsatz.

Die Geschichte des Micro STM's wurde im weiteren wesentlich von R.Hild bestimmt, der das höchst empfindliche Gerät in einem an Federn aufgehängten Kupferblock mit 3D-Wirbelstromdämpfung versenkte und somit zu Höchstleistung brachte.

Die Zeit war gegen das wunderbar stabile OS/2, es wurde zum Außenseiter und eigentlich benötigte man es nur noch zum Messen ...

Windows erschien mir als durchaus erfahrenem und gebräuchlichem DOS/Win3.X Programmierer als völlig ungeeignete Platform, so hatte ich doch die Vorteile einer stabilen Betriebssystemplatform mit OS/2 zu schätzen gelernt. Ich konnte jedoch mit meinen Unix/Linux Grundkenntnissen schnell eine zukunftssichere Alternative finden und entschied, erste Versuche mit einer neuen Software für mein SPA-LEED zu unternehmen. Das SPA-LEED sollte nämlich ebenfalls mit einer Signalprozessorkarte gesteuert werden, um später ggf. ohne zusätzlichen Aufwand ein STM nachrüsten zu können.

Das Resultat war ein Programm namens xspa, welches unter Verwendung der xforms Library unter X11 lief.

Aus diesen Erfahrungen schöpfend entwickelte ich ein völlig neues Konzept für eine grundlegend neue Struktur eines neuen STM-Programmes – der alte Code war zu steif und beinhaltete viel zu viele globale Variablen. Auch das fixe Datenformat dat wurde als historisch abgelegt und es fand ein Übergang zu dem NetCDF-Format statt. Das neue Konzept ist objektorientiert und ermöglicht erstmals eine flexible Mehrkanal- Verwaltung und -Messung (dies wurde in PMSTM nur per Trick in unflexibler Weise zur Datenaufnahme hineigebastelt, da das AFM die gleichzeitige Aufzeichnung von Kraft und Reibungssignal ermöglicht).

Es entstand xxsm, welches strukturell die Grundlage des heutigen gxsm darstellt. Dieses Programm entstand im Verlauf der Diplomarbeit von R.Hild, der die Entwicklung life mit seinem STM verfolgen mußte bzw. durfte :=)

Auch unser neues Spielzeug, ein Luft-AFM, wurde von xxsm gesteuert sowie eine Streulichaparatur (SARLS).

Die Computertechnologie schreitet unaufhaltsam voran und es ist zu befürchten, daß bald

keine ISA-Slots mehr verfügbar sind. So entschieden wir für neue Geräte, eine PCI-Version der DSP-Karte (PCI32) zu kaufen. Diese Karte schien der PC31, mal von dem PCI-Bus abgesehen, doch sehr vergleichbar. Es gab jedoch einige Unwegsamkeiten, die mich einige Nerven kosteten...

Jedenfalls entwickelte ich ein Kernelmodul für diese Karte und später eine Variation für die alte PC31, denn auf User-Space IO konnte wegen PCI Konfiguration, etc. nicht mehr ausgewichen werden. Zusätzlich mußten alle Utilities (Loader, Terminal) an die neue Karte angepasst werden. Eine neue Library machte das anfängliche Chaos komplett – die Hoffnung, das DSP- Programm nicht umschreiben zu müssen, war vergebens. Jedoch konnte mit einigen Tricks weiterhin eine, zwar neue, aber gemeinsame Version erhalten bleiben.

Die alte xforms Libary ist zwar extrem effizient, insbesondere meine sehr schnellen MIT-SHM Bilddarstellungsrouterien, aber die Oberfläche und Menüdarstellung lassen einige Wünsche offen. Eine modernes Toolkit Namens Gtk+/Gnome weckte mein Interesse im Sinne des Fortbestandes und der Weiterentwicklung dieser unterdessen mächtigen Mikroskopie- Software. Ein Kraftakt von diversen Nächten zwischen Juni und Dezember 1999 brachte xxsm im neuen Gewand als gnomified xxsm - kurz gxsm hervor.

Das objektorientierte Konzept von Gtk+/Gnome vereinfachte es auch erheblich, die SPA–LEED-Ansteuerung in gxsm mitaufzunehmen. Darüberhinaus entstanden gleichzeitig einige Tools wie Gfit, Goszi und dsp-applet.

Dieses Werk soll im weiteren all denjenigen helfen, die einerseits mit gxsm arbeiten, aber auch etwas mehr über die Internas erfahren wollen. So gliedert sich das folgende Manual in

- einen Anwendungsbezogenen Teil
- nützliche Tips (HOWTOs zu STM und AFM) und
- einen Versuch das Programm-Konzept zu erläutern.

Ich möchte hier meinen Dank an alle aussprechen, die zu meiner Arbeit beigetragen haben, insbesondere jedoch:

M.Henzler für das immer gute Instituts-Klima und die schönen "Almen".

M.Horn-von-Hoegen dafür, daß er mir die Zeit zum ständigen Arbeiten an diesem Projekt gelassen hat.

G.Meyer für seine Starthilfe bei der DSP Programmierung und diversen Gesprächen.

H.Bethge für Ihre Gedult mit mir im Keller.

R.Kumpe für seine Mitarbeit an PMSTM und Diskussionen.

H.Pietsch für sein Mitwirken an PMSTM.

L.Anderson, den ich leider nie persönlich kennengelernt habe, für seine Arbeit an PMSTM.

U.Köhler und seine Truppe für einige Diskussionen.

A.Meier für seine endlose Gedult mit mir ...

F.J.Meier zu Heringdorf für immerwieder freundlich und fröhliche BS Discussions mit diversen Guinness.

Preface to the GXSM-3.0 manual

R.Hild und M.Bierkandt für deren Ausdauer mit den ewig neuen Versionen.

A.Klust für alle Beiträge zu diesem Projekt.

H.Goldbach für SPA–LEED Discussions und für den (noch nicht) herausgesuchten BF krams.

Heilo für die Gewährung einer Mehraufwandzulage.

Negenborn Januar 2000

Percy Zahl

... Gxsm has been licensed as GPL and goes to SourceForge.net ...

Es ist Sonntag, der 21.1.2001, es schneit draußen und die Zeit ist mal wieder ein Jahr fortgeschritten; Ich stecke mitten in meinen Vorbeitungen zum Ortswechsel von Negenborn/Hannover nach Denver/Colorado und möchte ein paar Bemerkungen zum Stand des Gxsm-Projekts festhalten, nachdem meine Dissertation mit der Veröffentlichung meines Werkes zum Stress auf Oberflächen abgeschlossen ist.

Das Gxsm-Projekt ist seit Herbst 2000 offiziell als Open-Source auf <http://sourceforge.net> via Web-Interface und CVS-Access verfügbar. Die Verzeichnishirarchie wurde überarbeitet und "Gxsm" ist nun das Projekthauptverzeichnis.

Im Dezember 2000 wurde ein flexibles Plug-In Interface für Gxsm entworfen und damit begonnen alle speziellen Erweiterungen in Plug-Ins auszulagern. Damit wird eine erhebliche Flexibilität erzielt. Im Januar 2001 sind ein SPA-LEED Simulationsmodul, Peak-Finder, Fokus-Tool und Oszi-Plugin hinzugekommen. Ein Support für die alte Burr-Brown Karte für SPA-LEED ist ebenfalls verfügbar und bereits im Einsatz.

Negenborn Januar 2001

Percy Zahl

... Gxsm goes international...

Just a few up-to-date remarks:

As far as I know Gxsm is used now in more than four countries around the globe.

Kernel 2.4.x support was implemented and is proofed to work well.

More and more Plug-Ins are added...

A Gnome Druid to guide the new Gxsm user along all most important settings was added.

Golden, Colorado USA, August 2001

Percy Zahl

... about one year later and the Gxsm user community is expanding worldwide – some success?

A lot of new features were added and a so called “multi-layer” capability was implemented, especially for use with 2D probing. The modularization via Plug-Ins is driven further, so the scanning and probing control was released from Gxsm core and turned into Plug-Ins. There is still a lot to do, but it is already producing valuable 2D STS data! Together with this development the probing modes are expanded and a experimental digital Lock-In was implemented on a combined DSP and host level.

In today ongoing process the DSP software undergoes a complete redesign. While the DSP hardware future is still not clear (some great options are in sight) – future DSP platform setups are very expensive due to hardware and development tools as well, and as long there is no funding there can't be a new implementation for this non profit project.

And today we can announce the beginning of the new composed Gxsm Manual. It is partly automatic generated from Plug-In source files. Lots of thanks to Andreas for initiating this development!

Golden, Colorado USA, June 16, 2002

Percy Zahl

... Gxsm-2.0 is comming and it also starts support for a new DSP platform...

It's Sunday, my official part of my two year Postdoc job at Colorado School of Mines in Golden has just passed and I'm working on my Gxsm project – it's snowing outside for the first time since about three month of no perception at all...

Two important milestones in the Gxsm history are just in progress:

Gnome 2 is now available and features several good changes and redesigns but also some non-compatible issues to be taking into account. This has now led to the new Gxsm-2.0 CVS branch and results into a new “gxsm2”. The port is completed, including over 70 PlugIns. Some minor issues are to be treated but a functional alpha version of Gxsm-2 is available.

The old PCI32,PC31 DSP cards are hopelessly out of date and not any longer easily available. And just in time the “SignalRanger” DSP occurred, it's a via USB connected standalone DSP board. So the Linux support for SRanger-USB and some new DSP tools were developed and are available via their own SF project (<http://SRanger.sf.net>). The 3rd DSP soft generation for the fixed point SRanger board DSP (TMS320C5402) is currently in development, in parallel the Gxsm2 SRanger support is created – No results yet, but it all looks promising.

Golden, Colorado USA, February 2, 2003

Percy Zahl

... GXSM-3.0-2 and Signal Ranger in daily data production!

It's a rainy Sunday, and the GXSM-3.0 manual is going to be revised to the GXSM-3.0-2 version. The Signal Ranger documentation part is to be started... but the Signal Ranger boards (SR-STD and SR-SP2) are now both supported – thanks to SoftDB for the friendly loan of the STD board!

Also I'm happy to say, the SR does a really good job in our lab as second/spare and testing DSP subsystem. The analog performance (noise level) is outstanding: Using a huge scanning tube (XY: 1000Å/V, Z: 200Å/V) it's possible to resolve the Au-herringbone and Buckyballs on Au(111)!

Adliswil, Switzerland, October 5th, 2003

Percy Zahl

... GXSM-3.0-2 V1.8.4 and Signal Ranger going Multidimensional and Vector Probing

Before all that, the SRanger kernel module for Linux Kernel 2.6.12 and higher was stabilized. The completion of the Vector Probe implementation just happened. The design of the low level (DSP) Vector Probe was finished for over a year but the streaming of the data was only limited working for some time, also a really user friendly GUI was pending. That all changed now. The invention of the GXSM Event mechanism finalized the new capabilities, including the raster probe mode. Also the approach and coarse motion control was polished up and now features an arbitrary wave form mode for best possible results for any inertial motion driven positioning and maximized the flexibility. And for all the future, the whole GXSM core undergo an extensive code cleanup which now totally removed all remainings of hardware close parameters. These are now banned into the corresponding HwI plugins.

Rocky Point, Long Island, NY, USA, November 16th, 2005

Percy Zahl

... GXSM-3.0-2 V1.12.0 releases full multidimensional data processing and visualization power

Several new add-ons were added, in brief:

- more pre defined Vector-Probe modes
- a SignalRanger/CoolRunner CPLD based and hardware or CPLD times (gated) 32bit counter channel to acquire pulse counts/rates of any source
- a event logging mechanism to attach important parameter changes and probe data to the eventually running scan
- full core 4-dim data support and visualization now enabled
- transition of several math plugins to run on multi layers and time dimension on demand
- marker object to manually count things of different flavor (color)

But the major achievement is that the GXSM core now allows handling of 2-dimensional images sets. In particular, a single scan channel can now hold stacks of images (layers) for multiple times making a true 4-dimensional data set. Data can be played like a movie in layer (at a specific time) or in time dimension (at a specific layer). Visualization of profiles (or series of profiles) can be navigated in real time with the “Show Point” or “Show Line” tool in any dimension, also image-slices in any dimension can be generated on the fly.

Export of movies is possible using the Quicktime library. The new OSD (On Scan Display) allows to overlay real-time informations, like time any other parameters.

Rocky Point, Long Island, NY, USA, September 18th, 2007

Percy Zahl

... GXSM-3.0-2 V1.22.0 releases and all new SPM dedicated SR-A810

The year 2009 is already going to its end, a nice sunny colorful fall day... This year has brought big changes or better upgrades on DSP level and in particular on the side of analog signal conversions. The new Signal Ranger Mark 2 (MK2) was already around for a while in 2008 and Gxsm supported with a test HwI and ported DSP code the new platform, which brought us USB-2.0 and more DSP speed and memory. But it had a drawback, even a newer revision of the AICs, it was a slow bottle neck, as it had un acceptable long loop delays. The need for something dedicated was even more pressing now. All started out in early summer 2008, discussions of the Gxsm team and affiliated leading Gxsm users and institutions (in particular myself and Rolf Möller from Department of Physics, University of Duisburg-Essen, Germany) with SoftdB and their hardware engineer B. Paillard were paving the way to the now available Analog-810 interface for the MK2 (MK2-A810). Final designs and decisions for the DA/AD converter were made about following the STM conference in Keystone, Colorado.

Just a few days before Christmas 2008 I received the first prototype for the MK2-A810 assembly – big thanks to SoftdB! As the MK2 was already fully supported by Gxsm and the driver infrastructure for the A810 was seamless to the previous, getting it going was done in a snap! This new thing was holding to its great specs greatly and things were moving along very well. By January the first working HwI release was going into CVS and very soon was also field/labpratory STM proven to work and perform very well. In all aspects it is superior, precision, stability and speed – 75kHz feed back loop with full loop delay less than 5 samples. Sampling rates at up to 150kHz possible (and used in latest versions). A little later two optional counter channels were added the FPGA logic.

The new MK2-A810 is available since February 2009 and now also 19" rack ready on a Gxsm/SPM customized enclosure.

All basics done, the new power of the A810 was getting explored in depth and a couple of new features and software based performance improvements were implemented, let me just list:

- 2 channel 32-bit counter timer, synchronized with sampling
- Evaluation of software based resolution enhancements, Z and X/Y HR-mode
- Real time magnitude dependent bandwidth adjusting via FIR for current input
- Transition to Float in Gxsm in conjunction with transfer of full statistical data (32-bit resolution summed values from DSP + normalization count)
- FIFO data transfer optimization: using custom byte packing, first order linear predictor...
- Real time configurable 4-channel feedback input mixer, lin/log/fuzzy modes
- Enhanced and expanded Vector-Probe modes and visualization modes (GUI)

- Many more details...

And: A update on Gxsm and teh new MK2-A810 is submitted/accepted to Journal of Vacuum Science and Technology B (JVST B), proceedings of the NC-AFM, Yale 2009.

Brookhaven National Laboratory - CFN, Upton, Long Island, NY, USA, October 29th, 2009
Percy Zahl

...from Version Numbers to “Battenkill Warrior”, “Lancaster Classics”, “Lindau Historic” to GXSM-3.0-2 V1.27.3 “Arosa Express” – get prepared for the high-end MK3Pro-A810/PLL and new GXSM optimized SPM HV-Amp what is now also available: the “Smart Piezo Drive” with serious DSP power and featured under the hood not yet seen!

Not functional but just getting a little more “social” – dedication names to major mile stone version changes – so you can chat better about versions!

“Battenkill Warrior” is introducing a separated Z-Offset signal for scan slope compensation on analog level. Also a new so far little used/tested feature tracking VP mode to follow in real time “gradient up” or “-down”. May also name it atom-tracker.

Full 3D Offset control and Linear Drift Correction via automated Offset adjusting. Also added helping aids to easily determine drift manually from manually to be identified features in scan(s) via "Global Position Reference Mark" and Time/Drift calculation for Point markers. (New options: *main menu/Scan/View/Coordinates/(Time + Relative)*) If previously a Global Reference was set via any Point/Marker Object/Global Reference Point.

Massive selections of hundreds of VectorProbe data files for example from or live while raster probing via easy DnD read back is not supported and very useful for life data inspection of long mapping runs.

Always on going: new additions to the universal GXSM-3.0 VP modes/tabs, including a dual folder with user arrangeable tabs for better work flow and overview. Example: segmented STS.

“Lancaster Classics” is not providing much visible additions but introduces now a higher precision of the DSP level integer math moving to 32- and temporary 64bit for vector scan signal generations. Also further enhancements of the HR signal output mode (native 16bit to near 20 (some limitations apply) on software level) and other optimizations are included. Additions and new indicators for the PanView to incorporate GPIO and some DPS statemachine status indications.

“Lindau Historic” – incorporating a set of GXSM-3.0 community ideas discussed and collected at the 2011 NC-AFM in Lindau. Most visible, the new a red-profile history. Some patches needed for newer Gkt+/Gnome/X11 releases. Few more options for VP-Z. New VP feature allows to program limits/triggers to stop a VP section, for example when a certain max force is reached. Also the Mk3-A810/PLL activities are evolving and a PLL prototype is getting available in early 2011! The PLL is only suitable and dedicated for tuning fork systems with up to 75kHz detection – it is all software based and developed by SoftdB. This means for the future hardware speeds are moving up we will be able to follow up the bandwidth! DSP code porting is under the way and this takes more hurdles than anticipated – however with some support by SoftdB it's getting finally to a working version what still is a little beta (by end of 2011). Also new on the communities demand: A GXSM-3.0 optimized Piezo / HV Amplifier “Smart Piezo Drive” what includes a separate DSP for several control and monitoring tasks – it works all standalone and fully analog, but can be hooked up via USB to a control computer for watching

signals and configuring all kind of features like gains and bandwidth – but as a novelty it also can perform linear drift corrections by itself and accepts digitally offset settings.

It's now 2012 and GXSM-3.0 "Arosa Express" is out – get ready for more remote and freely via Python programmable SPM actions! Now VP probing and Mover/Autoapproach can be fully controlled via Gxsm-Python-Remote (via the "emb" interface). Here is a little sneak peak script:

```
import emb as gxsm

gxsm.set ("DSP_Bias", "0.1")
gxsm.set ("DSP_SCurrent", "1.5")
gxsm.set ("OffsetX", "0")
gxsm.set ("OffsetY", "0")

gxsm.action ("DSP_CMD_AUTOAPP")

for x in range (0,1000,100):
    gxsm.set ("OffsetX", "%f"%x)
    gxsm.sleep (10)
    gxsm.set ("DSP_IV-Start-End", "-1")
    gxsm.set ("DSP_IV-End", "1")
    gxsm.action ("DSP_VP_IV_EXECUTE")
    gxsm.sleep (10)
```

PS: New Motto: "Gxsm is a glove to your SPM experiment" – this said, we claim you have free hands to do pretty much all you like in a instant real time feedback experience – but be warned, this also means you can dig your tip into the surface if you want to also if you do not take care as there is pretty much no way to distinguish unless introduction mostly annoying safety "blocks".

Brookhaven National Laboratory - CFN, Upton, Long Island, NY, USA,

March 6th, 2012

Percy Zahl

Signal Revolution: “Snowy Janus Hack” + “Snowy Janus Signal Warrior” with Mark3-Pro-A810-PLL

“And soon I realized the power of the new dimension I added!”

An evolving idea leading out of a dilemma unleashing not yet seen power and configurability – a digital patch-rack with a lot of transparency and insights. Double power for users and developers – live “signal” or variable debugging and monitoring.

Why I am doing this? How? What is it and where will it go and what can I do with it? Let me try to explain:

A need for even more real time flexibility and the need to access even more data channels/signals – to manage a huge expansion of the “signal space”. This is no more possible and not good to handle efficiently with a fixed signal to channel assignment due to a limited number of total channel I can handle for several reasons. However, there are more than enough channels, just a growing variety of different “data” or “signals” beyond the raw analog inputs and you will never need all of them the same time. So that lead me to a new approach I already started initially to get out of the dilemma I found myself getting into and soon I realized the power of the new dimension I added!

The “PAC/PLL” signals needed to be accessible... that we have now for a while (Mk3) via the configuration of a signal source via a pull down menu for now each Feedback Mixer Input 0..3.

And again those can be remapped (MIX0..3) – still names (PAC1..3 in the scan channel source selector for 2d scans). I need to clean up the signal naming conventions – part of the over haul. That’s the harder part on GUI level. Plan executed: Each signal got a unique name, a value range and real world unit associated.

You are still with me? I hope...

Signals are simply variables and connections are made via just pointing the input to the source – as simple as this – “signals” will be “hot pluggable”.

The current DSP code in state machine design will remain unchanged in it’s proven design form, only so far fixed signal paths will be “broken up” and made open to be hot reroutable with more connectivity options, more data sources exposed to Gxsm via existing channels and at other locations as needed. The existing data processing blocks starting with ADC inputs, PAC-signals, counters, control values (like Bias)..., Scan, Offset Move, Probe, LockIn, ... and finally a HR Output block with new options... I will call from now on modules. And data/variables resulting from module data processing I will name signals. The default power-up signal linkage will be pretty much the default as known from the past. A kind of netlist will be exposed to Gxsm and configuration tools to be manipulated hot. So we have module input “nodes” and module output “signals” made available for netting up!

Let me break up the existing design in existing modules as sketched:

Outputs will be more configurable and not any more hard assigned – just a default! They will get an input stage with the signal input and two additional adding inputs (modulation, etc.) with

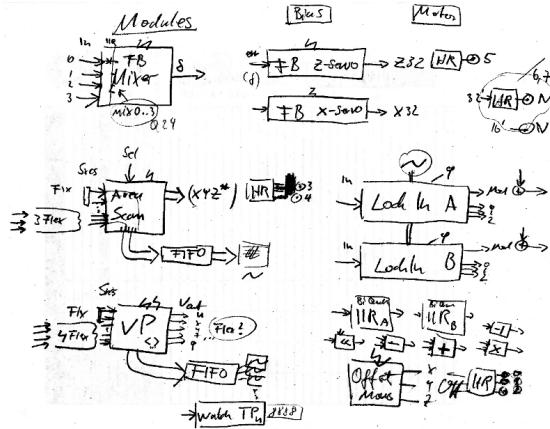


Figure 1.: Idea Sketch: Modules and signals.

optional gain.

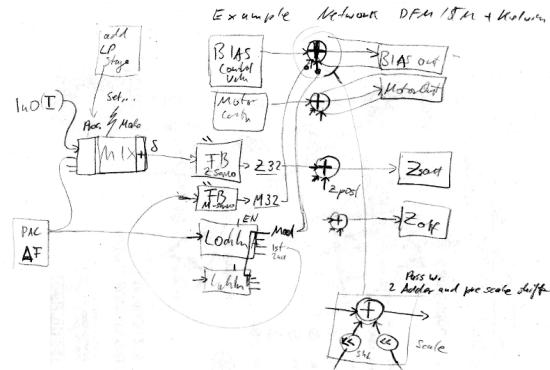


Figure 2.: Idea Sketch: Example SPM configuration.

To handle all this a new cleaned up python basic based support class and a configurator script with monitoring goodies and a signal graph visualization tool is now available. Also a signal configuration management with to flash store and auto restore there is.

On Gxsm level you will feel home pretty much right away – only several sources are now not fixed any more but allow to choose from multiple signals. Also signals are categorized in classes so not always all need to be exposed to prevent a “GUI over load”.

That said – enjoy the new next generation and a few handy and eye candy monitoring galvos, a signal scope and a new tuning tool with peak auto-fit.

Well, read about the actual outcome in the SRanger Mk2/3 HWI plugin chapter [23.9.8](#).

Gxsm Central, Rocky Point, Long Island, NY, USA,
March 29th, 2014

Percy Zahl

Got High Speed? Real Time Engine 4 GXSM-3.50: “Next Level RTime Engine” with Mark3-Pro-A810-PLL and RedPitaya

“It was long rocky way! It is way too long since myt last update here. But finally a e- or re-volution.”

We got a all new fully high definiton capable GTK3 compliant GUI! That was some act.

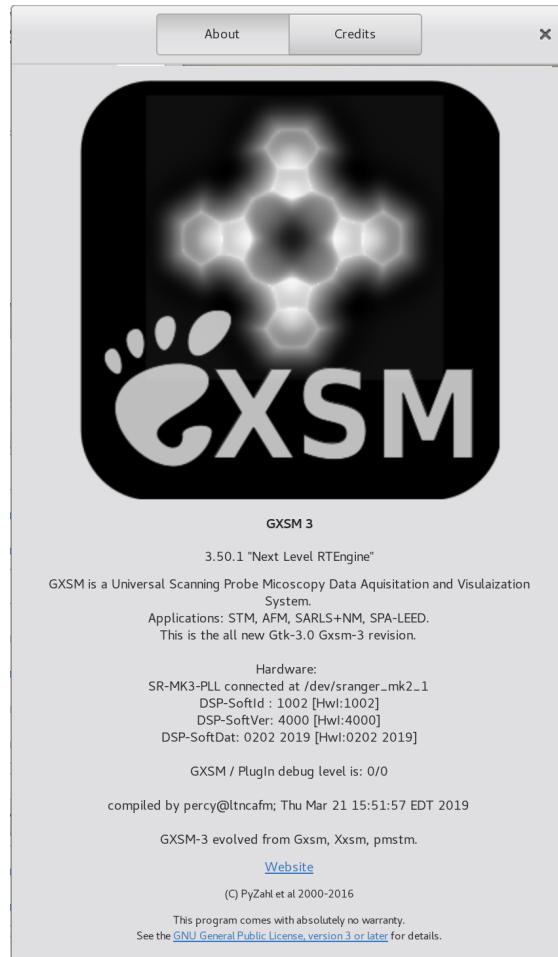


Figure 3.:

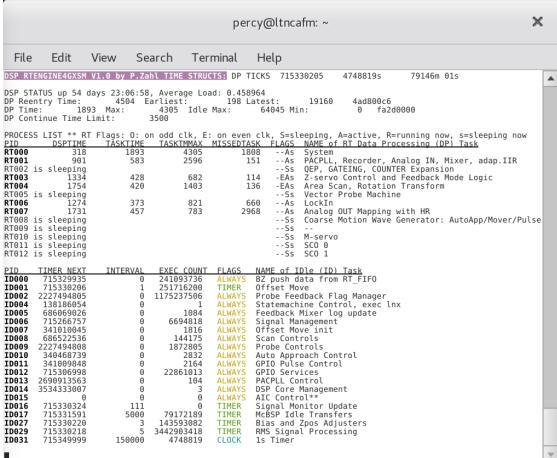
Beside many many new features, gxsm is now all into nc-AFM and provided dedicated tools for molecular imaging.

Alongside a in many aspects enhanced gxsm python console with gxsm utility libraraies and

include functionality and a growing python support library.

The MK3 convergence detector aka PAC-PLL is great, but suffers on bandwidth and statistics. So after a major act of learning and FPGA hacking the first ever 125MHz dual PAC-PLL and all bells and whistles including super fast amplitude controller, Q-control option and ultra wide range PLL operating with 48 bit phase/freq. control was born. After proof of principle some unexpected efforts needed to put in to create a reliable, fast and real time digital data link to the DSP – featuring now a McBSP serial link operating over a potentially longer set of two CAT5 cables (about 2m in use right now). This triggered some interrupt collision issues serving the McBSP on the DSP side fast and on time. With great help and insights from Alex at SoftdB the problem was tracked and a solution sketched up. May be for the good, a major revision of the historically more monolithic data processing scheme was on the table. A single hint... and I ended up creating my very own micro kernel and real time machine on a 1/150kHz time scale. Evaluating and optimizing tasks, moving every bit of code not fully real time critical into new idle tasks. Also new automatic RT task scheduling and enable/disable control – all DSP based – still provides a seamless and 100% backwards compatible DSP code from the “outside” point of view.

Oh no.... I need to rewrite the DSP under the hood section now.



The screenshot shows a terminal window titled "percy@ltncafm: ~". The window displays a process list from the GXSM tool. The columns in the list are: PID, TIMER, NEXT, INTERVAL, EXEC.COUNT, FLAGS, NAME of Idle (ID), Task. The list includes various tasks with their respective parameters and flags like ALWAYS, -As, -Ss, -Sc, etc.

PID	TIMER	NEXT	INTERVAL	EXEC.COUNT	FLAGS	NAME of Idle (ID)	Task
ID000			0	241093736	ALWAYS	B2 push data from RT_FIFO	
ID001			0	1175237596	ALWAYS	Probe Feedback Flag Manager	
ID002	2227494805		0	1175237596	ALWAYS	Statemachine Control, exec lnx	
ID004	138186854		0	1080	ALWAYS	Signal Management	
ID006	715266757		0	6694818	ALWAYS	Signal Management	
ID007	688522536		0	144175	ALWAYS	Scan Controls	
ID008	2227494806		0	1872805	ALWAYS	Probe Controls	
ID010	341089848		0	203	ALWAYS	GPIO Pulse Control	
ID011	341089849		0	22860	ALWAYS	ADC DAC Setups	
ID013	2699913563		0	104	ALWAYS	PACPL Control	
ID014	353433300		0	3	ALWAYS	DSP Core Management	
ID015	715330324		0	0	ALWAYS	Signal Monitor	
ID016	715330324		111	0	TIMER	Signal Monitor Update	
ID017	715330324		5800	0	TIMER	RTSP and B2P	
ID027	715330320		3	143593982	TIMER	Bias and Zeros Adjusters	
ID029	7153303218		5	344293418	TIMER	RMS Signal Processing	
ID031	715349999		150000	4748819	CLOCK	1s Timer	

Figure 4.: Real Time Engine 4 GXSM – a RT process view.

But for the good, things just got better and faster!

Gxsm Central, Upton, Long Island, NY, USA,
April 19th, 2019

Percy Zahl

Contents

Preface to the GXSM-3.0 manual	3
1. Front Matter	31
2. Introduction	33
3. GXSM-3.0 Project Installation	39
3.1. System requirements	39
3.2. Getting GXSM-3.0 from SVN	40
3.2.1. Configuration	40
3.2.2. Compilation	40
3.2.3. Installation	40
3.2.4. Updating	41
3.2.5. Trouble shooting	41
3.3. Getting GXSM-3.0 via APT	42
3.3.1. Adding package repositories	42
3.3.2. Package installation	43
3.3.3. Updating	43
3.3.4. Building packages	43
4. GXSM-3.0 Quick Start Guide	45
4.1. Storing and Restoring Settings	45
4.2. Delete Settings	45
5. GXSM-3.0 Tipps and Tricks	47
5.1. Running Two Instances of GXSM-3.0	47
I. The GXSM-3.0 core	49
6. Program Start	51
6.1. Command line parameters	51

Contents

7. The main window	53
7.1. Understanding the main window's entries: SPM mode	54
7.1.1. Scan parameters	54
7.1.2. File and user information	54
7.2. Drag and Drop	55
7.3. Keyboard-Accelerators	55
8. Channels	57
8.1. The channel dialog	57
9. Visualization	59
9.1. Data display modes	59
9.1.1. Scaling and shifting data view range	60
9.1.2. Using a palette: false color mapping	61
9.1.3. Custom and GXSM-3.0 provided palette	61
9.1.4. Additional Information	62
9.2. Visualization modes	62
9.2.1. View “No”	62
9.2.2. View “Grey 2D”	62
9.2.3. View “Surface 3D”	65
9.2.4. The general purpose “Profile 1D” view	77
10. Configuration	83
10.1. Hardware folder	83
10.2. Instrument folder	85
10.2.1. Inst-SPM folder	86
10.2.2. Inst-SPA folder	88
10.3. DataAq folder	89
10.4. Probe folder	91
10.5. Folder Adj.	91
10.6. Paths folder	91
10.7. User folder	91
10.8. GUI folder	92
II. Plug-ins	93
11. Plug-Ins: control	95
11.1. Inet JSON Scan Data Control	95
11.2. SPA–LEED simulator control	95
11.3. Nano HPLG plotter	97

11.4. SPA-LEED peak finder and monitor (OBSOLETE)	98
11.5. Nano Manipulator (to be ported)	99
11.6. SPM Scan Control	100
11.7. RHK Scan Control (to be ported)	103
12. Plug-Ins: scan	105
12.1. WIP (WiTeC) Import	105
12.2. Cube file Import	105
12.3. UKSOFT/U-view Import (ELMITEC LEEM)	106
12.4. Park Scientific (HDF) data Import	109
12.5. Quicktime	109
12.6. Import/Export of old (G-) dat files	112
12.7. Import/Export of old (G-) dat files	112
12.8. Import Tool for RHK STM-200	113
12.9. Converter	113
12.10UK2000 v3.4 import plug-in	114
12.11Import of RHK SPM32 files (STM-1000 electronics).	114
12.12external converter	115
12.13Import/export of non-SPM file formats	116
12.14ASCII data file Import/Export	117
12.15Import of Surface Data Format files	118
12.16Import/Export of G.Meyer STM/AFM Dat files	119
12.17Import/export of Scala SPM files (Omicron)	119
12.18Import of Digital Instruments Nanoscope files	120
12.19PNG image Import/Export	121
12.20Binary file Import	122
12.21SPA-LEED (SPA4) d2d data file Import	123
12.22Import/export for WSxM Nanotec Electronica SPM data	123
13. Plug-Ins: math/background	125
13.1. Gamma correction	125
13.2. Plane Regression	126
13.3. Lineregression	126
13.4. Line Regression	127
13.5. Plane max. propability	128
13.6. Remove Line Shifts	128
13.7. 2nd order scanline correction	129
13.8. Multiply Const Background correction	130
13.9. Waterlevel	130
13.10Stop band removal	131
13.11Template	132

Contents

13.12Plane three points	132
13.13Full Timescale FFT	133
13.14Plane regression	133
13.15Pass band copy	134
13.16Sub Const Background correction	135
13.17Smooth Z drift correction	136
14. Plug-Ins: math/convert	137
14.1. Convert to byte	137
14.2. Convert to short	138
14.3. Convert to complex	138
14.4. Generate test data	139
14.5. Convert Unsigned to float	139
14.6. Convert to short, apply custom fix filter	140
14.7. Convert to long	141
14.8. Convert to double	141
14.9. Convert to float	142
15. Plug-Ins: math/filter1d	145
15.1. Repair filter	145
15.2. Despike 1d	146
15.3. Template	146
15.4. Koehler filter	147
15.5. 1dim differentiation	148
15.6. Time Domain FFT Filter	149
15.7. Linear stationary differentiation	149
15.8. Ft1d filter	150
16. Plug-Ins: math/filter2d	153
16.1. Curvature	153
16.2. Line Interpolation	154
16.3. T derive	155
16.4. Smooth	156
16.5. Stat Diff	156
16.6. Convolution with 3x3 kernel	157
16.7. Lineinterp1	158
16.8. FT 2D	158
16.9. inverse FT	159
16.10Despike 2d	159
16.11Template	160
16.12Local height	160

16.13Edge	161
16.14Normal Z	162
17. Plug-Ins: math/probe	163
17.1. Probe Image Extract	166
17.2. AFM (NC-AFM) mechanical tip apex/molecule imaging simulations	166
18. Plug-Ins: math/misc	169
18.1. Smoothing layers in 3D	169
18.2. Absolute value	170
18.3. Cut spectra from 3D data	170
18.4. Smoothing layers in 3D	171
18.5. Create a shape polygon from lines	171
18.6. PSD add – SARLS	172
18.7. Find local maxima – SARLS	172
18.8. Smoothing layers in 3D	173
19. Plug-Ins: math/transform	175
19.1. 90° clockwise rotation	175
19.2. Shear along X	175
19.3. Quench Scan	176
19.4. Multi Dimensional Transposition	176
19.5. Shear along Y	177
19.6. Scale scan in X and Y	177
19.7. Movie Concat	178
19.8. Flip diagonal	178
19.9. Auto Align Multi Dimensional Movie	179
19.10Mirror X	180
19.11Rotate a scan area	180
19.12SPA–LEED octopole distortion correction	181
19.13Auto Align	182
19.14Volume Transform	183
19.15Unwraps Z data in given range	183
19.16Multi Dimensional Layer Reverse	184
19.17Horizontal merge	184
19.18Transformation Shift-Area	185
19.19Affine Transformation	185
19.20Mirror Y	186
19.21Vertical merge	187

Contents

20. Plug-Ins: math/statistik	189
20.1. Opencvmatch	189
20.2. Vacancy Line Analysis	190
20.3. Generate a polar histogramm	192
20.4. Nndistribution	193
20.5. SPA–LEED simulation	194
20.6. OpenCV Re-Center Feature	195
20.7. Vorlage (Template) PlugIn	196
20.8. SPA–LEED 1D profile simulation k_z	198
20.9. Crosscorrelation	200
20.10. Stepcount	201
20.11. Autocorrelation	202
20.12. Calculate in plane direction of gradient	203
20.13. Histogram	203
20.14. Angular Analysis	204
20.15. Calculate gradient (slope)	205
20.16. Add Trail	206
20.17. Average X Profile	206
21. Plug-Ins: math/arithmetic	207
21.1. Multiply scans	207
21.2. Max of two sources	208
21.3. Add two scans	208
21.4. Multiply scans	209
21.5. Invert Z	210
21.6. Divide scans	210
21.7. Subtract scans	211
21.8. Z Limiter	212
21.9. Multiply scans	213
21.10. Absolute Value	213
21.11. Logarithm transform of Z-values	214
22. Plug-Ins: common	215
22.1. View NetCDF file data	215
22.2. Probe Indicator	215
22.3. Pan View	217
22.4. Python remote control	220
22.4.1. Reference	220
22.5. Query Hardware/DSP software information	254
22.6. Postscript printing tool	254
22.7. Query Hardware/DSP software information	255

22.8. Show plug-in details	255
22.9. MkIcons – Make icons	256
23. Plug-Ins: hard	257
23.1. SPM SIM and Template Hardware Interface	257
23.2. Signal Ranger Hardware Interface	257
23.2.1. SR-DSP Feedback and Scan Control	258
23.2.2. Trigger – Multi-Volt and -Current Control	259
23.2.3. Advanced Feedback and Probe Control	260
23.2.4. SR-DSP Mover and Approach Control	270
23.2.5. Extra Python SR-DSP Control and Configuration Scripts	272
23.3. Local Area Network (Internet) RHK Controller Hardware Interface (to be ported)	278
23.4. TC211 CCD Interface	279
23.5. Interface to the spm2 scan device	280
23.6. Innovative DSP Hardware Interface for PC31 and PCI32 (OBSOLETE or to be ported)	281
23.7. Demo Hardware Interface	281
23.8. Video4Linux Grabber (experimental, to be ported)	282
23.9. Signal Ranger MK2/3-A810 Hardware Interface	282
23.9.1. SR-DSP Feedback and Scan Control	287
23.9.2. Advanced Feedback and Probe Control	289
23.9.3. SR-DSP Mover and Approach Control	313
23.9.4. SR-DSP Phase/Amplitude Convergent Detector (PAC) Control	317
23.9.5. Setup – Gxsm Configurations for the MK2-A810.	321
23.9.6. Sample STS IV,dIdV	324
23.9.7. Sample STS along line demo setup	325
23.9.8. Signal Ranger MK3-Pro-A810/PLL Additional Features Guide	326
23.9.9. Optional MK3-Pro-SmartPiezoDrive with Gxsm Link	358
24. Plug-Ins: hard/MK3-A810_spmcontrol/experimental_python_scripts	361
25. Plug-Ins: hard/MK3-A810_spmcontrol/python3_scripts	363
26. Using plug-ins	365
III. Digital signal processing subsystem	367
IV. Appendix	369
A. Terms and conditions	371

Contents

References	385
V. Index	387
Index	389

1. Front Matter

The Gxsm Project: Gxsm itself, drivers, utilities, demos and documentation

Copyright (C) 1999 - 2017 Percy Zahl, Andreas Klust, et al

Email: zahl@users.sourceforge.net WWW: <http://gxsm.sourceforge.net>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

See the end of this document for complete license: Appendix A

2. Introduction



GXSM-3.0¹ – Gnome X Scanning Microscopy – is a powerful graphical interface for any kind of 2D and 3D (multi layered 2D mode) data acquisition methods, especially designed for scanning probe microscopy (SPM).



Figure 2.1.: Gxsm3 (version 3.47.0 Action Eclipse) Main Scan Control Window.

The aim of the GXSM-3.0-project is to provide a versatile control system being suitable to operate all different kinds of scanning probe microscopes. This includes in particular scanning tunneling microscopes (STMs) and atomic force microscopes (AFMs), but it is not restricted to these types. In principle, it is also flexible enough to operate scanning angle resolved light scattering (SARLS) experiments or spot profile analysis of low-energy electron diffraction (SPA-LEED) optics. One reason for this versatility is

¹The Project can be found in the Internet at <http://gxsm.sourceforge.net>

2. Introduction

that all these instrument have in common a 2D-data acquisition by scanning sequentially points in the xy -plane.

Certainly, GXSM-3.0 gains also versatility, because of its design concept: a DSP (digital signal processor) based hardware is used for the data acquisition, the scan signal generation and various feedback loops (z distance between tip and sample but also the oscillation control of an NC-AFM). The graphical user interface (GUI) provides not only tools for 1D (line profiles), 2D (images), 3D (morphology), and even 4D (time series) data visualization, but also for manipulation and analysis. As the GUI is based on plug-ins, it can be easily extended to new tasks.

A third aspect is the python interface of GXSM-3.0 allowing to control the GUI remotely at almost real-time level. The performance of the python scripting is more than sufficient to provide a functionality far beyond batch acquisition and processing of data, but also allowing to code complex data acquisition tasks without programming anything directly on DSP level.

The project was founded at the Institute for Solid State Physics² Leading developer of GXSM-3.0 is Percy Zahl³, but more than 50 SPM groups world wide are not just using GXSM-3.0 but also contribute to its further development. The program was developed for Linux⁴ using the Gtk+/Gnome libraries⁵ for the GUI.

And best of all: it's free! GXSM-3.0 is licensed under the terms of the GNU General Public License (GPL, see A). Therefore, everyone can copy, use, and modify GXSM-3.0 for his/her needs provided that the resulting software is again published under the GPL licence.

The following list gives a short overview on the main features of GXSM-3.0:

- Support for STM, AFM, and any other 2D (2D layered and multi-channel) data-acquisition method can be supported by GXSM-3.0.
- The GXSM-3.0 core handles multiple channels of 2D (layered) data fields of arbitrary type and unlimited size and a grey-scale or false color view of 2D data image, using "on the fly" data transformation as there are:
 - "Quick": a line regression and subtraction is performed on each scan-line
 - "Direct": only contrast and brightness adjustments
 - "Direct HiLit": same as direct but marks the lowest and highest values
 - "Plane": on-the-fly 3-point defined plane subtraction

²Institut für Festkörperphysik, Universität Hannover, Appelstraße 2, D-30167 Hannover, Germany
www.fkp.uni-hannover.de

³e-mail: zahl@users.sourceforge.net

⁴For example Debian 9.3

⁵GXSM-3.0 currently requires Gnome 3.2 and GTK+3.22

-
- "Logarithmic": logarithmic scaling mode, almost used by diffraction methods
 - "Horizontal": automatic line average subtraction
 - "Differential": view of X-derivative, $[Z(X + 1) - Z(X)]$
 - "Periodic": like "Direct" Mode, but grey-scale is applied modulo #grey-levels
 - More sophisticated background correction and data analysis methods are implemented as filters.

For more details on the different modes of the visualization of the data see Chap. 9.

- Data presentation is by default a (grey or false color) image but it can be switched to a profile view (1d), profile extraction on the fly... Or you can use a 3D shaded view (using OpenGL 4.6) which now offers a sophisticated scene setup.
- The "high-level" scan controller is now separated from the GXSM-3.0 core and is build as PlugIn, while the real-time "low-level" scanning process, data-acquisition and feedback loop (if needed), runs on the DSP – if present, else a dummy image is produced. The current scan-line, marked in red, can be viewed simultaneously as profile. (View→red Profile)
- Extremely flexible configuration of user settings and data acquisition and probe modes.
 - Easy to extend by Plug-ins, some examples of existing Plug-ins:
 - * Background correction methods
 - * Image filtering 1D and 2D, including several Fourier transformation methods
 - * Image analysis/statistics: histogram, step analysis, ...
 - * Geometric transformations: scaling, rotation, affine, ...
 - * and more, write and contribute your favorite math Plug-in for GXSM-3.0! Don't be afraid, there is a step by step instruction tutorial and a math Plug-in generator, all you need to do is to add your math code!
 - Special datafile/formats import/export filters, (extendable via PlugIns):
 - * a set of simple raw formats (.byt,.sht,.flt,...), see 12.13
 - * Digital Instruments/Veeco Metrology Group, NanoScope (import) (<http://veeco.com>, see 12.18)
 - * Omicron NanoTechnology, Scala (<http://www.omicron.de>, see 12.17)

- * WSxM/Nanotec Electronica SPM, WSxM (<http://www.nanotec.es>, see [12.22](#))
 - * SDF (Surface Data Format, see [12.15](#))
 - * UK2000 v3.4, see [12.10](#)
 - * G. Meyer STMAFM, see [12.16](#)
 - * Park Scientific (AFM, basic import support, see [12.13](#))
 - * “d2d” data format (SPA–LEED), used by “spa4” and “xspa”, see [12.21](#)
 - * any NetCDF file containing a 2D data array
 - * Grey Images in .pgm format (P5 type), see [12.19](#)
 - * PNG image format, see [12.19](#)
 - * Targa (.tga) export, in 8/16/24bits color depth, [12.13](#)
 - Special instrument control Plug-Ins:
 - * a Scan Control Panel
 - * SPM: DSP Control: Feedback and Scan Characteristics (Speed, ...)
 - * SPM: Mover/slider and auto approach controls
 - * SPM: flexible Probing: Spectroscopy (STS), Force-Distance Curves (AFM), using the DSP also Digital LockIn Probing (e.g. dI/dU (U)) is possible without any additional hardware! But not only SPM, I already ran our Quadrupole Mass Analyzer with it!
 - * Phase Lock Loop (PLL) operation in combination with the SR-MK3Pro & A810 boards (with enabled PLL option)
 - NanoPlotter Plug-in: reads simple HPGL files and moves along the plot path using predefined DSP settings (U, I, Feedback Parameters, Speed, ...) for “Pen-Up” and “Pen-Down” movements. This was in principle already possible via the remote control and a script, but now it’s much more convenient and user friendly! Even the path is shown ... can be modified, saved and re plotted!
 - A Plug-in categorizing mechanism automatically only loads the required Plug-Ins for the actual setup: E.g. no Hardware Control Plug-ins are loaded in “offline” Data Analysis Mode.
 - At the time there are more than 80 Plug-ins.
- GXSM-3.0 itself is fully hardware independent. It provides a generic hardware-interface (HwI) plugin infrastructure to attach any kind of hardware. The HwI has

to manage low level tasks and pass the data to the GXSM core, plus, it has to provide the necessary GUI to provide user access and control to hardware/instrument specific parameters and tasks.

- Scan parameter changing on the fly – you can modify the feedback parameters or switch the tunneling voltage while scanning in between scan lines. For example: Imagine you are scanning a STM topography and current Image, the surface looks flat, then just change the feedback parameters to CP=0 and CI=1e-5 (something small) and now you are in constant height mode!
- On the fly, even while scanning is in progress, you can view profiles, extract data parts, re-scale – just do all you like!
- Event mechanism: User “Events” like bias change are now logged and attached to the scan data and can be visualized (Position marker). Other events like “Probe” (any kind of spectroscopy or manipulation) while or after scanning are automatically attached to the scan with all data and can be visualized via a single mouse click. The same for automatic rastered “Probe” scanning.
- Python Remote Control Interface: The GXSM-3.0 scanning progress is fully scriptable using python. Or may be used to automatize math/data analysis tasks.
- Cross Platform: works on i386 and PPC based Linux distributions; a Windows port besides the main stream is available⁶.
- GXSM-3.0 takes full advantage of the NetCDF data format.
- Scan auto saving, session logging, Plug-in details browser, NC-View, PS-Printing and a Icon generator are available too.

The GXSM-3.0 software can be divided into three parts: First, the GXSM-3.0 core providing the main functionality for handling and visualization of data described in the first part of this manual. The basic functions of the GXSM-3.0 core can be extended using plug-ins. Plug-ins are small pieces of software dynamically linked to the core. The plug-ins are described in the second part of the manual. The third part documents the digital signal processing (DSP) software needed to carry out actual measurements. The DSP software is not necessary for applications using GXSM-3.0 only for data analysis purposes.

⁶<http://gxsm4w.sf.net>

3. GXSM-3.0 Project Installation

3.1. System requirements

GXSM-3.0 needs a reasonably up-to-date machine running a recent version of almost any Linux variant. GXSM-3.0 is always developed on a Debian Testing. Some description to install GXSM-3.0 an a clean Debian Jessie are found in this forum posting: <https://sourceforge.net/p/gxsm/discussion/297458/thread/3f0faafe/> It also runs on Ubuntu 20.04 LTS for which dedicated binary packages are available.

There are many more forum entries related to the installation processes on different linux distributions.

System memory requirements are ranging from little to several gigabytes if you want to deal with big scans, movies or multidimensional data sets ☺.

The GXSM-3.0 Windows port supports SRanger based hardware. Binaries are available via a separate project hosted on sourceforrge.net: <https://sourceforge.net/projects/gxsm4w/>.

The GXSM for Windows project was not updated since 2013.



To install/compile GXSM-3.0 on Debian Testing install these package:

```
apt-get_install_subversion_libgnomeui-dev_intltool_yelp-tools
 gtk-doc-tools_gnome-common_libgail-3-dev_libnetcdf-dev
 libnetcdf-cxx-legacy-dev_libfftw3-dev_libgtk-3-0
 libgtksourceview-3.0-dev_gsettings-desktop-schemas-dev
 python-gobject-2-dev_libgtksourceviewmm-3.0-dev
 libquicktime-dev_libglew-dev_freeglut3-dev_libgl1-mesa-dev
 libopencv-core-dev_libopencv-features2d-dev_libopencv-highgui-dev
 libopencv-objectdetect-dev_libnlopt-dev_libglm-dev_fonts-freefont-ttf
```

gtk+-3.0 >=3.22.0 is not included in Ubuntu 16.04 LTS. If you want to compile GXSM-3.0 with prior versions of gtk3, you have to patch the file configure.ac.



3.2. Getting GXSM-3.0 from SVN

Go to the GXSM-3.0 Project’s home page at <http://sourceforge.net/projects/gxsm> and follow the link almost at the bottom “SVN Repository” and have a look at the instructions there.

The “module” contains the C++ source code for “Gxsm-3.0”. It can be retrieved with subversion anonymously:

```
$ svn co svn://svn.code.sf.net/p/gxsm/svn/trunk/Gxsm-3.0 Gxsm-3.0
```

The module “Gxsm-3.0-Manual” can be checked out, too. It contains the L^AT_EX-sources for the manual that you are currently reading.

```
$ svn co svn://svn.code.sf.net/p/gxsm/svn/trunk/Gxsm-3.0-Manual Gxsm-3.0-Manual
```

3.2.1. Configuration

Change into the gxsm3-svn directory and type

```
$ ./autogen.sh
```

This should finally create all Makefiles in several sub-directories.

3.2.2. Compilation

Type make in the gxsm3-svn directory:

```
$ make
```

This command will compile everything, if you want to compile it step by step, you can run this command in the src sub-directory first separately to see if all works OK. But later you want to compile all plug-ins and tools as well – if it works in src it is most likely it works OK for the rest, so try make in the gxsm-svn dir again!



To speed the make process up on multi core systems, use **\$ make -j2**, **\$ make -j4...** for spawning multiple compile jobs.

3.2.3. Installation

Finally you need to run the install command:

```
$ [sudo] make install
```

(root privileges needed!)

If there are any errors, in most cases there are missing development packages (e.g. missing header files/libraries), have a look at the first occurring error and figure out what’s missing, refer to [3.2.5](#).



For automatic thumb-nailing within the Nautilus file manager and setting up other good-

ies, check out the GXSM-3.0 *Tips and tricks* forum:

http://sourceforge.net/forum/forum.php?forum_id=302195

3.2.4. Updating

For later updates just run

\$ svn update

in the gxsm3-svn directory and remake/install all again!

If you want to keep your previous binary version/installation, use a different installation path (prefix):

\$./configure --prefix=/usr/local/gxsm3-test ; make; make install

3.2.5. Trouble shooting

If your configuration or compilation fails, most likely a dependency is missing and needs to be installed before trying again. We try hard to keep the SVN source tree in a sane state, one which should always allow an error free compilation. All developers are strongly encouraged to adhere to this golden-rule of SVN usage – and as far as I remember the occurrences of broken check-ins are very close to zero and for sure of a very short lasting. So if you experience errors, they are unlikely to be found in the source itself, but rather than in the development environment. A known issue is the fact, that the current GXSM-3.0 configuration script does not cover all checks and error reports if a needed include file or library is missing, thus you will get to the point to start the compilation and then missing include files may be reported.

However, this does not mean it's all *bug-free*. If you are experiencing runtime bugs or crashes, please report them here:

http://sourceforge.net/tracker/?group_id=12992&atid=112992

The following list may help a bit to locate/solve trouble at compilation/installation time:

configuration trouble if the ./autogen.sh fails:

Check the long output for error/missing messages. Check if you have all necessary libraries and development files installed! Especially the gnome-autogen.sh script must be available.

compilation/make/cc Assuming the ./autogen.sh finished with the creation of all Makefiles: At least the first gcc call should be started... Then the most likely errors are generated by missing include files, check for the first occurring error/missing includes!

3. GXSM-3.0 Project Installation

Second, have a look if the files, that are noted missing in your error messages are really absent or just installed in unusual places.

compilation/linking If this fails, something is messed up with your development packages and library installation, usually unlikely to happen. Make sure there is no version mismatch between development/include files and libraries installed/configured!

installation Very unlikely this fails, except your permissions are not sufficient (not root?) or you are running out of disk space (check with df -h).

call gxsm3 Please do not mix up different installations of GXSM-3.0. The installation from SVN and from the APT repository usually use different folders to store the plug-ins and so on. If you intend to use the latest version from the SVN repository, deinstall your deb-packages

```
$ [sudo] apt-get purge gxsm3
```

Also, check the online GXSM-3.0-Forum https://sourceforge.net/forum/?group_id=12992 for additional information and reports by other users! If you need further assistance/help, please make use of the *Help* forum there!

3.3. Getting GXSM-3.0 via APT

If you want to contribute to the GXSM-3.0-project, we strongly recommend to use the latest source code available via the SVN repository on sourceforge.net. But for just using GXSM-3.0 (on an Ubuntu based system) using your package manager APT is the easier alternative.

3.3.1. Adding package repositories

First you have to add the PPA (Personal Package Archive) hosted on launchpad:

```
$ [sudo] apt-add-repository ppa:totto/gxsm
```



On Ubuntu 16.04 LTS you will need to add also the PPA from the Gnome3-team to obtain recent version of the GTK+3.0 libaries.

```
$ [sudo] apt-add-repository ppa:gnome3-team/gnome3  
$ [sudo] apt-add-repository ppa:gnome3-team/gnome3-staging
```

3.3.2. Package installation

Refresh your local database:

\$ [sudo] apt update

and install the GXSM-3.0 package via

\$ [sudo] apt install gxsm3 [sranger-modules-std-dkms sranger-modules-mk23-dkms]

The last command will not only install GXSM-3.0 (and the optional sranger packages in the [] brackets) but also fulfill the required dependencies.

Once you have installed GXSM-3.0 you will be noticed by your package manager (in the latest Ubuntu version this is called Ubuntu Software Center) about updates.

In any case, please do not mix up SVN and APT installation.



3.3.3. Updating

No further action is required. The package manager usually checks frequently for new versions of the GXSM-3.0 package and will offer to update it.

3.3.4. Building packages

The SVN also contains all files to make your own deb-package. To do so checkout the source via subversion as described above. Then enter the folder gxsm3-svn and the subfolder debian (assuming that you checked out the source code in gxsm3-svn) and rename control_<your ubuntu> to control.

Then you can use

\$ dch -i

\$ dch -r

to update the changelog file of the package. In particular, you might want to increase the version number. Then use

\$ debuild -b -I -uc -us

to make your (unsigned) package. To make a signed package use

\$ debuild -b -I -k<your email>

The last command will require the variables DEBEMAIL and DEBFULLNAME to be set, i.e., you may want to add to your .bashrc

DEBEMAIL=<your email>"

DEBFULLNAME=<your full name>"

export DEBEMAIL DEBFULLNAME

Package built this way can be installed via **\$ sudo dpkg -i <package-name>**.

4. GXSM-3.0 Quick Start Guide

In this chapter, we will briefly described to configure GXSM-3.0 after installation to operate one of SoftdB's Open Source SPM controller, i.e. MK2-A810 or MK3Pro-PLL. Please, make sure that you have installed a recent version of GXSM-3.0 and the proper kernel modules before continuing. For further instructions to install GXSM-3.0 see Chap. 3. There is also a Chap. explaining how to upload the DSP binary to your Open Source SPM controller.

In case, you want to start over with ‘fresh’ settings, please follow the instructions below  Hint in Sec. 4.2

4.1. Storing and Restoring Settings

To save and archive/backup simply do this to save it in a file named for example gxsm3-dconf-dump:

```
$ dconf dump /org/gnome/gxsm3/ > gxsm3-dconf-dump
```

and to restore simply read it back – sure this will overwrite any current settings:

```
$ dconf load /org/gnome/gxsm3/ > gxsm3-dconf-dump
```

It is strongly adviced to exit GXSM-3.0 while storing/restoring settings.

To manually inspect it you may use (sure with caution) the dconf-editor as well.



Note

4.2. Delete Settings

If required you can delete all settings stored by GXSM-3.0 in the dconf by

```
$ dconf reset -f /org/gnome/gxsm3/
```

The / at the command is important to manipulate the whole tree and not just a single entry.



Note

5. GXSM-3.0 Tipps and Tricks

5.1. Running Two Instances of GXSM-3.0

Because of GXSM-3.0 and glib's new application management, any attempt to start a new "instance" on the same user session/account will NOT span a new process per default, but connect to an eventually existing GXSM-3.0process – no matter if running with hardware connected or not.

This is unfortunate in case that you want a independent process for just reviewing data on the same login/X11 display while running GXSM-3.0 in data aquisition mode the same time.

To circumvent this complication create a new user account (here simply named "gxsm@localhost") for analysis and run a **\$ ssh -X** (with auto X forwarding) session on your current account while GXSM-3.0 may be running with hardware connection on "this localhost" like shown below. Here "user@spm" is the actual user and machine running the X11 session on this machine with hardware.

If no alternative account exists, simply run as root **\$ adduser gxsm** before.

Terminal output/sesper defaultlsion example:

```
user@spm$ ssh -X gxsm@localhost
```

Within this new session/terminal you can safely start a new instance of GXSM-3.0.

```
gxsm@localhost$ gxsm3 -h no
```

.... now you have a new GXSM-3.0 application running and it will also accept drag and drop from nautilus while in no means disturbing any existing gxsm3 process :)

Part I.

The GXSM-3.0 core

6. Program Start

This chapter describes the very basics of GXSM-3.0 usage: the start of the program. In a modern Gnome3 (gnome-shell based) desktop environment with a proper installed gxsm3 you will find the GXSM-3.0 in the Application menu, can seek via gnome-search command, etc. and place it into your quick start application bar.

However, some of its behavior can be modified using the command line parameters described in section [6.1](#).

6.1. Command line parameters

To start from a command line just type – no special options at all:

\$ gxsm3

Or for temporary hardware disabling, very handy for data analysis on your system configured for hardware usage by default:

\$ gxsm3 -h no

It is recommended to use this as default handler of *.nc Mime-File types, i.e. used by Nautilus to launch GXSM-3.0 while clicking a data file. To list all understood command line options type:

\$ gxsm3 --help

~\$ gxsm3 --help

Usage:

gxsm3 [OPTION...] List of loadable file(s) .nc, ...

Help Options:

-?, --help	Show help options
--help-all	Show all help options
--help-gtk	Show GTK+ Options

Application Options:

-V, --version	Show the application's version
-h, --hardware-card	Hardware Card: no ... (depends on available H)
-d, --Hardware-DSPDev	Hardware DSP Device Path: /dev/sranger0 ... (



6. Program Start

```
-u, --User-Unit
-L, --logging-level
-m, --load-files-as-movie
--disable-plugins
--force-configure
--force-rebuild-configuration-defaults
--write-gxsm-preferences-gschema
--write-gxsm-gl-preferences-gschema
--write-gxsm-pcs-gschema
--write-gxsm-pcs-adj-gschema
-D, --debug-level=DN
-P, --pi-debug-level=PDN
-s, --new-instance
--display=DISPLAY
```

XYZ Unit: AA | nm | um | mm | BZ | se
Set Gxsm logging/monitor level. omit
load file from command in one channel
Disable default plugin loading on sta
Force to reconfigure Gxsm on startup
Forces to restore all GXSM values to
Generate Gxsm preferences gschema fil
Generate Gxsm GL preferences gschema
Generate Gxsm pcs gschema file on sta
Generate Gxsm pcs adjustements gschem
Set Gxsm debug level. 0: no debug out
Set Gxsm Plug-In debug level. 0: no c
Start a new instance of gxsm3 -- not
X display to use

Parameter	Description
-h, --Hardware-Card= <i>type</i>	set up type of hardware no ... depends on available HwIs
-u, --User-Unit= <i>unit</i>	XYZ unit Å, $1e-10$ m nm, $1e-9$ m
--disable-plugins	Disables the loading of plugins on startup mainly for debugging.

Table 6.1.: commando line parameters for calling Gxsm

Right after the options shown above you can list files for opening to the free channels.
All known formats ([12.2212.1812.1712.1312.1512.1012.1612.2112.712.412.19](#)) are autodetected.

¹This is the default, the device location can be configured using the *GXSM preferences dialog, folder Hardware/Hardware/Device*

²The device location can be configured using the *GXSM preferences dialog, folder Hardware/Hardware/Device*, use '/dev/sranger0'.

7. The main window

After startup, the main window appears. The actual user interface provided by the main window depends on the configuration (compare Fig. 7.1. GXSM-3.0 can be configured for use with SPM techniques, which is the default.



Figure 7.1.: Gxsm3 Main Scan Control Window.

The main window provides two different functions: Firstly, it has a menu bar with pull-down menus. These menus provide the user with the usual **File** and **Help** menus which can be found in practically every mouse-driven software piece. Some of these pull-down menus interact with (**Math**) or start-up (**Windows**) other windows. Secondly, the main menu contains a large number of control fields which can be used, e.g., to control an instrument, or just display certain parameters. These control fields are described in the following two sections.

7.1. Understanding the main window's entries: SPM mode

This section explains the contents of the main window (see Fig. 7.1) for Gxsm running in SPM mode. The main window contains from top to bottom the menubar, a taskbar, the scan parameter, view mode, file, and info/comment sections, and a status and progress bar. The scan parameter and info sections of the main window are used both for entering parameters during data taking and displaying them after loading data.

7.1.1. Scan parameters

Each scan or image is characterized by its size and resolution. The size, or *Range XY*, gives the scale of the image like the scale of a city map and denotes the height and width of the scanned area. The resolution is determined by either the distance between the single scan points/pixels given by *Steps XY* or the number of points in X and Y direction given by *Points XY*. Given two of these parameters, the third one can be computed. The check box *Calculate* determines, which of them is calculated by GXSM-3.0. For instance, if *Steps* is checked, a change of *Range XY* results automagically in a new value for *Steps XY*.

The parameter *Offset XY* determines the distance of the zeropoint of the image coordinates from the center of the physical scanrange. The actual location of the zeropoint within the scan depends on the source of the data. If the data was acquired using GXSM-3.0, the zeropoint is the middle of the topmost line. Using *Rotation*, the imaged area can be rotated. Both inputs using numeric values and the scrollbar are possible.

GXSM-3.0 can be used to do spatially resolved spectroscopy (“probing”) and time dependent measurements (“movies”). Channels containing probing (or time dependent) data are essentially three-dimensional (3D) datasets. In these 3D datasets the X and Y coordinates correspond to the 2D position like in conventional SPM images. The third dimension can be the voltage V or the time t. GXSM-3.0 displays only one slice corresponding to one V or one t value at a time. *Layers* denotes the number of points in the V direction, *Time* in the temporal direction.

VRange Z and *VOffset Z* are used for the visualization of the scan data. They do not influence the data itself. See also Sec. 9.1.1.

7.1.2. File and user information

For the users convenience, the filenames for saving new data are automatically generated. The filename is set together using the *Basename* and the scan number. The default for the *Basename* is the login name of the GXSM-3.0 user. The scheme used for generating the filename from the scan number can be configured in the Preferences on the

tab *User*. The scan number is followed by “-M-” if the image contains additional information besides the bare 2D image like events and point probes. The next part of the file name indicates the scan direction: Xp or Xm. Finally, the channel name is attached to the file name. The channel name can be configured on the tab *DataAq* of the Preferences window. *Auto Save* is checked, each new scan is automatically saved after the scan is finished.

During image analysis it is often convenient to save the “enhanced” images using an easy to memorize name. Nevertheless, it is often necessary to get back to the original data. For this purpose, *Originalname* shows the name of the original data file. This feature works only for files saved using GXSM-3.0’s NetCDF format.

The *Comment* field allows adding comments to scan data, e.g. the name of the sample. Again, saving this information is supported best for the NetCDF file format.

If your dataformat is not natively supported by GXSM-3.0, but can be exported to ASCII, consider using `ncdump` and `ncgen` to create a nc-file from your data. Running `ncdump` on any GXSM-3.0-nc-file shows you which parameters are necessary, insert your data in the ASCII output and revert the ASCII-file back to NetCDF using `ncgen`.



Hint

7.2. Drag and Drop

Gxsm accepts all loadable files via ‘drag and drop’, e.g. from the Nautilus and understands VFS file paths. Even dragging URL’s pointing to loadable files on the web is possible.

If you drop a file on a channel-window, it is loaded into that channel. To create a new window with a new channel bound to it, drop the file above the main window.

7.3. Keyboard-Accelerators

Most common used action on a scan-view are assigned to keyboard accelerators, this is indicated by the Key-Symbol on the right side of a menu entry. (See pull down/pop-up menus on scan 2D view). F2 for example triggers a auto-display (auto scale to min-max of all data or via active rectangle area selection).

8. Channels

One of the most important features of Gxsm is the multichannel-capability. Multichannel-capability describes the simultaneous data acquisition and display from different sources. You may e.g. at the same time measure topography and friction-forces with the AFM. Additionally to the simultaneous acquisition of different signals, the multi-channel feature of GXSM-3.0 can be used to load multiple images and, e.g. compare them, or apply more complex operations to them. Furthermore, it serves as a history mechanism during image manipulation, because the result of any mathematical operation on one channel does not overwrite it but is stored in a new channel.

Channel Selector					
Ch	View	Mode	Dir	AS	
1	Grey 2D	Topo	->	<input checked="" type="checkbox"/>	
2	Grey 2D	Mix-In 0-ITunnel	->	<input type="checkbox"/>	
3	Grey 2D	McBSP Freq	->	<input type="checkbox"/>	
4	Grey 2D	McBSP Ampl	->	<input type="checkbox"/>	
5	Grey 2D	McBSP Phase	->	<input type="checkbox"/>	
6	Grey 2D	Off	->	<input checked="" type="checkbox"/>	
7	Grey 2D	Off	->	<input checked="" type="checkbox"/>	
8	Off	Off	Off	Off	

Figure 8.1.: Channelselector Window.

8.1. The channel dialog

The channel dialog pops up selecting Ch. Sel. in the Windows menu of the main window. You can use it, to select the displaymode (View) and the source for data acquisition (Mode) for any channel.

One of the following modes can be chosen, see [9.2](#) for details:

8. Channels

1. **No:** During data acquisition no data is displayed – background storage/saving only. You can switch view mode any time.
2. **Grey 2D:** The data are displayed as a grayscaled/false-color image.
3. **Surface 3D:** Three/multi dimensional data/scan viewer. OpenGL (4.0 minimum) based. Can display volume data and slices of multilayered data.
4. **Profile 1D:** Profile view of the current or all lines.

Usually you will use the mode **Grey 2D** for data acquisition. If you want to see the line profiles of the actual scan line, right-click on the window of the channel, select **view** and activate **red Profile**.

For processing of data several modes are available:

1. **Off** or **On**. Off deletes the channel.
2. **Active.** The most important mode, it sets the active channel. All image manipulation is done using this channel. Only one channel can be **Active** at a time.
3. **Math Channel**, which stores the result of the last operation is automatically called **Math**.
4. **X** Needed for several math/image-manipulation, that need more than one source.

If you want to activate a channel for data acquisition, please select in the second column the channel name, i.e. **Topo** or **ADC0_ITunnel**. This channel will be used as a target for a scan. The toggle $->$ and $<-$ in the third column defines the scanning direction at which data are collected. Thus **+Topo** means measurement of your topography during movement of your scanhead in **+X** direction. By choosing **>** or **<** an *experimental* mode is activated in which each scanline is scanned twice, i.e. this mode is used for magnetic force measurements.



Using the MK3-A810 DSP with flexible signal configuration 4 special modes can be configured to acquire any available signal. See **DSP-Control**.



The names of the input channels can be customized (see **10**).

9. Visualization

As a multi purpose 2D/3D data visualization, acquisition and manipulation system GXSM-3.0 provides a set of powerful graphical presentation modes. Available are 2D grey/false color, profile and 3D representations. The internal objective structure of GXSM-3.0 allows to switch in between all available “View-Modes” on the fly and at all times, even while scanning. In addition, on the fly data to color space mappings are provided for viewing data.

9.1. Data display modes

Due to a possibly huge Z-value range and often small local signal variation special data transformations (e.g. for mapping to color space) are needed to visualize these features. Therefore from *Quick*, *Direct*, *PlaneSub*, *Logarith.*, *Horizontal Diff* and *Periodic* view modes can be selected via GXSM-3.0 main window in “View” or via the pop-up window of the data window itself. The raw data which are saved for example in the nc-files are not affected by a change of the data display mode.

Quick for each line a line regression is estimated from a subset of points (for speed) and data is slope and offset corrected for visualization.

Direct data is displayed ‘as is’. Only a linear transformation for shifting and scaling data into view range is performed using the view range “VRange Z” and view offset “VOffset Z” controls located context menu of the scan window.

Direct HiLit Same as Direct but marks the lowest and highest values.

Plane data is displayed after background correction by a bi-linear/offset function (plane-correction) defined by three Point-Objects. The resulting Z values are linear transformed to shifting and scale data into view range as it is defined by the view range “VRange Z” and view offset “VOffset Z” controls located in the context menu of the scan window or automatically calculated to fit a selected rectangular area.

Horizontal shifts the lines, so that their average is zero.

9. Visualization

Periodic works like *Direct*, but the available colors are used periodically repeating (mapping: linear Z transform as in *Direct* modulo number of colors).

The periodic mode is only via. pop up menu accessible.



Logarithmic a logarithmic scaling can be applied to data. It used for very high dynamic data such as diffraction data (e.g. SPA–LEED or XRD) where background variation should be visible, while high intensity peaks are not out of display range. The translation happens in the following way:

$$\text{Grey/Color number} \propto \log(1 + |\text{Data} - \text{Min}| \times \text{Contrast} + \text{Bright}),$$

with *Min* being the smallest value (omits negative values), Contrast and Bright are calculated from VRange Z and VOffset Z to fit data into desired Z Range window.

Differential displays a averaged and weighted X-derivative, similar to the Koehler filter 15.4, but using a smaller averaging range (16 pixels).

To change the view mode just select a *main window radio button Quick / Direct / Plane /Logarith. / Diff..*. These data display modes are applied to all visualization modes, see 9.2.

9.1.1. Scaling and shifting data view range

View Range Z* and *View Offset Z

The parameters *VRange Z* and *VOffset Z* found in Gxsm main window are controlling the always applied linear transformation of the data to grey or false color mapping.

VRange Z entry sets the data Z-Range which should be mapped to full color space.

VOffset Z entry sets the offset relative to averaged data range. E.g. if you data represents a stepped surface (terraces assumed to be horizontal aligned) and you set the *VRange Z entry* to approx the step height you will get only one terrace in view range, others are pinned at max/min color, using the *VOffset* you can select a terrace to be viewed with high contrast.

Usually using the *toolbar/Autodisp* button will do the job automatically for the whole scan or if a rectangle object 9.2.2 is selected the selected area will be scaled automatically to full color space available. *VOffset Z entry* is always set to zero by *toolbar/Autodisp*.



If your scan has some distortions/spikes messing up the automatic min-max range, try enabling the *main menu/View/Tolerant Auto Display*-option. This will compute the view-range via an automatic histogram analysis in a progressive way.

In SPA-LEED mode: CPS High – Low

In SPA-LEED mode the scaling of CPS-High/Low can be set.

9.1.2. Using a palette: false color mapping

9.1.3. Custom and GXSM-3.0 provided palette

The color palette is simply a one dimensional PNM-bitmap file, that can be selected through an entry in the preferences dialog, call *main menu/Settings/Preferences*, and is located there *GXSM preferences dialog, folder User/User/Palette*. Activate *main menu/View/Palette* to make use of the selected palette and refresh the view via the *GXSM-3.0 toolbar/Autodisp*.

GXSM-3.0 will make use of up to 8192 palette entries automatically.

Here a short description of the PNM file format:

```
P3
# CREATOR: The GIMP's PNM Filter Version 1.0
1024 1
255
R
G
B
R
G
B
...
.
```

The file is expected to have at least 1024 RGB-entries (R, G, B values in the range 0 ... 255) just following the header, but no more than 8192 are accepted. Use *gimp* to generate this conveniently.¹

GXSM-3.0 can (re)load a new palette at any time. Choose a new one as described above and refresh via the *GXSM-3.0 toolbar/AutoDisp*.

Additional custom palette files can be loaded from any location, put the full path to it into *GXSM preferences dialog, folder Paths/Paths/UserPalette* and press OK, then you will find it in the list of available palette files in *main menu/View/Palette* on next call of the preferences.

¹1. Palette Editor, 2. fill the picture with 1024x1 pixel, 3. save as *MyPalette.pnm* (ascii-type).

9.1.4. Additional Information

In particular during data acquisition it is convenient to have a 2D representation of the data and a line graph of the last scan line(s). You can get this by activating *main menu/View/Red profile*. You can also activate the display of additional meta-data of the image by *main menu/View/Side Pan*. That will show you on the different tabs the parameters/variables of the nc-file, Probe Events, User Events, and a list of the Objects.².

9.2. Visualization modes

The visualization mode can be switched at all times, see 8.1 for details. The default mode is always *Grey 2D*.

9.2.1. View “No”

This is not really a visualization mode, it just prevents any display and saves memory and CPU power if a data channel is just to be acquired in background in a blind manner.

9.2.2. View “Grey 2D”

The default mode for viewing 2D data. It allows using a simple grey scale data representation or using of any available false color palette. By default the image size is scaled to achieve a good fit on your screen. But any *zoom* (magnifying by number) or *quench* (down-size by 1/number) is possible.

Window title information

In the window title of the channel view shows for example:

Ch2 : X+ Topo, * Q1/5 Short [2]

Ch2 channel number is 2.

X+ Topo, * this channel is in data acquiring mode, else here appears the path/filename

Q1/5 the data view is currently down-sized by factor of 5.

Short [2] the current data is of type short (2 bytes).

²The list of objects is currently not auto-updating (Gxsm-2 1.40.0 ‘Sandy Experiment’)

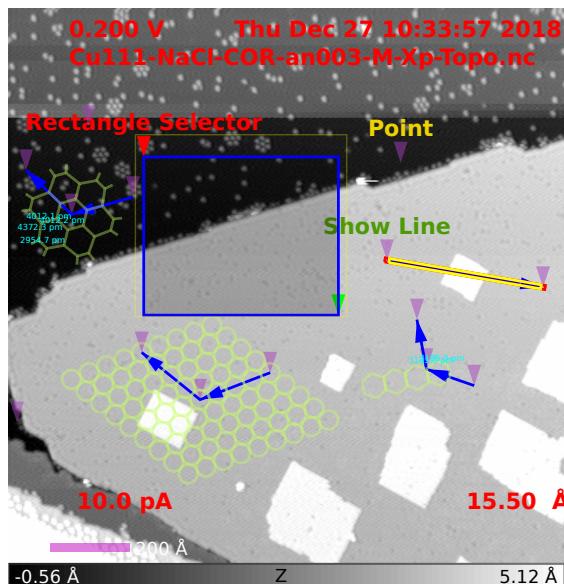


Figure 9.1.: Grey 2D view with commonly used objects and other gizmos OSD scan info of selected parameters is enabled here (red text). The currently active object (colored nodes) is the Rectangle object. Also enabled at time of this pdf data view export is the legend (scale bar and Z legend, below option).

Popup Menu

The *image popup window* by clicking *mouse button 3* on the image window activates several convenient options and displaying tools:

Activate set the current window (channel) to active state (used for math, etc.).

Autodisp activates this window and performs an automatic scaling.

Mode switch the channel mode, see 8.1.

File load/save scan data in this channel,

load/save objects (9.2.2),

Get Info (about this scan, it has to be (re)loaded before, if it was currently acquired),

bring up a print dialog for this image,

close channel (removes this data from memory and closes the window).

The current view including all objects, legned(s), etc. can be exported as png bitmap and also as vector graphics as pdf - -this allows for example to use inkscape to edit the meta data and objects or simply remove them.

Edit Copy, Crop (a marked rectangle, creates a new scan), Zoom In/Out (use a rectangle object to select area to zoom in or zoom out to, it resizes data into a new scan).

View change view mode, see [9.1](#).

Objects set type of object to create, see [9.2.2](#). *Remove all* removes all objects.

Events controls what kind of Events are displayed.

Objects

Only in the Grey 2D mode are coordinate measurements and selections available. Basic objects are Point, Line, Rectangle, PolyLine, Circle, Ksys (Coordinate System used for atom grid overlays/measuring and also new basic PAH molecule models can be selected) and Parabel. Some are available with the prefix “Show” in the menu, which means the data along a path, such as line, circle and in a case of a point the (possibly) available 3rd layered dimension in depth of scan, is shown using the profile view ([9.2.4](#)).

Select between the different types of objects by the *Objects popup window*. You add an object by left-clicking into the 2D image. You can remove an object by clicking with the middle mouse button on it.³ You can remove all visible objects by *Objects/Remove all popup window*.

By left-clicking on the object handlers you can modify them. In the case that the handlers are two small you can change their size in the menu *main menu/Settings/Prefereces*. Look for the settings *GXSM preferences dialog, folder GUI/HandleLineWidth*, *GXSM preferences dialog, folder GUI/HandleSize* and *GXSM preferences dialog, folder GUI/ObjectLineWidth*.

For getting the coordinates and Z-value of the mouse position, just hit the middle mouse button to get a cross hair pointer and move around to measure. You can switch the shown units from default unit to pixels using the toggle switch *Grey 2D view popup window View/Pixels*.

While moving objects, the objects properties coordinates (angle, length, ...) are shown in the status line of the window. This is useful for measurements, etc.. More precise measurements are possible using the *Show Line* object, which is updated on the fly while moving the line interactively across the image.

Objects are also used to provide some math actions with coordinates.

All object coordinates can be viewed and changed using the object popup menu (middle mouse button on object). The objects save/load (File menu of view popup) allows saving objects and reloading those later to any scan at absolute coordinates (including offset).

³If your notebook or mouse has just two buttons click both at the same time.

While scanning is in progress the current scan-line can be viewed as profile, use *Grey 2D view popup window View/red Profile* to toggle it on and off⁴.

Scan Events

Scan Events, available since GXSM-2.0 V1.6.0, can be any kind of Event or Experiment related to the current tip position and time. Per definition a GXSM-3.0 “Scan Event” holds a position and possibly more specific information like what happened. I.e a simple bias change (User-Event-Entry) or a full set of probe data (Probe-Event-Entry).

The number of Events per scan is unlimited. And Events are always getting attached to the current active scan. The Scan Pop-up menu Events allows to show Probe Events or User Events. Because there can be many thousands of Probe Events using the automatic rastered probing, the displayed markers can be limited in number and distance from a certain point, indicated by middle-mouse clicking.

Probe events are sourced by the SRanger Vector Probing (spectroscopy, manipulation, etc.), single and automatic (rastered).

Enabling the displaying of Probe Events will add some controls to the scan window for selecting the range and number of Events to be displayed. Also the setup for data plotting shows up.

– to be completed –

Tips

Define your personal hot-keys with the pop-up, just select an pop-up entry and press a key!

E.g: Set $\boxed{=}$, $\boxed{-}$ for *Grey 2D view popup window View/zoom in,out!*

9.2.3. View “Surface 3D”

A 3D model of the data, using the Z-value as height, can be viewed using the GL/Mesa renderer. Hardware acceleration is used by the GL subsystem if the X-server provides it. This works even while scanning, but consumes a lot of CPU power and slows the scanning process down.

Pop-up Menu

The *3D view popup window* is activated by clicking *mouse button 3 on the image window*. It includes some of the options already known from the Grey 2D popup menu,

⁴Showing the profile while scanning can delay the scanning process on slower machines, so switch it off if not needed! To save CPU power you can disable auto-scaling and tick marks in the profile view.

9. Visualization

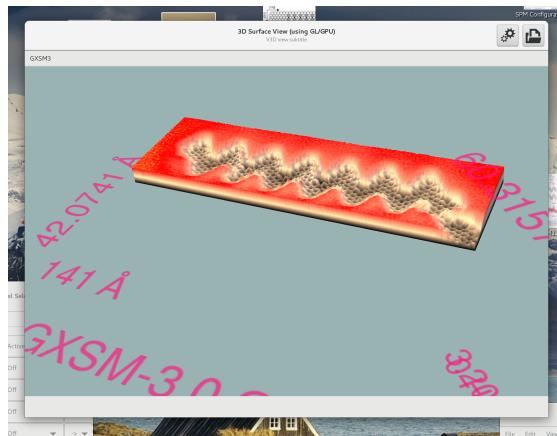


Figure 9.2.: 3D-view example.

see 9.2.2. The popup offers some options for controlling the 3D view:

GL Options Quick access to the display options

- Zero Plane* (enable/disable showing of the “bottom box”),
- Mesh* (use meshing instead of solid rendering),

Scene Setup Open the control panel for 3D Scene Setup, refer to next section.

Using the mouse for rotation and translation is possible, but a bit unusual. First, set the window size not too huge and the rendering resolution (see Scene Setup) above 1 to speed up things. Then press the left mouse button and move the mouse carefully around for rotation. Use the middle button for translation. If you find this impractical, use the scene setup to set rotation and translation!

Scene Setup



few preliminary notes on 3D view setup

This dialog allows a sophisticated 3D rendering configuration.

Mouse/wheel motions: rot, distance, translation. See "Help" buttons!

View Setting of the model (surface) orientation and view. The coordinate system and view controls: scan xyz dimensions are mapped into a 1x1x1 cube (with y may have a aspect if non square) in GL coordinates (GL-XZ is Gxsm-surface plane), Z-scale (absolute, 1=max-value-1x1x1 cube dim or relative angstroem, 1 will scale Z the same as X.)

RotationX/YZ Setting of the model (surface) orientation in world space.

RotationPreset Select "Top-View", then elevation is above, and distance "in front of center"... if you rotate object/surface, it changes accordingly. And more default views as named. The "Manual" view will leave XYZ rotation settings untouched vs. pre sets which will reset the rotations to the selected view at any update.

LookAt Predefines look at positions.

Translation model translation in world space.

FoV Field of View.

Distance Distance from model center in "Top-View".

Elevation Elevation above model center in "Top-View".

HeightScaleMode Height scaling mode. Relative range: 1 equals full data range fits 1x1x1 cube. Absolute Range: only meaningful for XYZ data with same units. 1 equals Z scale matching X scale to fit 1x1x1 cube.

fix typo in GUI



Tskl N/A

SliceOffset Z-Offset of slicing selection in volume mode.

VertexSource Flat: no displacement, Direct Height: data is used directly as displacement source, ViewMode Height: view mode transformed data is used. You must select the view mode before entering the 3D View as data is loaded to the GPU only once. Note: Toggle 3D view back to 2D to update. y: experiments (N/A), Channel-X: Channel X data is used if available. X/Y/Z-slice: slicing views for volume data with value dimension. Switch to Volume Shader (ShadeModel=Volume) in Surface Material Tab. Volume: render 3D data as volume, also use Volume Shader. Scatter: experimental / cube projection, N/A.

SliceLimiter : Volume view define the detail and range for view aligned projection planes to be generated. A default cube data set will fit into a 0 .. 1000 range, typically 50..100 slices are sufficient, but you can opt for more detail. Computation intense. Example: `[start = 0, end = 1000, step = 10, 0]` To project beyond you can extend the ranges to for example `[-500, 1500, 10, 0]` – may be necessary to cover odd aspect ratio data. Also you can intentionally use it to "cut-open" a volume and only render part of it - i.e. `[500, 1000, 10, 0]`!

SlicePlanes not used at this time.

Light Light sources are ambient and a single "sun" defined by a light direction vector. TipView: enable/disable rendering of a virtual "tip".

Surface Material Color mapping/shading: data is always automatically mapped into a 0..1 full scale range equivalent. Gxsm palette is mapped to 0..1 full range. Controls are offset and scale.

Advanced multi dim data handling (you need a value-dimension with data) if data is in "time" you can run the transformation plugin "multi-dim transpose" to swap time to value.

Slicing and volume view, and the use of transparency settings. Additional transparency for blending. The transfer function for (1-alpha) blending is transparency-offset+transparencyscale*value. Using Palette color as above.

Annotations Title, Labels, etc.

Render Opt. Ortho Orthographic projection is used when set to true.

TessLevel Use the render option "TessLevel" to control the tessellation level, i.e. the detail of rendering. A default triangulated flat base plane spawning a 128x128 grid is used and is subdivided on GPU level up to $64 \times$ controlled by the tessellation factor. On GPU level the selected data as used for height field displacements any any resulting given resolution given by the tessellation factor. Only if that happens not to be sufficient you should increase the "BasePlaneGrid" size, but there is usually no need to do so as 128×64 is bigger than most data sets pixel resolution. Not recommended for performance reasons. You may even want to reduce this for less powerful GL hardware. For this change (BasePlaneGrid only) to take effect you have to click apply and close/reopen the 3D view to rebuild the new geometry.

BasePlaneGrid See TessLevel.

ProbeAtoms Number "atoms" created for tip visualization.

Mesh false: regular 3D rendering. true: use a mesh.

ShadingMode Debug Shader only. Debugging and special shading mode/experimental and customizable for GL experts to play with. You can hack even the tess-fragment.glsl shader for custom fun!

ClearColor Select the scene background color.

Fog... Fog settings – debug shader only, use debug shader, and debug mode "5" to apply fog.

Cull Culling means back side triangles of the surface (hidden in top view) are never drawn, mean if you look from below, it's like an roof with no shingles.. faster and normally better. Just disabled in volume/slicing mode obviously..

9.2. Visualization modes

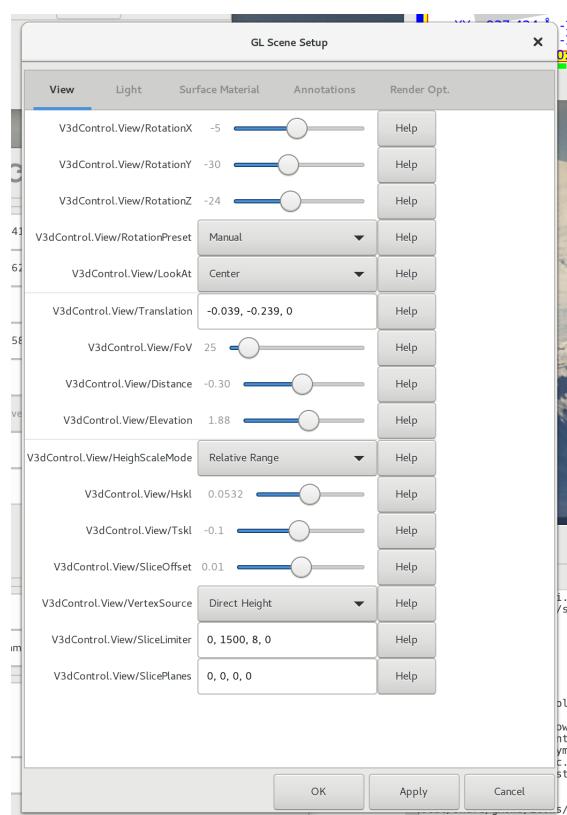


Figure 9.3.: 3D Scene View Control, typical settings for a surface terrain like rendering.

9. Visualization

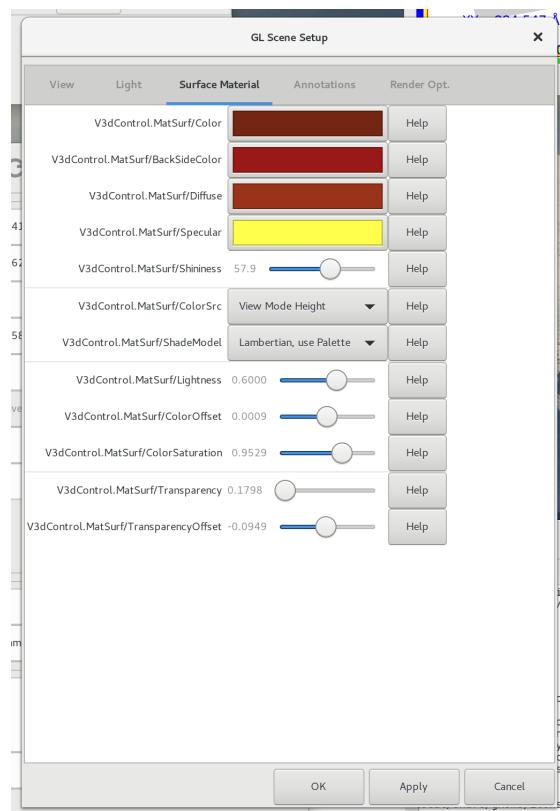


Figure 9.4.: 3D Scene View Material/Color control, typical settings for a surface terrain like rendering.

9.2. Visualization modes

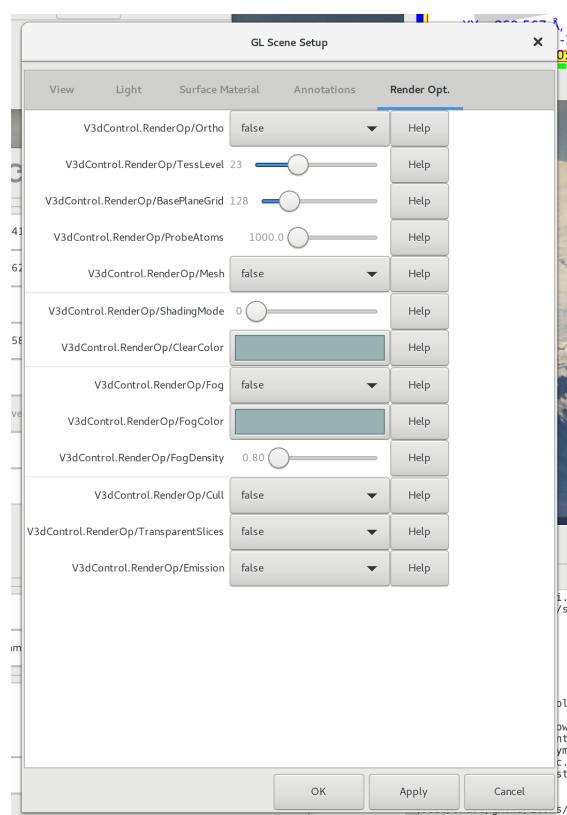


Figure 9.5.: 3D Scene View Options, typical settings for a surface terrain like rendering.

9. Visualization

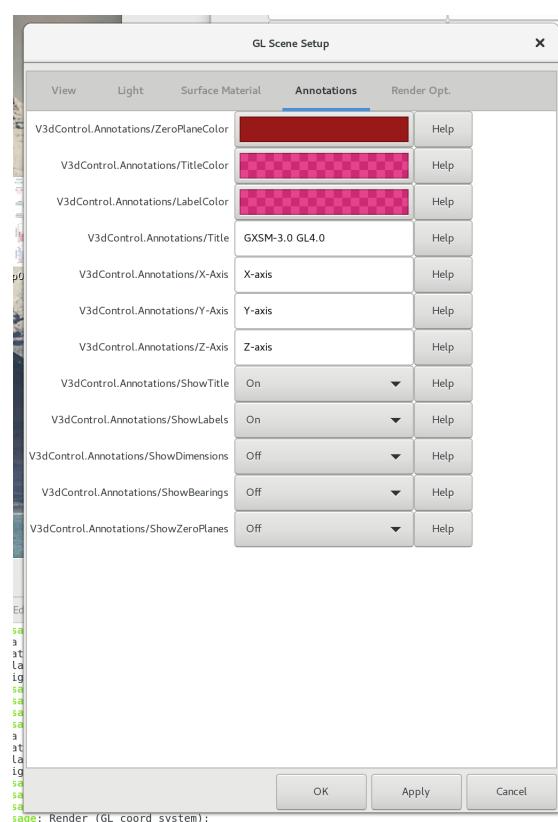


Figure 9.6.: Scene annotations.



Figure 9.7.: Volume rendering example.

9. Visualization

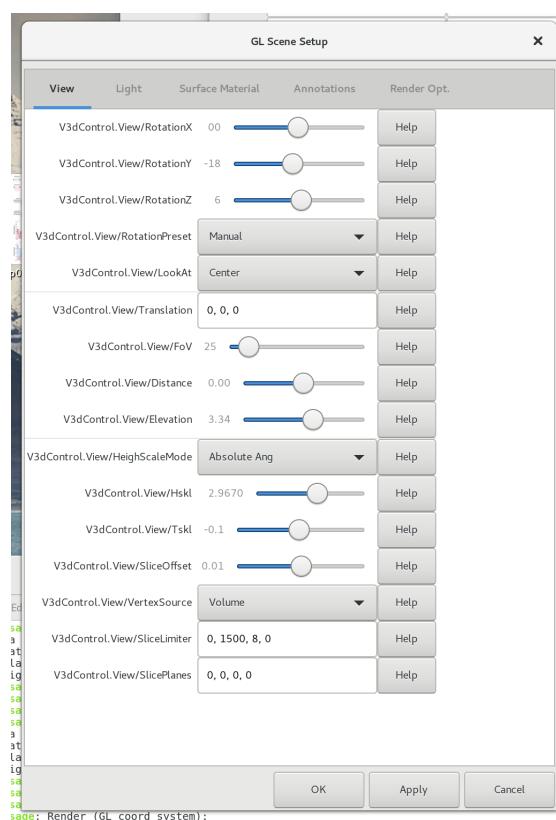


Figure 9.8.: 3D Scene View Control, typical settings for a volume rendering.

9.2. Visualization modes

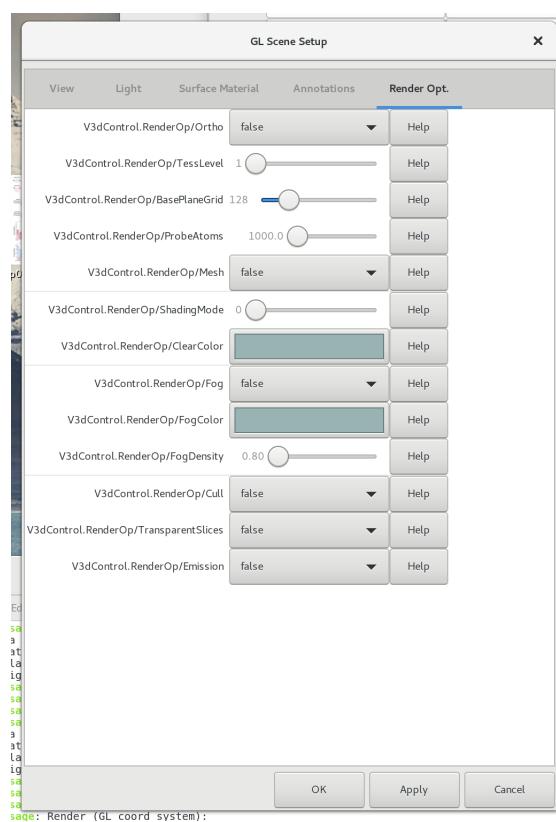


Figure 9.9.: 3D Scene View Option, typical settings for a volume rendering.

9. Visualization

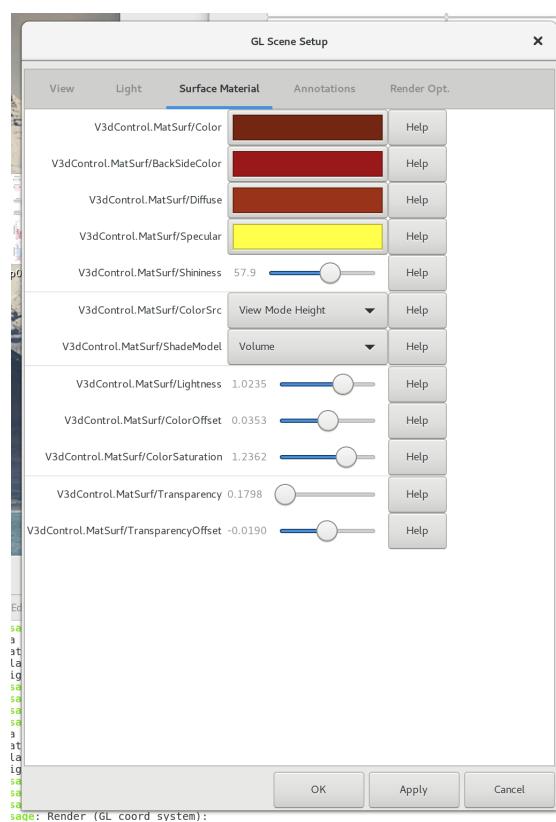


Figure 9.10.: 3D Scene View Material/Color control, typical settings for a volume rendering.

9.2.4. The general purpose “Profile 1D” view

For several purposes Gxsm can show data using the a general purpose *Profile-View* representation (XY-plot).

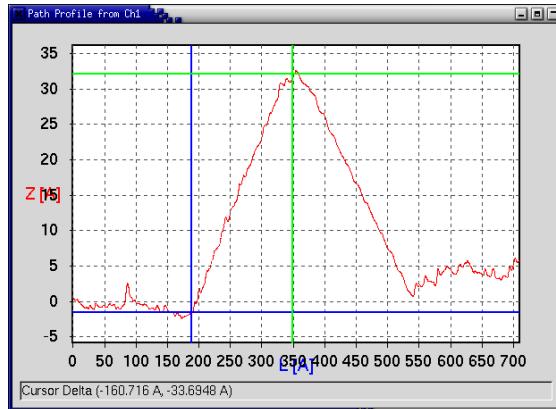


Figure 9.11.: Profile as used by the Show-Line object (old version gxsm2).

In the view-mode *Profile 1D* it shows the currently measured line or the first or via popup menu (*Data Select...*) selected line of the data set. The *All* option loads all profiles at once, be careful using this with huge scan (lots of memory needed). The scaling is (with respect to shown tickmarks) only correct for all profiles, if the option *Y Scaling/hold* or *Y Scaling/expand only* is activated⁵.

Pop-up Menu

The popup menu of the *Profile View* offer the following options:

File File operations:

Open, Save File handling.

Print Printing via script. See subsection Profile Printing.

Activate If used as “channel view”, to activate this scan.

Options This menu provides toggle buttons for the following display options:

Y lineregression Enable to use a “on-the-fly” line regression.

⁵There is a bug with refreshing the toggle indicators to the initial setting upon first menu activation, to make sure the setting is correct please toggle the option once! Gxsm saves the last setting of all these option.

9. Visualization

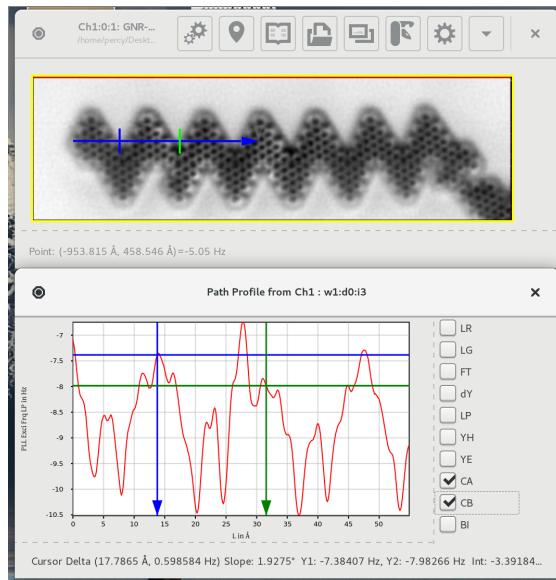


Figure 9.12.: Gxsm3 Profile View of data section with cursors A,B enabled for measuring.

Symbols – not yet available –

Legend – not yet available –

Tickmarks Enable display of tickmarks

no Gridlines Disable gridlines

Y Scaling Several Y scaling options⁶:

Y logarithmic Set Y axis to logarithmic scale.

Y hold Hold Y scale (prevents automatic rescaling).

Y expand only Expand Y scale only.

Y auto Force auto scale.

... A set of manual Y offset shift (move upper/lower bound) and Y zooming (in/out) options.

Hint For quicker adjusting pull of the menu (drag the menu apart by clicking the dashed line) and place it beside!

X Scaling – not yet available –

⁶in case no data is appearing, one of the following conditions may be present: negative values (using log.) or zero Y or X range.

Cursor This menu allows to enable up to two cross hair cursors (blue and green). Use the left mouse button to drag the cursors around, click on the vertical cursor line to grab it! The status line will show (one cursor) the coordinates or (two cursors) the differential coordinates for measurements.

The Cursor menu (tear it off!) allows to auto position the cursor on local minima or maxima to the left or right of the current position. Or just to skip to the next/previous data point.

Data Select Select/walk through a set of multiple data lines or show all.

Once selected all, you can't undo this yet, reopen the “Profile View” therefore.



1D data file format

Via the *File* menu the profile data can be saved in Ascii format. The file is preceded by a simple header using commentary lines and a keyword. The data itself follows using the following format:

Height Profile Data:

```
# ASC LineProfile Data
# Date: Tue Jun 18 17:21:42 2002
# FileName: /tmp/HutProfile.asc
# Len      = 405.843A
# tStart= 1016686983s
# tEnd   = 1016687662s
# phi    = 39.9441\B0
# dX     = 0.235244A
# dY     = -0.197002A
# dZ     = 0.0139465A
# Xo     = -1079.55A
# Yo     = 0A
# Anz    = 1323
# NSets  = 1
# Xunit  = A
# Xalias= AA
# Xlabel= L
# Yunit  = A
# Yalias= AA
# Ylabel= Z
# Title = title not set
```

9. Visualization

```
# Format= X[index*LineLen/Nx] Z[DA]... InUnit: X[Xunit] Z[Zunit]...
0 -103 InUnits: 0 -1.43649
0.30676 -105 InUnits: 0.306992 -1.46439
0.613519 -107 InUnits: 0.613983 -1.49228
0.920279 -107 InUnits: 0.920975 -1.49228
1.22704 -106 InUnits: 1.22797 -1.47833
1.5338 -104 InUnits: 1.53496 -1.45044
...
```

STS Data:

```
# ASC LineProfile Data
# Date: Tue Jun 18 17:24:59 2002
# FileName: /tmp/STS.asc
# Len    = 0A
# tStart= 1024442686s
# tEnd   = 1024442686s
# phi    = 0\B0
# dX     = 0A
# dY     = 0A
# dZ     = 1A
# Xo     = 0A
# Yo     = 0A
# Anz    = 500
# NSets  = 2
# Xunit  = V
# Xalias= V
# xlabel= Bias
# Yunit  = nA
# Yalias= nA
# ylabel= I
# Title = STS [local Spectroscopy]
# Format= X[index*LineLen/Nx] Z[DA]... InUnit: X[Xunit] Z[Zunit]...
0 0 0 InUnits: -2 0 0
0.02 0 0 InUnits: -1.99198 0 0
0.04 0 0 InUnits: -1.98397 0 0
0.06 0 0 InUnits: -1.97595 0 0
0.08 0 0 InUnits: -1.96794 0 0
...
```

Simple Shell Script for extracting just X and Y data:

```
#!/bin/bash

for file do
    awk '{ if ($1 != "#") { print $5, $6}}' $file > $file.dat1
done
```

Gxsm can reload this type of files using just the *Open* command. The file should have the `.asc` extension to be recognized as a profile data set.

Profile Printing

Starting from the *File* menu you will find the *Print*-entry. From here you have access to six user commands. These commands are simple shell scripts that receive a temporarily created version of the 1D profile as argument and can do various things with this profile (not only print). There are six example scripts in the `profileplot` subdirectory.

1. The first script runs *gri* to process your profile and will create a postscript file of your print.
2. The second script runs *gri* to process your profile and will create a postscript file of your print. The data are modified with an embedded Python script to calculate STS.
3. The third script runs *gri* to process your profile and will create a postscript file of your print. The y-axis is log'ed here.
4. The fourth script runs *gri* to process your profile and will create a postscript file of your print. This is a copy of the first script. Use this to experiment.
5. The fifth script runs *xmgrace* to show your profile. Remember that Gxsm will freeze while *xmgrace* is running, so don't use this option as long as a scan is running.
6. The sixth script runs *python-matplotlib* to show your profile. You can export to many bitmap formats from this. Python-matplotlib is not (yet; check it out) part of debian testing (Feb 05)⁷.

These scripts can run standalone (i. e. without Gxsm running). Simply run them with an ASCII-1D-profile as argument like this:

```
/path/gxsmplot1d-1.gri.sh myprofile.asc
```

⁷Follow the instructions on the matplotlib-website for installing. It's easy. Simply add one line to `/etc/apt/sources.list` and run `apt-get update && apt-get install python-matplotlib`

9. Visualization

One can think of many different things these scripts are capable of. They can run any calculation on the profiles. These scripts are designed to run together with the *Show Line* option, not with the 1-D profile view in the channel selector. If you use the latter you will always receive a plot of line 0.

You can run your own scripts by modifying the entries User/command[1..6] in the preferences.

10. Configuration

All your settings are saved using the gconf database¹.

The configuration druid uses the gconf-database and the save-values command will rewrite it, just like the Accept and OK button in the preferences.

With special care the gconf database can be edited, to set defaultvalues. To restore a totally screwed configuration simply remove it. The next time you start GXSM-3.0 the configuration wizard will pop up to create your settings from the defaults.

With Settings/Preferences (dt: Einstellungen/Einstellungen) GXSM-3.0 can be configured during runtime, the changes are (as seen above) saved in `~/.gnome/gxsm`. This file can be copied to any new user, who wants to share the same instrument.

Any of the settings provides a small help text for your information.

10.1. Hardware folder

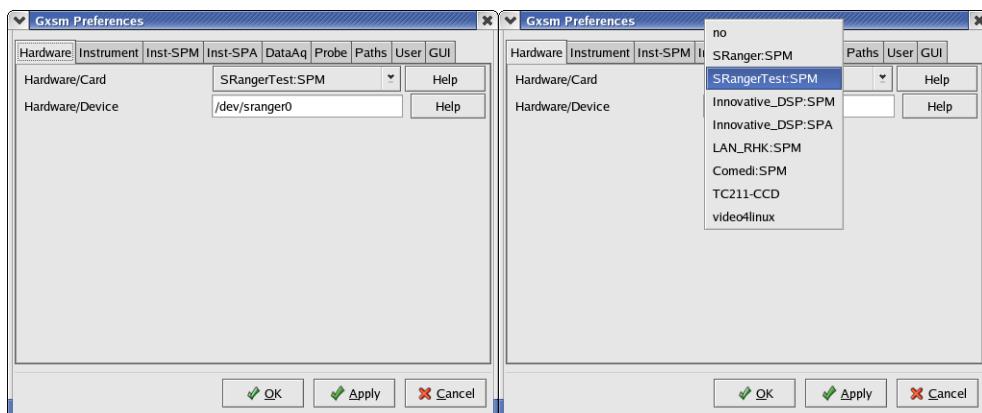


Figure 10.1.: Screenshot of the GXSM-3.0 preferences folder, hardware page. Here the connected hardware type and device is specified. Shown are the settings for the Signal Ranger.

¹GConf: try the gconf-editor to explore the GXSM settings, path: /apps/gxsm2!

Hardware/Card Choose hardware: see also section [23](#) about Hardware-Interface (HwI) plug-ins for specific details.)

no no hardware connected, for data analysis, using internal dummy-mode.

SRanger: SPM Signal Ranger is connected, a running SPM DSP code is expected.

Innovative_DSP: SPM A Intelligent Scanning Probe Hardware Module is expected at the device. (pci32.o, pc31.o at /dev/pcdsp (see later) + running xafm.out on DSP interface board

LAN_RHK: SPM use the RHK internet device

... some more experimental HwI modules may show up.

Hardware/Device Path to the device used by the specific hardware. Typical are /dev/sranger0, /dev/pcdsp, ...

10.2. Instrument folder

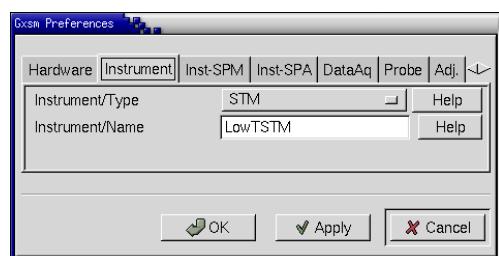


Figure 10.2.: Gxsm preferences folder, instrument page

Instrument/Type Select one of STM, AFM, SARLS, SPALEED, CCD. This determines the "look" of the main window.

Instrument/Name Any name you want to associate with your instrument. Please limit to 30 characters.

10.2.1. Inst-SPM folder

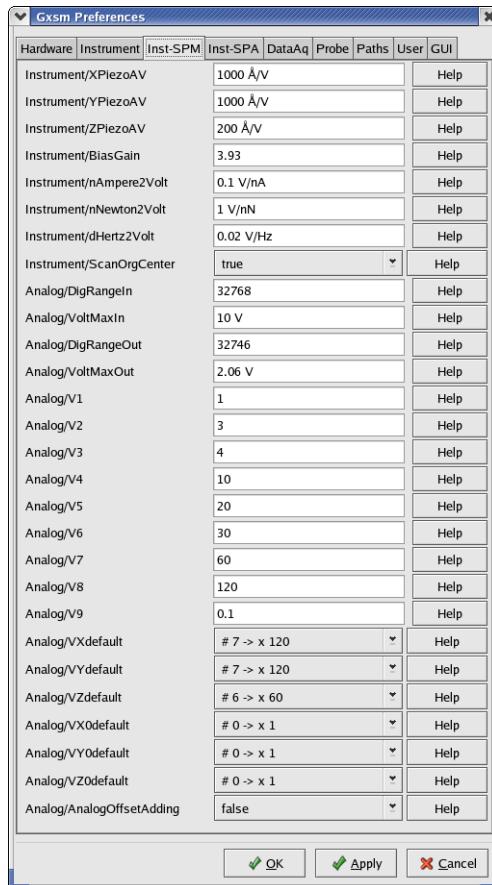


Figure 10.3.: GXSM-3.0 preferences folder, instrument SPM page

Instrument/X,Y,ZPiezoAV SPM Piezo sensitivity in A/V.

Instrument/BiasGain Gain of the bias-volatge.

Instrument/nAmpere2Volt Tunnel amplifier sensitivity: factor*Volt=nA

Instrument/nNewton2Volt for AFM Setpoint in nN, 1 for 1nN = 1V

Instrument/dHertz2Volt NC AFM Setpoint

Instrument/ScanOrgCenter Specific to hardware setup

Analog/DigRangeIn positive maximum value for X,Y,Z,... AD-conversion with respect to VoltMaxIn. The converters have to be bipolar (e.g. ± 10 V)

Analog/VoltMaxIn AD Voltage corresponding to DigRangeIn

Analog/DigRangeOut positive maximum value for X,Y,Z,... DA-conversion with respect to VoltMaxOut. The converters have to be bipolar (e.g. ± 10 V)

Analog/VoltMaxOut DA Voltage corresponding to DigRangeOut

Analog/V1-V9 Piezoamplifier Settings, typically: 1,2,5,10,15.

Analog/VX/Y/Zdefault Preferences at for programm startup gain selections

Analog/VX0/Y0/Z0default Preferences for programm startup gain selections for Offset, only essential if analog offset adding is used.

10.2.2. Inst-SPA folder

Instrument/X,YCalibV SPALEED: X,Y calibration factor, 1V at DA → ±10 V resp. 15 V at octopol front/back.

Instrument/EnergyCalibVeV factor*Volt = energy in eV

Instrument/Sensitivity $BZ = U \cdot \text{Sensitivity} / \sqrt{\text{Energy[eV]}}$

Instrument/ThetaChGunInt Half angle between channeltron and electron gun (intern).

Instrument/ThetaChGunExt unused.

Sample/LayerDist Atom layer distance of the sample in Ångstroem, used for calculation of phase (energy in S).

Sample/UnitLen unused.

10.3. DataAq folder

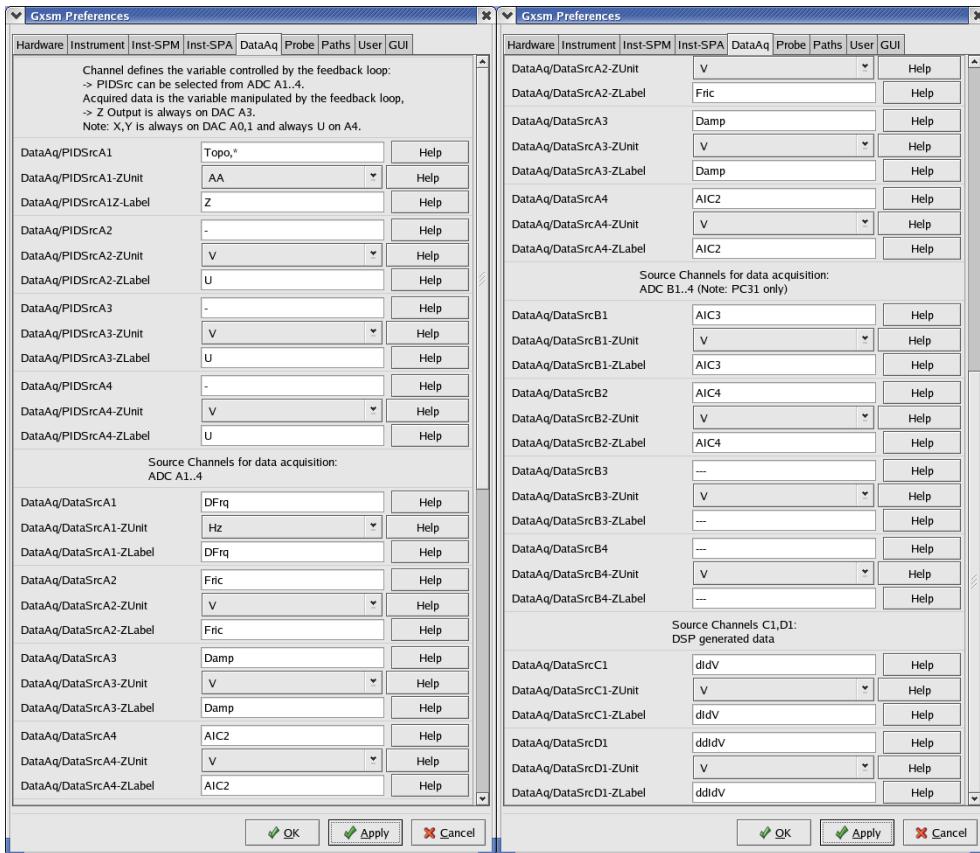


Figure 10.4.: Gxsm preferences folder, DataAq page. The shown settings are for a AFM setup driven by the Signal Ranger.

In this section deals with the assignment of data sources to GXSM-3.0 channels. The particular configuration depends on your data acquisition hardware and is unique for each instrument. The following explanations are based on the Signal Ranger DSP board.

For historic reasons the analog data source channels are grouped in blocks of four (A,B) inputs, this was due to the old PCI32 DSP board ² has four A/D-converters with four input-lines directly connected. The even older PC31 has only two A/D-connectors,

Note

²The old PCI32 DSP card is still supported by GXSM-3.0, but some limits apply and it's less and less tested with newer GXSM-3.0 versions.

10. Configuration

but four input-lines are addressed to each of them via a 4x-multiplexer, which results in eight usable analog input lines. Name convention in the x-resources is A and B for the converters and 1 to 4 for the four multiplexer input lines (PC31). The PCI32 has four converters which are called A1, A2, A3 and A4.

For the Signal Ranger board the hardware channel assignment to the GXSM-3.0 data sources is defined as shown here:

Channel Descriptor	Name	Notes
DataAq/PIDSrcA1 DataAq/PIDSrcA2...4	Topo,* —	Z: generated by the Feedback not used
DataAq/DataSrcA1 DataAq/DataSrcA2 DataAq/DataSrcA3 DataAq/DataSrcA4	AIC5 AIC0 AIC1 AIC2	AIC5: I, Force, dFrq, ... what is used as feedback signal AIC0 input (Fric) AIC1 input (Damp., opt. FUZZY source) AIC2 input
DataAq/DataSrcB1 DataAq/DataSrcB2 DataAq/DataSrcB3 DataAq/DataSrcB4	AIC3 AIC4 AIC5 AIC6	AIC3 input AIC4 input AIC5 input AIC6 input
DataAq/DataSrcC1 DataAq/DataSrcD1	dIdV ddIdV	LockIn 1st of AIC5 (32bit) LockIn 2nd of AIC5 (32bit)
DataAq/DataSrcE1	I0	test:FB-Integrator (LockIn-I0 avg) (32bit)
DataAq/DataSrcF1	Counter	SR-CoolRunner Counter if equipped (experimental) (32bit)

Table 10.1.: Signal Ranger board the hardware channel to GXSM-3.0 data sources assignment definitions. I0, Counter are experimental and may change any time

The field “Name” can be used to give the input a real name instead of “AIC0” you can used “PLL-dF” or what ever you like. Do not use a “,” except for “,*” to set a default, the text behind it will be ignored.

For you hackers, check out the SR code, push_area_scan_data() in SRanger/TiCC-project-files/FB_spmcontrol/FB_spm_areascan.c.



10.4. Probe folder

– obsolete –

10.5. Folder Adj.

– obsolete – see Entry-Popup menu!

Configuration of Value-Slider:

Adjustments.Name/min,max for Area (please care for min < max !!), Adjustments.Name/step,page for stepwidth at Cursor, Click.

10.6. Paths folder

Path/Logfiles

Path/Data

Path/RemoteFifo path to remote fifo (read only by GXSM-3.0)

Path/RemoteFifoOut path to control fifo Echo (wo by GXSM-3.0)

Path/Plugins Additional Plugin searchpath

10.7. User folder

User/SaveWindowGeometry Always false, for future use...

User/Unit XYZ-Einheit: Choice of AA, nm, um, mm, BZ, sec, V, 1

User/HiLoDelta Checking distance of array for calculation of Min/Max for Autoplay, 1 = all points visited.

User/FileType nc (dat possible, but out of date.)

User/NameConvention digit: Auto enumeration with 001, 002, ..., alpha: Auto enumeration with aaa, aab, ...

User/SliderControlType slider: Omicron Slider Control, mover: Besocke

User/Palette path to palette image (may be altered at runtime) See also ...

10.8. GUI folder

GUI/layerfields Select here, if you want use layered (3d) scans.

Part II.

Plug-ins

11. Plug-Ins: control

The *control* plugins are those used to control the instrument and perform special data acquisitions tasks such as DSP control, mover/slider control, probing (spectroscopy) and things like that.

11.1. Inet JSON Scan Data Control

RP data streaming

Description

Usage

References



Info for Plug-In: windows-section Inet JSON Scan External Data

Plug-In name: inet_json_external_scandata
Author: Percy Zahl

File: control/inet_json_external_scandata.C
Email: zahl@users.sf.net

11.2. SPA–LEED simulator control

Description

A frontend for SPA–LEED and E-gun parameter control, currently used to configure the SPA–LEED simulator kernel module. But it has the potential for remote controlling a future SPA–LEED unit :-)

11. Plug-Ins: control

Usage

Open *main menu/windows-sectionSPA-LEED Ctrl* and play.

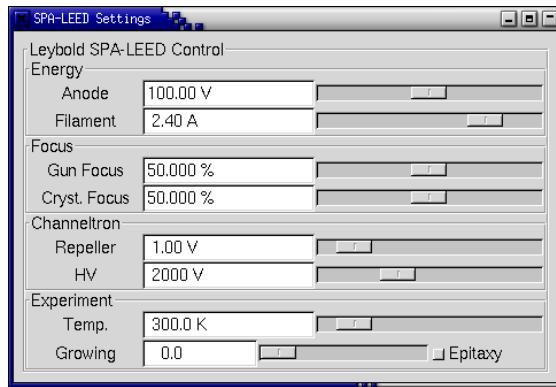


Figure 11.1.: The SPA-LEED Control window.

Not intuitive?

Note

The SPA-LEED simulator kernel module is here:
`Gxsm/plug-ins/hard/modules/dspspaemu.o`

Info for Plug-In: windows-sectionSPA-LEED Ctrl

Plug-In name: SpaLeedControl
Author: Percy Zahl

File: control/SpaLeedControl.C
Email: zahl@users.sf.net

11.3. Nano HPLG plotter

Description

This is a tool to use your tip for writing or manipulating by moving the tip (i.e. the scan-piezo) along an arbitrary programmable path. There are two DSP-parameter sets for bias, feedback and scan settings which can be used for to assign two "writing" modes called PU (pen up) and PD (pen down). The path is read from a plotter-file in a very simple HPGL-plotter language style. This file can be created by hand (editor) or from line or polyline object(s) drawn in a scan.

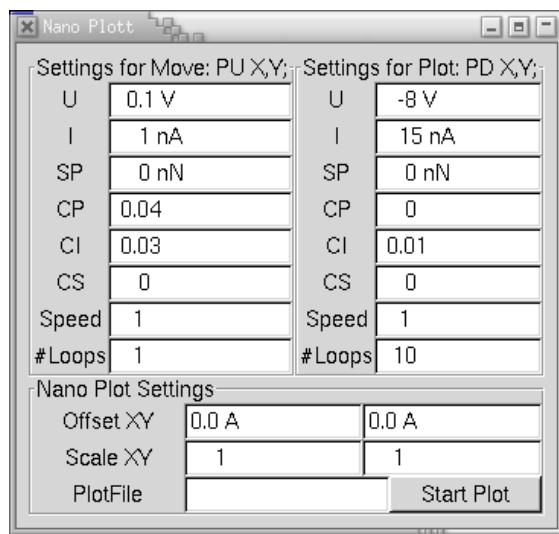


Figure 11.2.: The Nano-Plot window.

Usage

Set desired parameters for Pen Up and Pen Down, put in the path/file to your plotfile (.plt) and press start... You can translate and scale you plot data using the parameters Offset and Scale. Make sure the plot file name is correct in can be found from current working directory or use an absolute path.

Objects

Poly line objects are used for display only – not for input of path!

Destination

The plotted path is shown on the active scan using poly line objects.

Files

Create a simple HPGL plott file for input. You can do this with Gxsm itself: place a bunch of PolyLine objects and use *Grey 2D popup window File/Save Objects* to create a HPGL file, enter therefore a file with a .plt extension! Use this as input for the NanoPlotter.

Known Bugs

Not 100% tested, beta state.



PU 0.0,0.0;
PD 0.0,5000.0;

Disclaimer

This tool may ruin your tip and sample, so take care!

Info for Plug-In: windows-sectionNano Plotter

Plug-In name: NanoPlott	File: control/NanoPlott.C
Author: Percy Zahl	Email: zahl@users.sf.net

11.4. SPA–LEED peak finder and monitor (OBSOLETE)

Description

For SPA–LEED users Gxsm offers a peak intensity monitor and peak finder to follow a shifting peak. It is designed for both: Focus adjustment and peak intensity monitoring. It can handle a unlimited number of peaks simultaneous.

Usage

Open *main menu/windows-sectionDSP PeakFind*.

Configuration

Lots of controls... Since I do not know about other SPA–LEED users I'll not spend more time here!

Known Bugs

The long term peak intensity monitor is unstable and beta. I need more feedback from SPA–LEED users...



Info for Plug-In: windows-sectionDSP PeakFind

Plug-In name: DSPPeakFind
Author: Percy Zahl

File: control/DSPPeakFind.C
Email: zahl@users.sf.net

11.5. Nano Manipulator (to be ported)

Description

This is a tool to use your tip for nano manipulating.

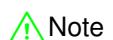
It will connect to the DSP/SPM and provide a realtime force feedback (using a modern Force Feedback Joystick Control Device) while moving/pushing things around...

Usage

Objects

Shows a trace and current tip position...

This PlugIn is work in progress.



Disclaimer

This tool may ruin your tip and sample if imoproper used!

Info for Plug-In: windows-sectionNano Manipulator

Plug-In name: nano_manipulator
Author: Percy Zahl

File: control/nano_manipulator.C
Email: zahl@users.sf.net

11.6. SPM Scan Control

Description

Provides a SPM Scan Control Window and connects to the GXSM-3.0 toolbar buttons *Scan*, *Movie*, *Stop* for quick access. The control panel offers in addition to *Scan*, *Movie*, *Stop* a *Pause* and *HS Capture* button. Use the *Pause* button to pause the scanning process and press it again for continuation. (PCI32 only)

The *HS Capture* button starts a continuous (movie) high speed (HS) frame capturing process on the DSP, a refresh is done after the whole scan data is received from the DSP. The scan size is limited by the available memory (SRAM) on the DSP platform. The *HS Capture* mode assures precise real time inter line timing on DSP level and allows maximum frame rates due to minimized communication between Gxsm and the DSP. (applies to PCI32 only)

Using this panel it is now possible to set the scan Y direction from *TopDown* (default, from Top to Bottom) to *TopDownBotUp* (alternating from Top to Bottom and vice versa) or *BotUp* (Bottom to Top).

If the *Repeat* option is checked, scanning will repeat until released (at scan end) or stop is pressed to cancel scanning at any time. Files are automatically saved if the *AutoSave* check-button in the main window is checked.

Using the *Movie* mode, no single frames are saved while scanning, but the whole movie is appended into the time-dimension of the scan (this need sufficient amount of memory, as GXSM-3.0keeps all data in memory.). Use the Movie-Control/Player window to play it. If stopped, the last frame will be incomplete, you can truncate it using the *main menu/Edit/Copy* tool and copy all but the last frame. Multidimensional data is saved into the NetCDF file.

This Gxsm PlugIn module actually provides not only the the scanning controls, it does the whole job of high-level data acquisition/sorting itself.

Usage

Used for advanced SPM data acquisition control, open the control panel via *main menu/windows-section SPM Scan Control*.

The *HS Capture* scan mode is always in *TopDown* mode and no *Pause* will be accepted, use *Stop*. This feature is implemented for the PCI32 only (old) and is obsolete for all newer HWIs using direct data streaming (FIFO).

 Note

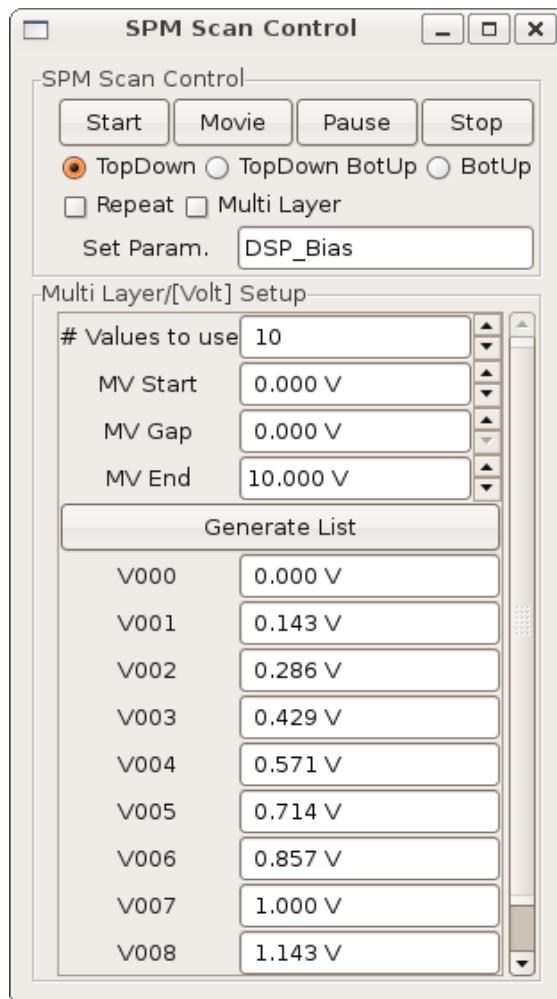


Figure 11.3.: The SPM Scan Control window.

References

About pthreads hacking, this is object of future use:

http://java.icmc.sc.usp.br/library/books/ibm_pthreads/



In *TopDownBotUp* mode always the last scan direction is remembered (even if the scan was stopped in between) and the opposite scan direction is used at next *Scan Start*.



You can switch the scan direction mode while running a movie or single scan, it will be used as soon as the next scan is started!



Hacker notes: This plugin is responsible for the high level scanning process, background/idle display refresh. It does the initial scan creation and setup of all data types due to the configuration as provided by the user via preferences. This setup need to match the current hardware configuration.

Info for Plug-In: windows-section SPM Scan Control

Plug-In name: spm_scancontrol

File: control/spm_scancontrol.C

Author: Percy Zahl

Email: zahl@users.sf.net

11.7. RHK Scan Control (to be ported)

Description

Provides a Scan Control Window for the RHK STM 100 electronics. It connects to the GXSM-3.0 toolbar buttons *Scan*, *Movie*, *Stop* for quick access. The control panel offers in addition to *Scan*, *Movie*, *Stop* a *Pause* and *HS Capture* button. Use the *Pause* button to pause the scanning process and press it again for continuation.

The RHK electronics is used through a stand-alone program (*rhk_controller*) that has to be started before Gxsm. Use the computer and port in which that program is running as the Hardware/Device setting (localhost:5027 is the default).

The *HS Capture* button starts a continuous (movie) high speed (HS) frame capturing process, where a refresh is done after the whole scan data is received from the STM100 electronics. The scan size is limited by the available driver memory (currently 2Mb) on the *rhk_controller* program. The *HS Capture* mode assures precise real time inter line timing and allows maximum frame rates due to minimized communication between Gxsm and the *rhk_controller* program.

This plugin is also the main RHK control panel. The RHK interface is slightly different from the rest of the SPM hardware supported by Gxsm. The difference is due to the fact that the scan generator is inside the RHK STM-100, so it is not under the control of Gxsm. The offset and scan size can only be read, as are the sample bias and the tunneling current. For now they are only updated when the "update" button is pressed, and before acquiring an image (the image size is also read from the RHK so this is a must). An additional option is the automatic update of the parameters (every second or so), which can be turned on or off by a button.

This RHK Gxsm PlugIn module actually provides not only the scanning controls, it does the whole job of data acquisition itself, and is a substitute for the standard Scan Control plugin.

On the left side of the panel, there is a pixmap that shows the current scan area size and offset relative to the total range available.

Info for Plug-In: windows-sectionRHK Scan Control

Plug-In name: *rhk_scancontrol*

Author: Farid El Gabaly, Juan de la Figuera

File: control/rhk_scancontrol.C

Email: farid.elgabaly@uam.es, juan.delafiguera@uam.es

12. Plug-Ins: scan

The *scan* plugins are those used to handle import/export of scan data – not to do scanning as it may sound like, these things are located in the *control* section.

12.1. WIP (WiTeC) Import

Description

The *wip_im_export* plug-in allows importing selected data sets of WIP (WiTeC-Project) files.

Usage

The plug-in is called by *main menu/File/Import/WIP*.

Info for Plug-In: File/Import/WIP

Plug-In name: wip_im_Export
Author: Percy Zahl

File: scan/wip_im_export.C
Email: zahl@users.sf.net

12.2. Cube file Import

Description

The *Cube* plug-in supports reading of a cube volumetric data file format. <http://paulbourke.net/dataformats/cube/>

CPMD CUBE FILE.

OUTER LOOP: X, MIDDLE LOOP: Y, INNER LOOP: Z

```
3 0.000000 0.000000 0.000000
40 0.283459 0.000000 0.000000
40 0.000000 0.283459 0.000000
40 0.000000 0.000000 0.283459
8 0.000000 5.570575 5.669178 5.593517
```

1 0.000000 5.562867 5.669178 7.428055

1 0.000000 7.340606 5.669178 5.111259

-0.25568E-04 0.59213E-05 0.81068E-05 0.10868E-04 0.11313E-04 0.35999E-05

: : : : :

: : : : :

: : : : :

In this case there will be 40 x 40 x 40 floating point values

: : : : :

: : : : :

: : : : :

Usage

The plug-in is called by *main menu/File/Import/Cube*. Only import direction is implemented.

Info for Plug-In: File/Import/Cube

Plug-In name: cube_import

File: scan/cube_import.C

Author: Percy Zahl

Email: zahl@users.sf.net

12.3. UKSOFT/U-view Import (ELMITEC LEEM)

Description

The *uksoft_im_export* plug-in allows importing of single and multidimensional sets of .dat and multiple image/movie .dav files in the UKSOFT-2001 data format generated by U-view/Elmitec-LEEM Control Software.

It is recommended to configure *GXSM preferences dialog, folder Instrument/Type* to "CCD" to make use of a simplified CCD-Hi/Low scaling method. Use the "Direct" view mode.

Usage

The plug-in is called by *main menu/File/Import/UKSOFT*. Select a single .dat or .dav file.

For automatic reading of multidimensional image series the files has to be in one director and the file names have to be numbered in one or two dimensions like this or similar:

```
RuO2_CO_10mu_000_000.dat  
RuO2_CO_10mu_000_001.dat  
RuO2_CO_10mu_000_002.dat  
RuO2_CO_10mu_001_000.dat  
RuO2_CO_10mu_001_001.dat  
RuO2_CO_10mu_001_002.dat  
...  
Select the first file and replace the relevant digits by a valid integer C format string identifier, i.e.:  
GTK3: use CTRL-L after selecting a file to allow (show the edit field) edit the file pattern!!
```



RuO2_CO_10mu_%03d_%03d.dat

Click OK and the Multi-File import setup dialog will show up. Setup the number of images for each dimension (Max Index) to read and use Offset and Step for optional skipping of images. If only one index is present, it is assumed to be the Layer (Value) dimension.

Info for Plug-In: File/Import/UKSOFT

Plug-In name: uksoft2001_im_Export
Author: Percy Zahl

File: scan/uksoft2001_im_export.C
Email: zahl@users.sf.net

12. Plug-Ins: scan

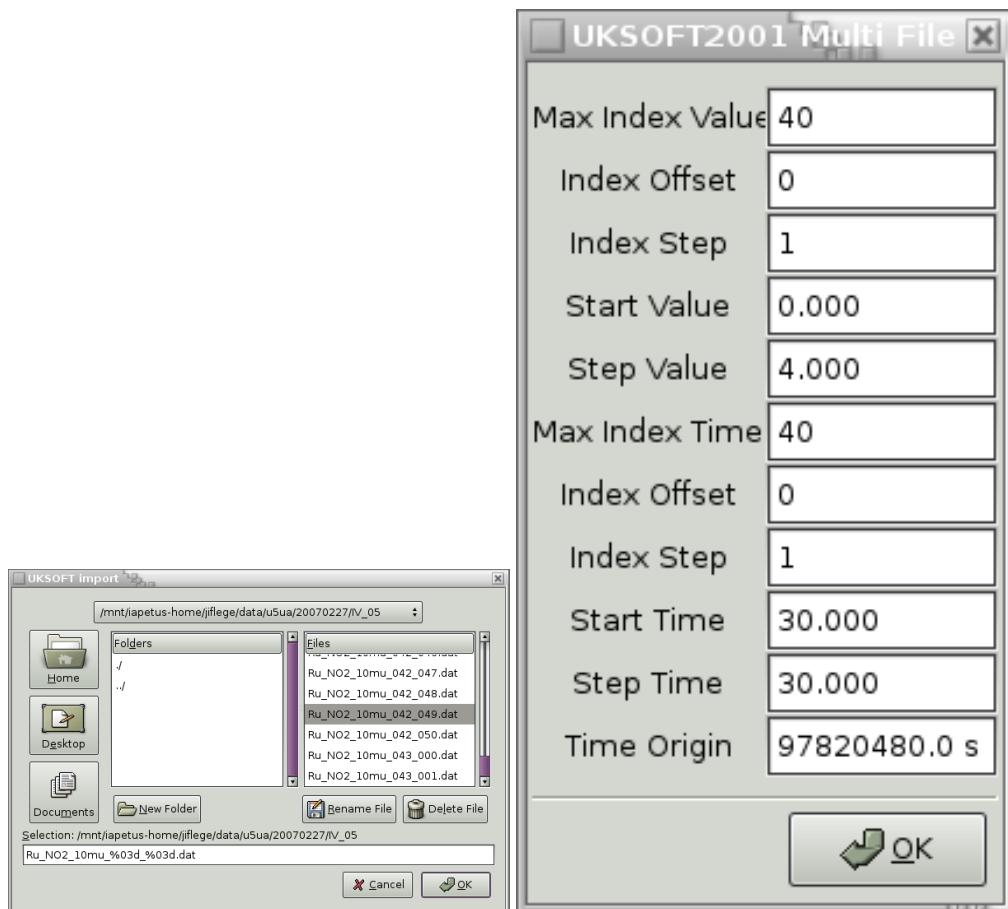


Figure 12.1.: left: UKSOFT/U-view import file dialog, right: setup for multi dimensional image processing

12.4. Park Scientific (HDF) data Import

Description

The *PsiHDF_im_export* plug-in supports reading of Psi-HDFe files used by Park-Scientific AFM.

Usage

The plug-in is called by *main menu/File/Import/PsiHDF*.

Known Bugs

Not yet tested, porting to GXSM-2 in progress..

Info for Plug-In: File/Import/PsiHDF

Plug-In name: PsiHDF_im_export
Author: Percy Zahl

File: scan/PsiHDF_im_export.C
Email: zahl@users.sf.net

12.5. Quicktime

Description

The *quicktime_im_export* plug-in allows exporting of single and multidimensional image data sets as Quicktime Movie.

Usage

The plug-in is called by *main menu/File/Import/Quicktime*.

Recommended Format: MJPEG-A, works fine with Power-Point and most Movie Players.  Note

There seem to be much more formats available from the lib-quicktime, but for an unknown reason some are just not working or crashing the program if used. This seems to depend on the system and libquicktime version used, so please try it out.

12. Plug-Ins: scan

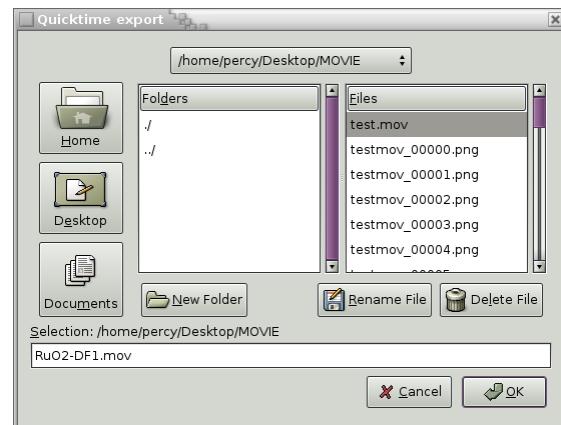


Figure 12.2.: QT Export File Dialog

Info for Plug-In: File/Import/Quicktime

Plug-In name: quicktime_im_Export
Author: Percy Zahl

File: scan/quicktime_im_export.C
Email: zahl@users.sf.net

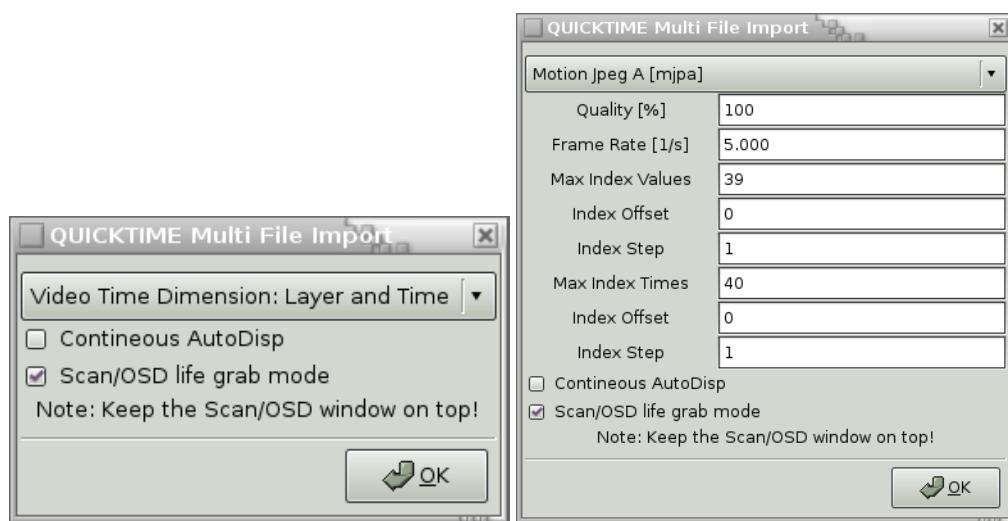


Figure 12.3.: left: QT Export Setup step one., right: QT Export Setup step two.

12.6. Import/Export of old (G-) dat files

Description

The *g_dat_im_export* plug-in supports reading and writing of the old .dat fileformat used in Hannover at former times. It was used by the very first xxsm, pmstm and even older OS/2 software in Hannover, mostly by Köhler et al. Also the so called *gnutools* are can use this 16-bit data format. To distiguish different dat files, I call it *G-dat*.

Usage

The plug-in is called by *main menu/File/Import/G-dat*.

Known Bugs

Not yet tested.

Info for Plug-In: File/Import/G-dat

Plug-In name: v5d_Export
Author: Percy Zahl

File: scan/v5d_export.C
Email: zahl@users.sf.net

12.7. Import/Export of old (G-) dat files

Description

The *g_dat_im_export* plug-in supports reading and writing of the old .dat fileformat used in Hannover at former times. It was used by the very first xxsm, pmstm and even older OS/2 software in Hannover, mostly by Köhler et al. Also the so called *gnutools* are can use this 16-bit data format. To distiguish different dat files, I call it *G-dat*.

Usage

The plug-in is called by *main menu/File/Import/G-dat*.

Known Bugs

Not yet tested.

Info for Plug-In: File/Import/G-dat

Plug-In name: G_dat_Im_Export
Author: Percy Zahl

File: scan/g_dat_im_export.C
Email: zahl@users.sf.net

12.8. Import Tool for RHK STM-200

Description

This plug-in is responsible for importing files saved by RHK STM-2000 systems. The STM-2000 system is the STM electronics manufactured by RHK based on SGI workstations.

Usage

Registers itself for loading files with the filename suffix ".Stm".

Info for Plug-In: File/Import/RHK STM-200 import

Plug-In name: RHK2000-Import
Author: Peter Wahl

File: scan/rhk2000_import.C
Email: wahl@fkf.mpg.de

12.9. Converter

Description

Simple file conversion utility, which converts files all kinds of data files supported by GXSM-2

Usage

Registers itself for converting files. Just choose the right file extensions for the conversion (e.g. .nc to .top). You can also select the source and destination folder by the browse buttons.

Info for Plug-In: Tools/Converter

Plug-In name: Converter

Author: Kristan Temme and Thorsten Wagner

File: scan/converter.C

Email: stm@users.sf.net

12.10. UK2000 v3.4 import plug-in

Description

This plugin is responsible for reading UK2000 v3.4 images, with the following limitations so far: only topography, and only the forward scan. To add support for the back scan would be pretty simple, if someone is interested. This STM electronics is a DSP based system with transputers as CPUs (yes, it has several). It was made by Uwe Knipping from Arizona State University and sold by CVS (Custom Vacuum Systems Ltd.). There are a few systems around, still running after more than 10 years. The PC version of the electronics are the basis of the DSP STM unit sold (and further developed) by Molecular Imaging.

The files are an ASCII file with the scan details (extension .stp), and an 16bit binary dump of the image (extension .std).

Usage

Registers itself for loading files with the filename suffix ".std".

Info for Plug-In: File/Import/Uk2k import

Plug-In name: UK2k_import

Author: Juan de la Figuera

File: scan/UK2k_import.C

Email: juan.delafiguera@uam.es

12.11. Import of RHK SPM32 files (STM-1000 electronics).

Description

The *rhk_spm32_import* plug-in supports reading of files generated with the SPM32 software distributed by RHK Technology Inc. (Troy, MI, USA). This is the software that is usually used to control the RHK STM-1000 electronics (unless you use GXSM-3.0 to control it :-).

Usage

The plug-in is called by *main menu/File/Import/RHK SPM32*. It is also automatically invoked by GXSM-3.0 when opening RHK SPM32 files.

Known Bugs

Not yet all very well tested. At the moment, it is somewhat restricted: i) SPM32 files can contain several pages with data from, e.g., forward and backward scans. The plug-in only reads the first page. ii) only pages with pure images can be imported, no spectroscopy data.

Info for Plug-In: File/Import/RHK SPM32

Plug-In name: rhk_spm32_import
Author: Andreas klust

File: scan/rhk_spm32_import.C
Email: klust@users.sf.net

12.12. external converter

Description

Simple plugin to call an external converter

Usage

Registers itself. Select source file,a destination folder, a suitable suffix, the external converter (full path) and press okey. You can also path some options to the external program

Info for Plug-In: Tools/external_converter

Plug-In name: external_converter
Author: Thorsten Wagner

File: scan/external_converter.C
Email: stm@users.sf.net

Suffix	description
.dat	(old) STM data format, used by e.g. PMSTM, discontinued.
.byt	byte format: raw 8 Bit data
.sht	Short format: raw 16 Bit data
.flt	Float format: floats, single precision
.dbl	Double format: floats, double precision
.pgm	Portable Greymap (P5) format
Read-only supported formats:	
.d2d	(old) SPA-LEED data format, used by SPA V
.nsc	a very old version of nanoscope import filter (obsolete), use the new Nanoscope Import PlugIn!
.h16	Markus/Mats simulations program data import
Write-only supported formats:	
.tga	TARGA bitmap format (8, 16 and 24 Bit colordepth) 24bit: only option to write color images (usage of palette)

Table 12.1.: file formats supported by the *primitiveimexport* plug-in.

12.13. Import/export of non-SPM file formats

Description

This plug-in is responsible for importing and exporting a wide variety of file formats. Most of them are file formats not specifically designed for storage of scientific data such as the Portable Graymap (pgm) format. Nevertheless, the support of these formats allows to exchange data between GXSM-3.0 and countless other programs, e.g. The GIMP for the final "touch up" of images.

Supported file formats

The file formats supported by the *primitiveimexport* plug-in are listed in Tab. 12.1. The ".dat" format refers to the old, STM specific data format used by the OS/2 software PM-STM and the Linux based Xxsm, both predecessors of GXSM-3.0. The ".d2d" format is used for SPA-LEED by the SPA V program. The primitive formats ".byt", ".sht", ".flt", and ".dbl" are described in more detail below.

The primitive file formats

The primitive file formats ".byt", ".sht", ".flt", and ".dbl" are supported by the *primitiveimexport* plug-in for both reading and writing. These file formats have a simple

structure and therefore present a considerable alternative to more complex formats for quickly hacked programs. They are binary files in PC byte order starting with two 16 bit integer numbers denoting the number of rows and columns in the image. Following them, the image data is written in raw binary form using the following data types to represent a single pixel: char ("byt"), 16 bit integer ("sht"), single precision floating point ("flt"), or double precision floating point ("dbl") numbers.

For the manipulation of data in these formats, the GXSM-3.0 project provides an extensive set of small tools included in the GXSM-3.0 source code. For instance, these tools include programs to create, filter, and doing simple algebra operations. All these tools are designed for the use in shell scripts.

Usage

The file formats are recognized by their filename suffix. For instance, exporting to `my-image.pgm` results in a greymap image in the binary Portable Greymap (pgm) format. For other supported formats and their correspondent suffixes see Tab. 12.1.



Hint

Info for Plug-In: File/Import/Primitive Auto Import and File/Export/Primitive Auto Export

Plug-In name:	primitiveimexport	File:	scan/primitiveimexport.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

12.14. ASCII data file Import/Export

Description

The `ascii_data_im_export` plug-in supports reading of ASCII data files.

It auto detects the text header "Reuben-NSNOM" as described below – if this fails, it attempts to auto read a plain matrix of values, line by line. Space or "CSV" separated numbers, dimensions are auto determined by max elements per line and number of lines found. File extension must be ".asc" or ".csv".

Special but simple text header "Reuben-NSNOM" is suggested. Here is a sample file:

```
scan length = 25.000000
points/line = 25
point spacing = 1.041667
# lines = 25
line spacing = 1.041667
points in average = 1
```

12. Plug-Ins: scan

```
velocity = 2.000000
lock-in time constant (msec) = 30.000000
X origin = -1883.596000
Y origin = 1321.116000

end

-1.5850 -1.5889 -1.5967 -1.5737 -1.5610 -1.5581 -1.5518 -1.5537 -1.9194 -2.2808 -2.2148 -2.1709 -2.1382 -2.1367 -2.1123 -2.0864 -2.06
-1.5796 -1.5811 -1.5684 -1.5718 -1.5625 -1.5552 -1.5425 -1.5488 -1.9272 -2.2764 -2.2104 -2.1660 -2.1387 -2.1294 -2.1094 -2.0801 -2.06
-1.5776 -1.5747 -1.5854 -1.5718 -1.5610 -1.5488 -1.5400 -1.5474 -1.9253 -2.2651 -2.2124 -2.1597 -2.1338 -2.1309 -2.0996 -2.0728 -2.06
-1.5688 -1.5840 -1.5825 -1.5640 -1.5518 -1.5488 -1.5361 -1.5430 -1.9395 -2.2607 -2.2070 -2.1548 -2.1309 -2.1250 -2.0942 -2.0708 -2.05
-1.5674 -1.5776 -1.5776 -1.5610 -1.5474 -1.5459 -1.5288 -1.5347 -1.9136 -2.2549 -2.1963 -2.1519 -2.1250 -2.1182 -2.0913 -2.0664 -2.05
-1.5610 -1.5732 -1.5732 -1.5566 -1.5439 -1.5391 -1.5259 -1.5337 -1.9502 -2.2471 -2.1919 -2.1445 -2.1196 -2.1123 -2.0850 -2.0620 -2.04
-1.5610 -1.5640 -1.5684 -1.5547 -1.5381 -1.5332 -1.5229 -1.5308 -1.9697 -2.2422 -2.1782 -2.1387 -2.1157 -2.1094 -2.0820 -2.0576 -2.04
-1.5562 -1.5625 -1.5640 -1.5483 -1.5308 -1.5283 -1.5171 -1.5107 -1.9854 -2.2378 -2.1797 -2.1367 -2.1123 -2.1030 -2.0742 -2.0542 -2.03
-1.5503 -1.5562 -1.5610 -1.5410 -1.5283 -1.5244 -1.5073 -1.5181 -1.9443 -2.2300 -2.1768 -2.1309 -2.1094 -2.1030 -2.0664 -2.0464 -2.03
-1.5459 -1.5532 -1.5581 -1.5400 -1.5239 -1.5171 -1.5073 -1.5044 -1.9302 -2.2227 -2.1738 -2.1260 -2.1021 -2.0967 -2.0693 -2.0435 -2.02
-1.5410 -1.5483 -1.5483 -1.5317 -1.5181 -1.5181 -1.5015 -1.4585 -1.9780 -2.2178 -2.1670 -2.1216 -2.0972 -2.0894 -2.0645 -2.0391 -2.02
-1.5337 -1.5410 -1.5425 -1.5244 -1.5137 -1.5132 -1.4951 -1.4966 -1.9224 -2.2114 -2.1597 -2.1172 -2.0957 -2.0864 -2.0601 -2.0327 -2.01
-1.5322 -1.5332 -1.5425 -1.5210 -1.5107 -1.5093 -1.4907 -1.4917 -1.9575 -2.2061 -2.1567 -2.1104 -2.0894 -2.0845 -2.0527 -2.0278 -2.01
-1.5308 -1.5181 -1.5361 -1.5088 -1.5059 -1.5015 -1.4873 -1.4873 -1.9653 -2.2036 -2.1523 -2.1060 -2.0864 -2.0786 -2.0464 -2.0229 -2.01
-1.5259 -1.5229 -1.5317 -1.5181 -1.4985 -1.4922 -1.4780 -1.4834 -1.9834 -2.1978 -2.1509 -2.1021 -2.0820 -2.0708 -2.0420 -2.0176 -2.00
-1.5229 -1.5229 -1.5273 -1.5117 -1.4873 -1.4937 -1.4780 -1.4785 -2.0308 -2.1885 -2.1489 -2.0952 -2.0757 -2.0679 -2.0405 -2.0127 -2.00
-1.5244 -1.5054 -1.5181 -1.4937 -1.4771 -1.4873 -1.4722 -1.4766 -2.0483 -2.1855 -2.1401 -2.0923 -2.0708 -2.0591 -2.0327 -2.0098 -1.99
-1.5229 -1.4893 -1.5103 -1.4180 -1.4687 -1.4771 -1.4687 -1.4722 -2.0327 -2.1812 -2.1279 -2.0864 -2.0635 -2.0542 -2.0234 -2.0000 -1.99
-1.4497 -1.4487 -1.4858 -1.4414 -1.4614 -1.4736 -1.4614 -1.4644 -2.0039 -2.1704 -2.1187 -2.0757 -2.0605 -2.0493 -2.0229 -2.0020 -1.98
-1.5229 -1.4478 -1.4917 -1.4893 -1.4766 -1.4736 -1.4497 -1.4429 -1.9790 -2.1523 -2.1050 -2.0786 -2.0566 -2.0513 -2.0205 -1.9990 -1.98
-1.5088 -1.3389 -1.4658 -1.4321 -1.4687 -1.4507 -1.4385 -1.4121 -2.0327 -2.1611 -2.1182 -2.0728 -2.0513 -2.0420 -2.0127 -1.9912 -1.97
-1.4707 -1.4678 -1.4507 -1.4443 -1.4707 -1.4067 -1.4521 -1.4463 -2.0117 -2.1494 -2.1123 -2.0684 -2.0508 -2.0376 -2.0083 -1.9897 -1.97
-1.4565 -1.5117 -1.4946 -1.4824 -1.4497 -1.4541 -1.4463 -1.4189 -2.0576 -2.1431 -2.0894 -2.0664 -2.0435 -2.0264 -2.0034 -1.9824 -1.96
-1.4893 -1.5044 -1.4858 -1.4658 -1.4722 -1.4551 -1.4307 -1.2671 -2.0464 -2.1357 -2.0967 -2.0576 -2.0356 -2.0205 -1.9990 -1.9780 -1.96
-1.4902 -1.5044 -1.4824 -1.4707 -1.4663 -1.4404 -1.4141 -1.3501 -2.1660 -2.1338 -2.0952 -2.0513 -2.0356 -2.0269 -1.9961 -1.9717 -1.96
```

Usage

The plug-in is called by *main menu/File/Import/ASCII*. Only import direction is implemented.

Info for Plug-In: File/Import/ASCII

Plug-In name: ascii_data_im_export
Author: Percy Zahl

File: scan/ascii_data_im_export.C
Email: zahl@users.sf.net

12.15. Import of Surface Data Format files

Description

The *sdfimport* plug-in supports reading files as defined by the surfstand group (search internet for 'surfstand'), a group, sponsored by the EU to develop a basis for 3D surface roughness standards.

Usage

The plug-in is called by *main menu/File/Import/SDF-import*.

No endianess independent code – no cross platform data exchange possible.



Note

Known Bugs

The scaling of the data, especially in Z direction is not well tested.

Info for Plug-In: File/Import/SDF-import

Plug-In name: sdfimport
Author: Stefan Schroeder
File: scan/sdfimport.C
Email: stefan_fkp@users.sf.net

12.16. Import/Export of G.Meyer STM/AFM Dat files

Description

The *gmeyer_im_export* plug-in supports reading and writing of Dat files used by the G.Meyer STM/AFM software.

Usage

The plug-in is called by *main menu/File/Import/GME Dat*.

Known Bugs

Not yet all very well tested.

Info for Plug-In: File/Import/GMeyer Dat

Plug-In name: gmeyer_im_export
Author: Percy Zahl
File: scan/gmeyer_im_export.C
Email: zahl@users.sf.net

12.17. Import/export of Scala SPM files (Omicron)

Description

This plug-in is responsible for the import and export of data in the SPM format used by Omicron's Scala software. It supports both conventional, two-dimensional data such

as topographic images and gridded spectroscopic data. Data acquired using the point spectroscopy mode of the Scala system is not written in the SPM format. Therefore, point spectroscopy data is ignored by this plug-in.

In the Scala SPM format, data is saved in two parts: Firstly, the data itself as binary file. Secondly, additional information, e.g. scan size, in an extra ASCII file with the filename suffix ".par". This parameter file can contain information on several images. Therefore, for importing SPM files, the binary file must be selected. The parameter file contains many comments making it human readable.

Usage

When the Omicron_IO plug-in is installed, Scala SPM files can be loaded like any other files supported by GXSM-3.0. Alternatively, *main menu/File/Import/Omicron SPM Import* can be used.

Known Bugs

Exporting data in the Scala SPM format is not yet implemented.

Info for Plug-In: File/Import/Omicron_SPM_Import

Plug-In name: Omicron_IO
Author: Andreas Klust

File: scan/omicron_io.C
Email: klust@users.sf.net

12.18. Import of Digital Instruments Nanoscope files

Description

The *nanoimport* plug-in supports reading files of Digital Instruments' Nanoscope systems (Veeco Metrology Group).

This plugin can import original Nanoscope-data files, for older versions below 0x04300000 (Version: 0x04220200 tested) single channel data is assumed, for all newer versions (Version: 0x04320003, 0x04430006 tested) multiple channels of data can be present and are all imported at once using different channels in GXSM. Version below 0x03000000 are rejected (do not know about). The channel type is shown as scan title. The full ASCII header is appended to the comment.

Usage

The plug-in is called by *main menu/File/Import/Nano Scope*.

Known Bugs

In some cases the Z-scaling is wrong. The reason for that is, I did not yet figured out the correct way how to interpret several numbers given in the di-file header. Any help to fix it is welcome! – I changed the calculation once more, look OK for newew files and is still off for some others, I hate it...

According to the manual (V4.1):

$$H_z = \frac{\text{DAC-value}}{65536} Z_{\text{scale value}}$$

Info for Plug-In: File/Import/Nano Scope

Plug-In name:	nanoimport	File:	scan/nanoimport.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

12.19. PNG image Import/Export

Description

The *png_im_export* plug-in allows reading and writing of images using the Portable Network Graphics (PNG) image format.

The currently set palette is used if non is used or available a grey scale image is generated. The current set view mode (Direct/Quick/...) is used for automatic data transformation. The resulting image will appear like the active view but it has the size of the original scan (no zoom/quench applies).

A special feature: If the scan has type "RGBA" (4 layers) the raw data of the channels is written without any transformation or scaling to the PNG file in RGB mode.

In case of image series/movies this export filter automatically generates a series of png images: bild0001.png ... bild0099.png. To import image series replace 0000 ... 0099 in your file name by any valid C-format string like bild%04d.png.

To scale (call AutoDisp function) each frame incude the string äutodisp in the export filename.

Usage

The plug-in is called by *main menu/File/Import/PNG* and *main menu/File/Export/PNG*.

References

pnglib documentation: <http://www.libpng.org/pub/png/libpng-manual.html>

Info for Plug-In: File/Import/PNG

Plug-In name: png_Im_Export

File: scan/png_im_export.C

Author: Percy Zahl

Email: zahl@users.sf.net

12.20. Binary file Import

Description

The *Binary* plug-in supports reading of a simple binary volumetric data file format.

Data Set File Format:

The data files for rectilinearly sampled scalar data are written in the following format (all fields big-endian binary):

Resolution (number of grid points in x, y and z direction): three 32-bit int values. Let's refer to them as numX, numY, numZ.

Size of saved border around volume: one 32-bit int value. This value is not used in the provided data sets and is set to zero.

True size (extent in x, y and z direction in some unit of measurement): three 32-bit float values. Treat these fields like a sort of "3D aspect ratio" - usually, medical data sets are sampled as a stack of slices, where the distances between slices is different from the distances between pixels in a slice.

Data values: All numX*numY*numZ data values of the volume stored as unsigned char values in the range (0 .. 256). The values are stored in x, y, z order, i.e., x varies slowest, z varies fastest. In other words, they are stored in the memory order of a standard C three-dimensional array unsigned char values

numX

numY

numZ

Usage

The plug-in is called by *main menu/File/Import/Binary*. Only import direction is implemented.

Info for Plug-In: File/Import/Binary

Plug-In name: bin_import File: scan/bin_import.C
Author: Percy Zahl Email: zahl@users.sf.net

12.21. SPA–LEED (SPA4) d2d data file Import

Description

The *spa4_d2d_im_export* plug-in supports reading of SPA–LEED (SPA4) d2d data files.
PlugIn in construction.

Usage

The plug-in is called by *main menu/File/Import/SPA4-d2d*.

Known Bugs

Not yet tested.

Info for Plug-In: File/Import/SPA4-d2d

Plug-In name: spa4_im_export File: scan/spa4_d2d_im_export.C
Author: Percy Zahl Email: zahl@users.sf.net

12.22. Import/export for WSxM Nanotec Electronica SPM data

Description

Data import/export of the WSxM data format (version 2) used by Nanotec Electronica.

Usage

Call it from *main menu/File/Im,Export/WXsM*. Also a normal open or drag- and drop action from gmc, nautilus or Netscape (URL) will automatic import WSxM files. Plugin uses nc2top to export WSxM files. Therefor, it saves a .nc file first, and then runs nc2top. If you want, you can change the path of nc2top in the configuration menu, and you can also set if the .nc files shall be deleted after conversion

References

<http://www.nanotec.es>

The import algorithm is not complete yet. You will get the image, but the scan data are only fake.



Note DnD and URL drops are not tested.

Info for Plug-In: File/Im,Export/WXsM

Plug-In name: WSxM_io
Author: Thorsten Wagner

File: scan/WSxM_io.C
Email: stm@users.sf.net

13. Plug-Ins: math/background

The *math/background* plugins are those used to correct data for background, e.g. subtraction of a regression plane or line.

13.1. Gamma correction

Description

Applies a gamma correction as it is defined here:

$$Z_{\text{range}} := Z_{\text{minval}} - Z_{\text{maxval}}$$

$$Z_{\text{math}} = \frac{Z_{\text{range}} Z_{\text{active}}^\gamma}{Z_{\text{range}}^\gamma}$$

Usage

Call *main menu/Math/Background/Gamma* and give the gamma value γ if prompted.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Gamma

Plug-In name: bggamma
Author: Percy Zahl

File: math/background/bggamma.C
Email: zahl@users.sf.net

13.2. Plane Regression

Description

Plane Regression.

Usage

Write how to use it.

Hacking Version only.

Info for Plug-In: Math/Background/Plane Regression

Plug-In name: plane_regression

File: math/background/plane_regression.C

Author: Percy Zahl

Email: zahl@users.sf.net

13.3. Lineregression

Description

Lineregression action.

Usage

Call *main menu/Math/Background/Lineregression*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Lineregression

Plug-In name: lineregression

File: math/background/lineregression.C

Author: Percy Zahl

Email: zahl@users.sf.net

13.4. Line Regression

Description

This applies the *Quick View* representation to the data. Therefore a least squares fit line regression is calculated on a subset of 30 points for each scan line. This line is subtracted from the data to correct for slope and offset.

Usage

This filter is used for a quick and easy background correction of sample tilt and possible offset changes inbetween lines.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

At this time this filter is not a real PlugIn (in principal it could be), but it resides in the Gxsm Core located in the file `Gxsm/src/xsmmath.C` as a subroutine and cannot be removed, because the core code is dependent on this subroutine.



Doing it better: try using a plane regression! If this works, great! In case there are offset changes inbetween lines, you can try getting better results with an proceeding *main menu/Math/Filter 2D/LineShifts*. If you have some spikes in you image, try removing those first with the *main menu/Math/Filter 2D/Despike*.



Info for Plug-In: Math/Background/Line Regress

Plug-In name: (DocOnly) LineRegress
Author: Percy Zahl

File: `math/background/DocOnlyLineRegress.C`
Email: `zahl@users.sf.net`

13.5. Plane max. propability

Description

Calculates a max propability plane and subtracts it. It's purpose is to find automatically the best fitting plane to orient a stepped/vicinal surface in a way, that the terraces are horizontal.

Usage

Call *main menu/Math/Background/Plane max prop.*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



This code in work in progress, it is originated from PMSTM mpplane.c and was rewritten as a Gxsm math PlugIn. It looks like something does not work like expected, the corrected plane is not right for some still not found reason.

Info for Plug-In: Math/Background/Plane max prop

Plug-In name: plane_max_prop

Author: Percy Zahl, L.Anderson, Greg P. Kochanski

File: math/background

Email: zahl@users.sf.net

13.6. Remove Line Shifts

Description

This filter removes Z line shifts from the image background – most commonly due to tip changes or any other sudden Z changes from one to the next line. It works by comparing the 2nd order change of average scan line Z value to an given threshold value, i.e. it determines a jump in the $\partial_Y Z$ (Y-slope), if so, this jump is evaluated and Z of all following lines is adjusted. The the filter asks for the treashold value.

Usage

Activate source scan and call it filter from *main menu/Math/Background/Rm Line Shifts* menu. Input the desired threshold value.

Info for Plug-In: _Math/Background/Rm Line Shifts

Plug-In name: removelineshifts File: math/background/removelineshifts.C
Author: percy Email: zahl@users.sourceforge.net

13.7. 2nd order scanline correction

Description

Second order line by line background correction: The 2nd order best fit of the Z data of each line is computed and subtracted as background.

This filter can be used for a quick and easy background correction of sample tilt and possible offset changes inbetween lines. Additionally, the correction of an overlaying parabola, e.g. due to the geometry of the experimental setup (bending piezo tube) is

Usage

Activate a scan and select *main menu/Math/Background/Line: 2nd order*.

Sources

The active channel is used as data source.

Destination

The result is put into a new created math channel.

The algorithm is unchecked.



Info for Plug-In: Math/Background/Line

Plug-In name: parabolregress File: math/background/parabolregress.C
Author: Stefan Schröder Email: stefan_fkp@users.sf.net

13.8. Multiply Const Background correction

Description

Multiply multi dim data (only apply to selected range!) with factor.

Usage

Call *main menu/Math/Background/Multiply Const* and give the value when prompted.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Multiply Const

Plug-In name: mulconst	File: math/background/mulconst.C
Author: Percy Zahl	Email: zahl@users.sf.net

13.9. Waterlevel

Description

This plugin adds a waterlevel to the active scan. Everything below this level in the resulting scan will become invisible. This is achieved by setting the z value of all points with original z values below the waterlevel to the waterlevel: if $z(x, y) < \text{waterlevel}$ then $z(x, y) = \text{waterlevel}$.

Usage

Write how to use it.

The computation result is placed into an existing math channel, else into a new created math channel.

This plug-in is still under construction!

Info for Plug-In: Math/Background/Waterlevel

Plug-In name: waterlevel
Author: Andreas Klust

File: math/background/waterlevel.C
Email: klust@users.sourceforge.net

13.10. Stop band removal

Description

Used for zeroing data in selected rectangles and automatically complex conjugated (CC) rectangles. It is in particular made to be used for marking areas in frequency space (e.g. in a calculated Filter 2D/Power Spectrum, this generates a PDS and preserves the full original Re/Im data for manipulation and optional back-transform) for filtering of data in frequency domain. The result can then be transformed back using Filter 2D/IFT 2D (inverse FT) $IFT(FT())$ 2D filter.

Usage

Call *main menu/Math/Background/Stop CC*.

Sources

The active channel is used as data source.

Objects

All data in marked rectangles and CC rectangles are zeroed.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Stop CC

Plug-In name: stop_ccr
Author: Percy Zahl

File: math/background/stop_ccr.C
Email: zahl@users.sf.net

13.11. Template

Description

Template action.

Usage

Call *main menu/Math/Background/Template*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Template

Plug-In name:	template	File:	math/background/template.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

13.12. Plane three points

Description

The filter removes a by three points defined plane from the scan.

Usage

Define the three points using the *Point* object.

Sources

The active channel is used as data source.

Objects

The *Ksys* object is needet to define the plane via the three points provided.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Plane 3 Points

Plug-In name: plane3pkte File: math/background/plane3pkt.C
Author: Percy Zahl Email: zahl@users.sf.net

13.13. Full Timescale FFT

Description

Linear FT of scan data in time, takes all scan data concat as one long stream.

Usage

Activate source channel.

Info for Plug-In: Math/Background/Timescale FFT

Plug-In name: timescalefft File: math/background/timescalefft.C
Author: Percy Zahl Email: zahl@users.sourceforge.net

13.14. Plane regression

Description

The filter calculates a regression plane using a selected rectangular area and subtracts this from the data

This is usually the best way to flatten a good SPM scan, because it sustains the offsets inbetween lines. It allows to select a individual area of the scan as reference area to be flattened.

Usage

Mark the reference area using the rectangle object and call *main menu/Math/Background/E Regression*.

Sources

The active channel is used as data source.

Objects

A optional rectangle needed to select the reference area for the plane regression.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

 Note

 Note

The "E" (E Regression) is historically and stands for German 'Ebene', what means plane. At this time this filter is not a real PlugIn (in principal it could be), but it resides in the Gxsm Core located in the file Gxsm/src/xsmmath.C as a subroutine and cannot be removed, because the core code is dependent on this subroutine.

Info for Plug-In: Math/Background/E Regression

Plug-In name: (DocOnly) Eregress
Author: Percy Zahl

File: math/background/DocOnlyEregress.C
Email: zahl@users.sf.net

13.15. Pass band copy

Description

Used for copying data in selected rectangles and automatically complex conjugated (CC) rectangles. It is in particular made to be used for marking areas in frequency space (e.g. in a calculated Filter 2D/Power Spectrum, this generates a PDS and preserves the full orgininal Re/Im data for manipulation and optional back-transform) for filtering of data in frequency domain. The result can then be transformed back using Filter 2D/IFT 2D (inverse FT) $IFT(FT())$ 2D filter.

Usage

Call *main menu/Math/Background/Pass CC*.

Sources

The active channel is used as data source.

Objects

All data in marked rectangles and CC rectangles is copied.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Pass CC

Plug-In name: pass_cc File: math/background/pass_cc.C
Author: Percy Zahl Email: zahl@users.sf.net

13.16. Sub Const Background correction

Description

Subtracts a constant value from data.

Usage

Call *main menu/Math/Background/Sub Const* and give the value when prompted.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Sub Const

Plug-In name: subconst File: math/background/subconst.C
Author: Percy Zahl Email: zahl@users.sf.net

13.17. Smooth Z drift correction

Description

Corrects a slow and smooth variing Z drift using a polynominal fit for averaged line heights.

Usage

Call *main menu/Math/Background/Z drift correct*. It asks for the degree (2,3,4,...13) used for the polynominal (least squares) fit, 5th order is the default.

Sources

The active channel is used as data source.

Objects

A optional rectangle can be used to restrict the X range used for calculation the average scan line height. The Y coordinates of the rectangle are ignored.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Z drift correct

Plug-In name: bg_z_drift File: math/background/bg_z_drift.C
Author: Percy Zahl Email: zahl@users.sf.net

14. Plug-Ins: math/convert

The *math/convert* plugins are used for scan data type conversion, such as converting from *short* to *float*.

14.1. Convert to byte

Description

Convert scan data type to byte.

Usage

Call *main menu/Math/Convert/to byte*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Conversion from higher to lower dynamic range type may result in overflow, e.g. value wrapping round. There is no saturation mode yet.

Info for Plug-In: Math/Convert/to byte

Plug-In name: to_byte File: math/convert/to_byte.C
Author: Percy Zahl Email: zahl@users.sf.net

14.2. Convert to short

Description

Convert scan data type to short.

Usage

Call *main menu/Math/Convert/to short*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Conversion from higher to lower dynamic range type may result in overflow, e.g. value wrapping round. There is no saturation mode yet.

Info for Plug-In: Math/Convert/to short

Plug-In name:	to_short	File:	math/convert/to_short.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

14.3. Convert to complex

Description

Convert scan data type to complex. Therefore layer 0 is used to hold the absolute value, layer 1 the real part and layer 2 for the imaginary part. A conversion to it will set the imaginary part to zero, a conversion from complex to a scalar type will compute the magnitude from re and im, ignoring the possible existing abs value!

Usage

Call *main menu/Math/Convert/to complex*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Convert/to complex

Plug-In name: to_complex File: math/convert/to_complex.C
Author: Percy Zahl Email: zahl@users.sf.net

14.4. Generate test data

Description

Makes a test data set – for test and demonstration purpose only.

Usage

Call *main menu/Math/Convert/make test*.

Sources

Destination

Info for Plug-In: Math/Convert/make test

Plug-In name: make_test File: math/convert/make_test.C
Author: Percy Zahl Email: zahl@users.sf.net

14.5. Convert Unsigned to float

Description

Convert scan data type to float. While converting a unsigned short in signed short type is expected and corrected. This is a hack for fixing imported data.

The transformation used, applied to all data in H:

$$\text{if } Z < 0 \text{ then } Z' = Z + 2^{16} \text{ else } Z' = Z$$

Usage

Call *main menu/Math/Convert/U to float*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Convert/to float

Plug-In name: uto_float File: math/convert/uto_float.C
Author: Percy Zahl Email: zahl@users.sf.net

14.6. Convert to short, apply custom fix filter

Description

Convert scan data type to short and apply a custom fix (-32768 → +32766).

Usage

Call *main menu/Math/Convert/to short fix*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

This is a special temporary hack, only loaded if Instrument is set to SNOM.

Info for Plug-In: Math/Convert/to short fix

Plug-In name: short_to_short
Author: Percy Zahl

File: math/convert/short_to_short.C
Email: zahl@users.sf.net

14.7. Convert to long

Description

Convert scan data type to long.

Usage

Call *main menu/Math/Convert/to long*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Conversion from higher to lower dynamic range type may result in overflow, e.g. value wrapping round. There is no saturation mode yet.

Info for Plug-In: Math/Convert/to long

Plug-In name: to_long
Author: Percy Zahl

File: math/convert/to_long.C
Email: zahl@users.sf.net

14.8. Convert to double

Description

Convert scan data type to double.

Usage

Call *main menu/Math/Convert/to double*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Convert/to double

Plug-In name: to_double

File: math/convert/to_double.C

Author: Percy Zahl

Email: zahl@users.sf.net

14.9. Convert to float

Description

Convert scan data type to float.

Usage

Call *main menu/Math/Convert/to float*.

Sources

The active channel is used as data source.

Destination

The conversion result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Convert/to float

Plug-In name: to_float File: math/convert/to_float.C
Author: Percy Zahl Email: zahl@users.sf.net

15. Plug-Ins: math/filter1d

The *math/filter1d* plugins are all type off math applied to scan data in a line by line manner – not needing to know what the previous or next line data is, such as 1D derivation, inversion...

15.1. Repair filter

Description

This filter is obsolete for all Gxsm only users. It just fixes a high/low words order bug – occurred while data transport from DSP to host – in line data, e.g. data points on columns X_{2n} and X_{2n+1} are swapped, which happened in one special old version of Xxsm (the predictor of Gxsm). It's here, because the original old data are backed up and may be needed to be processed with this filter to fix.

Usage

Call *main menu/Math/Filter 1D/Repair* to execute.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 1D/Repair

Plug-In name: repair_cs File: math/filter1d/repair_cs.C
Author: Percy Zahl Email: zahl@users.sf.net

15.2. Despike 1d

Description

Despike 1d filter.

Usage

Call *main menu/Math/Filter 1D/Despike*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Despike1d

Plug-In name: despike1d
Author: Percy Zahl

File: math/filter1d/despike1d.C
Email: zahl@users.sf.net

15.3. Template

Description

Template action.

Usage

Call *main menu/Math/Background/Template*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Template

Plug-In name: template File: math/filter1d/template.C
Author: Percy Zahl Email: zahl@users.sf.net

15.4. Koehler filter

Description

The Koehler filter differentiates the image data line by line (1D). It uses a local weightened differentiation and floating averagening.

Starting with the left value Z_0

$$V_0 = Z_0$$

and then using 0.92 of the left and 0.08 of the following Z value:

$$V_i = 0.92V_{i-1} + 0.08Z_i \quad \text{for all } i \in 1, 2, \dots, N_x - 1$$

The next iteration doese the same starting at the right site with the current results:

$$V_i = 0.92V_{i+1} + 0.08V_i \quad \text{for all } i \in N_x - 2, N_x - 3, \dots, 0$$

Finally the difference of all Z values with the weightened and averaged value is computed:

$$Z'_i = Z_i - V_i \quad \text{for all } i \in 0, 1, 2, \dots, N_x - 1$$

Usage

Call *main menu/Math/Filter 1D/Koehler*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

No – the koefficients are constants and can only be changed in the PlugIn itself.

References

Filter is originated to PMSTM and Ulli Koehler?



A similar effect (on a limited lenght) is used the the differential view now on the fly.

Info for Plug-In: Math/Filter 1D/Koehler

Plug-In name: koehler File: math/filter1d/koehler.C
Author: Percy Zahl Email: zahl@users.sf.net

15.5. 1dim differentiation

Description

One dimensional differentiation...

Usage

Call *main menu/Math/Filter 1D/Diff*. This plugin can be called from a remote-control script with the command `action('diff_PI')`. The kernel-size is then set to '5+1'.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 1D/Diff

Plug-In name: diff File: math/filter1d/diff.C
Author: Percy Zahl, Stefan Schroeder Email: zahl@users.sf.net

15.6. Time Domain FFT Filter

Description

This filter acts in one dimension on real time domain of the whole scan. To use it you need the gap-less data of all times, i.e. the forward and backward scan is needed and the scan data should also be gap-less and real time at the turn-over points at the end/start of every scan lines – this is the case for the "SignalRanger" implementation.

Backward scan data must be in reverse (mirrored in X), as provided by the SignalRanger.  Note

Any X-scan directional slope will be automatically removed in time domain by this filter, also the scan DC component will be eliminated.  Note

The filter assembles a one dimensional data set in time domain from scan start to scan end using forward and backward scan data sets. Transforms this into frequency domain, eliminates all multiples of scan-width frequency components (tilt in scan direction) and cuts out the user provided band, then transforms it back into time domain and reassembles one double width images containing forward and backward data.

Usage

Activate a channel containing the forward → scan data and put the channel containing the backward ← scan data into mode X. Then execute the filter *main menu/Math/Filter 1D/t dom filter*. It will ask for the stop band position and half width in inverse pixels.

This plugin is under construction.

Info for Plug-In: Math/Filter 2D/t dom filter

Plug-In name: timedomfftfilter
Author: Percy Zahl

File: math/filter1d/timedomfftfilter.C
Email: zahl@users.sourceforge.net

15.7. Linear stationary differentiation

Description

Edge enhancement via differentiation as follows:

$$I_i = \frac{1}{9} \sum_{k=i-4}^{i+4} Z_k$$

$$Z'_i = \frac{1}{4} \frac{Z_i - I_i}{\sqrt{\frac{1}{2} \sum_{k=i-4}^{i+4} (Z_k - I_k)^2}} + \frac{I_i}{2}$$

Usage

Call *main menu/Math/Filter 1D/Lin stat diff.*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 1D/Lin stat diff

Plug-In name: linear_stat_diff File: math/filter1d/linear_stat_diff.C
Author: Percy Zahl Email: zahl@users.sf.net

15.8. Ft1d filter

Description

The ft1d filter: line by line FT.

Usage

Call *main menu/Math/Filter 1D/Ft1d.*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 1D/Ft1d

Plug-In name: ft1d File: math/filter1d/ft1d.C
Author: Percy Zahl Email: zahl@users.sf.net

16. Plug-Ins: math/filter2d

The *math/filter2d* plugins are all type of math doing something in 2D with the data, e.g. a 2D powerspectrum or all types of 2D FFT-filters. But special 2D statistical analysis or geometric transformation filters are going into a separate sections *math/statistics*, *transformations*.

Gxsm core 2D convolution filter support

The Gxsm core provides support for a generalized convolution, just providing the convolution kernel function itself. The minimal kernel size is $2R + 1$, it is automatically increased until non zero matrix elements are appearing.

A convolution kernel is provided by a kernel function like this Gaus-Kernel:

$$K_{ij} = 4 * e^{-\frac{i^2+j^2}{r^2}}$$

The convolution itself is defined as:

$$Z'(n, m) = \sum_{-R \leq i \leq R} \sum_{-R \leq j \leq R} Z(n + i, m + j) \cdot K_{ij}$$

For all PlugIns using the convolution method, just the kernel will be documented in the following descriptions.

16.1. Curvature

Description

Curvature calculation.

Usage

Call *main menu/Math/Filter 2D/Curvature*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 2D/Curvature

Plug-In name: curvature
Author: Percy Zahl

File: math/filter2d/curvature.C
Email: zahl@users.sf.net

16.2. Line Interpolation

Description

Use this tool to replace distorted line(s) (i.e. caused by manual Z-Offset adjustments while scanning or tip changes) by interpolated data from the lines one before and one after the distortion.

If necessary you should remove line-shifts (use the Lens Shifts filter) before calling this tool. I.e. the average Z before and after the distortion should be about the same.

It can work manually on exactly one line or in an automatic mode with automatic line detection.

Usage

It works in two modes:

a) Assuming line 100 (in pixel coordinates) to be broken as example, mark exactly the line 99 and line 101 with exactly one rectangle objects. You may want to use the rectangle-properties to set the line numbers manually. Then execute it via *main menu/Math/Filter 2D/Line Interpol*.

b) The automatic mode assumes a fairly well flattened image with only distorted single lines. Mark the reference area including the distorted lines using the rectangle object and call *main menu/Math/Filter 2D/Line Interpol*. It will ask for a threshold, which is used to determine if a line is distorted, therefore it compares the average Z within the marked X range of any line in the marked Y range and the line before.

Sources

The active channel is used as data source.

Objects

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

The X position and size of the rectangle does not matter at all for method (a).



Info for Plug-In: Math/Filter 2D/Line Interpol

Plug-In name: (DocOnly) LineInterpol
Author: Percy Zahl

File: math/filter2d/DocOnlyLineInterpol.C
Email: zahl@users.sf.net

16.3. T derive

Description

Not sure what the purpose of this old filter T-derive is...

Usage

Call *main menu/Math/Filter 2D/T derive*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 2D/T derive

Plug-In name: Tderive
Author: Percy Zahl

File: math/filter2d/Tderive.C
Email: zahl@users.sf.net

16.4. Smooth

Description

A 2D Gausian smooth is calculated via convolution with a Gaus-Kernel:

$$K_{ij} = 4 * e^{-\frac{i^2+j^2}{r^2}}$$

Usage

Call *main menu/Math/Filter 2D/Smooth*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



Alternative: Use the Fourier-Filter methods Gaus-Stop/Pass for huge convolutions, it's faster!

Info for Plug-In: Math/Filter 2D/Smooth

Plug-In name: smooth	File: math/filter2d/smooth.C
Author: Percy Zahl	Email: zahl@users.sf.net

16.5. Stat Diff

Description

Stationary Differentiation in scan direction using a convolution kernel:

$$K_{ij} = C \cdot j \cdot e^{-\frac{i^2}{r_y} - \frac{j^2}{r_x}}$$

Usage

Call *main menu/Math/Filter 2D/Stat Diff*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Known Bugs

Crashes Gxsm – pending to fix.

Info for Plug-In: Math/Filter 2D/Stat Diff

Plug-In name: stat_diff File: math/filter2d/stat_diff.C
Author: Percy Zahl Email: zahl@users.sf.net

16.6. Convolution with 3x3 kernel

Description

This plug-in convolutes the active scan with a 3x3 matrix (kernel). The kernel can be changed using *main menu/Tools/Plugin Details* by calling the plug-ins configure function (c.f. [22.8](#)).

Usage

The *smallconvol* plug-in can be found in *main menu/Math/Filter 2D/Small Convol*. It acts on the active channel and the output is put in the math channel.

Info for Plug-In: Math/Filter 2D/Small Convol

Plug-In name: smallconvol File: math/filter2d/smallconvol.C
Author: Percy Zahl Email: zahl@users.sf.net

16.7. Lineinterpol

Description

Lineinterpol action.

Usage

Call *main menu/Math/Filter 2D/Lineinterpol*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 2D/Lineinterpol

Plug-In name: lineinterpol

File: math/filter2d/lineinterpol.C

Author: Percy Zahl

Email: zahl@users.sf.net

16.8. FT 2D

Description

Two dimensional forward Fourier Transformation. Results in three layers (PSD, Re, Im):

Layer 0 is the Power Spectral Density, layer 1 (Re) and 2 (Im) are the Complex numbers of the FT data. This data can be used for reverse transformation after (background) stopp and/or pass operations.

Usage

Info for Plug-In: math-filter2d-sectionFT 2D

Plug-In name: ft2d

File: math/filter2d/ft2d.C

Author: Percy Zahl

Email: zahl@users.sourceforge.net

16.9. inverse FT

Description

Two dimensional inverse (backward) Fourier Transformation of a three layer (PSD, Re, Im) data set, as generated by the forward FT:

Layer 0 is the Power Spectral Density (not used here), layer 1 (Re) and 2 (Im) are the Complex numbers used for the complex input for the IFT operation.

Usage

Activate a channel containing a complex data set (3 layers: PSD, Re, Im).

This plugin is under construction.

Info for Plug-In: Math/Filter 2D/IFT 2D

Plug-In name: ift2d File: math/filter2d/ift2d.C
Author: Percy Zahl Email: zahl@users.sourceforge.net

16.10. Despike 2d

Description

Despike 2d filter.

Usage

Call *main menu/Math/Filter 2D/Despike*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Despike2d

Plug-In name: despike2d
Author: Percy Zahl

File: math/filter2d/despike2d.C
Email: zahl@users.sf.net

16.11. Template

Description

Template action.

Usage

Call *main menu/Math/Background/Template*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Background/Template

Plug-In name: template
Author: Percy Zahl

File: math/filter2d/template.C
Email: zahl@users.sf.net

16.12. Local height

Description

Computes the local height via convolution.

Usage

Call *main menu/Math/Filter 2D/Local height*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 2D/Local height

Plug-In name: local_height

File: math/filter2d/local_height.C

Author: Percy Zahl

Email: zahl@users.sf.net

16.13. Edge

Description

A 2D Laplace of Gaussian (LoG) edge detect filter kernel is calculated and applied via convolution to teh source data set, feature size σ :

$$K_{ij} = -\frac{1}{\pi\sigma^4} \left(1 - \frac{i^2 + j^2}{2\sigma^2}\right) e^{-\frac{i^2+j^2}{2\sigma^2}}$$

Usage

Call *main menu/Math/Filter 2D/Edge*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Filter 2D/Edge

Plug-In name: edge File: math/filter2d/edge.C
Author: Percy Zahl Email: zahl@users.sf.net

16.14. Normal Z

Description

Z Normal component is calculated.

Usage

Call *main menu/Math/Filter 2D/Normal Z*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



Info for Plug-In: Math/Filter 2D/Normal Z

Plug-In name: normal_z File: math/filter2d/normal_z.C
Author: Percy Zahl Email: zahl@users.sf.net

17. Plug-Ins: math/probe

The *math/probe* plugins are designated for probe data analysis and extraction or separation. They deal with all kind of probe and event data attached to a scan.

A quick Overview of the ScanEvent interface: – to be translated –

Also zum Einstieg, solltet Ihr Euch ein wenig mit dem Event-Handling vertraut machen, mal `scan_event.[hC]` studieren -- die NetCDF load/save routinen könnt Ihr einfach irgnorieren. Diese wird dann von `mem2d.[hC]` verwendet, jedoch weniger wichtig, einzige was in `mem2d.h` von interesse ist, ist die eine (und einzige) Liste (public in `mem2d`)

```
GSList *scan_event_list;
```

die alle ScanEvents (Probe und User, etc...!!) enthält.

Also zum Überblick vielleicht gebe ich mal eine Kurzeinführung:

`GSList *scan_event_list;` ist eine `GSList` die pointer von Type `ScanEvent` enthält.

Class `ScanEvent` ist ein Abstrakter Event (jedes types), und `ScanEvent` hält nicht mehr als die Koordinaten XYV + Event(s) (eine Art Vierervektor, wenn man die Events selber noch mitnimmt und als t sieht-:). (V == Value, optional und zur Zeit noch nicht verwendet, ist jedoch nur logisch vorzusehen, wenn ich an Layered Scan denke (jeder Layer N hat einen zugeordneten Value "V"), XY Koordinate ist in absoluten Angstroems (incl Offset/Rotation!).

Also:

```
Der ScanEvent im Überblick, der Konstructor: (benötigt Ihr nicht)
class ScanEvent
ScanEvent (double xPos, double yPos, double Val=0.);
```

Wichtige Member Funktionen ggf. von Interesse:

```
void add_event (EventEntry *ee); // Einen realen Daten-Event hinzufügen
double get_position (double &x, double &y); // Position abrufen
double distance (double *xy); // Abstand zu Punkt xy, ist eine Funktion
quint get_event_count() ; // Anzahl Events in Liste
```

Data:

```
GSList *event_list; // Liste von Daten-Events "hier" bei X11
                     // i.a. nur ein DatenEvent, type der Point
```

Dann wäre da die Basis Class "EventEntry" (Ist Basis aller "Daten-Events")

```
class EventEntry
EventEntry (gchar *Name, time_t Time);
```

Ein EventEntry hat also einen Namen und eine Zeit (Zeit Koordinate, time_t)

Wichtig: Ich habe als "Konvention" Probe Events mit dem Namen "Probe" versehen und User Events mit dem Namen "User", ist zum Identifikationszweck vorgesehen. Und Weiterhin sollen die Anfangsbuchstaben von Probe und User eindeindeutig sein, d.h. 'P' für Probe:

```
gchar* description () { return name; }; // der volle Name
gchar description_id () { return name[0]; }; // Identifikator
```

Dann gibt es den ProbeEntry (etwas gestrippeder Code) für alle Probenartigen Daten:

```
class ProbeEntry : public EventEntry{
public:
    ProbeEntry (gchar *Name, time_t Time, GPtrArray *Labels, GPtrArray
*Unitssymbols, int Chunk_size); // Name, zur Zeit nur "Probe", time_t Time
// dann ein Array von Pointern auf Labels und auf Unitssymbols, und
```

```
// Chunk_size ist die Anzahl der aufgezeichneten Channels des Probe, aus
// Performance gründen speichere ich den ganze Probe-Datensatz von
// N-Punkten * J-Channels in einem linearen GArray ab mit folgender
// linearisierter Indizierung: (n*chunk_size+j) -- OK?

    virtual double get (int n, int j) // gibt den Probe Wert des N-ten
// Punktes, Channel j zurück -- die j-indizierung stimmt mit der "j"
// Indizierung von Labels und Unitssym überein. - OK?

    int get_num_sets () { return num_sets; }; // Anzahl Daten Punkte (N)
    int get_chunk_size () { return chunk_size; }; // Anzahl Channels (J)
    gchar *get_label (int j);
    gchar *get_unit_symbol (int j);
}
```

Analog gibt es dann noch

```
class UserEntry : public EventEntry{
public:
    UserEntry (gchar *Name, time_t Time, GPtrArray *Messages=NULL, int
num=0); // Name == "User" per default
...}
```

Wichtig ist, das man bevor man einen Event aus der EventListe als
ProbeEvent interpretiert den Type checked via der "ID" Funktion! Klar?

That's it.

17.1. Probe Image Extract

Description

Extract selected probe data/values and generates a new image from that. Select a *Rectangle*.

Usage

Call *main menu/Math/Probe/Probe_Image_Extract*

Sources

The active channel is used as data source.

Objects

A rectangle object is needed. Probe events must be present in the active scan.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

The PlugIn configurator allows to set a default index and a channel. If this is non zero the user is not prompted!

Info for Plug-In: Math/Probe/Probe_Image_Extract

Plug-In name: probe_image_extract
Author: Bastian Weyers

File: math/probe/probe_image_extract.C
Email: weyers@users.sf.net

17.2. AFM (NC-AFM) mechanical tip apex/molecule imaging simulations

Description

Simulating NC-AFM images and force curves. Based on publication:

PHYSICAL REVIEW B 90, 085421 (2014) Mechanism of high-resolution STM/AFM imaging with functionalized tips Prokop Hapala –Institute of Physics, Academy of Sciences of the Czech Republic, v.v.i., Cukrovarnick a 10, 162 00 Prague, Czech Republic; Georgy Kichin, Christian Wagner, F. Stefan Tautz, and Ruslan Temirov – Peter Grünberg Institut (PGI-3), Forschungszentrum Jülich, 52425 Jülich, Germany and Jülich Aachen Research Alliance (JARA), Fundamentals of Future Information Technology, 52425 Jülich, Germany; Pavel Jelinek – Institute of Physics, Academy of Sciences of the Czech Republic, v.v.i., Cukrovarnick a 10, 162 00 Prague, Czech Republic and Graduate School of Engineering, Osaka University 2-1, Yamada-Oka, Suita, Osaka 565-0871, Japan

High-resolution atomic force microscopy (AFM) and scanning tunneling microscopy (STM) imaging with functionalized tips is well established, but a detailed understanding of the imaging mechanism is still missing. We present a numerical STM/AFM model, which takes into account the relaxation of the probe due to the tip-sample interaction. We demonstrate that the model is able to reproduce very well not only the experimental intra- and intermolecular contrasts, but also their evolution upon tip approach. At close distances, the simulations unveil a significant probe particle relaxation towards local minima of the interaction potential. This effect is responsible for the sharp submolecular resolution observed in AFM/STM experiments. In addition, we demonstrate that sharp apparent intermolecular bonds should not be interpreted as true hydrogen bonds, in the sense of representing areas of increased electron density. Instead, they represent the ridge between two minima of the potential energy landscape due to neighboring atoms.

and

related Supplementary Material: The mechanism of high-resolution STM/AFM imaging with functionalized tips.

Usage

Call *main menu/Math/Probe/AFM mechanical sim*

Sources

The active channel is used as geometry/size/offset template only. Must be of type DOUBLE.

Objects

Input data file to load as external molecule/structure model.

"model.xyz" type file. 1st line: "N" number of atoms, 2nd line comment/name/info – ignored. Then atom 1...N in following lines. Format:

17. Plug-Ins: math/probe

El X Y Z C 0 0 0 O 2.365 0.213 0.03 Cu 0.1 0.7 -3.5 ...

El=Element Symbol, must be two characters like " C" " O" "Cu", then X Y Z coordinates in Angstroem

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

The PlugIn configurator...

Info for Plug-In: Math/Probe/AFM_mechanical_sim

Plug-In name: Unnamed	File: math/probe/afm_lj_mechanical_sim.C
Author: Percy Zahl	Email: zahl@users.sf.net

18. Plug-Ins: math/misc

The *math/misc* plugins are all types of math not fitting well into all other sections here.

18.1. Smoothing layers in 3D

Description

This filter extracts from a series of images in energy (layer-dimension) the energy (layer value lookup) finding the MIN index/ image intensity at each pixel. Resulting energy (layer corresponding value) is placed into the destination image.

Usage

Call *main menu/Math/Misc/Minzlayer...*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



Info for Plug-In: Math/Misc/Minzlayer

Plug-In name: minzlayer
Author: Percy Zahl

File: math/misc/minzlayer.C
Email: zahl@users.sf.net

18.2. Absolute value

Description

Takes the absolute value.

Usage

Call *main menu/Math/Arithmetic/Absolute Value*.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Info for Plug-In: Math/Arithmetic/Absoluet Value

Plug-In name: make_volume
Author: Percy Zahl

File: math/misc/make_volume.C
Email: zahl@users.sf.net

18.3. Cut spectra from 3D data

Description

This plug-in is made for choosing areas from a topographic scan using the *rectangle* object. The spectra in the correspondent spectroscopic data taken simultaneously with the topographic data are then cut out and saved in a file. Currently, only an index vector is saved in the GNU Octave ASCII format for further analysis of the data. Its filename is hard coded ("tmp/spec.ivec").



This plug-in is in a very alpha stage.

Usage

The topographic scan must be the active, the scan with the spectroscopic data the X channel. The plug-in is called via *main menu/Math/Misc/Spectrocut*.

Info for Plug-In: Math/Misc/Spectrocut

Plug-In name: spectrocut
Author: Andreas Klust

File: math/misc/spectrocut.C
Email: klust@users.sf.net

18.4. Smoothing layers in 3D

Description

This filter applies a gaussian smooth to all layers of a scan. It can also smooth accross layers in 3D, using a smoothing radius in layer-dimension.

Usage

Call *main menu/Math/Misc/Layersmooth...*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Not yet finished, the 3D smooth (accross layers) is not yet implemented. Please set  Note Radius in Layer Dim to zero!

Info for Plug-In: Math/Misc/Layersmooth

Plug-In name: layersmooth
Author: Percy Zahl

File: math/misc/layersmooth.C
Email: zahl@users.sf.net

18.5. Create a shape polygon from lines

Description

A brief description goes here.

Usage

Call *main menu/Math/Misc/Shape*.

Info for Plug-In: Math/Misc/Shape

Plug-In name: shape File: math/misc/shape.C
Author: Percy Zahl Email: zahl@users.sf.net

18.6. PSD add – SARLS

Description

Experimental filter for adding PSD-slope-signals.

Usage

It is only available from *main menu/Math/Misc/PSD Add* if the instrument type is set to SARLS or a reload of all PlugIns is forced.

Sources

The active and X channel is used as data sources.



I would appreciate if one of the current SARLS group members could figure out more about this piece of code.

Info for Plug-In: Math/Misc/PSD Add

Plug-In name: psdadd File: math/misc/psdadd.C
Author: Martin Langer, PZ Email: stefan_fkp@users.sf.net

18.7. Find local maxima – SARLS

Description

Finds local maxima in a scan. One (unknown to me, PZ) unknown guy (M.Langer?) of the SARLS group in hannover wrote this PlugIn for searching of local maxima. It looks somehow specialized to me.

Usage

It is only available from *main menu/Math/Misc/Find Loc Max* if the instrument type is set to SARLS or a reload of all PlugIns is forced.

Sources

The active channel is used as data source.

Files

Saves a list of local maxima in Ascii format (List of X Y Z) to a file.

I would appreciate if one of the current SARLS group members could figure out more about this piece of code.  Note

Info for Plug-In: Math/Misc/Find Loc Max

Plug-In name: findlocmax File: math/misc/findlocmax.C
Author: M. Langer? Email: stefan_fkp@users.sf.net

18.8. Smoothing layers in 3D

Description

This filter extracts from a series of images in energy (layer-dimension) the workfunction assuming the image intensity dropping all sudden by a given percentage from it's near zero energy value. (LEEM Mirror-Mode to Regular imaging transition detection) and puts teh resulting energy (layer corresponding value) into the destination image.

Usage

Call *main menu/Math/Misc/Workfuncextract...*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



Note

Info for Plug-In: Math/Misc/Workfuncextract

Plug-In name: workfuncextract
Author: Percy Zahl

File: math/misc/workfuncextract.C
Email: zahl@users.sf.net

19. Plug-Ins: math/transform

The *math/transform* plugins are all type of math doing geometric transformations with the image like rotation, scaling, etc.

19.1. 90° clockwise rotation

Description

This plug-in rotates the active scan clockwise by 90°.

Usage

background corrections such as line regression of scans with the scan direction up-down instead of left-right. Just rotate the scan and apply the usual background correction functions.

Info for Plug-In: math-transformations-sectionRotate 90deg

Plug-In name: rotate90 File: math/transform/rotate90.C
Author: Andreas Klust Email: klust@users.sf.net

19.2. Shear along X

Description

To shear a image along X use this transformation.

Usage

Call *main menu/Math/Transformation/Shear X* and fill in the shear angle as prompted.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformation/Shear X

Plug-In name: shear_x File: math/transform/shear_x.C
Author: Percy Zahl Email: zahl@users.sf.net

19.3. Quench Scan

Description

This filter quenches the scan to half size, therefor 2x2 pixels are averaged.

Usage

Call *main menu/Math/Transformation/Quench Scan*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Quench Scan

Plug-In name: quenchscan File: math/transform/quenchscan.C
Author: Percy Zahl Email: zahl@users.sf.net

19.4. Multi Dimensional Transposition

Description

For now this tool swappes time and layer dimensions.

Usage

Activate a channel and run it.

Info for Plug-In: math-transformations-sectionMulti Dim Transpose

Plug-In name: multi_dim_transpose
Author: Percy Zahl

File: math/transform/multi_dim_transpose.C
Email: zahl@users.sf.net

19.5. Shear along Y

Description

To shear a image along Y use this transformation.

Usage

Call *main menu/Math/Transformation/Shear Y* and fill in the shear angle as prompted.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformation/Shear Y

Plug-In name: shear_y
Author: Percy Zahl

File: math/transform/shear_y.C
Email: zahl@users.sf.net

19.6. Scale scan in X and Y

Description

This filter scales the scan by given factors in X and Y. The resulting pixel value is computated by using a 2d linear approximatione inbetween neigbour points.

Usage

Call *main menu/Math/Transformations/Scale Scan*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Scale Scan

Plug-In name: scalescan	File: math/transform/scalescan.C
Author: Percy Zahl	Email: zahl@users.sf.net

19.7. Movie Concat

Description

Movie Concat allows to concatenate two movie data set in time with a chosen range in time and in layers of both sources. The number of layers chosen for each must match.

Usage

The active and X marked channels are used as data sources.

Info for Plug-In: math-transformations-sectionMovie Concat

Plug-In name: movieconcat	File: math/transform/movieconcat.C
Author: Percy Zahl	Email: zahl@users.sourceforge.net

19.8. Flip diagonal

Description

Flips an image along its diagonale.

Usage

Call *main menu/Math/Transformations/Flip Diagonal*.

Sources

The active channel is used as data source.

Info for Plug-In: Math/Transformations/Flip Diagonal

Plug-In name: flip_diagonal
Author: A. Klust, P. Zahl

File: math/transform/flip_diagonal.C
Email: zahl@users.sf.net

19.9. Auto Align Multi Dimensional Movie

Description

The purpose of this method is to autoalign image series in multiple dimensions, i.e. to correct for image drift. The GXSM autoalign Plug-In does automatic image alignment for translation, scaled rotation, rigid body and affine transformation, of an image series in time and layer domain (i.e. for drift correction). The underlying algorithm of this code is based on the following paper, implemented as GXSM-Plugin, ported from JAVA to C++, optimized and multithreaded by P. Zahl.

Usage

Activate the movie and run it.

This work is based on the following paper:

P. Thevenaz, U.E. Ruttimann, M. Unser A Pyramid Approach to Subpixel Registration Based on Intensity IEEE Transactions on Image Processing vol. 7, no. 1, pp. 27-41, January 1998.

This paper is available on-line at <http://bigwww.epfl.ch/publications/thevenaz9801.html>
Other relevant on-line publications are available at <http://bigwww.epfl.ch/publications/>
Additional help available at <http://bigwww.epfl.ch/thevenaz/turboreg/>

You'll be free to use this software for research purposes, but you should not redistribute it without our consent. In addition, we expect you to include a citation or acknowledgment of both projects whenever you present or publish results that are based on it.

Info for Plug-In: Math/Transformations/Auto Align

Plug-In name: autoalign
Percy Zahl, P. Thevenaz, U.E. Ruttimann, M. Unser

File: math/transform/autoalign.C
Email: zahl@users.sf.net

19.10. Mirror X

Description

Mirrors a image along X.

Usage

Call *main menu/Math/Transformations/Mirror X*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Mirror X

Plug-In name: mirror_x
Author: A. Klust, P. Zahl

File: math/transform/mirror_x.C
Email: zahl@users.sf.net

19.11. Rotate a scan area

Description

To rotate a selected area select a *Rectangle* of the area to be rotated into. Think inverse, the result is the cropped area of the source scan, which is rotated before around the center of the selected area. If needed data points are outside, they are replaced by the value found at the closed edge.

Usage

Place a rectangle object and call *main menu/Math/Transformation/Rotate*. It prompts for the rotation angle (clockwise) if not set to any other default than zero via the PlugIn configurator.

Sources

The active channel is used as data source.

Objects

A rectangle object is needed.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

The PlugIn configurator allows to set a default angle. If this is non zero the user is not prompted for a rotation angle!

Info for Plug-In: Math/Transformation/Rotate

Plug-In name:	rotate	File:	math/transform/rotate.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

19.12. SPA–LEED octopole distortion correction

Description

This is a experimental filter to reduce the SPA–LEED typical octopole cushion distortion using a "ab-initio" like approch. This plugin applies the inverse of this distortion to any image.

Input parameters are: Energy, StepSize and Origin (invariant Point, if not set the user is prompted for coordinates!). A free parameter b (distortion strength, may be left at default value (-0.4)).

Usage

Place a point object and call *main menu/Math/Transformation/Octo Corr.*

Sources

The active channel is used as data source.

Objects

A point object is needed to mark the center of distortion symmetry (invariant point).

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformation/Octo Corr

Plug-In name: OctoCorr	File: math/transform/OctoCorr.C
Author: Percy Zahl	Email: zahl@users.sf.net

19.13. Auto Align

Description

Apply a simple drift in pixels per frame.

Usage

Figure out the drift inbetween frames in pixels in X and Y dimension to compensate for.

Info for Plug-In: Math/Transformations/Auto Align

Plug-In name: mandriftfix	File: math/transform/mandriftfix.C
Author: Percy Zahl	Email: zahl@gxsm.sf.net

19.14. Volume Transform

Description

Transform/Rotate a XY-Layer set/volume along any given axis laying in XY plane by theta. A new data set with same layer number (or any given) is computed my linear (1st order) interpolation of data.

Usage

Activate a channel and run it. Needs volumetric data, i.e. a set of images in layer dimension

Info for Plug-In: math-transformations-sectionVolume Transform

Plug-In name: volume_transform File: math/transform/volume_transform.C
Author: Percy Zahl Email: zahl@users.sf.net

19.15. Unwraps Z data in given range

Description

Flips an image along it's diagonale.

Usage

Call *main menu/Math/Transformations/Unwarp*.

Sources

The active channel is used as data source.

Info for Plug-In: Math/Transformations/Unwrap

Plug-In name: unwrap File: math/transform/unwrap.C
Author: P. Zahl Email: zahl@users.sf.net

19.16. Multi Dimensional Layer Reverse

Description

Reverse Layer Order.

Usage

Activate a channel and run it.

Info for Plug-In: math-transformations-sectionReverse Layers

Plug-In name: reverse_layers	File: math/transform/reverse_layers.C
Author: Percy Zahl	Email: zahl@users.sf.net

19.17. Horizontal merge

Description

Used to horizontal merge two scans together. The scan are expected to have same height.

Usage

Call *main menu/Math/Transformations/Merge H*.

Sources

The active and X channel are merged.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Merge H

Plug-In name: merge_h	File: math/transform/merge_h.C
Author: Percy Zahl	Email: zahl@users.sf.net

19.18. Transformation Shift-Area

Description

This plugin shifts the lower part of a scan with respect to the upper part, according to a chosen line.

Usage

Choose a line Object an connect the two points with your line, that shall be brought together. The green point must be lower than the red one.

Sources

You need one active scan and a line object. Rectangle works two, the diagonal will work like the line.

Objects

If a rectangle is selected the calculated information applies to the content of the rectangle. Otherwise, the whole scan is analyzed.

Destination

A new scan will be created, which contains the unchanged upper part and the shifted lower part, connected.

Info for Plug-In: Math/Transformation/Shift Area

Plug-In name: shiftarea
Author: Stefan Schröder

File: math/transform/shiftarea.C
Email: stefan_fkp@users.sf.net

19.19. Affine Transformation

Description

In case you want to undo a linear distortion of an image (e.g. slow and continuous drift/creep) this transformation helps. It applies a affine transformation to the image, e.g. two arbitrary oriented vectors (and the image) are transformed to be a orthogonal system afterwards.

Usage

Use a *Ksys* to set the vectors. The relative length between both is ignored, they are normalized before.

Sources

The active channel is used as data source.

Objects

A *Ksys* object is used for setting the two vectors.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



If the image is rotated or flipped more than expected, try flipping the *Ksys*! I know it's a bit tricky, good ideas to fix this are welcome!

Info for Plug-In: Math/Transformation/Affine

Plug-In name: affine File: math/transform/affine.C
Author: Percy Zahl Email: zahl@users.sf.net

19.20. Mirror Y

Description

Mirror scan along Y.

Usage

Call *main menu/Math/Transformations/Mirror Y*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Mirror Y

Plug-In name: mirror_y
Author: A. Klust, P. Zahl

File: math/transform/mirror_y.C
Email: zahl@users.sf.net

19.21. Vertical merge

Description

Used to vertically merge two scans together. The scan are expected to have same width.

Usage

Call *main menu/Math/Transformations/Merge V*.

Sources

The active and X channel are merged.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Transformations/Merge V

Plug-In name: merge_v
Author: Percy Zahl

File: math/transform/merge_v.C
Email: zahl@users.sf.net

20. Plug-Ins: math/statistik

The *math/statistik* plugins are all type of math doing some statistical analysis like a histogram creation.

20.1. Opencvmatch

Description

The OpenCV Match...

Usage

Call it from Gxsm Math/Statistics menu. It will prompt for the number of bins and provides a estimated number as default. Also the current min/max Z-value limits and range is shown for informative purpose.

Sources

The active channel is used as data source.

Objects

A optional rectangle can not be used. Use crop before!

Destination

The computation result is placed into an new profile view.

Find out what happens with more or less bins!



Info for Plug-In: Math/Statistics/Opcvmatch

Plug-In name: opencvmatch
Author: Percy Zahl

File: math/statistik/OpenCV_match.C
Email: zahl@users.sf.net

20.2. Vacancy Line Analysis

Description

This plugin is used to find the location of dimer vacancy lines. It was designed to determine statistics of thin layers of SiGe. A very important feature to note is that scans must be rotated so that the vacancy lines are vertical. Lines are searched for vertically down the scan. Line objects are used to mark the start and stop positions for the vacancy lines

Four output results can be created. The most apparent is the copy of the scan with the position of the vacancy lines standing out as raised up. A second layer of that scan shows a 3D histogram of the relative horizontal positions of two vacancies on the same vacancy line separated by n dimer rows. $n = 0$ is shown at the top of the screen and the separations are all 0. The next row down shows a histogram for $n = 1$, the next is $n = 2$ and so forth up to N rows down defined in the configuration (Dimer Rows included in histogram). The positions of the vacancies are also saved to a file. The last output is an histogram of the space between vacancy lines.

Usage

Call it from *main menu/Math/Statistics/Vacancy Line Analysis*.

Sources

The active channel is used as data source.

Objects

Lines must be used to show where the vacancy lines should start and end. Use one line for each vacancy line to be located.

Destination

The position of the vacancy lines is displayed in the math channel or a new channel if a math channel is not open. This channel will have a second layer showing a histogram of the vacancy line straightness. If a histogram showing the vacancy line separation is chosen, a new profile window is also opened.

Configuration

Several parameters are adjustable in the configuration window.

Dimer Spacing The spacing of the dimers – is 7.68 Å on Si.

Max Vacancy Line Shift The horizontal distance searched in each direction for the position on the dimer vacancy. Typical values are 2...3 atomic rows.

Pixels to Average Number of pixels in horizontal direction to average in searching for vacancy.

Dimer rows included in 3D Histogram The Maximum number of dimer rows between two vacancies on the same vacancy line to be included in the histogram. (N as described above in the Description area)

3D Histogram Size The size can be either 1 : 1 (default), meaning that the horizontal scale is correct. If 0 is entered the output is scaled to fill the window.

Line-Spacing Histogram Bin Width The width of each bin in the histogram showing the vacancy line spacing. If 0 is entered no histogram is created.

Files

The user is prompted for an output file. A matrix is saved in the file containing the positions of the vacancy lines. Each column in the matrix represents a vacancy line. Each row represents a dimer row. The value of a matrix position is the horizontal pixel coordinates of the dimer vacancy. A value of -1 means is before the beginning of the line or after the end of the line. Two rows before the matrix are used to designate the row number for the first and last dimer vacancy in that column (vacancy line). The first row is designated as 0. Also included in the heading is the dimer spacing used, which gives the conversion for rows to vertical position, and the x and y pixel spacing. The x pixel spacing provide a conversion from pixel position to horizontal position.

Here is a quick check list:



1. Rotate the scan so the Vacancy lines are as close to vertical as possible.
2. Move start and stop positions or divide a line in two using two
3. Save your line objects – it is time consuming to put them in and
4. Determine the correct dimer spacing for your image using a

Info for Plug-In: Math/Statistics/Vacancy Line Analysis

Plug-In name: VacancyLineAnalysis
Author: J.S. Palmer

File: math/statistik/VacancyLineAnalysis.C
Email: jspalmer@mines.edu

20.3. Generate a polar histogramm

Description

This PlugIn generates from two data sets one (polar) histogram...

Usage

Example:

1. open the demo Ge pyramid image
2. optional for slow machines, scale by x and y 0.2 (*main menu/Math/Transformation/Scan Scan*), then activate this channel
3. call *main menu/Math/Statistics/Slope Abs*, use Facet Radius = 5 (with scale down image)
4. do *Autodisp* on resulting *Math* channel
5. activate the original (e.g. scaled down image)
6. set a free channel to *Math*
7. call *main menu/Math/Statistics/Slope Dir*
8. the “Slope Abs” resulting channel to *Mode-Active*
9. the “Slope Dir” resulting channel to *Mode-X*
10. call *main menu/Math/Statistics/Polar Hist*
angular slices 180
data channels 200
data start 0
data end 45
Vmode 1

data start = data end = 0 : auto ranging (min/max) is used

Sources

The active channel is used as histogramm data source, the X-channel is used as bin

Objects

A optional rectangle is used for data extraction...

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Known Bugs

Produces sometimes strange output... pending to fix!

Docu not finished yet, PlugIn makes Gxsm unstable after usage – work in progress.



Info for Plug-In: Math/Statistics/Polar Histogram

Plug-In name: polarhist File: math/statistik/polarhist.C
Author: Percy Zahl Email: zahl@users.sf.net

20.4. Nndistribution

Description

The NN-distribution plugin calculates the nearest neighbour (lateral) distribution of marker groups manually placed in the active channel using a default (estimated from data) or user provided number of bins. Option to auto-recenter markers on max or min locally.

Usage

Call it from Gxsm Math/Statistics menu. It will prompt for the number of bins and provides a estimated number as default. Also the current min/max Z-value limits and range is shown for informative purpose.

Sources

The active channel is used as data source.

Objects

A optional rectangle can not be used. Use crop before!

Destination

The computation result is placed into an new profile view.



Find out what happens with more or less bins!

Info for Plug-In: Math/Statistics/Nndistribution

Plug-In name: nndistribution

File: math/statistik/nndistribution.C

Author: Percy Zahl

Email: zahl@users.sf.net

20.5. SPA–LEED simulation

Description

This Plugin simulates a SPA-LEED measurement (asks for the electron wave lenght λ as input) using the following transformation:

1. phase transformation, using

$$e^{\frac{2\pi i Z(x,y)}{\lambda}}$$

2. 2 dim. fourier transformation of the phase transformed image

3. the resulting intensity is stored: $|FT(\text{phase trans. image})|^2$

The resulting scan is scaled to have a size of ± 100 (full width is one. To automatically calculate realspace dimensions Γ from inverse of spot separation (e.g. from center (0,0) to some spot or lenght of the line object) enable *InvAng* with the PlugIn configurator! By default coordinates in pixels are used (center is (0,0)).



Remember: One pixel distance (e.g. in X) corresponds to the full width of the original picture!

Usage

Call from *main menu/Math/Statistics/SPALED Sim.* and input λ in Angstroems.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

Using the Plug-In configurator you can preset the wave lenght λ .

Known Bugs

The show-line object did not work with the *InvAng* setting – sorry, but you can still save the data, but it will not show any profile.

You may want to carefully background correct your image before! Check also for correct step heights, if applicable.  Hint

Info for Plug-In: Math/Statistik/SPALED Sim.

Plug-In name:	spasim	File:	math/statistik/spasim.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

20.6. OpenCV Re-Center Feature

Description

The OpenCV Recenter Feature identifies the most likely position of a given template feature (hold in a Channel set to Mode-X) in the active channel and sets the Scan-Offset to the resulting position.

Usage

Call it from Gxsm Math/Statistics menu.

Sources

The active channel is used as data source. Channel set to X-Mode is used as template.

Objects

Destination

The computation result of matching thresholds is placed into an new math channel for reference.



Info for Plug-In: Math/Statistics/Opencvrecenter

Plug-In name: opencvrecenter

File: math/statistik/OpenCV_recenter.C

Author: Percy Zahl

Email: zahl@users.sf.net

20.7. Vorlage (Template) PlugIn

Description

This is a “Vorlage” (German for template) PlugIn. It’s purpose is to be a template for easy start with writing a new PlugIn and for demonstration how a simple PlugIn works as well. The code is extensively commented and it includes a nine step instructions list for starting your new PlugIn.

Usage

This PlugIn is not build and loaded, because it is not listed in Makefile.am.

Building a new PlugIn in nine steps

1. Make a copy of this vorlage.C to your_plugins_name.C!

2. Replace all “vorlage” by “your_plugins_name”
→ please do a search and replace starting here (top of file) NOW!! (Emacs does preserve caps!)
3. Decide: One or Two Source Math:
search for “#define GXSM_ONE_SRC_PLUGIN_DEF”
4. Fill in GxsmPlugin Structure, see below
5. Replace the “about_text” below a desired
6. Optional: Start your Code/Vars definition below (if needed more than the run-fkt itself!), search for “6.”. please, and see comment there!!
7. Fill in math code in vorlage_run(), have a look at the Data-Access methods infos at end
8. Add vorlage.C to the Makefile.am in analogy to others
9. Make a “make; make install”
 - A. Call *main menu/Tools/reload Plugins*, be happy!
 - B. Have a look at the PlugIn Documentation section starting at the beginning (this is, what you are reading here!) and please fill out this section to provide a proper documentation.
→ rebuild the Gxsm manual in Gxsm/Docs/Manual:
run ./docuscangxsmplugins.pl; latex Gxsm-main there!
- ... That’s it!

Sources

The active channel is used as data source.

Objects

A optional rectangle is used for data extraction...

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

Describe the configuration options of your plug in here!

Files

This PlugIn is located here: Gxsm/plug-ins/math/statistik/vorlage.C
If your PlugIn uses, needs, creates any files, then put the info here!

References

Any references about algorithm sources, etc.?

Known Bugs

Are there known bugs? List! How to work around if not fixed?



If you have any additional notes, place them here!



Any hints, tips or tricks? Yeah!

Check out the more automatic math PlugIn building script: Go to dir Gxsm/plug-ins and run generate_math_plugin.sh there!

And never mind, use any existing PlugIn as template as well, but please please copy and rename it properly before!

Info for Plug-In: Math/Misc/Vorlage

Plug-In name:	vorlage	File:	math/statistik/vorlage.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

20.8. SPA–LEED 1D profile simulation k_z .

Description

Calculates SPA–LEED Profiles over phase (S) e.g. a k_z -plot:

For each phase S (S=2 StepHeight/WaveLength) a 1D fourier transformation is calculated for all image lines, which are phase transformed before. All transformed lines for this phase are summed up and stored to the new image as Line 'S'.

Def. Phase Transformation:

$$e^{2\pi i SZ(x,y)}$$

Algorithm (shortened, extracted from source):

```

for(i=0, S=PhaseStart; i<Dest->data.s.ny; S+=PhaseStep, ++i) {
    // PhaseTrans:
    // transform scan data to complex data with correct phase
    double sf = 2. * M_PI * S * Src->data.s.dz / StepHeight;
    for (int line=0; line < Src->mem2d->GetNy(); line++) {
        Src->mem2d->data->SetPtr(0, line);
        for (int col=0; col < Src->mem2d->GetNx(); col++) {
            double arg = sf * Src->mem2d->data->GetNext();
            c_re(htrans[col]) = cos(arg);
            c_im(htrans[col]) = sin(arg);
        }
    }

    // do FFT
    fftw( plan, 1, htrans, 1, 0, hkspc, 1, 0);

    // StoreAbsolute, Add to Dest [double]
    Dest->mem2d->data->SetPtr(0, i);
    for (int j = 0; j<Src->mem2d->GetNx(); ++j) {
        int k=QSWP(j, Dest->mem2d->GetNx());
        Dest->mem2d->data->SetNext(
            Dest->mem2d->data->GetThis()
            + c_re(hkspc[k])*c_re(hkspc[k])
            + c_im(hkspc[k])*c_im(hkspc[k])
        );
    }
}
}
}

```

Usage

Call from *main menu/Math/Statistics/SPALED Sim k_z* . and input the step height, phase range and phase step size.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Configuration

Use the Plug-In configurator to set default values. Use the entry *Ask Next* to prevent or reenable further asking for parameters (1 will ask, 0 not).

Info for Plug-In: Math/Statistics/SPALEED Simkz

Plug-In name: spasimkz File: math/statistik/spasimkz.C
Author: Percy Zahl Email: zahl@users.sf.net

20.9. Crosscorrelation

Description

Computes the crosscorrelation of two images using a masked area of first source (the active scan).

– WORK IN PROGRESS –

Usage

Call *main menu/Math/Statistic/Cross Correlation* to execute.

Sources

The active and X channel are used as data source, a rectangular selection mask is used for feature selection.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



Info for Plug-In: Math/Statistic/Cross Correlation

Plug-In name: crosscorrelation
Author: Erik Muller

File: math/statistik/crosscorrelation.C
Email: emmuller@users.sourceforge.net

20.10. Stepcount

Description

This is a primitive plugin for the analysis of artificially generated scans. It counts the number of steps in x-direction, higher than 255 counts.

Usage

Use with active scan or a selected rectangle within the active scan.

Sources

You need one active scan.

Objects

If a rectangle is selected the calculated information applies to the content of the rectangle. Otherwise, the whole scan is analyzed.

Destination

The result is printed on the console, so you better have one open!

Configuration

None.

Known Bugs

None

Is there interest in a more general approach?

 Note

Info for Plug-In: Math/Statistics/stepcounter

Plug-In name: baseinfo
Author: Stefan Schröder

File: math/statistik/stepcount.C
Email: stefan_fkp@users.sf.net

20.11. Autocorrelation

Description

Computes the autocorrelation of an image.

$$Z' = |\text{IFT}(\text{FT}(Z))|$$

Usage

Call *main menu/Math/Statistic/Auto Correlation* to execute.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.



The quadrants of the resulting invers spectrum are aligned in a way, that the intensity of by it self correlated pixels (distance zero) is found at the image center and not at all four edges.

Info for Plug-In: Math/Statistic/Auto Correlation

Plug-In name: autocorrelation
Author: Erik Muller

File: math/statistik/autocorrelation.C
Email: emmuller@users.sourceforge.net

20.12. Calculate in plane direction of gradient

Description

Calculation of the direction of the local slope (gradient) using a user defined facet size at each pixel as reference area. A plane regression is performed at each pixel to find the best matching local facet of the given size. Its normal is used to find the gradients direction in plane.

Usage

Activate channel to use and call it from Menu *Math/Statistics/Slope Dir.*

Sources

The active channel is used.

Destination

Existing Math channel, else newly created Math channel.

Configuration

Can set a default facet size, if set to zero it will ask at each call.

Known Bugs

No bugs known.

Info for Plug-In: Math/Statistics/Slope Dir

Plug-In name:	SlopeDir	File:	math/statistik/slopedir.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

20.13. Histogram

Description

The Histogram plugin calculates the Z-value distribution (typically a height histogram) of the active channel using a default (estimated from data) or user provided number of bins.

Usage

Call it from Gxsm Math/Statistics menu. It will prompt for the number of bins and provides a estimated number as default. Also the current min/max Z-value limits and range is shown for informative purpose.

Sources

The active channel is used as data source.

Objects

A optional rectangle can not be used. Use crop before!

Destination

The computation result is placed into an new profile view.



Find out what happenes with more or less bins!

Info for Plug-In: Math/Statistics/Histogram

Plug-In name: histogram

File: math/statistik/histogram.C

Author: Percy Zahl

Email: zahl@users.sf.net

20.14. Angular Analysis

Description

Calculate all local gradients and presents those in a polar histogram of slope as radius and direction as polar angle.

Usage

Call *main menu/Math/Statistics/Angular Analysis*.

Info for Plug-In: Math/Statistics/Angular Analysis

Plug-In name: AngularAnalysis
Author: Percy Zahl

File: math/statistik/AngularAnalysis.C
Email: zahl@users.sf.net

20.15. Calculate gradient (slope)

Description

Calculation of the absolute local slope (gradient) using a user defined facet size at each pixel as reference area. A plane regression is performed at each pixel to find the best matching local facet of the given size. Its normal is used to find the gradient.

Usage

Activate channel to use and call it from Menu *Math/Statistics/Slope Abs.*

Sources

The active channel is used.

Destination

Existing Math channel, else newly created Math channel.

Configuration

Can set a default facet size, if set to zero it will ask at each call.

Known Bugs

No bugs known.

Info for Plug-In: Math/Statistics/Slope Abs

Plug-In name: SlopeAbs
Author: Percy Zahl

File: math/statistik/slopeabs.C
Email: zahl@users.sf.net

20.16. Add Trail

Description

Add's trail local height to Z at positions in scan.

Usage

Call *main menu/Math/Statistics/Add Trail*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Statistics/Add Trail

Plug-In name:	add_trail	File:	math/statistik/add_trail.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

20.17. Average X Profile

Description

Compute the average X profile of all scanlines.

Info for Plug-In: Math/Statistics/Average X Profile

Plug-In name:	average_profile	File:	math/statistik/average_profile.C
Author:	P. Zahl	Email:	zahl@gxsm.sf.net

21. Plug-Ins: math/arithmetic

The *math/arithmetic* plugins are those used for simple arithmetic, such as applying a log function to Z values, adding scans, etc.

21.1. Multiply scans

Description

Multiplys the Z-values of two scans.

Usage

Call *main menu/Math/Arithmetic/Mul X*.

Sources

The active channel is multiplied with the X-channel.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Both scans are required to have the same size in pixels.



Info for Plug-In: Math/Arithmetic/Mul X

Plug-In name: mul_scan File: math/arithmetic/mul_scan.C
Author: Percy Zahl Email: zahl@users.sf.net

21.2. Max of two sources

Description

This filter merges two (same sized and aligned) scans by using the max Z value of source one (active) and two (X) as resulting Z .

Usage

Select two same sized sources: One should be "Active" and the other in mode "X", assure there is only one mode "X" channel around – always the first "X" marked channel (lowest channel number) will be used! And run *main menu/Math/Arithmetic/Max*

Sources

The active channel and X-channel are used.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Arithmetic/Max

Plug-In name:	max	File:	math/arithmetic/max.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

21.3. Add two scans

Description

Adds the Z-values of two scan.

Usage

Call *main menu/Math/Arithmetic/Add X*.

Sources

The active channel is added to the X channel.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Both scans are required to have the same size in pixels.



Info for Plug-In: Math/Arithmetic/Add X

Plug-In name: add_scan File: math/arithmetic/add_scan.C
Author: Percy Zahl Email: zahl@users.sf.net

21.4. Multiply scans

Description

Linear transformation of the Z-values of scans offset and factor.

Usage

Call *main menu/Math/Arithmetic/Z Usr Rescale*.

Sources

The "Z" of the active channel, all layers is rescaled by a given factor.

Objects

Requests a input coefficients file, one line per layer:

 offset0 factor0

 offset1 factor1

 ...

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Info for Plug-In: Math/Arithmetic/Z usr rescale

Plug-In name: Z_usr_rescale

File: math/arithmetic/Z_usr_rescale.C

Author: Percy Zahl

Email: zahl@users.sf.net

21.5. Invert Z

Description

Invert the Z values of a scan.

$$Z_{\text{math}} = -Z_{\text{active}}$$

Usage

Call *main menu/Math/Arithmetic/Invert*.

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Arithmetic/Invert

Plug-In name: invert_z

File: math/arithmetic/invert_z.C

Author: Percy Zahl

Email: zahl@users.sf.net

21.6. Divide scans

Description

Divide the Z-values of two scans.

Usage

Call *main menu/Math/Arithmetic/Div X*.

Sources

The active channel is divided by the X channel.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Both scans are required to have the same size in pixels. There is an $\epsilon = 10^{-8}$ defined as minimal divisor, if the absolute value of the divisor is smaller than ϵ the original dividend data is kept unchanged.



Info for Plug-In: Math/Arithmetic/Div X

Plug-In name: div_scan File: math/arithmetic/div_scan.C
Author: Percy Zahl Email: zahl@users.sf.net

21.7. Subtract scans

Description

Subtracts the Z-values of two scan from each other.

Usage

Call *main menu/Math/Arithmetic/Sub X*.

Sources

The X channel is subtracted from the active channel.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Both scans are required to have the same size in pixels.



Info for Plug-In: Math/Arithmetic/Sub X

Plug-In name: sub_scan
Author: Percy Zahl

File: math/arithmetic/sub_scan.C
Email: zahl@users.sf.net

21.8. Z Limiter

Description

The Z Limiter limits the Z range to a given range defined by an selected area (rectangle object used before by *AutopDisplay*).

Usage

Call *main menu/Math/Arithmetic/Z Limiter*.

Sources

The active channel is used as data source.

Objects

The range withing an rectangle (i.e. the current *AutoDisp* settings) is used to obtain Z min/max for limiting.

Destination

The computation result is placed into an existing math channel, else into a new created math channel.

Info for Plug-In: Math/Arithmetic/Z Limiter

Plug-In name: Z_limiter
Author: Percy Zahl

File: math/arithmetic/Z_limiter.C
Email: zahl@users.sf.net

21.9. Multiply scans

Description

Multiplys the Z-values of scans bt factor. May limit action to rectange – will be used if any is found.

Usage

Call *main menu/Math/Arithmetic/Z Rescale*.

Sources

The "Z" of the active channel is rescaled by a given factor.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Info for Plug-In: Math/Arithmetic/Mul X

Plug-In name: Z_rescale	File: math/arithmetic/Z_rescale.C
Author: Percy Zahl	Email: zahl@users.sf.net

21.10. Absolute Value

Description

Takes the absolute value for the data.

Usage

Call *main menu/Math/Arithmetic/Absolute Value*.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Info for Plug-In: Math/Arithmetic/Absoluet Value

Plug-In name: abs_scan File: math/arithmetic/abs_scan.C
Author: Percy Zahl Email: zahl@users.sf.net

21.11. Logarithm transform of Z-values

Description

Apply a logarithm transform to the Z values. It subtracts the Z minimum, adds one computes the logarithm of that:

$$Z_{\text{math}} = \log(Z_{\text{active}} - \text{minval}(Z_{\text{active}}) + 1)$$

Usage

Call *main menu/Math/Arithmetic/Log.*

Sources

The active channel is used as data source.

Destination

The computation result is placed into an existing math channel, else into a new created math channel. The result is of type *float*.

Info for Plug-In: Math/Arithmetic/Log

Plug-In name: log_z File: math/arithmetic/log_z.C
Author: Percy Zahl Email: zahl@users.sf.net

22. Plug-Ins: common

Here are the GXSM-2 *common* type plugins. These plugins are of general purpose such as plugin management itself, printing, data viewing and other tasks not fitting into the following sections.

22.1. View NetCDF file data

Description

It shows all information stored in any NetCDF file. Huge data fields are truncated and only the first few values are shown.

Usage

Call it from Tools menu and select a NetCDF file when the file open dialog is presented.



Info for Plug-In: Tools/NetCDF-View

Plug-In name: editnc File: common/editnc.C
Author: Percy Zahl Email: zahl@users.sf.net

22.2. Probe Indicator

Description

The Probe Indicator (Note: head up display (HUD) as future option was designed to be overlayed on any scans canvas) is intended to give a real time feedback and provide dedicated monitoring of the tip/probe conditions. Visually indicating probe postion (Z), current and frequency. It also provides a contineous rolling graphical view of Z and current plus a (work in progress) spectral analysis. Plus a recording button allows to gapless record two channels (Current, Z,...) as selected for the Recorder Signals1 and

2 inputs. You will have to use currently the python spm control app/Oszi if you like to change the default channels used here.

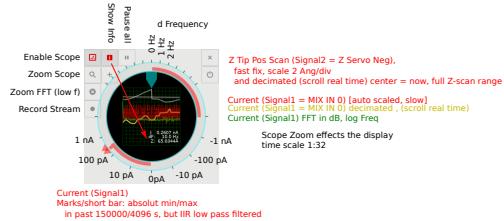


Figure 22.1.: The ProbeIndicator window explained.

Usage

Although this is a plugin it is opened automatically upon startup of GXSM automatically, you will need to open first time via Main menu Window/Probe-Hud.

Please have a look for a demo:

[Video Link: https://youtu.be/eB1FO76M7gI](https://youtu.be/eB1FO76M7gI)

[Video Link: https://youtu.be/vTrldyKxrZ4](https://youtu.be/vTrldyKxrZ4)

Note

not included: a signal selector for Signal 1 and 2 (recorder). Use the spm control python app, oscilloscope and setup teh channels there: Select "MIX IN 0" for CH1 and "Z Servo Neg" for CH2.

What the recorder does excatly:

It records 2x 4096 data points (scope channel) at full BW, they are plotted in a 4096 to 128 decimated graph.

And it records in a constant gap less continuous stream a 1:256 decimated data stream for both signals.

Checking the [o] record button this data is (for ever, until you delete it) appended into two files located in the working directory (where you start gxsm3). It writes away plain integer numbers and occasional (every 60s) a line starting with a double comment ## nnnn absolute system time stamp in us. And for every read block a index # n...m range. That may be used to precisely align both data sets in two individual files. Sample rate is exactly 150kHz/256.

Using the "Magnifying Button" (Zoom) you can choose to plot at FULL BW the first 128 points.

The Signal Button turns the scope on.

The Info Button toggles some text info overlay.

The [x] Button (left) selects FULL BW FFT display.

Scope scaling is fully automatic...
 FFT scale in double log dB(Freq).... but needs still some optimizations. FFT is run either on decimated 1:256 or FULL BW data.
 Also it shows a symbolic tip... indication the actual Z-position, up it tip "up".
 And a logarithmic current scale bar polar graph and indicators on the outside. Bottom is Zero, 1st tic is 10p, then 100p, ... left / right = pos/neg
 The markers and short polar indicator at the position indicated the current absolute min/max (low pass filtered) for the current as read by the scope!

MK3 only.



Note



Hint

Known Bugs

None

Info for Plug-In: Tools/Probe Indicator

Plug-In name: ProbeIndicator
 Author: Percy Zahl

File: common/ProbeIndicator.C
 Email: zahl@users.sf.net

22.3. Pan View

Description

This is a handy tool which shows you, where your current scan is in the range of the maximum scan. Especially it gives you an error message if you leave the scan area. It also shows the position of rotated scans.

In addition the current realtime XY position of the tip plus it indicates the Z-position of the tip is visualized by the marker on the right edge of the window.

Usage

Although this is a plugin it is opened automatically upon startup of GXSM. There is not interaction with the user.

1.) Indicator of the state machine on the DSP. In general the colors indicate green=ON/in progress, red=OFF/inactive. From left to right the boxed indicate the status of the feed-back, scan in progress, vector proce in progress, mover in progress (coarse approach)

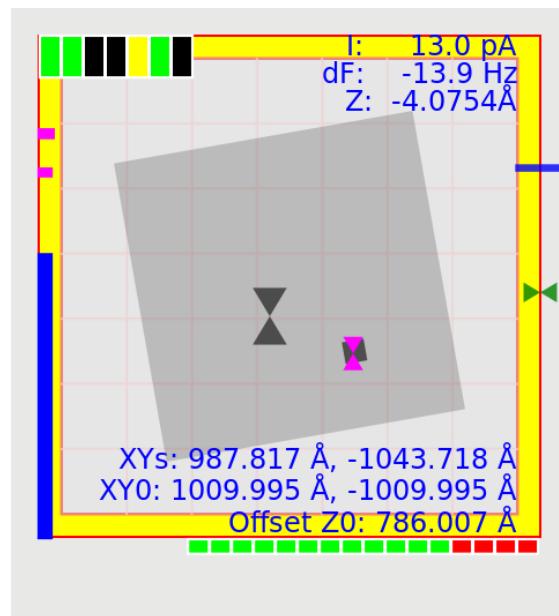


Figure 22.2.: The PanView window.

- 2.) Indicators of the 8 GPIO channels. They can be read on/off (red/black) or write on/off (green/white) giving you four possible states.
- 3.) Indicator of the Z position (Z-offset/z-scan)

Note

Hint

Known Bugs

None

Info for Plug-In: Tools/Pan View

Plug-In name: PanView

Author: Kristan Temme, Thorsten Wagner, Percy Zahl

File: common/PanView.C

Email: stm@users.sf.net

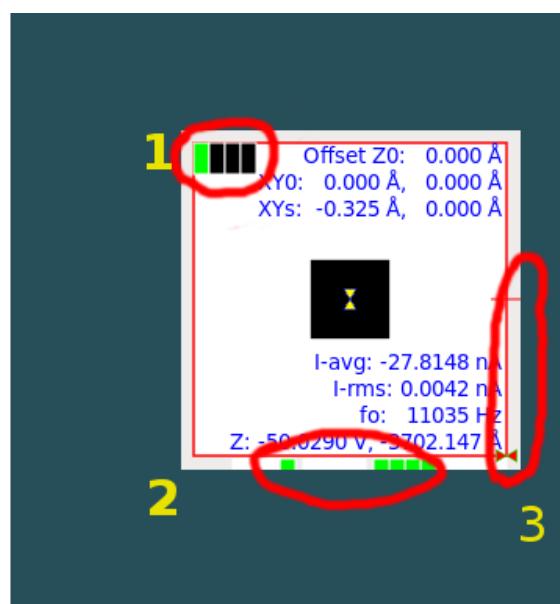


Figure 22.3.: The indicators of the PanView window.

22.4. Python remote control

Description

This plugin is an interface to an embedded Python Interpreter.

Usage

Choose Pyremote Console from the *main menu/Tools* menu to enter a python console. A default command script is loaded when the console is started for the first time, but is not executed automatically. In the appendix you will find a tutorial with examples and tips and tricks.

22.4.1. Reference

The following script shows you all commands that the gxsm-module supports:

```
# list core gxsm module functions:  
print dir(gxsm)  
  
# list buildin help on functions:  
for h in gxsm.help ():  
print (h)  
  
# list all by gxsm known entry (set/get) reference names:  
for h in gxsm.list_refnames ():  
print (' => ^'.format(h, gxsm.get(h)))  
  
# list all action hooks to activate function  
# via a script triggered button press  
# or call plugin hook enabled math plugins:  
for h in gxsm.list_actions ():  
print (h)
```

The result will look like this (with added notes):

```
** gsm python module:  
['__doc__', '__loader__', '__name__', '__package__', '__spec__',  
'add_marker_object', 'autodisplay', 'autosave', 'autoupdate',  
'chfname', 'chmodea', 'chmodem', 'chmoden', 'chmodeno', 'chmodex',
```

```
'createscan', 'createscanf', 'crop', 'da0', 'direct', 'echo', 'export', 'ge
'get_differentials', 'get_dimensions', 'get_geometry', 'get_object', 'get_s
'get_v_lookup', 'get_x_lookup', 'get_y_lookup',
'gets', 'help', 'import', 'list_actions', 'list_refnames', 'load', 'log', '
'moveto_scan_xy', 'progress', 'put_data_pkt', 'quick', 'rtquery', 'save', '
'scaninit', 'scanline', 'scanupdate', 'scanylookup', 'set', 'set_scan_lookup
'set_v_lookup', 'set_view_indices', 'set_x_lookup', 'set_y_lookup', 'signal'
'startscan', 'stopscan', 'unitbz', 'unitev', 'units', 'unitvolt', 'waitscan

***  

(1) Gxsm3 python remote console -- gxsm.help on build in commands  

The following list shows a brief explanation of the commands, together with  

the signature (that is the type of arguments).  

'()' equals no argument. E.g. startscan()  

'(N)' equals one Integer argument. E.g. chview1d(2)  

'(X)' equals one Float argument. No example.  

'(S)' equals a string. Often numbers are evaluated as strings first. Like  

'(S,N)' equals two parameters. E.g. gnuexport("myfilename.nc", 1)

-----
gxsm.help : List Gxsm methods: print gxsm.help ()
gxsm.set : Set Gxsm entry value, see list_refnames: gxsm.set ('refname', 'va
gxsm.get : Get Gxsm entry as value, see list_refnames. gxsm.get ('refname')
gxsm.gets : Get Gxsm entry as string. gxsm.gets ('refname')
gxsm.list_refnames : List all available Gxsm entry refnames (Better: pointe
to see tooltip with ref-name). print gxsm.list_refnames
gxsm.action : Trigger Gxsm action (menu action or button signal), see list_
gxsm.list_actions : List all available Gxsm actions (Better: pointer hover
to see tooltip with action-name): print gxsm.list_actions
gxsm.rtquery : Gxsm hardware Real-Time-Query: svec[3] = gxsm.rtquery('X|Y|Z
gxsm.y_current : RTQuery Current Scanline.
gxsm.moveto_scan_xy : Set tip position to Scan-XY: gxsm.moveto_scan_xy (x,y,
gxsm.createscan : Create Scan int: gxsm.createscan (ch,nx,ny,nv pixels, rx,
gxsm.createscanf : Create Scan float: gxsm.createscan (ch,nx,ny,nv pixels,
```

```
gxsm.set_scan_unit : Set Scan X,Y,Z,L Dim Unit: gxsm.set_scan_unit
gxsm.set_scan_lookup : Set Scan Lookup for Dim: gxsm.set_scan_lookup
gxsm.get_geometry : Get Scan Geometry: [rx,ry,x0,y0,alpha]=gxsm.get_
gxsm.get_differentials : Get Scan Scaling: [dx,dy,dz,dl]=gxsm.get_
gxsm.get_dimensions : Get Scan Dimensions: [nx,ny,nv,nt]=gxsm.get_
gxsm.get_data_pkt : Get Data Value at point: value=gxsm.get_data_pk_
gxsm.put_data_pkt : Put Data Value to point: gxsm.put_data_pkt (valu_
gxsm.get_slice : Get Data Slice from Scan Imagein ch, values are so_
gxsm.get_x_lookup : Get Scan Data index to world mapping: x=gxsm.ge_
gxsm.get_y_lookup : Get Scan Data index to world mapping: y=gxsm.ge_
gxsm.get_v_lookup : Get Scan Data index to world mapping: v=gxsm.ge_
gxsm.set_x_lookup : Set Scan Data index to world mapping: x=gxsm.ge_
gxsm.set_y_lookup : Set Scan Data index to world mapping: y=gxsm.ge_
gxsm.set_v_lookup : Set Scan Data index to world mapping: v=gxsm.ge_
gxsm.get_object : Get Object Coordinates: [type, x,y,...]=gxsm.get_o_
gxsm.add_marker_object : Put Marker Object at pixel coordinates or 
                           gxsm.add_marker_object (ch, label=str|'xy'
gxsm.startscan : Start Scan.
gxsm.stopscan : Stop Scan.
gxsm.waitscan : Wait Scan. ret=gxsm.waitscan(blocking=true). ret=--1
gxsm.scaninit : Scaninit.
gxsm.scanupdate : Scanupdate.
gxsm.scanylookup : Scanylookup.
gxsm.scanline : Scan line.
gxsm.autosave : Save: Auto Save Scans. gxsm.autosave (). Returns cu
gxsm.autoupdate : Save: Auto Update Scans. gxsm.autoupdate (). Retu
gxsm.save : Save: Auto Save Scans: gxsm.save ()
gxsm.saveas : Save File As: gxsm.saveas (ch, 'path/fname.nc')
gxsm.load : Load File: gxsm.load (ch, 'path/fname.nc')
gxsm.export : Export scan: gxsm.export (ch, 'path/fname.nc')
gxsm.import : Import scan: gxsm.import (ch, 'path/fname.nc')
gxsm.save_drawing : Save Drawing to file: gxsm.save_drawing (ch, t
gxsm.set_view_indices : Set Ch view time and layer indices: gxsm.se
gxsm.autodisplay : Autodisplay active channel: gxsm.autodisplay ()
gxsm.chfname : Get Ch Filename: filename = gxsm.chfname (ch)
gxsm.chmodea : Set Ch Mode to A: gxsm.chmodea (ch)
gxsm.chmodex : Set Ch Mode to X: gxsm.chmodex (ch)
gxsm.chmodem : Set Ch Mode to MATH: gxsm.chmodem (ch)
gxsm.chmoden : Set Ch Mode to Data Channel <X+N>: gxsm.chmoden (ch)
```

```

gxsm.chmodeno : Set View Mode to No: gxsm.chmodeno (ch)
gxsm.chview1d : Set View Mode to 1D: gxsm.chmode1d (ch)
gxsm.chview2d : Set View Mode to 2D: gxsm.chmode2d (ch)
gxsm.chview3d : Set View Mode to 3D: gxsm.chmode3d (ch)
gxsm.quick : Quick.
gxsm.direct : Direct.
gxsm.log : Log.
gxsm.crop : Crop (ch-src, ch-dst)
gxsm.unitbz : UnitBZ.
gxsm.unitvolt : UnitVolt.
gxsm.unitev : UniteV.
gxsm.units : Units.
gxsm.echo : Echo string to terminal. gxsm.echo('hello gxsm to terminal')
gxsm.logev : Write string to Gxsm system log file and log monitor: gxsm.logev
gxsm.progress : Show/update gxsm progress info. fraction<0 init, 0..1 progress
gxsm.add_layerinformation : Add Layerinformation to active scan. gxsm.add_layerinformation
gxsm.da0 : Da0. -- N/A for SRanger
gxsm.signal_emit : Action-String.
gxsm.sleep : Sleep N/10s: gxsm.sleep (N)
*
-----
(2) Gxsm3 python remote console -- help on reference names
    used for gxsm.set and get, gets commands.
    Hint: hover the pointer over any get/set enabled Gxsm entry to see it's name
    Example: gxsm.set ("dsp-fbs-bias", "1.0")
-----
script-control => 0.0
TimeSelect => 0.0
Time => 1.0
LayerSelect => 0.0
Layers => 1.0
Rotation => 0.0
ScanY => 0.0
ScanX => 0.0
OffsetY => 0.0
OffsetX => 0.0
PointsY => 400.0
PointsX => 400.0
StepsY => 2.0050125313283207

```

22. Plug-Ins: common

```
StepsX  => 2.0050125313283207
RangeY  => 800.0
RangeX  => 800.0
dsp-pac-ph-bw-set  => 8000.0
dsp-pac-ph-ci-gain  => -127.2
dsp-pac-ph-cp-gain  => -61.6
dsp-pac-ph-set  => -100.0
dsp-pac-res-gain  => 1.0
dsp-pac-res-q-factor  => 30000.0
dsp-pac-am-bw-set  => 8.0
dsp-pac-am-ci-gain  => -77.5
dsp-pac-am-cp-gain  => 8.1
dsp-pac-am-set  => 0.1
dsp-pac-excitation-sine-freq  => 32766.4
dsp-pac-excitation-sine-amp  => 0.5
dsp-pac-tau  => 1.5e-05
dsp-pac-res-amp-max  => 5.0
dsp-pac-res-amp-min  => 0.0
dsp-pac-res-amp-range  => 1.25
dsp-pac-res-amp-ref  => 0.0
dsp-pac-exci-amp-max  => 1.0
dsp-pac-exci-amp-min  => -0.05
dsp-pac-exci-amp-range  => 0.625
dsp-pac-exci-amp-ref  => 0.0
dsp-pac-res-phase-max  => 180.0
dsp-pac-res-phase-min  => -180.0
dsp-pac-res-phase-range  => 7.16
dsp-pac-res-phase-ref  => 0.0
dsp-pac-exci-freq-max  => 41000.0
dsp-pac-exci-freq-min  => 29000.0
dsp-pac-exci-freq-range  => 187.0
dsp-pac-exci-freq-ref  => 0.0
dsp-VP-Lim-Val-Dn  => 1.0
dsp-VP-Lim-Val-Up  => 1.0
dsp-X-Final-Delay  => 0.01
dsp-AX-GateTime  => 0.001
dsp-AX-Final-Delay  => 0.01
dsp-AX-Slope-Ramp  => 100.0
dsp-AX-V-Slope  => 100.0
```

```
dsp-AXrep  => 1.0
dsp-AX-Points  => 100.0
dsp-AX-V-End  => 1.0
dsp-AX-V-Start  => 0.0
dsp-LCK-AC-Repetitions  => 1.0
dsp-LCK-AC-Final-Delay  => 0.01
dsp-LCK-AC-Slope  => 12.0
dsp-LCK-AC-Points  => 720.0
dsp-ALCK-C-Phase-Span  => 360.0
dsp-LCK-AC-avg-Cycles  => 32.0
dsp-LCK-AC-Phase-B  => 90.0
dsp-LCK-AC-Phase-A  => 0.0
dsp-LCK-AC-Frequency  => 1171.88
dsp-LCK-AC-Z-Amp  => 0.0
dsp-LCK-AC-Bias-Amp  => 0.02
dsp-LCK-CORRSUM-SHR  => 0.0
dsp-LCK-CORRPRD-SHR  => 14.0
dsp-Noise-Amplitude  => 0.0
dsp-TK-Delay  => 1.0
dsp-TK-Speed  => 1000.0
dsp-TK-Mode  => -1.0
dsp-TK-Reps  => 100.0
dsp-TK-Nodes  => 12.0
dsp-TK-Points  => 10.0
dsp-TK-rad2  => 0.0
dsp-TK-rad  => 2.0
dsp-GVP-GPIO-Lock-57  => 57.0
dsp-GVP-Final-Delay  => 0.01
dsp-gvp-pcj44  => 0.0
dsp-gvp-nrep44  => 0.0
dsp-gvp-data44  => 0.0
dsp-gvp-n44  => 0.0
dsp-gvp-dt44  => 0.0
dsp-gvp-dsig44  => 0.0
dsp-gvp-dz44  => 0.0
dsp-gvp-dy44  => 0.0
dsp-gvp-dx44  => 0.0
dsp-gvp-du44  => 0.0
dsp-gvp-pcj43  => 0.0
```

```
dsp-gvp-nrep43  => 0.0
dsp-gvp-data43  => 0.0
dsp-gvp-n43    => 0.0
dsp-gvp-dt43   => 0.0
dsp-gvp-dsig43  => 0.0
dsp-gvp-dz43   => 0.0
dsp-gvp-dy43   => 0.0
dsp-gvp-dx43   => 0.0
dsp-gvp-du43   => 0.0
dsp-gvp-pcj42  => 0.0
dsp-gvp-nrep42  => 0.0
dsp-gvp-data42  => 0.0
dsp-gvp-n42    => 0.0
dsp-gvp-dt42   => 0.0
dsp-gvp-dsig42  => 0.0
dsp-gvp-dz42   => 0.0
dsp-gvp-dy42   => 0.0
dsp-gvp-dx42   => 0.0
dsp-gvp-du42   => 0.0
dsp-gvp-pcj41  => 0.0
dsp-gvp-nrep41  => 0.0
dsp-gvp-data41  => 0.0
dsp-gvp-n41    => 0.0
dsp-gvp-dt41   => 0.0
dsp-gvp-dsig41  => 0.0
dsp-gvp-dz41   => 0.0
dsp-gvp-dy41   => 0.0
dsp-gvp-dx41   => 0.0
dsp-gvp-du41   => 0.0
dsp-gvp-pcj40  => 0.0
dsp-gvp-nrep40  => 0.0
dsp-gvp-data40  => 0.0
dsp-gvp-n40    => 0.0
dsp-gvp-dt40   => 0.0
dsp-gvp-dsig40  => 0.0
dsp-gvp-dz40   => 0.0
dsp-gvp-dy40   => 0.0
dsp-gvp-dx40   => 0.0
dsp-gvp-du40   => 0.0
```

```
dsp-gvp-pcjr39  => 0.0
dsp-gvp-nrep39   => 0.0
dsp-gvp-data39   => 0.0
dsp-gvp-n39     => 0.0
dsp-gvp-dt39    => 0.0
dsp-gvp-dsig39   => 0.0
dsp-gvp-dz39    => 0.0
dsp-gvp-dy39    => 0.0
dsp-gvp-dx39    => 0.0
dsp-gvp-du39    => 0.0
dsp-gvp-pcjr38  => 0.0
dsp-gvp-nrep38   => 0.0
dsp-gvp-data38   => 0.0
dsp-gvp-n38     => 0.0
dsp-gvp-dt38    => 0.0
dsp-gvp-dsig38   => 0.0
dsp-gvp-dz38    => 0.0
dsp-gvp-dy38    => 0.0
dsp-gvp-dx38    => 0.0
dsp-gvp-du38    => 0.0
dsp-gvp-pcjr37  => 0.0
dsp-gvp-nrep37   => 0.0
dsp-gvp-data37   => 0.0
dsp-gvp-n37     => 0.0
dsp-gvp-dt37    => 0.0
dsp-gvp-dsig37   => 0.0
dsp-gvp-dz37    => 0.0
dsp-gvp-dy37    => 0.0
dsp-gvp-dx37    => 0.0
dsp-gvp-du37    => 0.0
dsp-gvp-pcjr36  => 0.0
dsp-gvp-nrep36   => 0.0
dsp-gvp-data36   => 0.0
dsp-gvp-n36     => 0.0
dsp-gvp-dt36    => 0.0
dsp-gvp-dsig36   => 0.0
dsp-gvp-dz36    => 0.0
dsp-gvp-dy36    => 0.0
dsp-gvp-dx36    => 0.0
```

```
dsp-gvp-du36  => 0.0
dsp-gvp-pcj35  => 0.0
dsp-gvp-nrep35  => 0.0
dsp-gvp-data35  => 0.0
dsp-gvp-n35  => 0.0
dsp-gvp-dt35  => 0.0
dsp-gvp-dsig35  => 0.0
dsp-gvp-dz35  => 0.0
dsp-gvp-dy35  => 0.0
dsp-gvp-dx35  => 0.0
dsp-gvp-du35  => 0.0
dsp-gvp-pcj34  => 0.0
dsp-gvp-nrep34  => 0.0
dsp-gvp-data34  => 0.0
dsp-gvp-n34  => 0.0
dsp-gvp-dt34  => 0.0
dsp-gvp-dsig34  => 0.0
dsp-gvp-dz34  => 0.0
dsp-gvp-dy34  => 0.0
dsp-gvp-dx34  => 0.0
dsp-gvp-du34  => 0.0
dsp-gvp-pcj33  => 0.0
dsp-gvp-nrep33  => 0.0
dsp-gvp-data33  => 0.0
dsp-gvp-n33  => 0.0
dsp-gvp-dt33  => 0.0
dsp-gvp-dsig33  => 0.0
dsp-gvp-dz33  => 0.0
dsp-gvp-dy33  => 0.0
dsp-gvp-dx33  => 0.0
dsp-gvp-du33  => 0.0
dsp-gvp-pcj32  => 0.0
dsp-gvp-nrep32  => 0.0
dsp-gvp-data32  => 0.0
dsp-gvp-n32  => 0.0
dsp-gvp-dt32  => 0.0
dsp-gvp-dsig32  => 0.0
dsp-gvp-dz32  => 0.0
dsp-gvp-dy32  => 0.0
```

```
dsp-gvp-dx32 => 0.0
dsp-gvp-du32 => 0.0
dsp-gvp-pcj31 => 0.0
dsp-gvp-nrep31 => 0.0
dsp-gvp-data31 => 0.0
dsp-gvp-n31 => 0.0
dsp-gvp-dt31 => 0.0
dsp-gvp-dsig31 => 0.0
dsp-gvp-dz31 => 0.0
dsp-gvp-dy31 => 0.0
dsp-gvp-dx31 => 0.0
dsp-gvp-du31 => 0.0
dsp-gvp-pcj30 => 0.0
dsp-gvp-nrep30 => 0.0
dsp-gvp-data30 => 0.0
dsp-gvp-n30 => 0.0
dsp-gvp-dt30 => 0.0
dsp-gvp-dsig30 => 0.0
dsp-gvp-dz30 => 0.0
dsp-gvp-dy30 => 0.0
dsp-gvp-dx30 => 0.0
dsp-gvp-du30 => 0.0
dsp-gvp-pcj29 => 0.0
dsp-gvp-nrep29 => 0.0
dsp-gvp-data29 => 0.0
dsp-gvp-n29 => 0.0
dsp-gvp-dt29 => 0.0
dsp-gvp-dsig29 => 0.0
dsp-gvp-dz29 => 0.0
dsp-gvp-dy29 => 0.0
dsp-gvp-dx29 => 0.0
dsp-gvp-du29 => 0.0
dsp-gvp-pcj28 => 0.0
dsp-gvp-nrep28 => 0.0
dsp-gvp-data28 => 0.0
dsp-gvp-n28 => 0.0
dsp-gvp-dt28 => 0.0
dsp-gvp-dsig28 => 0.0
dsp-gvp-dz28 => 0.0
```

```
dsp-gvp-dy28  => 0.0
dsp-gvp-dx28  => 0.0
dsp-gvp-du28  => 0.0
dsp-gvp-pcjr27  => 0.0
dsp-gvp-nrep27  => 0.0
dsp-gvp-data27  => 0.0
dsp-gvp-n27  => 0.0
dsp-gvp-dt27  => 0.0
dsp-gvp-dsig27  => 0.0
dsp-gvp-dz27  => 0.0
dsp-gvp-dy27  => 0.0
dsp-gvp-dx27  => 0.0
dsp-gvp-du27  => 0.0
dsp-gvp-pcjr26  => 0.0
dsp-gvp-nrep26  => 0.0
dsp-gvp-data26  => 0.0
dsp-gvp-n26  => 0.0
dsp-gvp-dt26  => 0.0
dsp-gvp-dsig26  => 0.0
dsp-gvp-dz26  => 0.0
dsp-gvp-dy26  => 0.0
dsp-gvp-dx26  => 0.0
dsp-gvp-du26  => 0.0
dsp-gvp-pcjr25  => 0.0
dsp-gvp-nrep25  => 0.0
dsp-gvp-data25  => 0.0
dsp-gvp-n25  => 0.0
dsp-gvp-dt25  => 0.0
dsp-gvp-dsig25  => 0.0
dsp-gvp-dz25  => 0.0
dsp-gvp-dy25  => 0.0
dsp-gvp-dx25  => 0.0
dsp-gvp-du25  => 0.0
dsp-gvp-pcjr24  => 0.0
dsp-gvp-nrep24  => 0.0
dsp-gvp-data24  => 0.0
dsp-gvp-n24  => 0.0
dsp-gvp-dt24  => 0.0
dsp-gvp-dsig24  => 0.0
```

```
dsp-gvp-dz24  => 0.0
dsp-gvp-dy24  => 0.0
dsp-gvp-dx24  => 0.0
dsp-gvp-du24  => 0.0
dsp-gvp-pcj23  => 0.0
dsp-gvp-nrep23  => 0.0
dsp-gvp-data23  => 0.0
dsp-gvp-n23  => 0.0
dsp-gvp-dt23  => 0.0
dsp-gvp-dsig23  => 0.0
dsp-gvp-dz23  => 0.0
dsp-gvp-dy23  => 0.0
dsp-gvp-dx23  => 0.0
dsp-gvp-du23  => 0.0
dsp-gvp-pcj22  => 0.0
dsp-gvp-nrep22  => 0.0
dsp-gvp-data22  => 0.0
dsp-gvp-n22  => 0.0
dsp-gvp-dt22  => 0.0
dsp-gvp-dsig22  => 0.0
dsp-gvp-dz22  => 0.0
dsp-gvp-dy22  => 0.0
dsp-gvp-dx22  => 0.0
dsp-gvp-du22  => 0.0
dsp-gvp-pcj21  => 0.0
dsp-gvp-nrep21  => 0.0
dsp-gvp-data21  => 0.0
dsp-gvp-n21  => 0.0
dsp-gvp-dt21  => 0.0
dsp-gvp-dsig21  => 0.0
dsp-gvp-dz21  => 0.0
dsp-gvp-dy21  => 0.0
dsp-gvp-dx21  => 0.0
dsp-gvp-du21  => 0.0
dsp-gvp-pcj20  => 0.0
dsp-gvp-nrep20  => 0.0
dsp-gvp-data20  => 0.0
dsp-gvp-n20  => 0.0
dsp-gvp-dt20  => 0.0
```

```
dsp-gvp-dsig20  => 0.0
dsp-gvp-dz20   => 0.0
dsp-gvp-dy20   => 0.0
dsp-gvp-dx20   => 0.0
dsp-gvp-du20   => 0.0
dsp-gvp-pcjrl9  => 0.0
dsp-gvp-nrep19  => 0.0
dsp-gvp-data19  => 0.0
dsp-gvp-n19    => 0.0
dsp-gvp-dt19    => 0.0
dsp-gvp-dsig19  => 0.0
dsp-gvp-dz19   => 0.0
dsp-gvp-dy19   => 0.0
dsp-gvp-dx19   => 0.0
dsp-gvp-du19   => 0.0
dsp-gvp-pcjrl8  => 0.0
dsp-gvp-nrep18  => 0.0
dsp-gvp-data18  => 0.0
dsp-gvp-n18    => 0.0
dsp-gvp-dt18    => 0.0
dsp-gvp-dsig18  => 0.0
dsp-gvp-dz18   => 0.0
dsp-gvp-dy18   => 0.0
dsp-gvp-dx18   => 0.0
dsp-gvp-du18   => 0.0
dsp-gvp-pcjrl7  => 0.0
dsp-gvp-nrep17  => 0.0
dsp-gvp-data17  => 0.0
dsp-gvp-n17    => 0.0
dsp-gvp-dt17    => 0.0
dsp-gvp-dsig17  => 0.0
dsp-gvp-dz17   => 0.0
dsp-gvp-dy17   => 0.0
dsp-gvp-dx17   => 0.0
dsp-gvp-du17   => 0.0
dsp-gvp-pcjrl6  => 0.0
dsp-gvp-nrep16  => 0.0
dsp-gvp-data16  => 0.0
dsp-gvp-n16    => 0.0
```

```
dsp-gvp-dt16  => 0.0
dsp-gvp-dsig16 => 0.0
dsp-gvp-dz16  => 0.0
dsp-gvp-dy16  => 0.0
dsp-gvp-dx16  => 0.0
dsp-gvp-du16  => 0.0
dsp-gvp-pcj15  => 0.0
dsp-gvp-nrep15 => 0.0
dsp-gvp-data15 => 0.0
dsp-gvp-n15   => 0.0
dsp-gvp-dt15  => 0.0
dsp-gvp-dsig15 => 0.0
dsp-gvp-dz15  => 0.0
dsp-gvp-dy15  => 0.0
dsp-gvp-dx15  => 0.0
dsp-gvp-du15  => 0.0
dsp-gvp-pcj14  => 0.0
dsp-gvp-nrep14 => 0.0
dsp-gvp-data14 => 0.0
dsp-gvp-n14   => 0.0
dsp-gvp-dt14  => 0.0
dsp-gvp-dsig14 => 0.0
dsp-gvp-dz14  => 0.0
dsp-gvp-dy14  => 0.0
dsp-gvp-dx14  => 0.0
dsp-gvp-du14  => 0.0
dsp-gvp-pcj13  => 0.0
dsp-gvp-nrep13 => 0.0
dsp-gvp-data13 => 0.0
dsp-gvp-n13   => 0.0
dsp-gvp-dt13  => 0.0
dsp-gvp-dsig13 => 0.0
dsp-gvp-dz13  => 0.0
dsp-gvp-dy13  => 0.0
dsp-gvp-dx13  => 0.0
dsp-gvp-du13  => 0.0
dsp-gvp-pcj12  => 0.0
dsp-gvp-nrep12 => 0.0
dsp-gvp-data12 => 0.0
```

```
dsp-gvp-n12  => 0.0
dsp-gvp-dt12  => 0.0
dsp-gvp-dsig12  => 0.0
dsp-gvp-dz12  => 0.0
dsp-gvp-dy12  => 0.0
dsp-gvp-dx12  => 0.0
dsp-gvp-du12  => 0.0
dsp-gvp-pcjrl1  => 0.0
dsp-gvp-nrep11  => 0.0
dsp-gvp-data11  => 0.0
dsp-gvp-n11  => 0.0
dsp-gvp-dt11  => 0.0
dsp-gvp-dsig11  => 0.0
dsp-gvp-dz11  => 0.0
dsp-gvp-dy11  => 0.0
dsp-gvp-dx11  => 0.0
dsp-gvp-du11  => 0.0
dsp-gvp-pcjrl0  => 0.0
dsp-gvp-nrep10  => 0.0
dsp-gvp-data10  => 0.0
dsp-gvp-n10  => 0.0
dsp-gvp-dt10  => 0.0
dsp-gvp-dsig10  => 0.0
dsp-gvp-dz10  => 0.0
dsp-gvp-dy10  => 0.0
dsp-gvp-dx10  => 0.0
dsp-gvp-du10  => 0.0
dsp-gvp-pcjrl09  => 0.0
dsp-gvp-nrep09  => 0.0
dsp-gvp-data09  => 0.0
dsp-gvp-n09  => 0.0
dsp-gvp-dt09  => 0.0
dsp-gvp-dsig09  => 0.0
dsp-gvp-dz09  => 0.0
dsp-gvp-dy09  => 0.0
dsp-gvp-dx09  => 0.0
dsp-gvp-du09  => 0.0
dsp-gvp-pcjrl08  => 0.0
dsp-gvp-nrep08  => 0.0
```

```
dsp-gvp-data08 => 0.0
dsp-gvp-n08 => 0.0
dsp-gvp-dt08 => 0.0
dsp-gvp-dsig08 => 0.0
dsp-gvp-dz08 => 0.0
dsp-gvp-dy08 => 0.0
dsp-gvp-dx08 => 0.0
dsp-gvp-du08 => 0.0
dsp-gvp-pcj07 => 0.0
dsp-gvp-nrep07 => 0.0
dsp-gvp-data07 => 0.0
dsp-gvp-n07 => 0.0
dsp-gvp-dt07 => 0.0
dsp-gvp-dsig07 => 0.0
dsp-gvp-dz07 => 0.0
dsp-gvp-dy07 => 0.0
dsp-gvp-dx07 => 0.0
dsp-gvp-du07 => 0.0
dsp-gvp-pcj06 => 0.0
dsp-gvp-nrep06 => 0.0
dsp-gvp-data06 => 0.0
dsp-gvp-n06 => 0.0
dsp-gvp-dt06 => 0.0
dsp-gvp-dsig06 => 0.0
dsp-gvp-dz06 => 0.0
dsp-gvp-dy06 => 0.0
dsp-gvp-dx06 => 0.0
dsp-gvp-du06 => 0.0
dsp-gvp-pcj05 => 0.0
dsp-gvp-nrep05 => 0.0
dsp-gvp-data05 => 0.0
dsp-gvp-n05 => 0.0
dsp-gvp-dt05 => 0.0
dsp-gvp-dsig05 => 0.0
dsp-gvp-dz05 => 0.0
dsp-gvp-dy05 => 0.0
dsp-gvp-dx05 => 0.0
dsp-gvp-du05 => 0.0
dsp-gvp-pcj04 => 0.0
```

```
dsp-gvp-nrep04  => 0.0
dsp-gvp-data04  => 0.0
dsp-gvp-n04    => 0.0
dsp-gvp-dt04   => 0.0
dsp-gvp-dsig04  => 0.0
dsp-gvp-dz04   => 0.0
dsp-gvp-dy04   => 0.0
dsp-gvp-dx04   => 0.0
dsp-gvp-du04   => 0.0
dsp-gvp-pcjr03  => 0.0
dsp-gvp-nrep03  => 1.0
dsp-gvp-data03  => 0.0
dsp-gvp-n03    => 0.0
dsp-gvp-dt03   => 0.0
dsp-gvp-dsig03  => 0.0
dsp-gvp-dz03   => 0.0
dsp-gvp-dy03   => 0.0
dsp-gvp-dx03   => 0.0
dsp-gvp-du03   => 0.0
dsp-gvp-pcjr02  => 0.0
dsp-gvp-nrep02  => 1.0
dsp-gvp-data02  => 2.0
dsp-gvp-n02    => 100.0
dsp-gvp-dt02   => 0.2
dsp-gvp-dsig02  => 0.0
dsp-gvp-dz02   => -100.0
dsp-gvp-dy02   => 0.0
dsp-gvp-dx02   => 0.0
dsp-gvp-du02   => -1.0
dsp-gvp-pcjr01  => 0.0
dsp-gvp-nrep01  => 1.0
dsp-gvp-data01  => 0.0
dsp-gvp-n01    => 2000.0
dsp-gvp-dt01   => 1.0
dsp-gvp-dsig01  => 0.0
dsp-gvp-dz01   => 0.0
dsp-gvp-dy01   => 0.0
dsp-gvp-dx01   => 0.0
dsp-gvp-du01   => 0.0
```

```
dsp-gvp-pcjr00 => 0.0
dsp-gvp-nrep00 => 1.0
dsp-gvp-data00 => 1.0
dsp-gvp-n00 => 100.0
dsp-gvp-dt00 => 0.2
dsp-gvp-dsig00 => 0.0
dsp-gvp-dz00 => 100.0
dsp-gvp-dy00 => 0.0
dsp-gvp-dx00 => 0.0
dsp-gvp-du00 => 1.0
dsp-TS-Repetitions => 1.0
dsp-TS-Points => 2048.0
dsp-TS-Duration => 1000.0
dsp-SP-Repetitions => 1.0
dsp-SP-Delay => 0.01
dsp-SP-Flag-V-on-X => 1.0
dsp-SP-Ramp-Time => 10.0
dsp-SP-Volts => 2.0
dsp-SP-Duration => 10.0
dsp-LP-Repetitions => 1.0
dsp-LP-Slope => 10000.0
dsp-LP-Laser-Delay => 10.0
dsp-LP-Tip-Retract => 0.0
dsp-LP-Trigger-Time => 10.0
dsp-LP-Trigger-Volts => 2.0
dsp-LP-FB-Time => 10.0
dsp-PL-Repetitions => 1.0
dsp-PL-Final-Delay => 0.01
dsp-PL-Initial-Delay => 0.01
dsp-PL-Step-dZ => 0.0
dsp-PL-Step => 0.0
dsp-PL-SetStart => 0.1
dsp-PL-dZ-ext => 0.0
dsp-PL-dZ => 0.0
dsp-PL-Volts => 2.0
dsp-PL-Res => 0.0
dsp-PL-Slope => 10000.0
dsp-PL-Duration => 10.0
dsp-Z-Final-Delay => 0.01
```

```
dsp-Z-Slope-Ramp  => 100.0
dsp-Z-Reps  => 1.0
dsp-Z-Slope  => 100.0
dsp-Z-Points  => 100.0
dsp-Z-end  => 100.0
dsp-Z-start  => 0.0
dsp-IV-Recover-Delay  => 0.3
dsp-IV-Final-Delay  => 0.01
dsp-IV-Line-Final-Delay  => 1.0
dsp-IV-Line-slope  => 100.0
dsp-IV-Line-dM  => 0.0
dsp-IV-Line-dY  => 0.0
dsp-IV-Line-dX  => 50.0
dsp-IVrep  => 1.0
dsp-IV-Slope-Ramp  => 50.0
dsp-IV-Slope  => 10.0
dsp-IV-dz-rep  => 0.0
dsp-IV-dz  => 0.0
dsp-6-IV-Points05  => 10.0
dsp-6-IV-End05  => 1.0
dsp-6-IV-Start05  => -1.0
dsp-5-IV-Points04  => 10.0
dsp-5-IV-End04  => 1.0
dsp-5-IV-Start04  => -1.0
dsp-4-IV-Points03  => 10.0
dsp-4-IV-End03  => 1.0
dsp-4-IV-Start03  => -1.0
dsp-3-IV-Points02  => 10.0
dsp-3-IV-End02  => 1.0
dsp-3-IV-Start02  => -1.0
dsp-2-IV-Points01  => 10.0
dsp-2-IV-End01  => 1.0
dsp-2-IV-Start01  => -1.0
dsp-IV-Points00  => 100.0
dsp-IV-End00  => 1.0
dsp-IV-Start00  => -1.0
dsp-IV-Sections  => 1.0
dsp-fbs-scan-ldc-dz  => 0.0
dsp-fbs-scan-ldc-dy  => 0.0
```

```
dsp-fbs-scan-ldc-dx  => 0.0
dsp-fbs-vp-section   => 2.0
dsp-adv-scan-slope-y => 0.0
dsp-adv-scan-slope-x => 0.0
dsp-adv-scan-xs2nd-z-offset => 0.0
dsp-adv-scan-dyn-zoom  => 1.0
dsp-adv-scan-fwd-slow-down-2nd => 1.0
dsp-adv-scan-pre-pts   => 0.0
dsp-adv-scan-fwd-slow-down => 1.0
dsp-adv-scan-fast-return => 1.0
dsp-adv-scan-rasterb  => 0.0
dsp-adv-scan-raster   => 0.0
dsp-adv-iir3-fo       => 18000.0
dsp-adv-iir2-fo       => 18000.0
dsp-adv-iir1-fo       => 18000.0
dsp-adv-current-offset => 10.0
dsp-adv-current-crossover => 100.0
dsp-adv-iir-fo-max    => 8000.0
dsp-adv-iir0-fo-min   => 200.0
dsp-adv-dsp-freq-ref  => 75000.0
dsp-fbs-scan-speed-scan => 4094.1
dsp-fbs-scan-speed-move => 1567.2
dsp-fbs-ci  => 0.0
dsp-fbs-cp  => 15.4
dsp-fbs-mx3-level     => 0.0
dsp-fbs-mx3-gain      => 0.5
dsp-fbs-mx3-set        => 0.0
dsp-fbs-mx2-level     => 0.0
dsp-fbs-mx2-gain      => 0.5
dsp-fbs-mx2-set        => 1.0
dsp-fbs-mx1-freq-level => 0.0
dsp-fbs-mx1-freq-gain  => -0.5
dsp-fbs-mx1-freq-set   => 0.0
dsp-fbs-mx0-current-level => 0.0
dsp-fbs-mx0-current-gain => 0.5
dsp-fbs-mx0-current-set  => 0.1
dsp-adv-dsp-zpos-ref  => 0.0
dsp-fbs-motor  => 0.0
dsp-fbs-bias3  => 0.5
```

```
dsp-fbs-bias2  => 0.5
dsp-fbs-bias1  => 0.5
dsp-fbs-bias  => 0.08567931456548372
dspmover-config-GPIO-delay  => 250.0
dspmover-config-GPIO-tmp2  => 0.0
dspmover-config-GPIO-tmp1  => 0.0
dspmover-config-GPIO-scan  => 0.0
dspmover-config-GPIO-direction  => 15.0
dspmover-config-GPIO-reset  => 0.0
dspmover-config-GPIO-off  => 0.0
dspmover-config-GPIO-on  => 0.0
dspmover-config-wave-out5-ch-z  => 0.0
dspmover-config-wave-out5-ch-y  => 0.0
dspmover-config-wave-out5-ch-x  => 0.0
dspmover-config-wave-out4-ch-z  => 0.0
dspmover-config-wave-out4-ch-y  => 0.0
dspmover-config-wave-out4-ch-x  => 0.0
dspmover-config-wave-out3-ch-z  => 0.0
dspmover-config-wave-out3-ch-y  => 0.0
dspmover-config-wave-out3-ch-x  => 0.0
dspmover-config-wave-out2-ch-z  => 0.0
dspmover-config-wave-out2-ch-y  => 0.0
dspmover-config-wave-out2-ch-x  => 0.0
dspmover-config-wave-out1-ch-z  => 0.0
dspmover-config-wave-out1-ch-y  => 0.0
dspmover-config-wave-out1-ch-x  => 0.0
dspmover-config-wave-out0-ch-z  => 5.0
dspmover-config-wave-out0-ch-y  => 4.0
dspmover-config-wave-out0-ch-x  => 3.0
dspmover-config-besocke-t2  => 0.09
dspmover-config-besocke-t1  => 0.1
dspmover-config-besocke-z-jump-ratio  => 0.1
dspmover-config-IW-Phase  => 55.0
dspmover-config-Wave-Offset  => 0.0
dspmover-config-Wave-Space  => 0.0
dspmover-z0-goto  => 0.0
dspmover-z0-speed  => 500.0
dspmover-z0-range  => 500.0
dspmover-auto-axis-Z  => 0.0
```

```
dspmover-auto-axis-Y  => 0.0
dspmover-auto-axis-X  => 0.0
dspmover-config-Auto-App-Retract-CI  => 150.0
dspmover-config-Auto-App-Max-Settling-Time  => 1000.0
dspmover-config-Auto-App-Delay  => 50.0
dspmover-auto-gpio  => 0.0
dspmover-auto-duration  => 4.0
dspmover-auto-amplitude  => 1.0
dspmover-auto-max-steps  => 5.0
dspmover-lens-axis-Z  => 0.0
dspmover-lens-axis-Y  => 0.0
dspmover-lens-axis-X  => 0.0
dspmover-lens-gpio  => 0.0
dspmover-lens-duration  => 5.0
dspmover-lens-amplitude  => 1.0
dspmover-lens-max-steps  => 100.0
dspmover-psd-axis-Z  => 0.0
dspmover-psd-axis-Y  => 0.0
dspmover-psd-axis-X  => 0.0
dspmover-psd-gpio  => 0.0
dspmover-psd-duration  => 5.0
dspmover-psd-amplitude  => 1.0
dspmover-psd-max-steps  => 100.0
dspmover-rot-axis-Z  => 0.0
dspmover-rot-axis-Y  => 0.0
dspmover-rot-axis-X  => 0.0
dspmover-rot-gpio  => 2.0
dspmover-rot-duration  => 3.0
dspmover-rot-amplitude  => 1.0
dspmover-rot-max-steps  => 2.0
dspmover-xy-axis-Z  => 0.0
dspmover-xy-axis-Y  => 0.0
dspmover-xy-axis-X  => 0.0
dspmover-xy-gpio  => 1.0
dspmover-xy-duration  => 3.0
dspmover-xy-amplitude  => 1.0
dspmover-xy-max-steps  => 1000.0
SPMC_SLS_Yn  => 0.0
SPMC_SLS_Ys  => 0.0
```

```
SPMC_SLS_Xn  => 0.0
SPMC_SLS_Xs  => 0.0
rp-pacpll-RP-VERBOSE-LEVEL  => 0.0
rp-pacpll-SCOPE-HEIGHT  => 256.0
rp-pacpll-SCOPE-WIDTH  => 1024.0
rp-pacpll-DFREQ-CONTROL-MONITOR  => 0.0
rp-pacpll-CONTROL-DFREQ-FB-UPPER  => 500.0
rp-pacpll-CONTROL-DFREQ-FB-LOWER  => -500.0
rp-pacpll-DFREQ-FB-CI  => -143.0
rp-pacpll-DFREQ-FB-CP  => -76.0
rp-pacpll-DFREQ-FB-SETPOINT  => 0.0
rp-pacpll-DFREQ-MONITOR  => 0.0
rp-pacpll-PHASE-HOLD-AM-NOISE-LIMIT  => 0.0
rp-pacpll-DDS-FREQ-MONITOR  => 32768.0
rp-pacpll-FREQ-FB-UPPER  => 333000.0
rp-pacpll-FREQ-FB-LOWER  => 32000.0
rp-pacpll-PHASE-FB-CI  => -150.0
rp-pacpll-PHASE-FB-CP  => -95.0
rp-pacpll-PHASE-FB-SETPOINT  => 60.0
rp-pacpll-PHASE-MONITOR  => 0.0
rp-pacpll-EXEC-AMPLITUDE-MONITOR  => 0.0
rp-pacpll-EXEC-FB-UPPER  => 500.0
rp-pacpll-EXEC-FB-LOWER  => -300.0
rp-pacpll-AMPLITUDE-FB-CI  => -90.0
rp-pacpll-AMPLITUDE-FB-CP  => -60.0
rp-pacpll-AMPLITUDE-FB-SETPOINT  => 8.0
rp-pacpll-VOLUME-MONITOR  => 0.0
rp-pacpll-TUNE-SPAN  => 50.0
rp-pacpll-TUNE-DFREQ  => 0.1
rp-pacpll-VOLUME-MANUAL  => 0.0
rp-pacpll-AUX-SCALE  => 0.011642
rp-pacpll-FREQUENCY-CENTER  => 149470.0
rp-pacpll-FREQUENCY-MANUAL  => 149470.0
rp-pacpll-QC-GAIN  => 0.0
rp-pacpll-QC-PHASE  => 0.0
rp-pacpll-PACATAU  => 40.0
rp-pacpll-PACTAU  => 40.0
rp-pacpll-PAC-DCTAU  => 10.0
rp-pacpll-DC-OFFSET  => 0.0
```

*

(3) Gxsm3 python remote console -- help on action names used for gxsm.actions
Hint: hover the pointer over any Gxsm Action enabled Button to see it's action
Example: gxsm.action ("DSP_CMD_GOTO_Z0")

MATH_FILTER2D_Smooth
MATH_FILTER1D_Diff
MATH_FILTER2D_Edge
MATH_FILTER2D_Normal_Z
MATH_FILTER2D_Despike
DSP_CMD_GOTO_Z0
DSP_CMD_HOME_Z0
DSP_CMD_AUTOCENTER_Z0
DSP_CMD_DOWN_Z0
DSP_CMD_UP_Z0
DSP_CMD_STOP_Z0
DSP_CMD_STOPALL
DSP_CMD_AUTOAPP
DSP_CMD_MOV-ZM_Auto
DSP_CMD_MOV-ZP_Auto
DSP_CMD_MOV-YM_Lens
DSP_CMD_MOV-XP_Lens
DSP_CMD_MOV-XM_Lens
DSP_CMD_MOV-YP_Lens
DSP_CMD_MOV-YM_PSD
DSP_CMD_MOV-XP_PSD
DSP_CMD_MOV-XM_PSD
DSP_CMD_MOV-YP_PSD
DSP_CMD_MOV-YM_Rot
DSP_CMD_MOV-XP_Rot
DSP_CMD_MOV-XM_Rot
DSP_CMD_MOV-YP_Rot
DSP_CMD_MOV-YM_XY
DSP_CMD_MOV-XP_XY
DSP_CMD_MOV-XM_XY
DSP_CMD_MOV-YP_XY
DSP_VP_ABORT_EXECUTE
DSP_VP_AX_EXECUTE

```
DSP_VP_AC_EXECUTE  
DSP_VP_TK_EXECUTE  
DSP_VP_GVP_EXECUTE  
DSP_VP_RCL_V0  
DSP_VP_STO_V0  
DSP_VP_RCL_VPJ  
DSP_VP_STO_VPJ  
DSP_VP_RCL_VPI  
DSP_VP_STO_VPI  
DSP_VP_RCL_VPH  
DSP_VP_STO_VPH  
DSP_VP_RCL_VPG  
DSP_VP_STO_VPG  
DSP_VP_RCL_VPF  
DSP_VP_STO_VPF  
DSP_VP_RCL_VPE  
DSP_VP_STO_VPE  
DSP_VP_RCL_VPD  
DSP_VP_STO_VPD  
DSP_VP_RCL_VPC  
DSP_VP_STO_VPC  
DSP_VP_RCL_VPB  
DSP_VP_STO_VPB  
DSP_VP_RCL_VPA  
DSP_VP_STO_VPA  
DSP_VP_TS_EXECUTE  
DSP_VP_SP_EXECUTE  
DSP_VP_LP_EXECUTE  
DSP_VP_PL_EXECUTE  
DSP_VP_FZ_EXECUTE  
DSP_VP_IV_EXECUTE
```

The following list shows a brief explanation of the commands, together with the signature (that is the type of arguments).

- '()' equals no argument. E.g. `startscan()`
- '(N)' equals one Integer arument. E.g. `chviewld(2)`
- '(X)' equals one Float argument. No example.
- '(S)' equals a string. Often numbers are evaluated as strings first. Like in `set ("RangeX", "23")`
- '(S,N)' equals two parameters. E.g. `gnuexport ("myfilename.nc", 1)`

Scan operation	
startscan()	Start a scan.
stopscan()	Stop scanning.
waitscan	is commented out in app_remote.C
initscan()	only initialize.
scanupdate()	Set hardware parameters on DSP.
setylookup(N, X)	?
scanline	Not implemented.
File operation	
save()	Save all.
saveas(S, N)	Save channel N with filename S.
load(S, N)	Load file S to channel N.
import(S, N)	Import file S to channel N.
export(S, N)	Export channel N to file S.
Channel operation	
chmodea(N)	Set channel(N) as active.
chmodex(N)	Set channel(N) to X.
chmodem(N)	Set channel(N) to Math.
chmoden(N), N	Set channel(N) to mode(N).
chmodeno(N)	Set channel(N) to mode 'No'.
chview1d(N)	View channel(N) in 1d-mode.
chview2d(N)	View channel(N) in 2d-mode.
chview3d(N)	View channel(N) in 3d-mode.
Views	
autodisplay()	Autodisplay.
quick()	Set active display to quickview.
direct()	Set active display to directview.
log()	Set active display to logview.
Units	
unitbz()	Set units to BZ.
unitvolt()	Set units to Volt.
unitev()	Set units to eV.
units()	Set units to S.
Others	
createscan(N, N, N, N, A)	Create scan from array.
list()	Get list of known parameters for get/set.
set(S, S)	Set parameter to value.
get(S)	Get parameter, returns floating point value in current user unit .
gets(S)	Get parameter, returns string with user unit.
action(S)	Initiate Action (S): trigger menu actions and button-press events (re)
rtquery(S)	Ask current HwI to run RTQuery with parameter S, return vector o
y_current()	Ask current HwI to run RTQuery what shall return the actual scanl
echo(S)	Print S to console.
logev(S)	Print S to logfile.
sleep(N)	Sleep N/10 seconds.
add_layerinformation(S, N)	Add Layerinformation string S to active scan, layer N.
da0(X)	Set Analog Output channel 0 to X Volt. (not implemented).

The set-command

The set command can modify the following parameters:

ACAmp	ACFrq
ACPhase	
CPShigh	CPSlow
Counter	Energy
Gatetime	Layers
LengthX	LengthY
Offset00X	Offset00Y
OffsetX	OffsetY
PointsX	PointsY
RangeX	RangeY
Rotation	
StepsX	StepsY
SubSmp	VOffsetZ
VRangeZ	ValueEnd
ValueStart	nAvg

These parameters are case-sensitive. To help the python remote programmer to figure out the correct set-names of all remote enabled entry fields a nifty option was added to the Help menu to show tooltips with the correct "remote set name" if the mouse is hovering over the entry.

The get-command

The get () command can retrieve the value of the remote control parameters. While get () retrieves the internal value as a floating points number, gets () reads the actual string from the text entry including units. The list of remote control accessible parameters can be retrieved with list () .

```
print "OffsetX = ", gxsm.get("OffsetX")
gxsm.set("OffsetX", "12.0")
print "Now OffsetX = ", gxsm.get("OffsetX")

for i in gxsm.list():
    print i, " ", gxsm.get(i), " as string: ", gxsm.gets(i)
```

On my machine (without hardware attached) this prints:

```
OffsetX = 0.0
Now OffsetX = 12.0
```

```
Counter    0.0  as string: 00000
VOffsetZ   0.0  as string: 0 nm
VRangeZ    500.0 as string: 500 nm
Rotation   1.92285320764e-304 as string: 1.92285e-304 Å°
TimeSelect  0.0  as string: 0
Time      1.0  as string: 1
LayerSelect 0.0  as string: 0
Layers     1.0  as string: 1
OffsetY    0.0  as string: 0.0 nm
OffsetX    12.0 as string: 12.0 nm
PointsY   1000.0 as string: 1000
PointsX   1000.0 as string: 1000
StepsY    0.519863986969 as string: 0.52 nm
StepsX    0.519863986969 as string: 0.52 nm
RangeY    64.9830993652 as string: 65.0 nm
RangeX    64.9830993652 as string: 65.0 nm
```

All entry fields with assigned id can now be queried.

Creating new scans

Pyremote can create new images from scratch using the `createscan` command. Its arguments are pixels in x-direction, pixels in y-direction, range in x-direction (in Angstrom), range in y-direction (in Angstrom) and finally a flat, numeric array that must contain as many numbers as needed to fill the matrix.

This example creates a new scan employing sine to show some pretty landscape.

```
import array    # for array
import numpy # for fromfunction
import math      # for sin

def dist(x,y):
    return ((numpy.sin((x-50)/15.0) + numpy.sin((y-50)/15.0))*100)

m = numpy.fromfunction(dist, (100,100))
n = numpy.ravel(m) # make 1-d
p = n.tolist()      # convert to list

examplearray = array.array('l', p) #
gxsm.createscan(100, 100, 10000, 10000, examplearray)
```

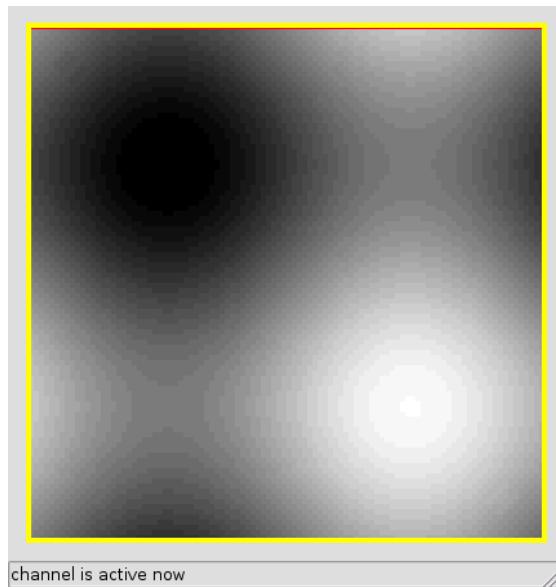


Figure 22.4.: An autogenerated image.

This command can be easily extended to create an importer for arbitrary file formats via python. The scripts directory contains an elaborate example how to use this facility to import the file format employed by Nanonis.

Menupath and Plugins

Any plugin, that has a menuentry can be executed via the menupath-action command. Several of them, however, open a dialog and ask for a specific parameter, e.g. the diff-PI in *main menu/Math/FilterID*. This can become annoying, when you want to batch process a greater number of files. To execute a PI non-interactively it is possible to call a plugin from scripts with default parameters and no user interaction.

The diff-PI can be called like this:

```
print "Welcome to Python."
gxsm.logev('my logentry')
gxsm.startscan()
gxsm.action('diff_PI')
```

The diff- and smooth-function are, at the time of this writing, the only Math-PI, that have such an 'action'-callback. Others will follow. See *diff.C* to find out, how to extend your favourite PI with action-capabilities.

The action-command can execute the following PI:

```
diff_PI      kernel-size set to 5+1  
smooth_PI   kernel-size set to 5+1  
print_PI     defaults are read from gconf
```

GXSM3 Menu Action Table Information – remote menu action/math/... call via action key, see table below for list, example:

```
gxsm.signal_emit("math-filter1d-section-Koehler")
```

22. Plug-Ins: common

Section ID	Menu Entry	Action Key
math-filter2d-section	Stat Diff	math-filter2d-section-Stat-Diff
math-convert-section	to float	math-convert-section-to-float
math-arithmetic-section	Z Rescale	math-arithmetic-section-Z-Rescale
math-statistics-section	Add Trail	math-statistics-section-Add-Trail
math-transformations-section	OctoCorr	math-transformations-section-OctoCorr
math-arithmetic-section	Mul X	math-arithmetic-section-Mul-X
math-filter2d-section	Edge	math-filter2d-section-Edge
math-arithmetic-section	Max	math-arithmetic-section-Max
math-statistics-section	Stepcounter	math-statistics-section-Stepcounter
math-convert-section	to double	math-convert-section-to-double
math-filter1d-section	Koehler	math-filter1d-section-Koehler
math-statistics-section	Histogram	math-statistics-section-Histogram
math-background-section	Line: 2nd order	math-background-section-Line: 2nd order
math-filter2d-section	Despike	math-filter2d-section-Despike
math-transformations-section	Auto Align	math-transformations-section-Auto Align
math-background-section	Plane Regression	math-background-section-Plane Regression
math-statistics-section	Cross Correlation	math-statistics-section-Cross Correlation
math-arithmetic-section	Z Usr Rescale	math-arithmetic-section-Z-Usr Rescale
math-filter1d-section	Diff	math-filter1d-section-Diff
math-filter2d-section	T derive	math-filter2d-section-T-derive
math-background-section	Pass CC	math-background-section-Pass CC
math-statistics-section	Auto Correlation	math-statistics-section-Auto Correlation
math-transformations-section	Rotate 90deg	math-transformations-section-Rotate 90deg
math-arithmetic-section	Invert	math-arithmetic-section-Invert
math-background-section	Plane max prop	math-background-section-Plane max prop
math-convert-section	U to float	math-convert-section-U to float
math-transformations-section	Movie Concat	math-transformations-section-Movie Concat
math-transformations-section	Shear Y	math-transformations-section-Shear Y
math-statistics-section	NN-distribution	math-statistics-section-NN-distribution
math-background-section	Waterlevel	math-background-section-Waterlevel
math-transformations-section	Quench Scan	math-transformations-section-Quench Scan
math-arithmetic-section	Div X	math-arithmetic-section-Div X
math-statistics-section	Vacancy Line Analysis	math-statistics-section-Vacancy Line Analysis
math-transformations-section	Shear X	math-transformations-section-Shear X
math-convert-section	to complex	math-convert-section-to-complex
math-convert-section	make test	math-convert-section-make test
math-misc-section	Spectrocut	math-misc-section-Spectrocut
math-arithmetic-section	Log	math-arithmetic-section-Log
math-statistics-section	Average X Profile	math-statistics-section-Average X Profile
math-filter2d-section	Lineinterp	math-filter2d-section-Lineinterp
math-background-section	Z drift correct	math-background-section-Z drift correct
250	Line Regression	math-background-section-Line Regression
math-background-section	SPALED Simkz	math-statistics-section-SPALED Simkz
math-statistics-section	to byte	math-convert-section-to-byte
math-convert-section	Slope Abs	math-statistics-section-Slope Abs
math-statistics-section	Small Convol	math-filter2d-section-Small Convolution
math-filter2d-section	to long	math-convert-section-to-long
math-convert-section	Multi Dim Transpose	math-transformations-section-Multi Dim Transpose
math-transformations-section	Sub X	math-arithmetic-section-Sub X
math-arithmetic-section	Stop CC	math-background-section-Stop CC
math-background-section	FT 2D	math-filter2d section-FT 2D
math-filter2d section		

DSP-Control

The DSP-Control is the heart of SPM activity. The following parameters can be set with set. (DSP2 commands are available in Gxsm 2 only)

Manual Hacker notes: list of DSP/DSP2 is deprecated. All entry fields with hover-over entry id is now remote capable.  Note

DSP_CI	DSP2_CI
DSP_CP	DSP2_CP
DSP_CS	DSP2_CS
DSP_I	DSP2_I
DSP_MoveLoops	DSP2_MoveLoops
DSP_MoveSpd	DSP2_MoveSpd
DSP_NAvg	DSP2_NAvg
DSP_Pre	DSP2_Pre
DSP_ScanLoops	DSP2_ScanLoops
DSP_ScanSpd	DSP2_ScanSpd
DSP_SetPoint	DSP2_SetPoint
DSP_U	DSP2_U

Manual Hacker notes: VP executes via hover-over ExecuteID and action command.



Peakfinder

Another plugin allows remote control. The plugin-functions are commonly executed by a call of the action-command. It is Peakfinder:

DSP Peak Find Plugin Commandset for the SPA-LEED peak finder:

Commands Plugin DSP Peak Find:

Cmd	Arg.	Values	Description
action	DSPPeakFind_XY0_1		Get fitted XY Position
action	DSPPeakFind_OffsetFromMain_1		Get Offset from Main
action	DSPPeakFind_OffsetToMain_1		Put Offset to Main
action	DSPPeakFind_EfromMain_1		Get Energy from Main
action	DSPPeakFind_RunPF_1		Run Peak Finder
action	DSPPeakFind_XXX_N		run action XXX (see above) on PF Folder N

The call is equivalent to the example above.

Peakfinder

Configuration

The plugin can be configured in the preferences. The default script that will be loaded when the console is entered for the first time is defined in the path-tab in item PyremoteFile. The name must be a qualified python module name. A module name is not a filename! Thus `remote.py` is not a valid entry, but `remote` (the default) is. The module is found by searching the directories listed in the environment variable `PYTHONPATH`. If no file is defined or a file matching the name cannot be found, a warning will be issued.

The module with GXSM internal commands is called `gxsm`.

To find the Python-script `remote.py`, the environment-variable `PYTHONPATH` is evaluated. If it is not explicitly declared, GXSM will set `PYTHONPATH` to your current working directory. This is equivalent to the following call:

```
$export PYTHONPATH=.'
$gxsm3
```

Thus, the script in your current working directory will be found.

If you want to put your script somewhere else than into the current directory, modify the environment variable `PYTHONPATH`. Python will look into all directories, that are stored there.

```
$export PYTHONPATH=/some/obscure/path/
$gxsm
```

Or you can link it from somewhere else. Or you can create a one line script, that executes another script or several scripts. Do whatever you like.

Files

Python precompiles your `remote.py` to `remote.pyc`. You can safely remove the file `remote.pyc` file at any time, Python will regenerate it upon start of the interpreter.

References

See the appendix for more information. Don't know Python? Visit python.org.

Known Bugs

The error handling is only basic. Your script may run if you give wrong parameters but not deliver the wanted results. You can crash Gxsm or even X! E.g. by selecting an illegal channel. Remember that channel counting in the scripts begins with 0. Gxsm's channel numbering begins with 1.

The embedded functions return -1 as error value. It's a good idea to attach `print` to critical commands to check this.

The `remote_echo` command is implemented via debug printing. Using Pythons `print` is recommended.

The view functions `quick`, `direct`, `log` change the viewmode, but not the button in the main window, don't be confused.

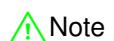
The `waitscan` and `da0` function are not yet implemented and likely will never be.

The library detection during compilation is amateurish. Needs work.

Python will check for the right type of your arguments. Remember, that all values in `set` are strings and have to be quoted. Additionaly care for the case sensitivity.

If you want to pause script execution, use the embedded sleep command `gxsm.sleep()` and not `time.sleep()`, because the function from the time library will freeze GXSM totally during the sleep. (This is not a bug, it's a feature.)

TODO: Add more action-handlers in Math-PI.



If you write a particularly interesting remote-script, please give it back to the community. The GXSM-Forums always welcome input.



Info for Plug-In: Tools/Pyremote Console

Plug-In name: pyremote	File: common/pyremote.C
Author: Stefan Schröder	Email: stefan_fkp@users.sf.net

22.5. Query Hardware/DSP software information

Description

This plugin looks for a file named `gxsm_extra_scan_info` in the current working directory at each file save action. Its content is added to the NetCDF file as variable named `extra_scan_info`.

Usage

Works automatic if loaded.



Note

Info for Plug-In: Tools/Hardware Info

Plug-In name: extra_scan_info	File: common/extra_scan_info.C
Author: Percy Zahl	Email: zahl@users.sf.net

22.6. Postscript printing tool

Description

This plugin replaces the GXSM core print utility.

Usage

Choose Print from the *main menu/File* menu. This Plugin can be run from the remote python script with `emb.action("print_PI")`. The default action is to print to a file with the same name as the nc-file. Other defaults are read from the gconf-registry. We don not print to a printer by default to allow mangling of the resulting PS-files after the PS-file creation. (It's a feature, not a bug. Later, configuration of the print action may be possible. Stay tuned.)

Info for Plug-In: File/Print

Plug-In name: printer	File: common/printer.C
Author: Stefan Schröder	Email: stefan_fkp@users.sf.net

22.7. Query Hardware/DSP software information

Description

This plugin asks the DSP, e.g the connected hardware or driver/module, to report informations about the current running software. This is used by Gxsm for automatic configuration of several features. – This new feature will be expanded and used more intens future versions.

Usage

Call it using the menu *main menu/Tools/Hardware Info*.

This works only with the second DSP software generation, starting with xsm CVS V1.20! And it works with the kernel DSP emulation modules!  Note

Info for Plug-In: Tools/Hardware Info

Plug-In name: queryDSPinfo	File: common/queryDSPinfo.C
Author: Percy Zahl	Email: zahl@users.sf.net

22.8. Show plug-in details

Description

This plugin lists all currently loaded plugins and shows all available information about those. In addition, it allows to call/run the About and Configure subroutines of each plugin – if available.

Usage

Call it using the menu *main menu/Tools/Plugin Details*.

Info for Plug-In: Tools/Plugin Details

Plug-In name: listplugins	File: common/listplugins.C
Author: Percy Zahl	Email: zahl@users.sf.net

22.9. MkIcons – Make icons

Description

This plugin helps you printing several nc-images on one page.

Usage

Choose Mkicons from the *main menu/Tools* menu.

Info for Plug-In: Tools

Plug-In name: mkicons
Author: Stefan Schröder

File: common/mkicons.C
Email: stefan_fkp@users.sf.net

23. Plug-Ins: hard

Here are the GXSM-2 *hard*, i.e. Hardware Interface (HwI), type plugins. These are a special subset of GXSM plugins and only one or none is selected on GXSM startup. HwI plugins are very similar to standard GXSM plugins from design point of view, except of a special and independent load strategy. A HwI plugin provides a **category** (same as used in standard type plugins for automatic plugin selection), this is used here to match the *GXSM preferences dialog, folder Hardware/Card*.

The hardware interface itself is created by providing one (or more) specialized derivation(s) of the GXSM core base class **XSM_Hardware**, which defines a default set of (virtual) generalized SPM-control functions to be overloaded later. A pointer to one instance of this new class has to be returned by the HwI plugins **get_gxsm_hwi.hardware_class(void *data)** function which is resolved by the GXSM core HwI plugin handler. And called to register the hardware for later use.

The current GXSM main stream and best supported hardware is the SignalRanger DSP board, see [23.2](#).

23.1. SPM SIM and Template Hardware Interface

Description

Info for Plug-In: Tools/SPM SIM Control

Plug-In name: spm_simulator_hwı
Author: Percy Zahl

File: hard/spm_simulator_hwı.C
Email: zahl@users.sf.net

23.2. Signal Ranger Hardware Interface

Description

This provides the Signal Ranger-STD and -SP2 hardware interface (HwI) for GXSM. It contains all hardware close and specific settings and controls for feedback, scanning, all kind of probing (spectroscopy and manipulations) and coarse motion control including

23. Plug-Ins: hard

the auto approach controller. Invisible for the user it interacts with the SRanger DSP, manages all DSP parameters and data streaming for scan and probe.

23.2.1. SR-DSP Feedback and Scan Control

The Feedback & Scan folder contains all necessary options regarding feedback and scan. Here the feedback parameters and tunneling/force settings are adjusted. The second purpose of this control panel is the adjustment of all scan parameters of the digital vector scan generator like speed.

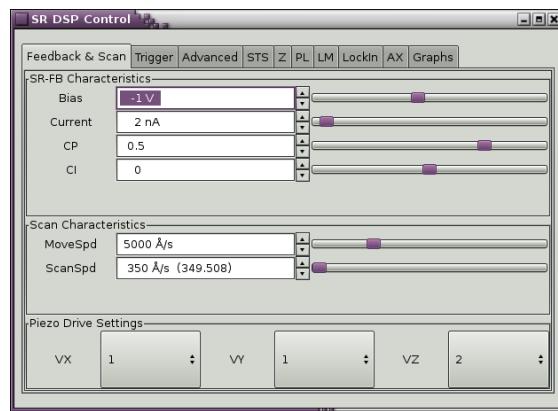


Figure 23.1.: GXSM SR DSP Control window: Feedback and Scan Control page

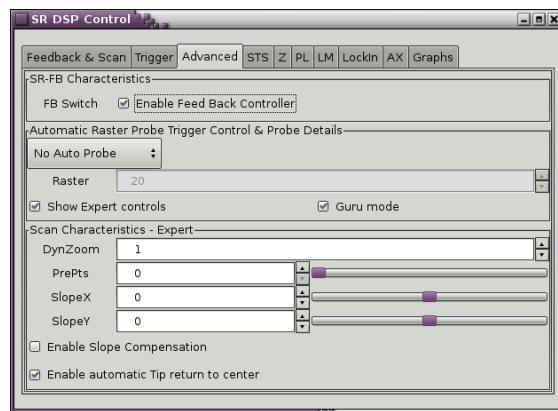


Figure 23.2.: GXSM SR DSP Control window: Advanced settings page

Bias Voltage that is applied to OUT6. Beware: The entered value will be divided by the

GXSM preferences dialog, folder InstSPM/BiasGain value. Without an additional amplification the *Instrument/BiasGain* entry value has to be set to 1.

Current(STM) / SetPoint(AFM) Set point the DSP is trying to reach. For a correct conversion of Current / SetPoint to incoming voltages at IN5 the values of *GXSM preferences dialog, folder InstSPM/nAmpere2Volt* and *GXSM preferences dialog, folder InstSPM/nNewton2Volt* have to be set correctly.

CP, CI Parameters for the feedback loop. CP is proportional to the difference between the set point and the current value at IN5. CI integrates this difference. Higher values for CP / CI mean a faster loop.

MoveSpd, ScanSpd MoveSpd is valid for movements without taking data (e.g. change xy-offset or moving to the starting point of an area scan). ScanSpd is valid if the DSP is taking data. PiezoDriveSetting: Usually a high voltage amplifier has different gains. A change of the gain can easily be mentioned in GXSM by activating the appropriate factor. A change of this values acts instantly. The available gain values can be defined in the *GXSM preferences dialog, folder InstSPM/Analog/V1-V9*. After changing the preferences GXSM needs to be restarted to take effect. Remark: Piezo constants and other parameters have to be charged. Please use the *GXSM preferences dialog, folder InstSPM/Instrument/PiezoAV* for this odd values.

Good conservative start values for the feedback loop gains CP and CI are 0.01 for both. Typically they can be increased up to 0.1, depending on the system, tip and sample. In general CP can be about 150% of CI to gain more stability with same CI. ⚠ Hint

23.2.2. Trigger – Multi-Volt and -Current Control

It is possible to trigger automatic changes of Bias or Current-Setpoint while scanning at up to four given X indices each, for Bias and Current for forward and backward scan direction.

If the index given is reached, the DSP will trigger a bias or current-setpoint change. The bias is never changed instantly, but is ramped with a fixed speed of (54 V/s @ 10V max. Bias) from that position to the new value. The index is actually down-counted to zero for each scan line, thus index 0 means the last point of every scan line.

The first line sets up the trigger points for X-forward (Xm) and X-backward (Xp) for bias changes: Index-Xp, new Bias, Index-Xm, new Bias. The second to fourth line does the same in an equivalent way. The next frame sets up the similar task for changes of

the current setpoint. Any Index never reached will just never trigger, thus an index of -1 will disable it.

To enable or update the trigger data table, toggle/enable the Trigger Enable checkbox. At startup, GXSM will always read the current DSP persistent trigger table and trigger-enable state.

While the trigger is enabled, the Bias and Current settings are locked. Any attempts to change it (i.e. the slider) will temporary enforce your setting until the next trigger hits.

You can turn the Trigger feature on or off at any time.

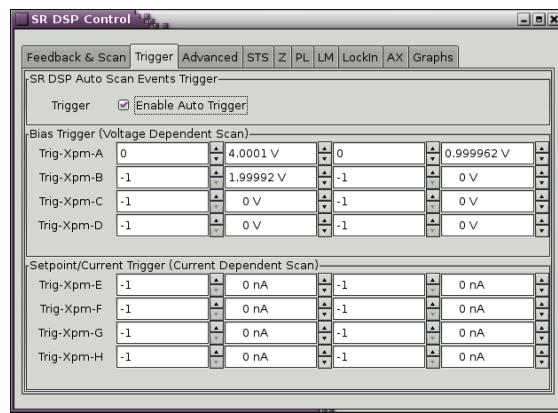


Figure 23.3.: Trigger Control window

23.2.3. Advanced Feedback and Probe Control

Advanced is a collection of different settings, to handle the advanced capabilities of the Signal Ranger DSP. It is quite a mixture of different tools.

FB Switch This option enables or disables the feedback loop. Disabling will keep the current Z-position.

Automatic Raster Probe In general spectra can easily be taken while doing an area scan. Activating this feature forces the DSP to take spectra every line with an intended misalignment. A value of 1 means a spectrum will be taken every scan point. A value of 9 means a distance of 9 area scan points between two spectra. The driven spectrum depends on the probe control that is selected (STS, Z, PL). Single spectra of a Raster Probe Scan easily can be handled by using the *Events popup window Show Probe* feature in the area plot. For a conversion into a layer based image please use the *main menu/Math/Probe/ImageExtract* plug-in.

Show Expert Controls This option hides or reveals some advanced options in different folders. In this help file controls will be mentioned as Expert Controls, if they are revealed by this option.

DynZoom With the Signal Ranger DSP it is possible to dynamical zoom while scanning.

PrePts Problems with drift in x-direction can be reduced by scanning a larger distance. The DSP adds equivalent points at both ends of a line which will be ignored in the resulting scan data. The total scan size will get larger in x by a factor of $(1 + 2 * PrePts / ScanPointsX)$. Use the Pan-View plug-in to prevent trouble with the maximum scan size.

IV-Type Spectroscopy

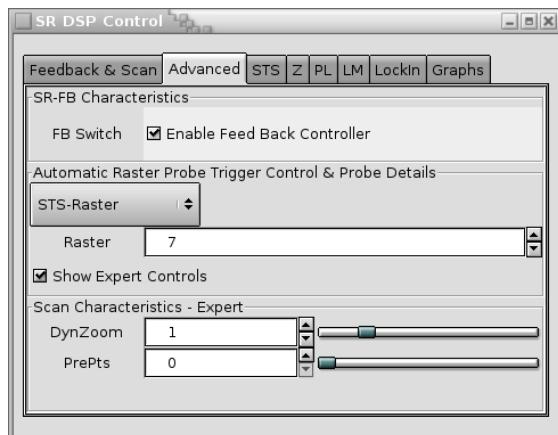


Figure 23.4.: Advanced setup with Raster enabled

This is the dialog for scanning tunneling spectroscopy (STS). Sources can be chosen in the Graphs Folder (Please have a look at the Graphs section for details).

IV-Start-End These are the start and the end values of the spectrum. Optional a repetition counter is shown (Expert Control option). This forces the Signal Ranger to do the STS spectrum n times.

IV-dz Lowering of the tip can improve the signal to noise ratio especially at low voltages. IV-dz is the maximum lowering depth which is reached at 0V. The lowering depth is equal to zero if the I/V curve meets the bias voltage. Lowering the tip means a decrease of the tunneling gap. An automatic correction of the resistance

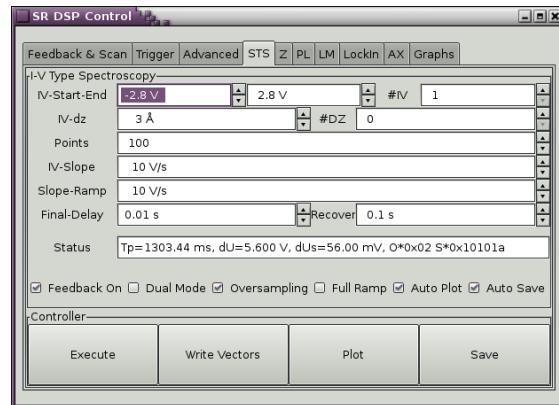


Figure 23.5.: STS folder

is implemented by means of several (#) dI/dz spectra at the bias voltage (Expert Control option).

Points Number of points the spectrum will have. Please have a look at *Int entry* for additional informations.

IV-Slope Slope while collecting data.

Slope-Ramp Slope while just moving.

Final-Delay After running a spectrum this is the time the DSP waits to enable the feedback again.

Recover This is the time between two spectra where the feedback is activated for a readjustment of the distance (Expert Control option).

Status Gives information about the ongoing spectrum:

T_p Total time the probe needs.

dU Maximum difference of the voltages that will be applied.

dUs Stepsize of the data points.

Feedback On Decides whether the feedback will be on or off while taking a spectrum.

Dual When activated the DSP will take two spectra. One spectrum is running from Start to End directly followed by a spectrum from the End to the Start value.

Int When activated the DSP will average all fetched data between two points. It can easily be seen, that decreasing the values of IV-Slope or Points will increase the oversampling and therefore will improve the quality of the spectrum. See note (*) below.

Ramp This option forces the DSP to stream all data to the PC including the Slope-Ramps.

Save When activated, GXSM will save spectra automatically to your home directory.

Plot When activated, GXSM will automatically show/update the plots chosen in the Graphs dialog.

* The sampling rate of the Signal Ranger is 22.1 kHz so the time between two points of a spectrum leads directly to the number of interim points that can be used for oversampling. total time for the spectrum:

$$ts = dU/IV - Slope$$

time per point:

$$tp = ts/Points = dU/(IV - Slope * Points)$$

number of samples at one point:

$$N = tp * 22100Hz = 22100Hz * dU/(IV - Slope * Points)$$



Note

Vertical (Z) Manipulation

Manipulation in general is controlled or forced top motion in one or more dimensions for any desired purpose. This is the dialog for distance spectroscopy and forced Z/tip manipulation.

Z-Start-End These are the start and the end values of the spectrum in respect to the current position.

Points Number of points the spectrum will have. Please have a look at *Int entry* for additional informations.

Z-Slope Slope while collecting data.

Slope-Ramp Slope while just moving.

Final-Delay After running a spectrum this is the time the DSP waits to enable the feedback again.

23. Plug-Ins: hard

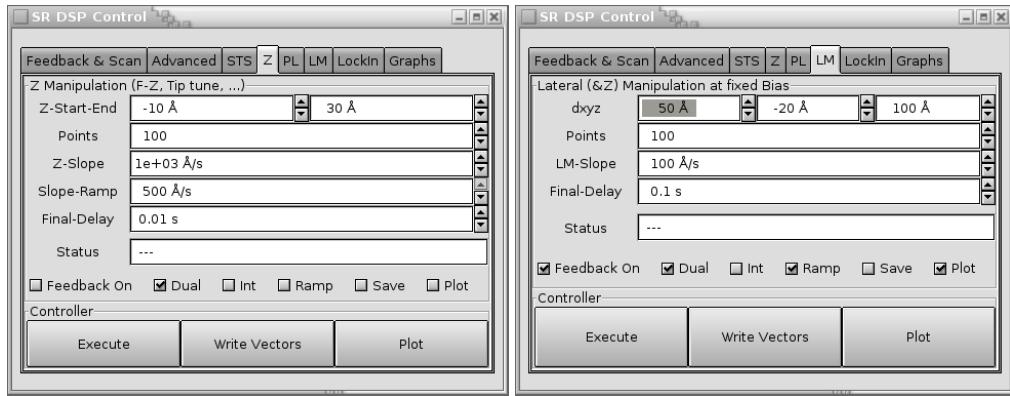


Figure 23.6.: left: GXSM SR DSP Control window, left: Z manipulation, right: LM, lateral and Z manipulation

Status Gives information about the ongoing spectrum:

T_p Total time the probe needs.

Informations about the check options can be found in STS.

Lateral Manipulation

With LM a lateral manipulation of the tip/sample is possible. But also the Z-dimension can be manipulated at the same time if dZ set to a non zero value.

dxz Distance vector that will be covered.

Points While moving it is possible to collect data. Points defines the number of collected data points.

LM-Slope Speed of the tip/sample.

Final-Delay Timeout after lateral manipulation.

Status Gives information about the ongoing move.

Informations about the check options can be found in STS.

Tip Enhancements and Field based Manipulation

There are several possibilities to prepare a tip. One is to dip the tip into the sample in a controlled manner (use *Z entry* for this). Another option is applying a charge pulse using this *PL entry dialog*.

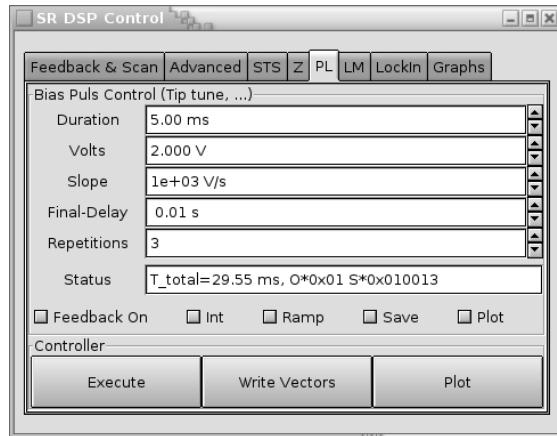


Figure 23.7.: GXSM SR DSP Control window: PL mode

Duration Determines the duration of the pulse.

Volts Applied voltage.

Slope Slope to reach *Volts entry*.

Final Delay Delay for relaxing the I/V-converter after pulsing.

Repetitions How many pulses are applied.

Status Gives information about the ongoing pulse.

Informations about the check options can be found in STS.

Control of the digital (DSP) Lock-In

The Lock-In folder provides all settings concerning the build in digital Lock-In. The Lock-In and with it the bias modulation is automatically turned on if any Lock-In data channel is requested, either for probing/STS (in Graphs) or as a scan data source (Channelselector) for imaging.

There are a total of five digital correlation sums computed:

Averaged Input Signal (LockIn0),

Phase-A/1st order (LockIn1stA), Phase-B/1st order (LockIn1stB),

Phase-A/2nd order (LockIn2ndA), Phase-B/2nd order (LockIn2ndB).

Please always select LockIn0 for STS.

 Note

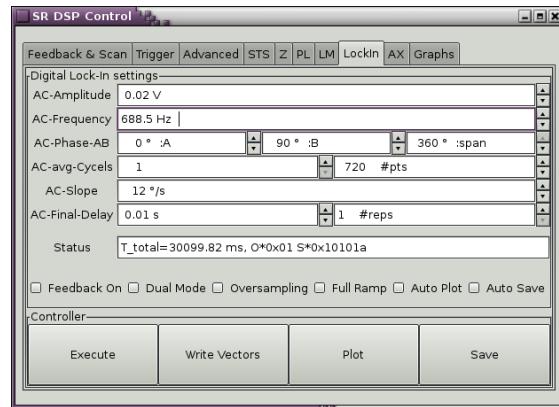


Figure 23.8.: GXSM SR DSP Control window: Lock-In settings

AC-Amplitude The amplitude of the overlaid Lock-In AC voltage.

AC-Frequency The base frequency of the Lock-In. There are four fixed frequency choices.

AC-Phase-AB Phase for A and B signal, applied for both, 1st and 2nd order.

AC-Avg-Cycles This sets the length for averaging, i.e. the corresponding time-constant.

For adjustments purpose only are the following parameters and the execute function here, not needed to run the Lock-In for all other modes. The special probe mode implemented in this section can actually sweep the phase of the Lock-In, it is useful to figure out the correct phase to use:

span Full phase span to sweep.

pts Number data points to acquire while phase sweep.

slope Phase ramp speed.

The digital Lock-In is restricted to a fixed length of base period (choices are 128, 64, 32, 16 samples/period with a fixed sample rate of 22100 samples/s) and a fixed number of 8 periods for computing the correlation sum: The total number of periods used for correlation of data can be increased by setting AC-Avg-Cycles greater than one, then overlapping sections of the 8 period long base window is used for building the correlation sum. Thus the total integration length (time constant) is

$$\tau = \frac{\text{AC-Ave-Cycles} \cdot 8}{\text{Frq}}$$

$$\text{Frq} = \frac{22100 \text{Hz}}{M = 128, 64, 32, 16}$$

There for the following discrete frequencies are available: 172.7Hz, 345.3Hz, 690.6Hz, 1381.2Hz.

The four correlation sums for A/B/1st/2nd are always computed in parallel on the DSP if the Lock-In is enabled – regardless what data is requested. The correlation length is given by:

$$N = 128 \cdot \text{AC-Ave-Cycels} \cdot 8$$

$$\omega = 2\pi \cdot \text{Frq}$$

Lock-In data calculations and reference signal generation is all in digital regime on the DSP in real-time. The modulation is applied to the Bias voltage by default automatically only if the Lock-In is active:

$$U_{\text{ref}} = \text{AC-Amp} \cdot \sin(\omega t) + \text{Bias}$$

Averaged signal and Lock-In output signals calculated:

$$\begin{aligned} U_{\text{LockIn0}} &= \sum_{i=0}^{N-1} U_{in,i} \\ U_{\text{LockIn1stA}} &= \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i \frac{2\pi}{M} + \text{Phase-A}) \\ U_{\text{LockIn1stB}} &= \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i \frac{2\pi}{M} + \text{Phase-B}) \\ U_{\text{LockIn2ndA}} &= \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i \frac{2\pi}{M} + \text{Phase-A}) \\ U_{\text{LockIn2ndB}} &= \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i \frac{2\pi}{M} + \text{Phase-B}) \end{aligned}$$

Implemented in FB_spm_probe.c, run_lockin() (C) by P.Zahl 2002-2007.



All Lock-In data is raw summed in 32bit integer variables by the DSP, they are not normalized at this time and moved to GXSM-3.0 via FIFO. GXSM-3.0 applies the normalization before plotting.



Informations about the check options can be found in STS.

Auxiliary Probe Control

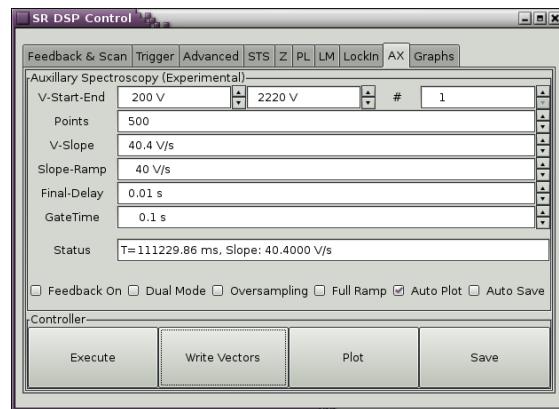


Figure 23.9.: GXSM SR DSP Control window: AX (Auxiliary) settings

This folder can be used for control and data acquisition from several kind of simple instruments like a QMA or Auger/CMA.

Note

Best is to setup a new user for this instrument and configure the Bias-Gain so the “Voltage” corresponds to what you need. As input you can select any channel, including Lock-In and Counter. Here the Gate-Time is used to auto set the V-slope to match V-range and points.

Data Sources and Graphing Control

In the Graphs folder all available data channels are listed. If a Source is activated, measured data will be transferred into the buffer. Saving the buffer will automatically save all activated sources. Additionally it is possible to define a source as to be displayed.

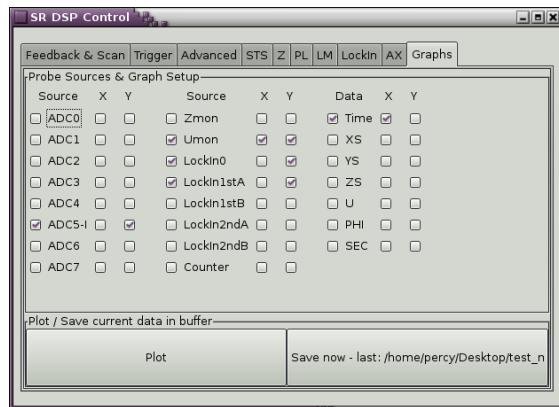


Figure 23.10.: GXSM SR DSP Control window, Graphs page: Plot and Data sources setup.

Beware: If a channel is not marked as a Source there will be no data to be displayed  Hint even if X or Y is checked.

23.2.4. SR-DSP Mover and Approach Control

GXSM with the SRanger also provides ramp like signal generation for slip-stick type slider/mover motions which are often used for coarse positioning and tip approach. Set *GXSM preferences dialog, folder User/User/SliderControlType* to *mover entry* to get the most configurable Mover Control dialog. If set to *slider entry* (default setting) the dialog will be simplified for Z/approach only. The different tabs are only for users convenience to store different speed/step values, the output will always occur as configured on the *Config entry* folder.

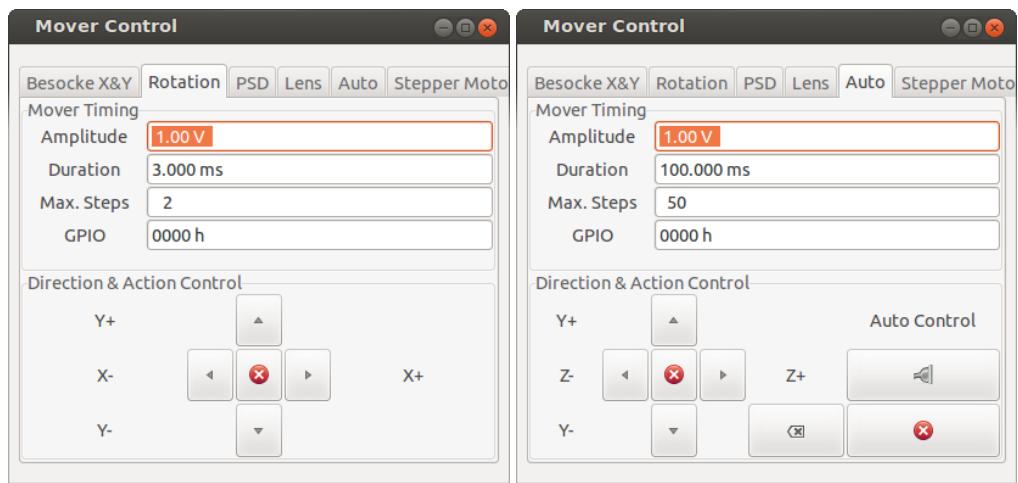


Figure 23.11.: left: GXSM SR generic coarse mover controller, right: Auto approach controller

23.2. Signal Ranger Hardware Interface

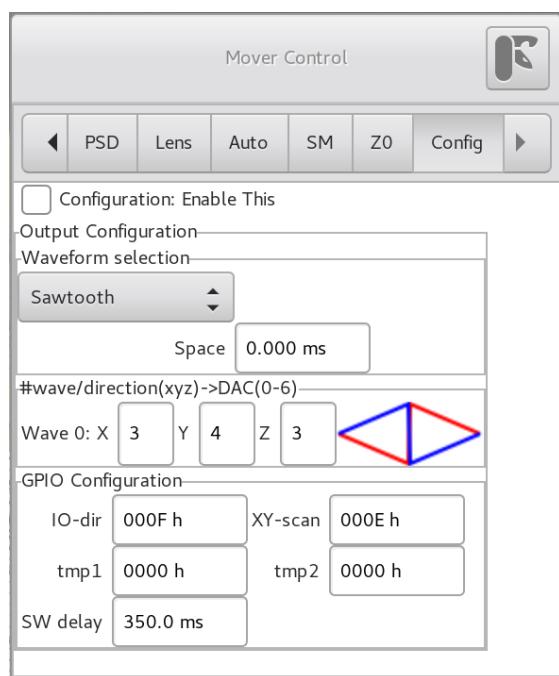


Figure 23.12.: GXSM SR DSP configuration of SRanger inertial driver engine.

23.2.5. Extra Python SR-DSP Control and Configuration Scripts

The Python script SRanger/TiCC-project-files/FB_spmcontrol/python_scripts/sr_spm_control can be used for inspection and DSP/SPM software configuartion:

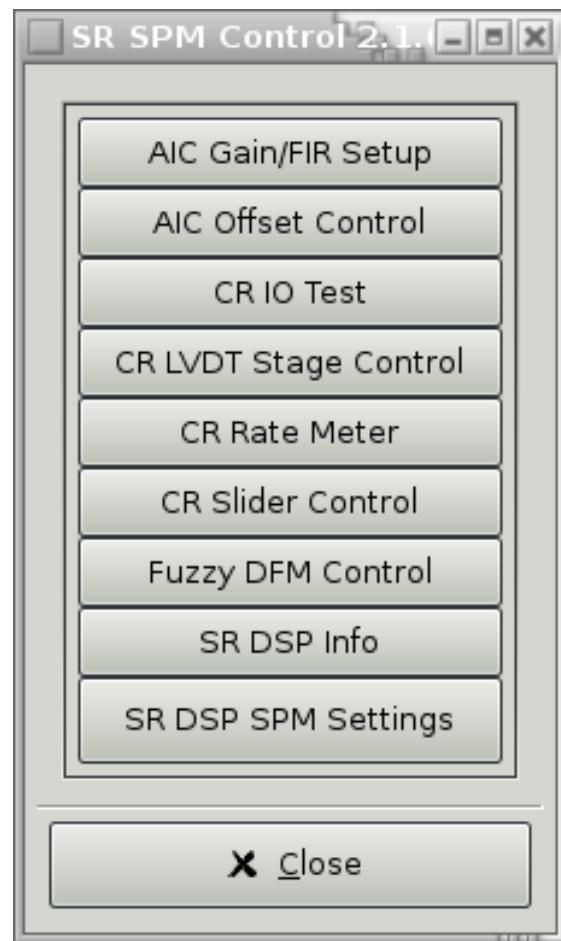


Figure 23.13.: Main Menu

23.2. Signal Ranger Hardware Interface

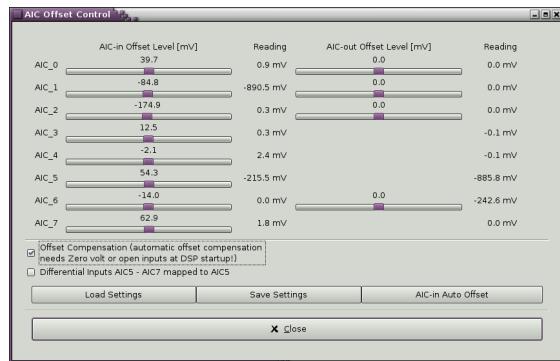


Figure 23.14.: AIC Offset Control

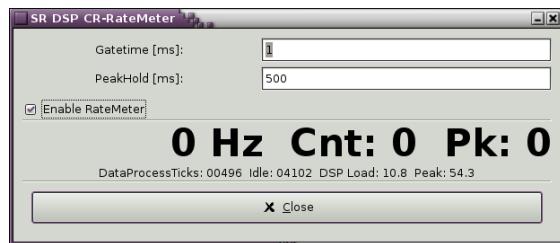


Figure 23.15.: CoolRunner Rate Meter

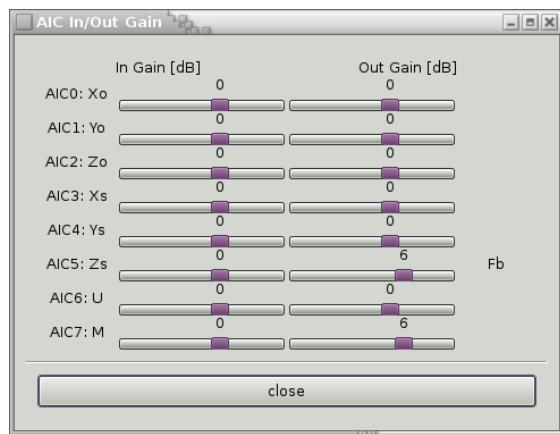


Figure 23.16.: AIC Gain Control



Figure 23.17.: SR-DSP/SPM software version info

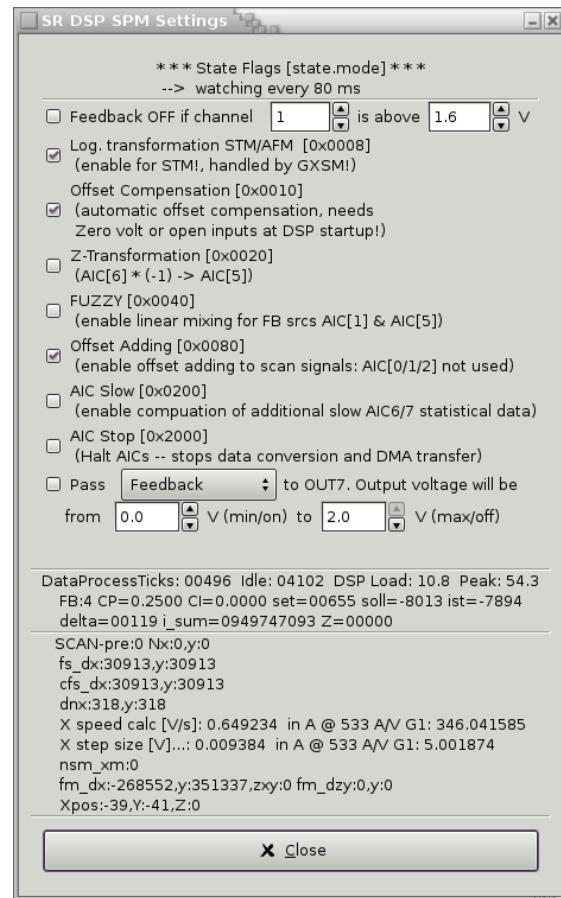


Figure 23.18.: SPM settings

23. Plug-Ins: hard

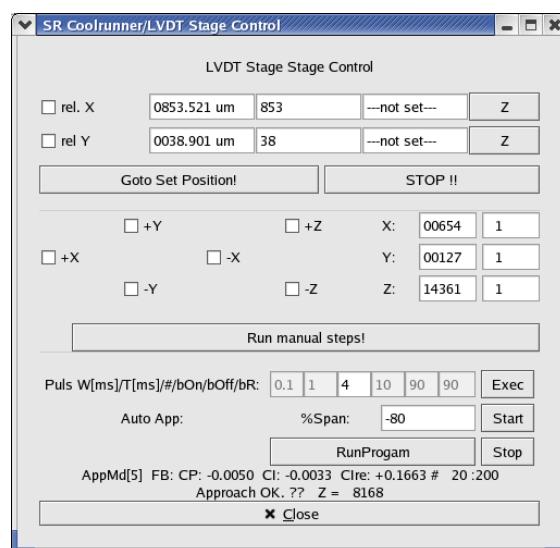


Figure 23.19.: DSP Stage/LVDT Control

Usage

Set the *GXSM preferences dialog, folder Hardware/Card* to "SRanger:SPM".

Launch

/SRanger/TiCC-project-files/FB_spmcontrol/FB_spmcontrol.out
on the SR before starting GXSM-3.0!



Note

Execute "loadusb" like this before starting GXSM-3.0 or any other DSP tool:

besocke@thundera:/SRanger/loadusb\$./loadusb/TiCC-project-files/FB_spmcontrol/FB_spmcontrol

Special features and behaviors to be documented here!

Info for Plug-In: Tools/SR-DSP Control

Plug-In name: sranger_hwi

File: hard/sranger_hwi.C

Author: Percy Zahl

Email: zahl@users.sf.net

23.3. Local Area Network (Internet) RHK Controller Hardware Interface (to be ported)

Description

This plugin/class provides a interface for SPM adquisition through a Local Area Network using sockets. In particular, this plugin provides the necessary interface to interact with the standalone `rhk_controller` program and thus use an *RHK STM-100* (the old stand-alone models before rev 7, not the new DSP based systems) with Gxsm.

It can be used as the foundation of a general SPM-through-the-internet plugin.

With this one, you can:

- read the settings of the RHK electronics (image size, offset, tunneling conditions, pixels in image)
- adquire images with the proper settings (either topographic, current or any other channel).
- change from XY to YX scanning

The current limitations are:

- Only forward image (it is not difficult to implement saving the backward image)
- Only a single channel can be adquiered. Again it is not too complex to change this.
- No spectroscopy. Actually we have the code for doing IV curves, but we do not expect we will implement it in the `rhk_controller` program anytime soon.

The ugliest code is due to Gxsm assuming that all settings (bias, scan size, etc) are set by Gxsm, not by the electronics itself (as is the case for the RHK electronics, which has its own scanning hardware). We have hacked the PutParameter call to also receive the data, and this is why a non standard scanning plugin is needed (`rhk_scancontrol`).

The `rhk_controller` program (available from the Gsxm CVS, module `RHK_controller`) handles all the low level details of data aquisition with an *RHK STM-100* <http://www.rhktech.com> with a simple DAQ card *I/O Tech Daqboard 2000* <http://www.Iotech.com>. A couple of cable adapters are needed to connect the electronics and the adquisition card: one was the *I/O Tech DBK202*, to separate the analogic and digital inputs and outputs of the andquisition card; the other was a home made one to adapt the DBK202 to the RHK unit (if anyone needs the pcb drawing, please contact Farid). The `rhk_controller` is run with simple ASCII commands to read the RHK settings, and select the adquisition channel, adquiere images and so on. Look at the program itself for more information

(it is a fairly simple C program which uses lex/yacc to parse the commands), or connect to it with telnet localhost 5027 and type "help". We use a Omicron Coarse approach controller which is interfaced by TTL with the DAQ card. The DSPMover plugin works with it.

To manage the *I/O Tech Daqboard 2000* under linux we use the I/O tech linux driver. The Daqboard2000.tgz file from IOTech has:

- the module with the device driver, db2k
- the lib to acces the device driver from a user site, libdaqx
- some examples

The version used for the *I/O Tech Daqboard 2000* driver is 0.1 and is GPL licenced. This library is not thread-safe and calls way too many "printk" to print debug messages. A big problem is that it tries to reserve the buffer memory before each adquisition. We have a patch (in the src RHK_controller) which eliminates the printk and reserves the memory buffer only at installation (2Mb).

Usage

Set the *GXSM preferences dialog, folder Hardware/Card* to "LAN_RHK:SPM", and the *GXSM preferences dialog, folder Device/Hardware* to "localhost:5027" if you are running rhk_controller in the same computer. Start the rhk_controller *before* the Gxsm program. The output of the rhk_controller program should say "someone connected" when starting Gxsm.

You can see some data taken with it in <http://hobbes.fmc.uam.es/loma>.

Info for Plug-In: Hardware/LAN_RHK:SPM-Hwi

Plug-In name: LAN_rhk_hwi

Author: Farid El Gabaly, Juan de la Figuera

File: hard/LAN_rhk_hwi.C

Email: johnnybegood@users.sourceforge.net

23.4. TC211 CCD Interface

Description

This is an experimental hardware interface plugin. Grabbing video data from a parport attached TC211 CCD camera based on the design presented in the "c't magazine für computer technik", Januar 1992, p.162ff.

Usage

Configure the *GXSM preferences dialog, folder Hardware/Card* to "TC211-CCD". Load the "ccd.o" kernel module before starting the GXSM, setup /dev/ccd correct. Check the module source code for correct LPT port.

Sources

Destination

Usual scan destination channel.

Used for my very old experimental Astro CCD camera. The special kernel module "ccd" is used for data transfer.

Info for Plug-In: Hardware/TC211-CCD-Hwi

Plug-In name: tc211_ccd
Author: Percy Zahl

File: hard/tc211_ccd.C
Email: zahl@users.sf.net

23.5. Interface to the spm2 scan device

Description

The spm2 package provides the control of a Scanning Probe Microscope through one or two common use DAQ boards (<http://spm.polosci.unifi.it>) and a software simulated DSP. This module supports the use of the spm2 package from within the GXSM program.

Info for Plug-In: Tools/SR-DSP Control

Plug-In name: kmdsp_hwi
Author: Marcello Carla'

File: hard/kmdsp_hwi.C
Email: carla@fi.infn.it

23.6. Innovative DSP Hardware Interface for PC31 and PCI32 (OBSOLETE or to be ported)

Description

This provides the Innovative DSP hardware interface for GXSM. Supported are PC31 and PCI32.

Usage

Set the *GXSM preferences dialog, folder Hardware/Card* to "Innovative_Dsp:SPM" or "Innovative_Dsp::SPA".

This is an experimental code since GXSM2 and HwI. Use GXSM-1 ("Gxsm" CVS branch) for a stable version.

Info for Plug-In: Hardware/Innovative_DSP:SPM:SPA-HwI

Plug-In name:	innovative_dsp_hw	File:	hard/innovative_dsp_hw.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.7. Demo Hardware Interface

Description

This provides a demonstration hardware interface (HwI) for GXSM. It contains all hardware close and specific settings and controls for feedback, scanning, all kind of probing (spectroscopy and manipulations).

All functions are simulated.

Info for Plug-In: Tools/SR-DSP Control

Plug-In name:	demo_hw	File:	hard/demo_hw.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.8. Video4Linux Grabber (experimental, to be ported)

Description

This is an experimental hardware interface plugin. Grabbing video data using the Video4Linux (v4l) device.

It's using v4l, so set Hardware/Device to the desired /dev/videoXX device!

Usage

Set the *GXSM preferences dialog, folder Hardware/Card* to "grab_v4l".

Sources

v4l device, i.e. (S)-Video/TV/...

Destination

Usual scan destination channel.

Experimental.

Info for Plug-In: Hardware/video4linux-Hwi

Plug-In name:	grab_v4l	File:	hard/grab_v4l.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.9. Signal Ranger MK2/3-A810 Hardware Interface

Description

This provides the SignalRanger-MK2 and -MK3-Pro (equipped with the A810 analog IO module) hardware interface (Hwi) for GXSM. It contains all hardware close and specific settings and controls for feedback, scanning, all kind of probing (spectroscopy and manipulations) and coarse motion control including the auto approach controller. Invisible for the user it interacts with the SRanger DSP, manages all DSP parameters and data streaming for scan and probe. The newer MK3-Pro (mark 3) DSP provides more computing power and a 32bit architecture and is capable of running a software PLL on top of the standart A810 architecture and a PLL optimized A810 module what is fully compatible but provided a more precise (stabilized) frequency reference for higher PAC/PLL precision.

THIS PLUGIN IS DERIVED FROM THE ORIGINAL SR MODULE AND IT STARTS WITH EXACTLY THE SAME FEATURES. THIS DOCUMENT SECTION IS STILL WRITING IN PROGRESS. We assume by now that the SR-MK2 or MK3-Pro is equipped with the Analog-810 (MK2-A810 or MK3-PLL SPM Controller), this is the new SPM dedicated analog module GXSM will supports best starting in 2009 (official introduction: Feb-2009) on. Main differences/upgrades are 75/150kHz loop, high stability and precision AD/DA, lowest possible loop delays (less than 5 samples total), no hardware FIR/IIR or oversampling any longer on any channel.

 Note

The MK2-A810 AD/DA channel configuration is by now equivalent to the old SRanger with exception of the new 4-channel feedback source mixer inputs. SR-AIC0..7-in/out corresponds to MK2-A810 AD/DA0..7 (See Appendix, SRanger DSP).

 Note

In brief:

DA0/1/2: X0/Y0/Z0, DA3/4/5: Xs/Ys/Zs, DA6/7: Bias, Motor.

AD0: real time IIR enables channel, Mixer input 0 (default feedback in for Current-Signal), AD1,2,3: additional input channels usable for feedback source mixer, AD4..7: Auxiliary.

Notes on the latest MK3-Pro-A810/PLL “Snowy Janus Hack”: Starting with this experimental SVN revision <https://sourceforge.net/p/gxsm/svn/HEAD/tree/branches/Gxsm-2x3-transition-sig> a totally real time and hot configurable so called signal configuration allows for any in- and output configuration and thus all there is a default assignment so far known from the MK2. Please refer to the MK3 subsection for deviations from the MK2 interface and setup.

 Note

New: The A810 DSP code implements a high resolution (HR) mode for the DAC. This increases the bit resolution of selected output channels up to 3 bits. This is possible on DSP software level now running sampling and data processing at full 150 kHz but limiting all other DSP tasks like scan and feedback to a reasonable fraction of this. For all input channels an automatic scan speed depending bandwidth adjustment (simple averaging) is used as before, but now the gained resolution due to statistics is not any longer thrown away (rounded off to integer), but the full 32 bit accumulated value and normalization count is kept and transferred. A normalization to the original 16 bit magnitude, but now as floating point number, is done by the HwI as data post processing. For performance reasons and future expansions the FIFO data stream consisting of a set of 32 bit signals is now compressed using first order linear predictor and custom encoded byte packed.

In particular the signal to noise ratio of small or noisy signals will increase automatically with lowering the scan speed (given here in pixels/s). All available input channels

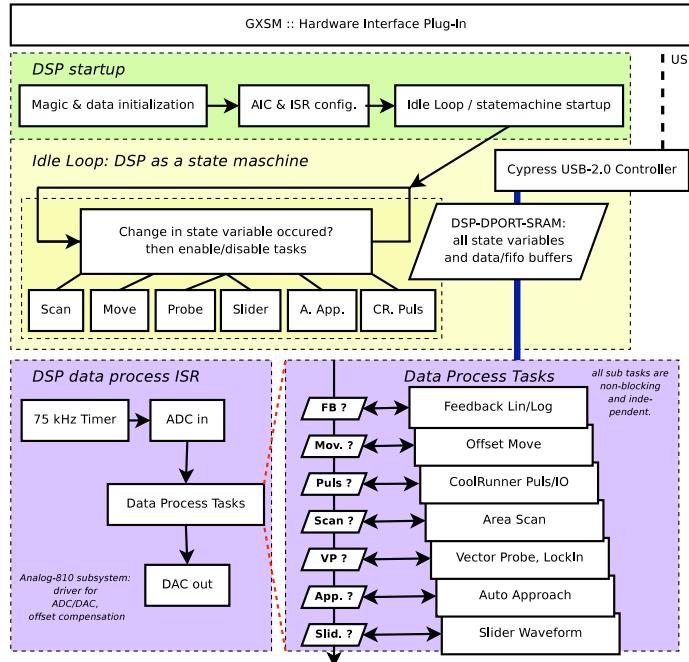


Figure 23.20.: Schematic diagram of the DSP code topology: Startup section configures the DSP system, sets up timers for data processing interrupt subroutine (ISR) and enters the never ending idle loop, which implements a state machine.

ADC_i are managed in this automatic scan speed matching bandwidth mode:

$$V = \frac{1}{N} \sum_{n=0}^{N-1} \text{ADC}_i(n), \quad N = \frac{75000 \text{ Hz}}{\text{Pixel/s}}$$

or in signal noise gain (S_N) terms

$$S_N = 20 \log \left(\frac{1}{\sqrt{N}} \right)$$

is the gain in signal to noise ratio – assuming statistical noise – on top of the available 16 bits (1 bit RMS) of the MK2-A810.

This now requires to adjust the Gxsm preferences as indicated on Gxsm startup if not setup correctly. Refer also to the sample configuration shown in the appendix 23.9.5 of this Hwi section.

Further the latest experimental release revises the feedback loop configuration and allows up to four signals (provided on ADC0...3) to be user configurable as feedback sources using linear or logarithmic signal transformations and even can be mixed and

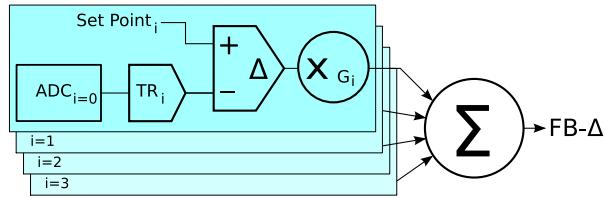


Figure 23.21.: Schematic diagram of the new DSP feedback with four signal source mixer. The input signals are transformed as requested (TR_i: linear/logarithmic/fuzzy) and the error signal Δ_i is computed for every channel i using individual set points. The sum of previously with gain G_i scaled delta signals is computed and feed into the feedback algorithm FB as Δ .

weighted in different ways via the gains G_i as illustrated in Fig. 23.21. It also includes a special “FUZZY” mix mode for signal level depended enabling of a particular chanqnel only if its signal is beyond a given level; only the “amount beyond” is used for Δ_i computation.

This multi channel feedback mode allows for example contineous transitions between STM and AFM or Dynamic Force Microscope (DFM) operation modes. The “FUZZY” mode can be used in many ways, one may be a kind of “tip guard” mechanism watching for special conditions, i.e. watching the power dissipation signal commonly available from Phase Locked Loop (PLL) controllers used for DFM.

For channel ADC₀ a real time self adaptive Infinite Response (IIR) filter which adjusts its frequency as function of the signal magnitude is implemented – assuming to be used for sampling the tunnel current in STM mode. The user selects a crossover current I_c and cut-off lower limit frequency f_{\min} , also the upper bandwidth can be limited to f_{\max} . This will then in real time limit the ADC₀ input bandwidth in dependence of the signal magnitude $|I_n|$ according to:

$$f_0(q) = -75000 \text{ Hz} \frac{\ln(q)}{2\pi}, \quad q(I_n) = 1 - \frac{\frac{f_{\min}}{f_{\max}} I_c + |I_n|}{I_c + |I_n|}$$

This real time computed q is limited to a q_{\min} matching the given f_{\max} before the bandwidth limited (IIR filter) current signal \tilde{I}_n which is recursively computed on the DSP according to:

$$\tilde{I}_n = q\tilde{I}_{n-1} + (1-q)I_n$$

Fig. 23.22 illustrates the IIR filter response to a logarithmic steepening stair case test input signal. This is a live IIR performance demonstration utilizing the Gxsm Vector

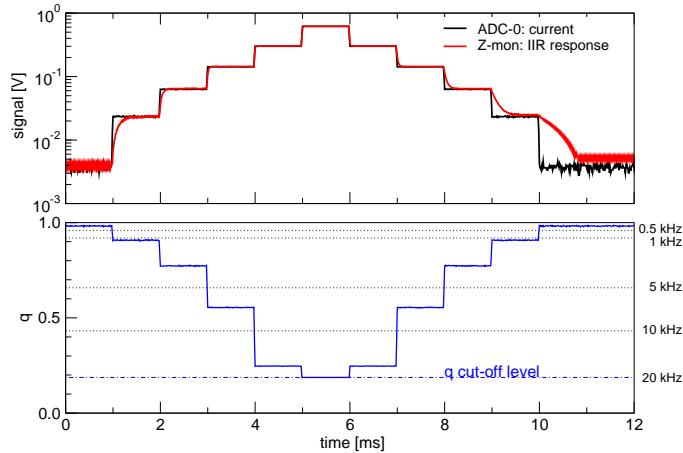


Figure 23.22.: Life IIR performance demonstration in Gxsm self-test configuration. A stair case like current input signal starting at 3.6pA noise level assuming 1V/nA (or a $\times 10^9$ gain) with logarithmic steps starting at 20 pA and scaled by $\times 2$ for the following steps (ADC-0 (black)) is generated using the Gxsm "PL"-Vector Probe mode. At the same time the IIR-response (via Z-mon (red)) and the real-time q (blue) is recorded. IIR settings used: $f_{\min} = 50$ Hz, $f_{\max} = 20$ kHz, $I_c = 500$ pA.

Probe Engine itself and having Gxsm and the MK2-A810 set up in a self-test configuration. The stair case like current input signal is starting at about 3.6 pA noise level assuming 1 V/nA (or a $\times 10^9$ gain). The logarithmic steps start at 20 pA and scale by $\times 2$ for the adjacent steps (ADC-0 (black)). The test signal is generated using the Gxsm "PL"-Vector Probe mode. At the same time the IIR-response (via Z-mon (red)) and the real-time q (blue) is recorded.

The IIR settings used for this demonstration are: $f_{\min} = 50$ Hz, $f_{\max} = 20$ kHz, $I_c = 500$ pA.

Looking at the IIR signal (red) the filter effectivity is clearly visible for small signals, also the fast "step-up" response is demonstrated by comparing with the slower "step-down" response. Also note the q cut-off indicated as the q cut-off level at 20 kHz.

For a low signal to noise ratio (given for tunneling currents in the pA regime) the feedback stability can be gained by a combination of IIR filtering and slower scanning. This digital self adapting IIR filter implementation allows full control of the frequency ranges and guarantees a fast tip response to a sudden increase of the tunnel signal (up to full band width) as needed to prevent the tip from crashing into step bunches or other "edges".

23.9.1. SR-DSP Feedback and Scan Control

The Feedback & Scan folder contains all necessary options regarding feedback and scan. Here the feedback parameters and tunneling/force settings are adjusted. The second purpose of this control panel is the adjustment of all scan parameters of the digital vector scan generator like speed.

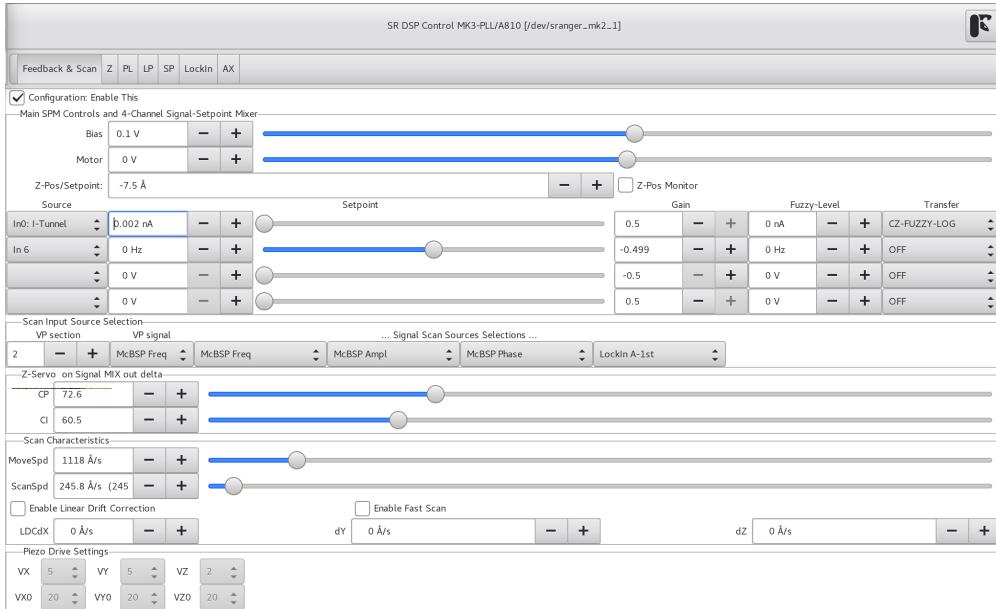


Figure 23.23.: GXSM3 MK3-PLL/A810 DSP Control window: Feedback and Scan Control page with activated configuration mode and internal offset-adding

Bias Voltage that is applied to OUT6. The entered value will be divided by the *GXSM preferences dialog, folder InstSPM/BiasGain* value. Without an additional amplification or attenuation the *Instrument/BiasGain* entry has to be set to 1.

Feedback Mixer Source: SCurrent The signal from ADC0 is feed to the self adaptive FIR (if enabled in Advanced Settings) and the processed as any other Mixer Source Input. However, this channel is expected to be used for (STM) Current Setpoint as it is converted due to the setting of *GXSM preferences dialog, folder InstSPM/nAmpere2Volt*.

Feedback Mixer Source: SSetPoint, SAux2,3 Further Mixer input signals from ADC1, ADC2 and ADC3.

23. Plug-Ins: hard

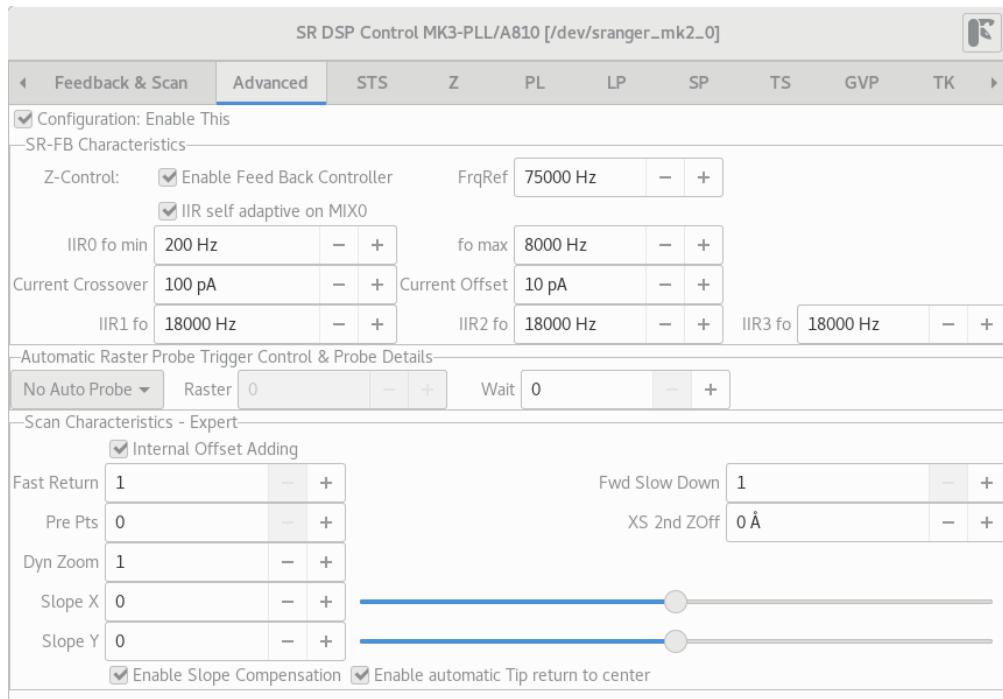


Figure 23.24.: GXSM3 MK3-PLL/A810 DSP Control window: Advanced settings page with activated configuration mode

Mixer Gains For scaling of the error signal (SetPoint - Source), use about 0.5 at max. Basically works as pre-scaling of CP and CI together if only one channel is used.

CP, CI Parameters for the feedback loop. CP is proportional to the difference between the set point and the current value at IN5. CI integrates this difference. Higher values for CP / CI mean a faster loop.

MoveSpd, ScanSpd MoveSpd is valid for movements without taking data (e.g. change xy-offset or moving to the starting point of an area scan). ScanSpd is valid if the DSP is taking data. PiezoDriveSetting: Usually a high voltage amplifier has different gains. A change of the gain can easily be mentioned in GXSM by activating the appropriate factor. A change of this values acts instantly. The available gain values can be defined in the *GXSM preferences dialog, folder InstSPM/Analog/V1-V9*. After changing the preferences GXSM needs to be restarted to take effect. Remark: Piezo constants and other parameters have to be charged. Please use the *GXSM preferences dialog, folder InstSPM/Instrument/PiezoAV* for this odd values.

LDCdx,dy Coefficients for Linear Drift Correction – active if *Enable LDC* is checked.

While checked, any manual Offset changes are prohibited (blocked) – uncheck to perform offset changes. This is the speed and direction the tip is moving to correct for drift. Use the negative number found via *Scan popup window View/Coordinates/Relative, check Time* and set a Globel Reference Point via Point-Object in a previous Scan to mark a feature.

Good conservative start values for the feedback loop gains CP and CI (on a MK2-A810 system, 32 bit internal handling) are 0.004 for both. Typically they can be increased up to 0.01, depending on the the system, tip and sample.



The MK3Pro-A810 handels the feedback internally with 64 bit. Therefore, the values for the feedback are roughly a factor of 256 higher than for a MK2 based system. Good standard values for the Mk3Pro-A810 based system will be between 20 and 40. A meaningfull maximum is 256. CP = 100 (MK3Pro-A810) results in 1 mV change per 1 V error signal.

In general CP can be about 150% of CI to gain more stability with same CI. The feedback transfer is 1:1 (error to output per loop) if all gains, CP and CI, are set to one in linear mode.

23.9.2. Advanced Feedback and Probe Control

Advanced is a collection of different settings, to handle the advanced capabilities of the Signal Ranger DSP. It is quite a mixture of different tools.

FB Switch This option enables or disables the feedback loop. Disabling will keep the current Z-position.

Frq-Ref must be set to 75000kHz and not changed at any time (guru mode only).

IIR ADC0 (Current) can be real time signal magnituden dependent IIR filtered. A self adjusting IIR algorithm is used. See introduction to this plugin for details.

Automatic Raster Probe In general spectra can easily be taken while doing an area scan. Activating this feature forces the DSP to take spectra every line with an intended misalignment. A value of 1 means a spectrum will be taken every scan point. A value of 9 means a distance of 9 area scan points between two spectra. The driven spectrum depends on the probe control that is selected (STS, Z, PL). Single spectra of a Raster Probe Scan easily can be handled by using the *Events popup window Show Probe* feature in the area plot. For a conversion into a layer based image please use the *main menu/Math/Probe/ImageExtract* plug-in.

Show Expert Controls This option hides or reveals some advanced options in different folders. In this help file controls will be mentioned as Expert Controls, if they are revealed by this option.

DynZoom With the Signal Ranger DSP it is possible to dynamical zoom while scanning – side effects on offest as zooming (data point separation adjustment) occurs in real time at the current tip position, guru mode only. Always set back to one at scan start.

PrePts Problems with piezo creep in x-direction can be reduced by scanning a larger distance. The DSP adds equivalent points at both ends of a line which will be ignored in the resulting scan data. The total scan size will get larger in x by a factor of $(1 + 2 * \text{PrePts}/\text{ScanPointsX})$. Use the Pan-View plug-in to prevent trouble with the maximum scan size.

STS

This is the dialog for scanning tunneling spectroscopy (STS). Sources can be chosen in the Graphs Folder (Please have a look at the Graphs section for details).

IV-Start-End, #IV These are the start and the end values of the spectrum. Optional a repetition counter is shown (Expert Control option enabled). This forces the Signal Ranger to do the STS spectrum n times.

IV-dz Lowering of the tip can improve the signal to noise ratio especially at low voltages. IV-dz is the maximum lowering depth which is reached at 0V. The lowering depth is equal to zero if the I/V curve meets the bias voltage. Lowering the tip means a decrease of the tunneling gap. An automatic correction of the resistance is implemented by means of several (#) dI/dz spectra at the bias voltage (Expert Control option).

Points Number of points the spectrum will have. Please have a look at *Int entry* for additional informations.

IV-Slope Slope while collecting data.

Slope-Ramp Slope while just moving.

Line-dXY, #Pts Only shown if Expert Control option is set. Only if #Pts is set to more than one the complete procedure previously defined will be repeated on #Pts points equally spread along the via Line-dXY defined vector starting at the current tip position. Markers will be placed into the active scan window once done if Auto Plot is selected or every time updated if Plot is hit. To find the propper Vector

23.9. Signal Ranger MK2/3-A810 Hardware Interface

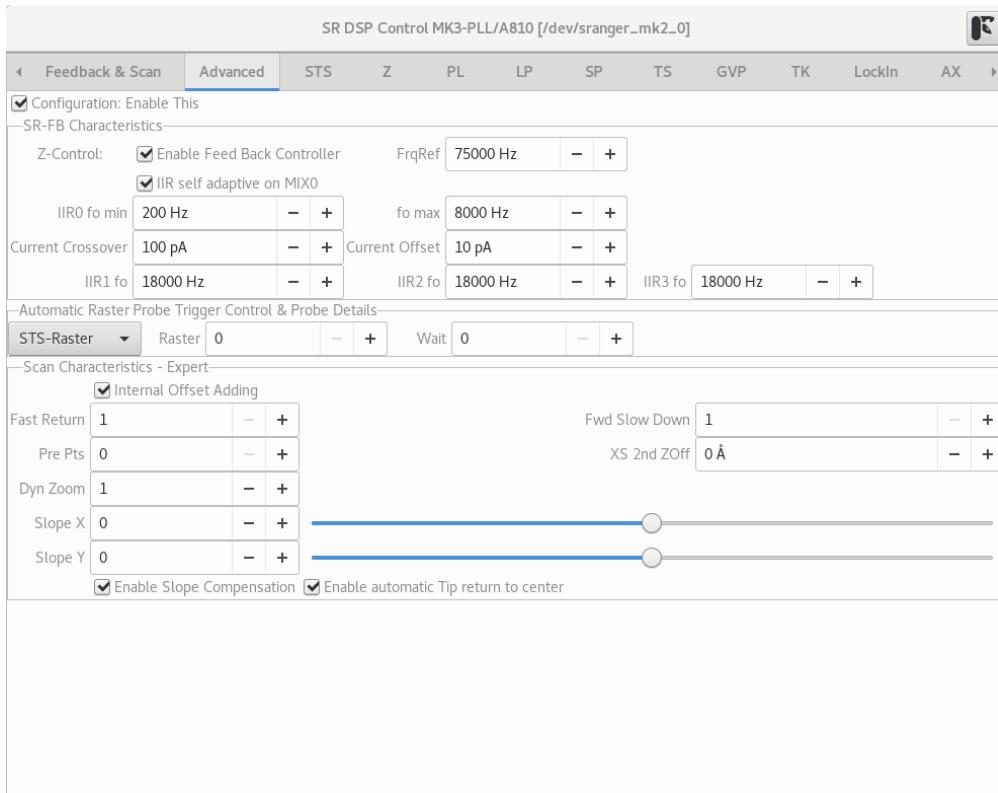


Figure 23.25.: Advanced setup with Raster enabled

coordinates, use the Line Object and read the dxy vector from the status line as indicated by $d(dx, dy)$. IV-Line Slope is used for moving speed from point to point – feedback will be enabled and the recovery delay is applied before the next STS probe cycle starts. Enable “Show Probe Events” (Scan side pane, Probe Events), else nothing will be shown on the scan. See demo setup in appendix 23.9.7 to this section.

Final-Delay After running a spectrum this is the time the DSP waits to enable the feedback again.

Recover This is the time between two spectra where the feedback is activated for a readjustment of the distance (Expert Control option).

Status Gives information about the ongoing spectrum:

T_p Total time the probe needs.

23. Plug-Ins: hard

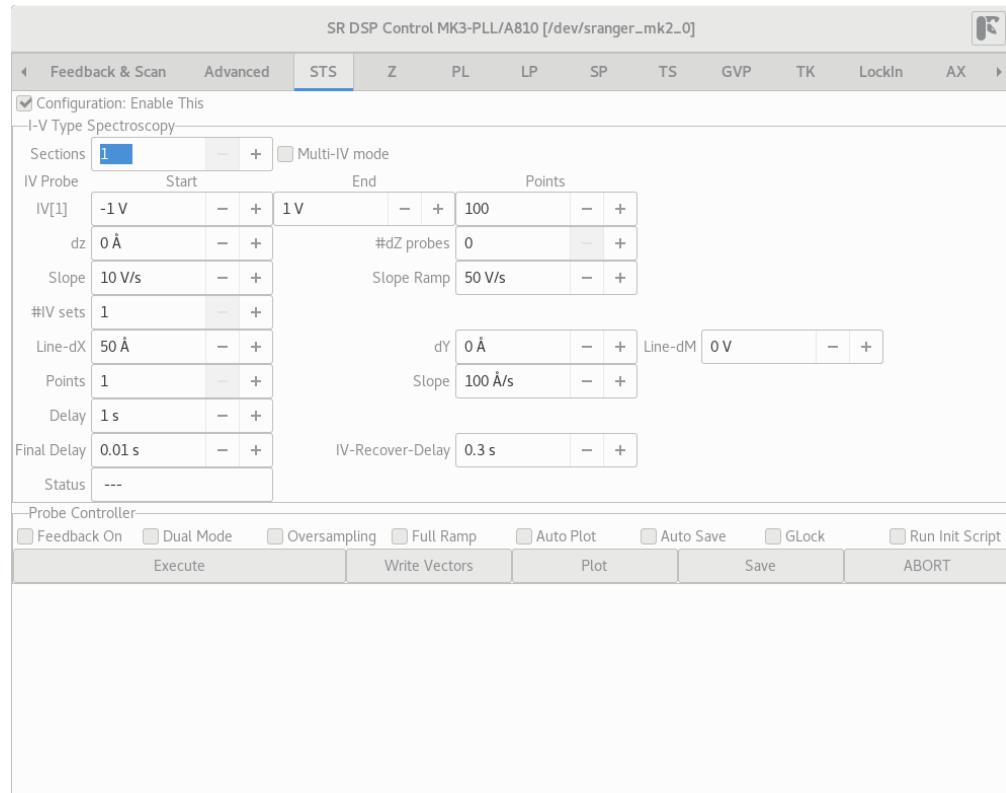


Figure 23.26.: STS tab of the MK3 DSP control

dU Maximum difference of the voltages that will be applied.

dUs Stepsize of the data points.

Feedback On Decides whether the feedback will be on or off while taking a spectrum.

Dual When activated the DSP will take two spectra. One spectrum is running from Start to End directly followed by a spectrum from the End to the Start value.

Int When activated the DSP will average all fetched data between two points. It can easily be seen, that decreasing the values of IV-Slope or Points will increase the oversampling and therefore will improve the quality of the spectrum. See note (*) below.

Ramp This option forces the DSP to stream all data to the PC including the Slope-Ramps.

Save When activated, GXSM will save spectra automatically to your home directory.

Plot When activated, GXSM will automatically show/update the plots chosen in the Graphs dialog.

* The sampling rate of the Signal Ranger is 22.1 kHz so the time between two points of a spectrum leads directly to the number of interim points that can be used for oversampling. total time for the spectrum:



$$ts = dU/IV - Slope$$

time per point:

$$tp = ts/Points = dU/(IV - Slope * Points)$$

number of samples at one point:

$$N = tp * 75000Hz = 75000Hz * dU/(IV - Slope * Points)$$

Vertical (Z) Manipulation

Manipulation in general is controlled or forced top motion in one or more dimensions for any desired purpose. This is the dialog for distance spectroscopy and forced Z/tip manipulation.

Z-Start-End These are the start and the end values of the spectrum in respect to the current position.

Points Number of points the spectrum will have. Please have a look at *Int entry* for additional informations.

Z-Slope Slope while collecting data.

Slope-Ramp Slope while just moving.

Final-Delay After running a spectrum this is the time the DSP waits to enable the feedback again.

Status Gives information about the ongoing spectrum:

Tp Total time the probe needs.

Informations about the check options can be found in STS.

23. Plug-Ins: hard

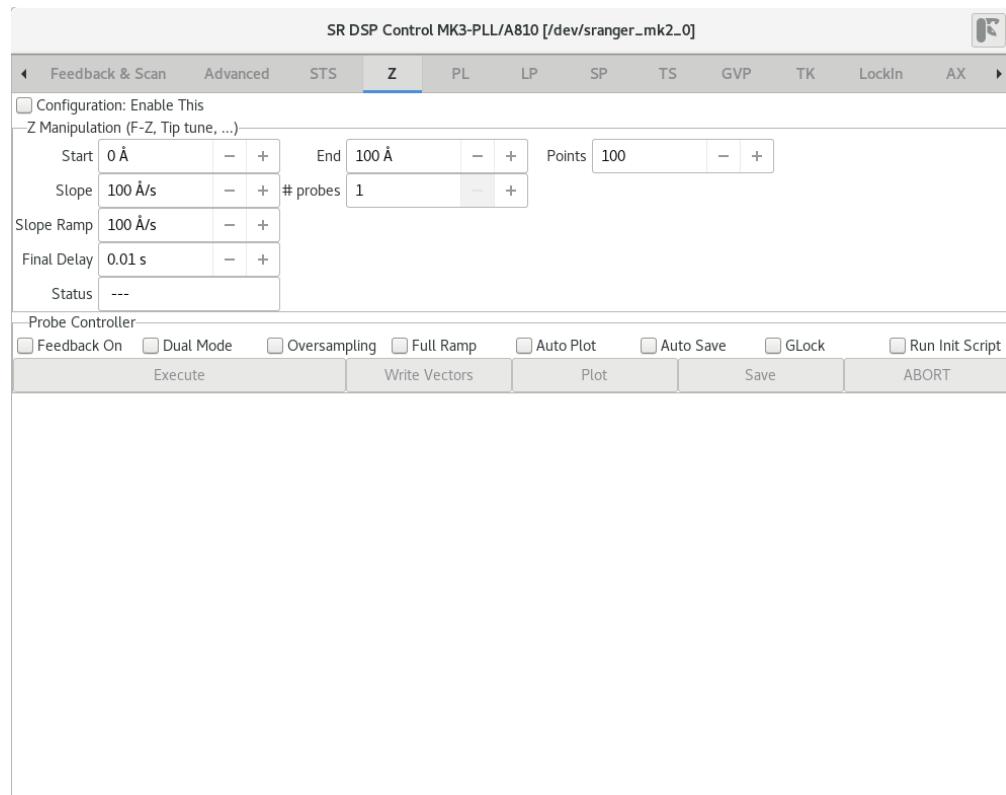


Figure 23.27.: GXSM SR DSP Control window, tab Z manipulation

Lateral Manipulation (LM)

With LM a lateral manipulation of the tip/sample is possible. But also the Z-dimension can be manipulated at the same time if dZ set to a non zero value.

dxyz Distance vector that will be covered.

Points While moving it is possible to collect data. Points defines the number of collected data points.

LM-Slope Speed of the tip/sample.

Final-Delay Timeout after lateral manipulation.

Status Gives information about the ongoing move.

Informations about the check options can be found in STS.

Pulse (PL) and Tip Forming or controlled crash Z manipulations

There are several possibilities to prepare or manipulate a tip or locale surface area. One is to dip the tip into the sample in a controlled manner, another option is applying a charge pulse or both combined.

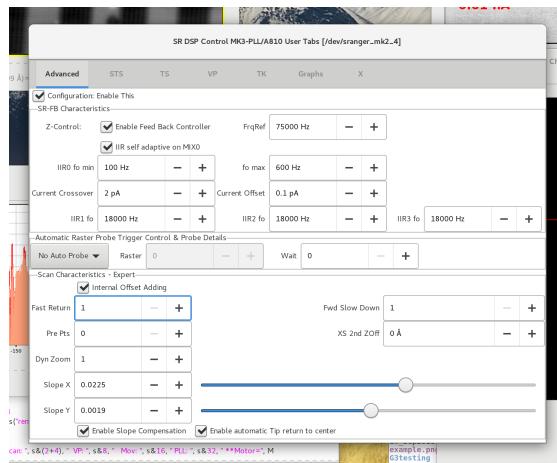


Figure 23.28.: SR-DSP-MK3-ADV-configure

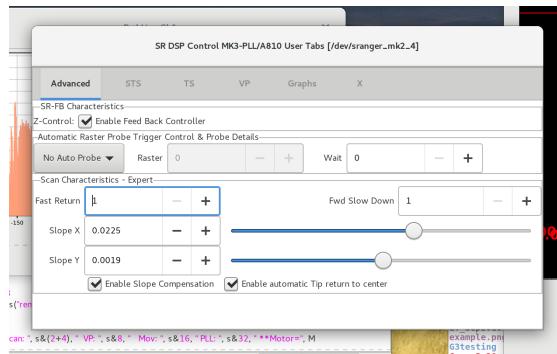


Figure 23.29.: SR-DSP-MK3-ADV-simple

23. Plug-Ins: hard

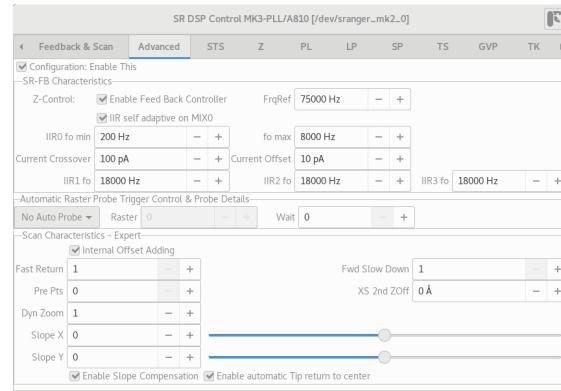


Figure 23.30.: SR-DSP-MK3-DSP-Advanced

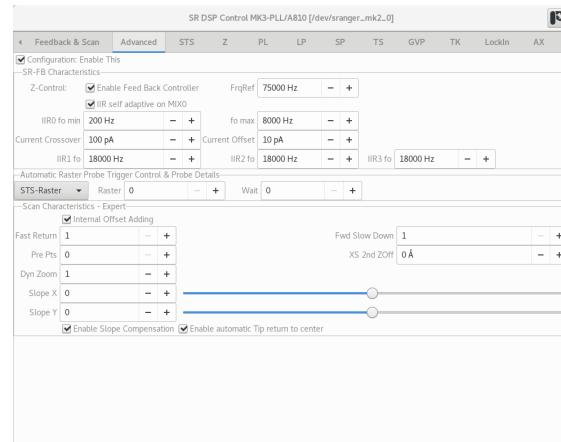


Figure 23.31.: SR-DSP-MK3-DSP-Advanced-Raster

23.9. Signal Ranger MK2/3-A810 Hardware Interface

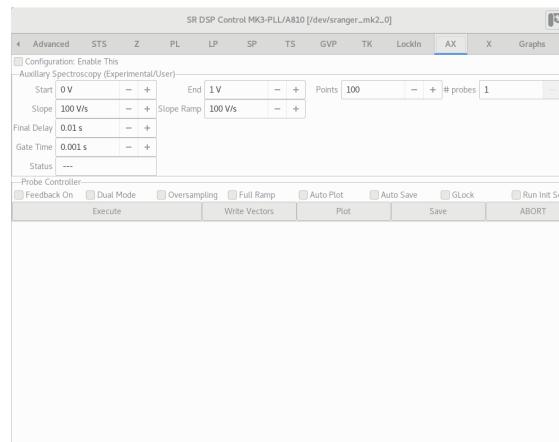


Figure 23.32.: SR-DSP-MK3-DSP-AX

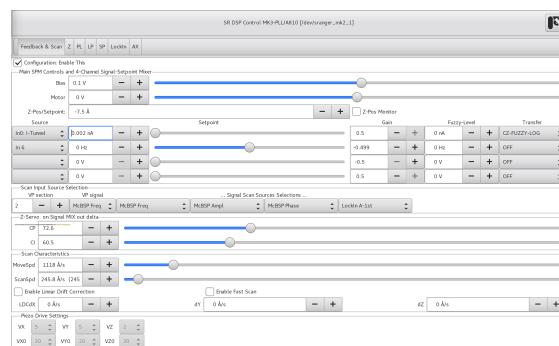


Figure 23.33.: SR-DSP-MK3-DSP-Feedback-Scan

23. Plug-Ins: hard

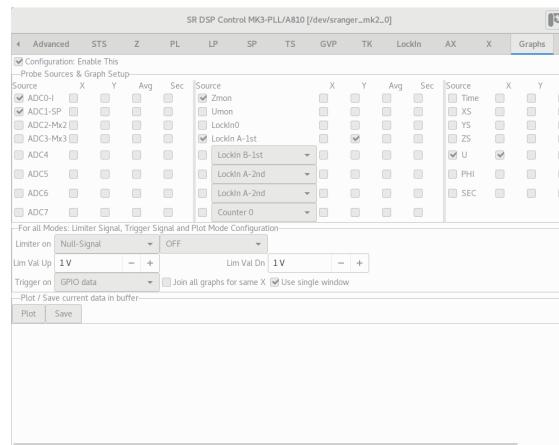


Figure 23.34.: SR-DSP-MK3-DSP-Graphs

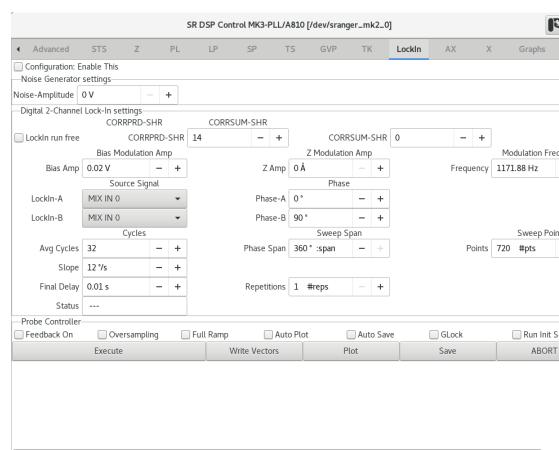


Figure 23.35.: SR-DSP-MK3-DSP-LockIn

23.9. Signal Ranger MK2/3-A810 Hardware Interface

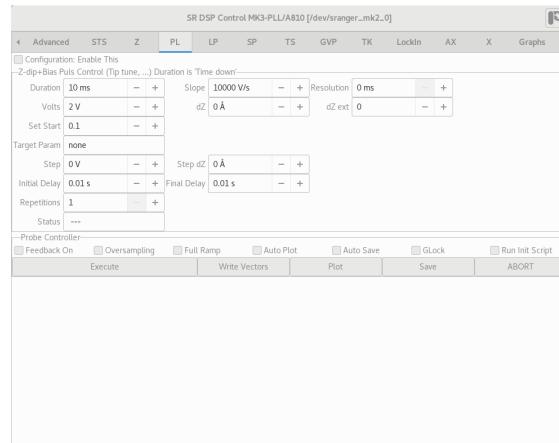


Figure 23.36.: SR-DSP-MK3-DSP-PL

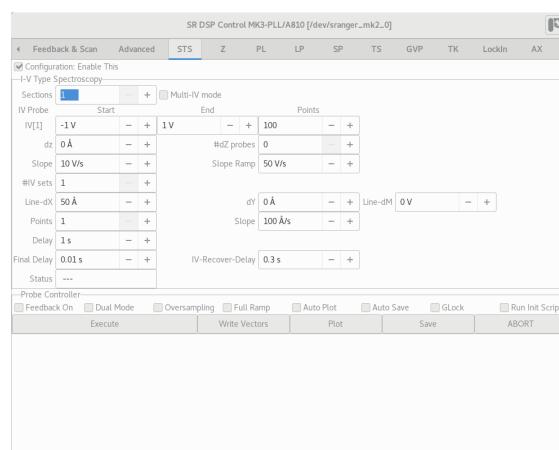


Figure 23.37.: SR-DSP-MK3-DSP-STS

23. Plug-Ins: hard

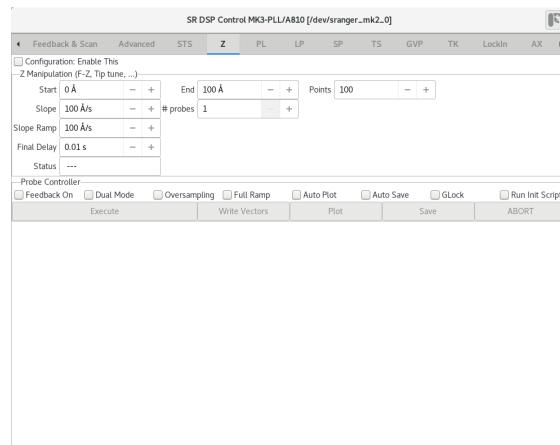


Figure 23.38.: SR-DSP-MK3-DSP-Z

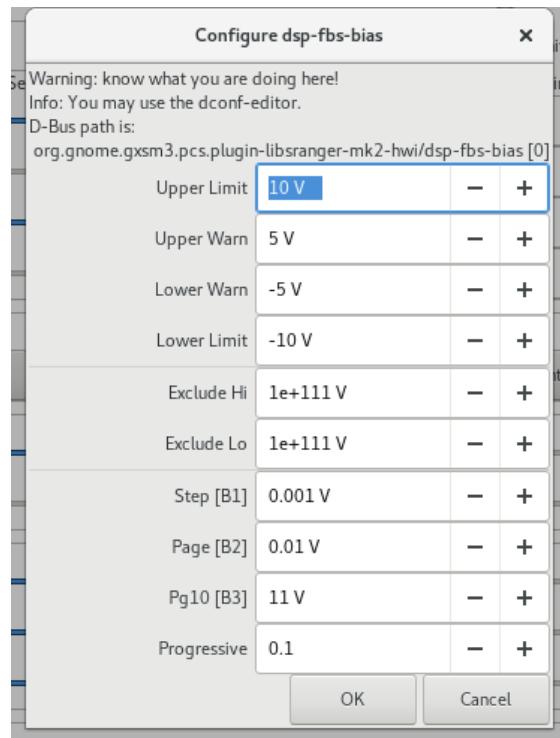


Figure 23.39.: SR-DSP-MK3-FBS-bias-log-configure

23.9. Signal Ranger MK2/3-A810 Hardware Interface

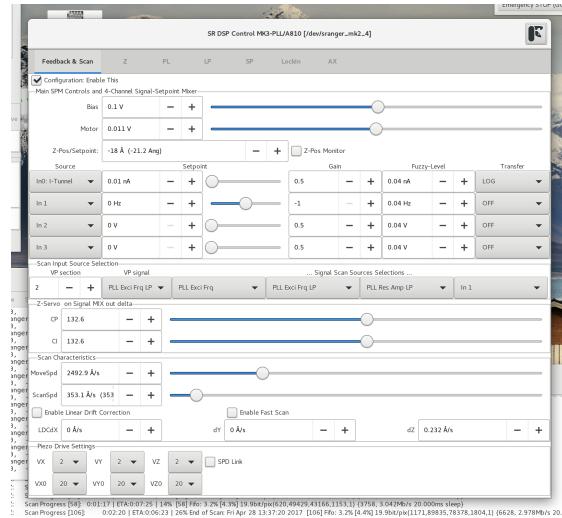


Figure 23.40.: SR-DSP-MK3-FBS-configure

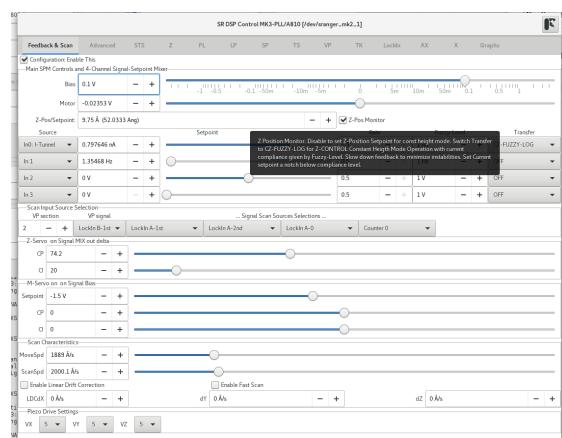


Figure 23.41.: SR-DSP-MK3-FBS-CZ-FUZZY-LOG-bias-log

23. Plug-Ins: hard

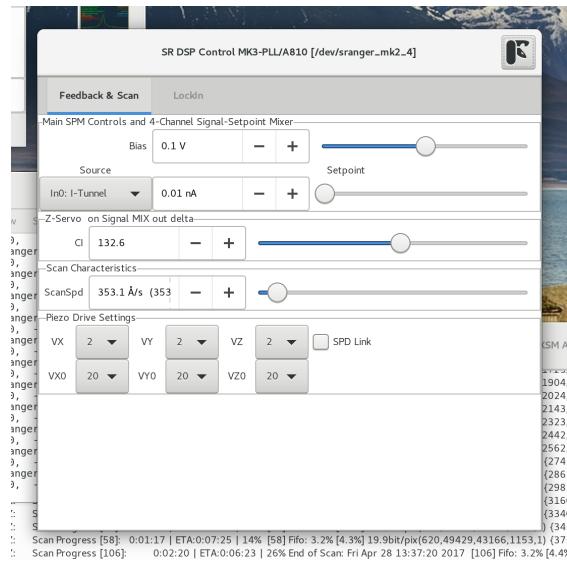


Figure 23.42.: SR-DSP-MK3-FBS-simple

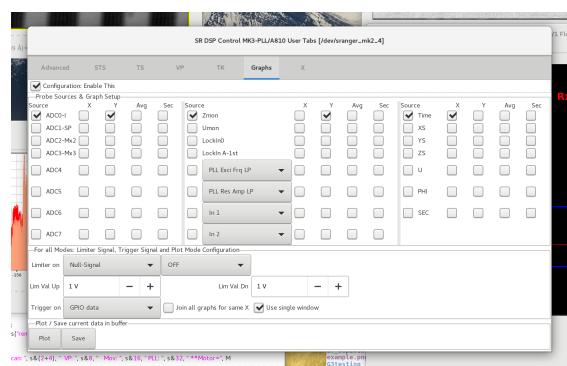


Figure 23.43.: SR-DSP-MK3-Graphs

23.9. Signal Ranger MK2/3-A810 Hardware Interface

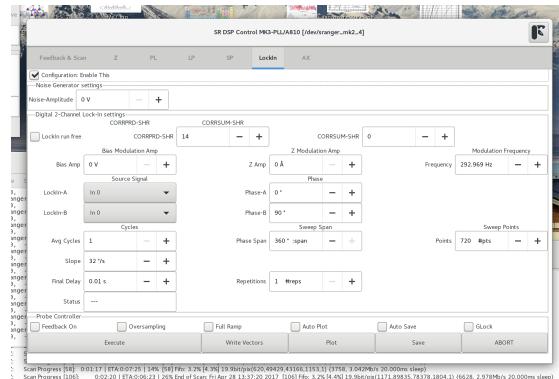


Figure 23.44.: SR-DSP-MK3-LockIn-configure

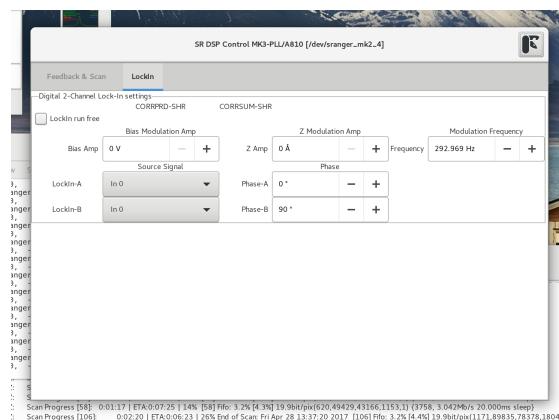


Figure 23.45.: SR-DSP-MK3-LockIn-simple

23. Plug-Ins: hard

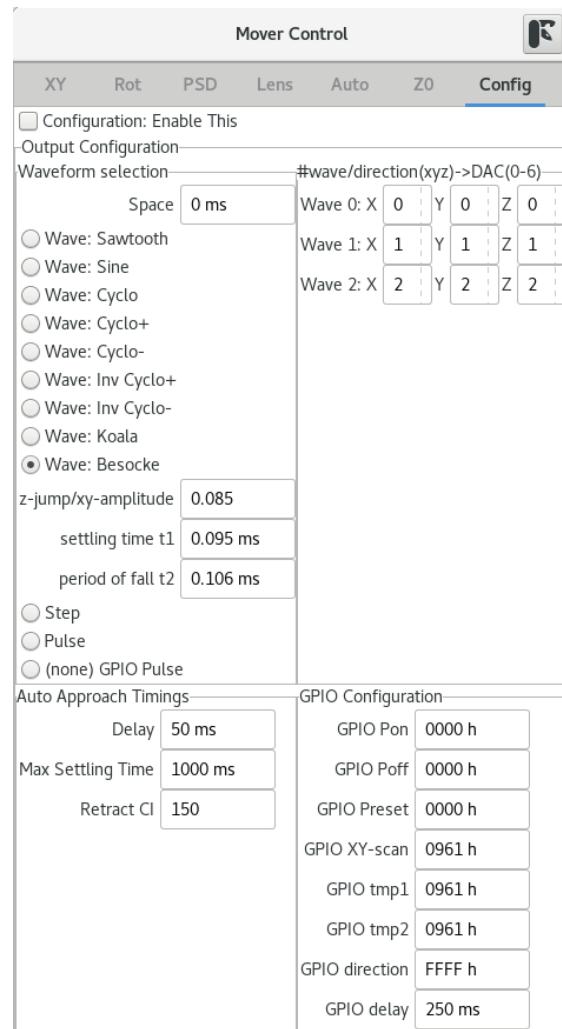


Figure 23.46.: SR-DSP-MK3-Mover-config

23.9. Signal Ranger MK2/3-A810 Hardware Interface

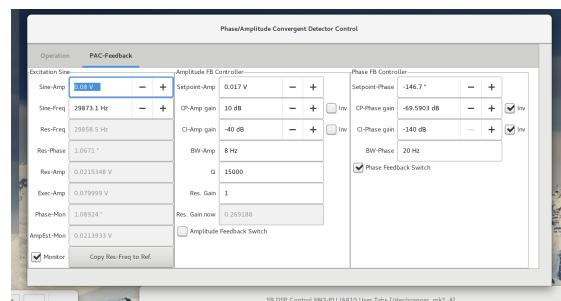


Figure 23.47.: SR-DSP-MK3-PAC-FB-Controllers

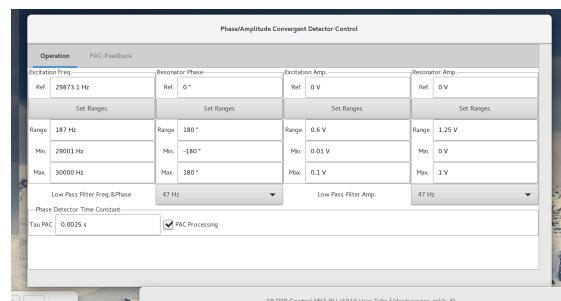


Figure 23.48.: SR-DSP-MK3-PAC-Operation

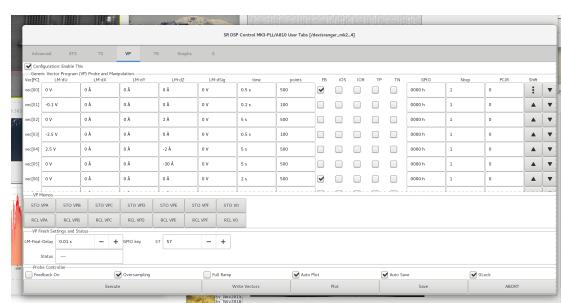


Figure 23.49.: SR-DSP-MK3-VP-configure

23. Plug-Ins: hard

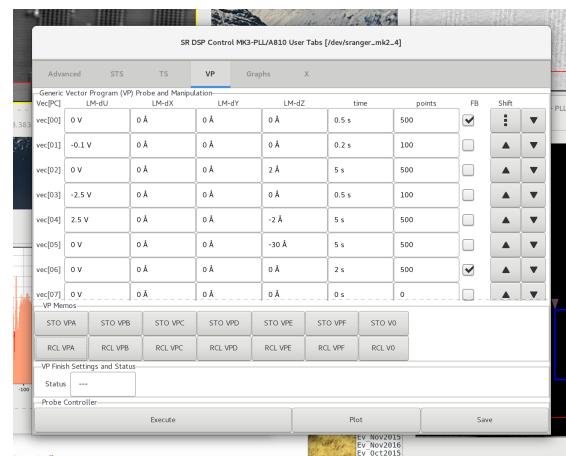


Figure 23.50.: SR-DSP-MK3-VP-simple

Duration Determines the duration of the pulse.

Slope Slope to reach *Volts and dZ entry*.

Volts Applied voltage.

dZ Gap or Z change.

Final Delay Delay for relaxing the I/V-converter after pulsing.

Repetitions How many pulses are applied.

Step,dZ Option to shift up/down every consecutive puls (for multiple repetitions)

Status Gives information about the ongoing pulse.

Informations about the check options can be found in STS.

Laser Pulse (LPC) user form

Slow Pulse (SP) user form

Time Spectrum (TS)

Tracking (TK) mode

Real time tracking of simple features like local maxima or minima in Topography (Z, feedback on), Current or other signal external sources on A810-IN1 or 2 (TK-ref option menu). This modes generated a special vector tracking sequence moving the tip starting at current tip position in circle (TK-nodes polygon (TK-points are used as step size each segment, moving at TK-Speed) with radius TK-rad, if TK-rad2 is > 0 it will use a double circle testing pattern) like motion. While moving on this pattern it will recognize the extrema (min/max) and if the gradient relative to the tips initial origin is up or down (as specified via TK-mode) it will relocate the tip and start over TK-Reps times.

Control of the digital (DSP) Lock-In

The Lock-In folder provides all settings concerning the build in digital Lock-In. The Lock-In and with it the bias modulation is automatically turned on if any Lock-In data channel is requested, either for probing/STS (in Graphs) or as a scan data source (Channelselector) for imaging.

There are a total of five digital correlation sums computed:

Averaged Input Signal (LockIn0),

Phase-A/1st order (LockIn1stA), Phase-B/1st order (LockIn1stB),

23. Plug-Ins: hard

Phase-A/2nd order (LockIn2ndA), Phase-B/2nd order (LockIn2ndB).



Please always select LockIn0 for STS.

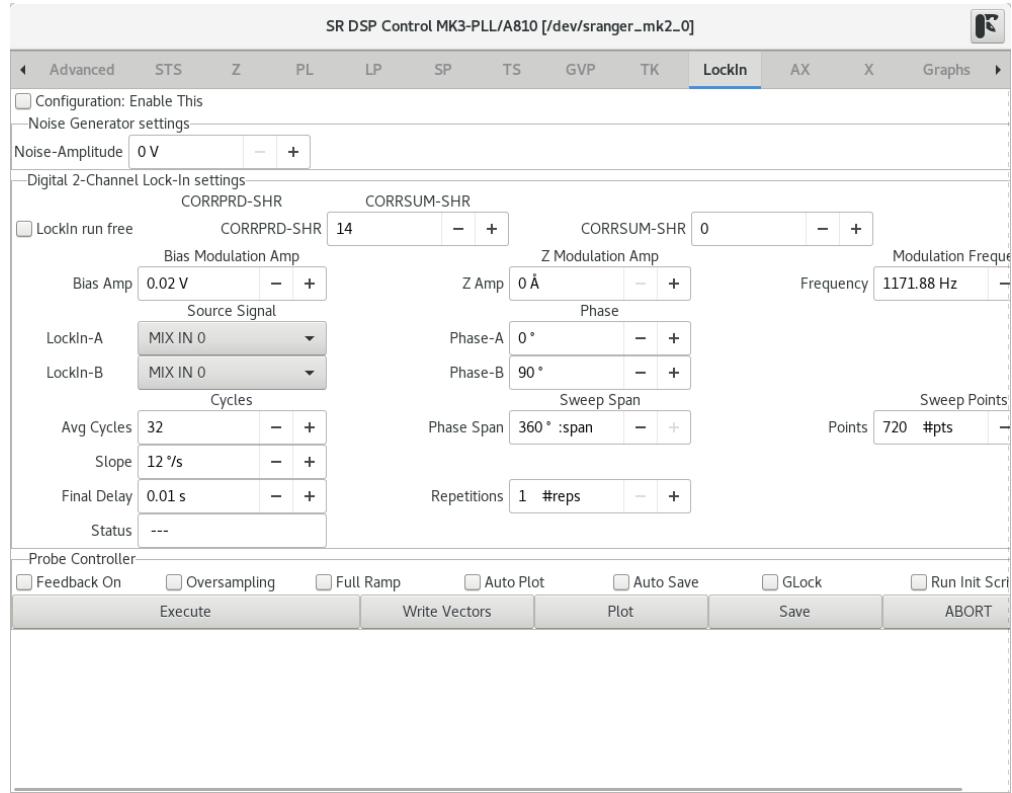


Figure 23.51.: GXSM SR DSP Control window: Lock-In settings and enabled configuration mode.

AC-Amplitude The amplitude of the overlaid Lock-In AC voltage.

AC-Frequency The base frequency of the Lock-In. There are four fixed frequency choices.

AC-Phase-AB Phase for A and B signal, applied for both, 1st and 2nd order.

AC-Avg-Cycles This sets the length for averaging, i.e. the corresponding time-constant.

For adjustments purpose only are the following parameters and the execute function here, not needed to run the Lock-In for all other modes. The special probe mode implemented in this section can actually sweep the phase of the Lock-In, it is useful to figure out the correct phase to use:

span Full phase span to sweep.

pts Number data points to acquire while phase sweep.

slope Phase ramp speed.

The digital Lock-In is restricted to a fixed length of base period (choices are 128, 64, 32, 16 samples/period with a fixed sample rate of 75000 samples/s) and a fixed number of 8 periods for computing the correlation sum: The total number of periods used for correlation of data can be increased by setting AC-Avg-Cycles greater than one, then overlapping sections of the 8 period long base window is used for building the correlation sum. Thus the total integration length (time constant) is

$$\tau = \frac{\text{AC-Ave-Cycels} \cdot 8}{\text{Frq}}$$

$$\text{Frq} = \frac{75000 \text{ Hz}}{M = 128, 64, 32, 16}$$

There for the following discrete frequencies are available: 585.9Hz, 1171.9Hz, 2343.7Hz, 4687.5Hz.

The four correlation sums for A/B/1st/2nd are always computed in parallel on the DSP if the Lock-In is enabled – regardless what data is requested. The correlation length is given by:

$$N = 128 \cdot \text{AC-Ave-Cycels} \cdot 8$$

$$\omega = 2\pi \cdot \text{Frq}$$

Lock-In data calculations and reference signal generation is all in digital regime on the DSP in real-time. The modulation is applied to the Bias voltage by default automatically only if the Lock-In is active:

$$U_{\text{ref}} = \text{AC-Amp} \cdot \sin(\omega t) + \text{Bias}$$

Averaged signal and Lock-In output signals calculated:

$$U_{\text{LockIn}0} = \sum_{i=0}^{N-1} U_{in,i}$$

$$U_{\text{LockIn}1\text{stA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i \frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn1stB}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i \frac{2\pi}{M} + \text{Phase-B})$$

$$U_{\text{LockIn2ndA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i \frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn2ndB}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i \frac{2\pi}{M} + \text{Phase-B})$$

 Note

Implemented in FB_spm_probe.c, run_lockin() (C) by P.Zahl 2002-2007.

 Note

All Lock-In data is raw summed in 32bit integer variables by the DSP, they are not normalized at this time and moved to GXSM-3.0 via FIFO. GXSM-3.0 applies the normalization before plotting.

Information about the check options can be found in STS.

Auxiliary Probe Control

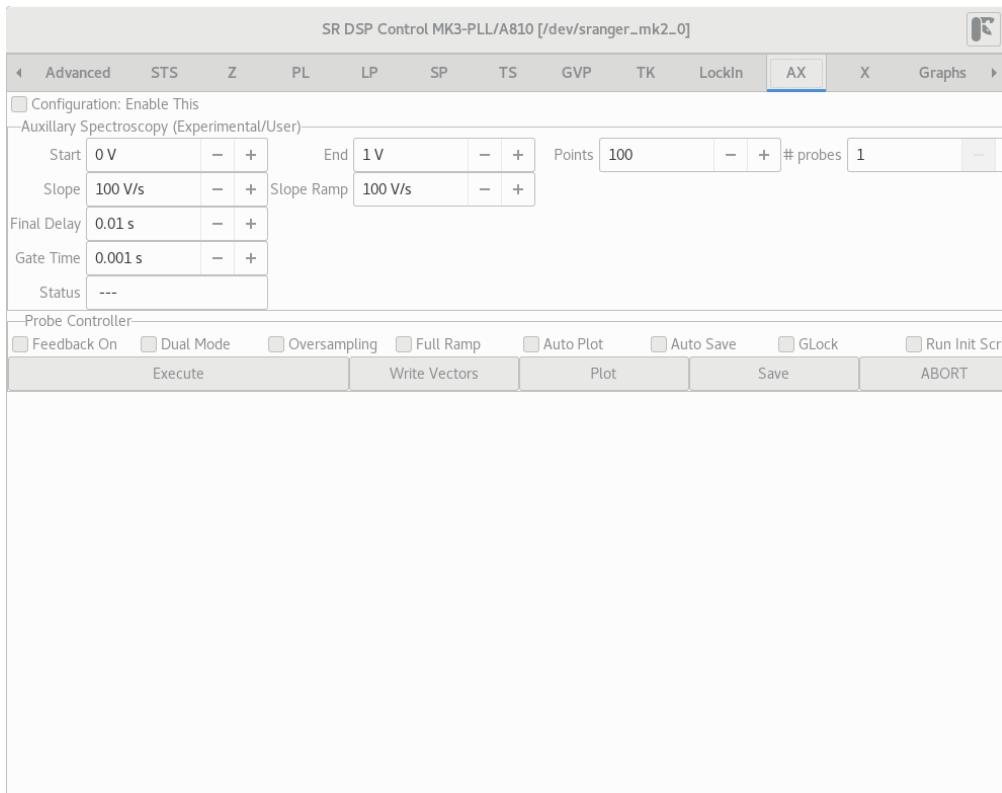


Figure 23.52.: GXSM SR DSP Control window: AX (Auxiliary) settings

This folder can be used for control and data acquisition from several kind of simple instruments like a QMA or Auger/CMA.

Best is to setup a new user for this instrument and configure the Bias-Gain so the “Voltage” corresponds to what you need. As input you can select any channel, including Lock-In and Counter. Here the Gate-Time is used to auto set the V-slope to match V-range and points.



Data Sources and Graphing Control

In the Graphs folder all available data channels are listed. If a Source is activated, measured data will be transferred into the buffer. Saving the buffer will automatically save all activated sources. Additionally it is possible to define a source as to be displayed.

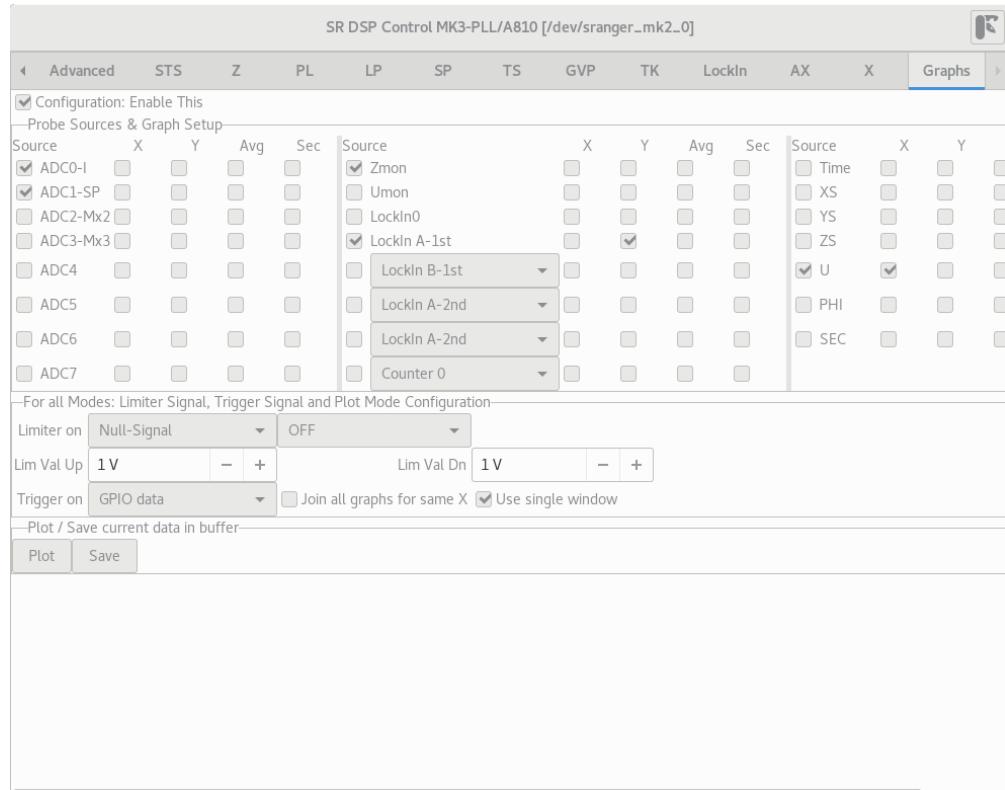


Figure 23.53.: GXSM SR DSP Control window, Graphs page: Plot and Data sources setup.



Beware: If a channel is not marked as a Source there will be no data to be displayed even if X or Y is checked.

23.9.3. SR-DSP Mover and Approach Control

GXSM with the SRanger also provides signal generation for slip-stick type slider/mover motions which are often used for coarse positioning and tip approach. Set *GXSM preferences dialog, folder User/User/SliderControlType* to *mover entry* to get the most configurable Mover Control dialog. If set to *slider entry* (default setting) the dialog will be simplified for Z/approach only. The different tabs are only for users convenience to store different speed/step values, the output will always occur as configured on the *Config entry folder*.

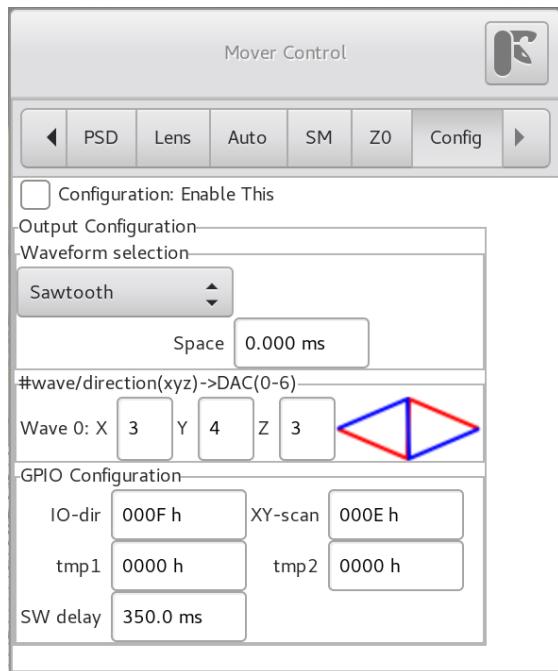


Figure 23.54.: SR-DSP-MoverConfig

To configure the mover output signal type and channels select the “Config” tab. Select the “Curve Mode”, normally a simple Sawtooth will do it. Then select the output configuration meeting your needs best. The MK3 “Signal Master” allows fully custom assignment of the generic “X” and “Y” mover actions to any available output channel. See below Wave[0,1] out on and Fig. 23.58.

More complex wave forms are available such as:

Wave: Sawtooth Just a simple linear ramp and then a jump back to the initial value.

Wave: Sine For testing and maybe some inch worm drive.

23. Plug-Ins: hard

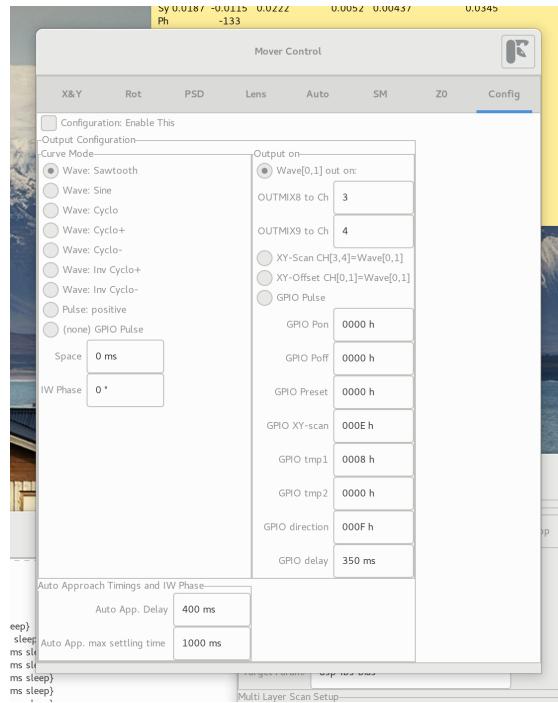


Figure 23.55.: SR-DSP-Mover-configure

Wave: Cyclo, Cyclo+, Cyclo-, Inv Cyclo+, Inv Cyclo- A cycloidal wave function (see

<http://dx.doi.org/10.1063/1.1141450>) should provide an even more abrupt change of the motion direction than the jump of the sawtooth and therefore work with lower amplitudes (or at lower temperatures) better. Depending on the option choosen, the signal is just limited to positive or negative values.

Wave: KOALA Intendet to be used with a KOALA drive (see <http://dx.doi.org/10.1063/1.3681444>). This wave form will require signal at two output channels with a phase shift of π . The wave form is shown in Fig. 23.59.

Wave: Besocke Intendet to be used with a Besocke style STM (see [https://doi.org/10.1016/0039-6028\(87\)90151-8](https://doi.org/10.1016/0039-6028(87)90151-8)): 3 piezos walk up and down a ramp. In this particular case, the piezos have three segments at their outer side (u, v, w). This coarse motion will require signals at three output channels. These signals vary for different directions of movement. The fundamental waveform is shown in Fig. 23.60. By an additional analog switch, which can be controlled by the GPIO ports, one can change between xy motion (translation) and z motion (rotation). The switch either routes

23.9. Signal Ranger MK2/3-A810 Hardware Interface

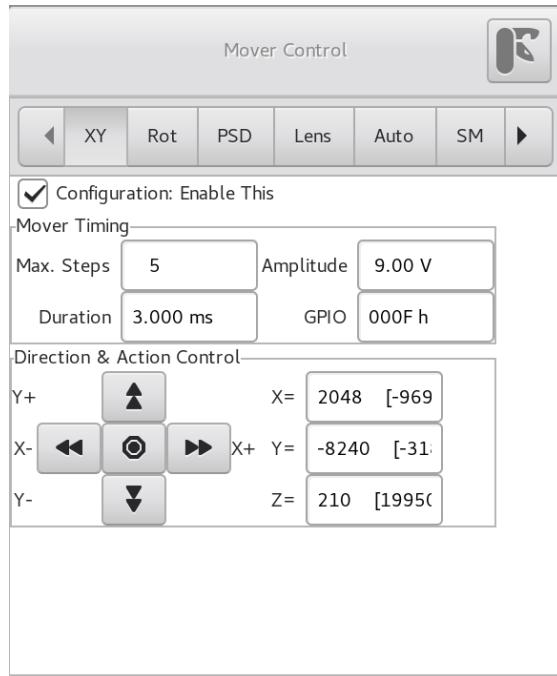


Figure 23.56.: SR-DSP-Mover-XY-configure

to the equivalent segments of the piezo the same signal (rotation) or projections on the three segments according to the 120° rotation between the three piezos (translation).

The time delay between the points A and B and also between D and E is named ‘settling time t1’, the time delay between B and C is ‘period of fall t2’. t2 should be shorter than the actual time for the scan head to fall back onto the ramp. If the z-jump corresponds to about $\Delta h = 1 \mu\text{m}$, the period of fall can be estimated according to the uniformly accelerated fall to be $t2 = \sqrt{\frac{\Delta h}{g}} \approx 0.44 \text{ ms}$.

These delays are both variable in the interface. The slip-stick amplitude is given by the voltage difference between C to D. The amplitude of the z-jump is defined as relative ratio to this value.

Pulse: positive Uses an analog channel to generate a simple on/off pulse similar to the GPIOs but you can controll the voltage range.

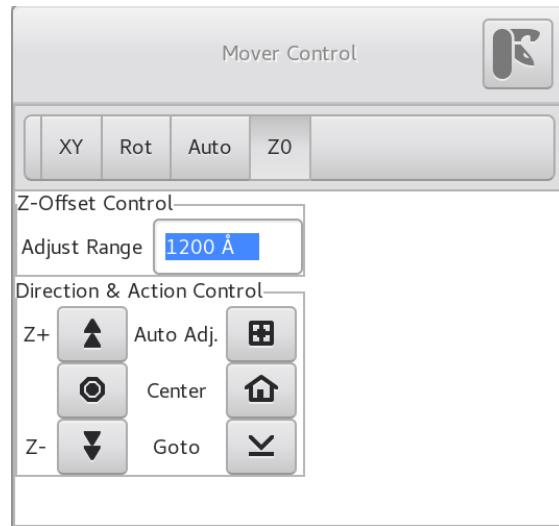


Figure 23.57.: SR-DSP-Mover-Z0-simple

Autoapproach

Main idea of autoapproach is to extract and retract scanner with tip (and enabled feedback) between each macro approach step to prevent destruction of the tip.

For correct use of wave outputs/coarse motion controls and autoapproach it is necessary to setup the output configuration and if needed the GPIO ports, which may control macro approach motor. In config tab is possible to set:

Wave(0,1) out on select the actual physical output channels used to output Wave X/Y vector. The defaults are: OUTMIX8 to Ch3 (normally X-Scan) and 9 to Ch4 (normally Y-Scan). You may alternatively also choose the pre set options below.

GPIO Pon Sets ports as logic 1 or 0 in point *e* - at the beginning of each pulse (see Fig. 23.61).

GPIO Poff Sets ports in point *f* -at the end of each pulse.

GPIO Preset Sets ports in point *g* - at the end of each pulse set.

GPIO XY-scan Sets ports while scanning.

GPIO tmp1,2 Sets ports while switching mode.

GPIO direction Sets ports as input or output pin. Logical 1 means output.

GPIO delay Sets time delay *d*.

All previous settings (except GPIO delay) must be set in hexadecimal format.

Autoapproach is stopped, when measured signal reaches setpoint value. This value is set in SR DSP Control window (SCurrent value).

In all tabs there is possible to set number of pulses in each step (input field *Max. Steps entry*). In the Fig. 23.61, it is marked with letter *a*. Length of each step (*c*) is determined by input field *Duration entry*.

 Note

23.9.4. SR-DSP Phase/Amplitude Convergent Detector (PAC) Control

GXSM with the SRanger MK3-Pro Phase/Amplitude Convergent Detector.... (PLL).

23. Plug-Ins: hard

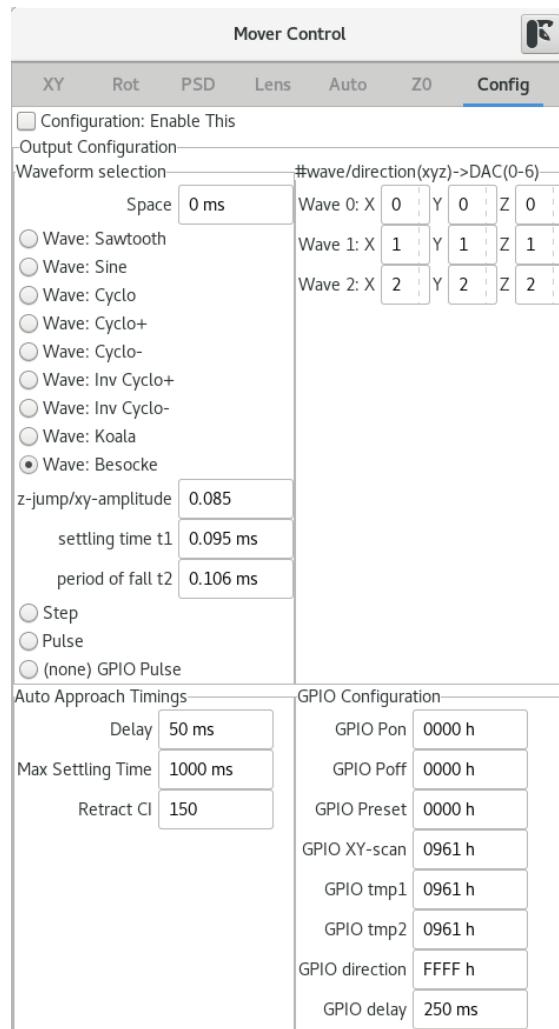


Figure 23.58.: GXSM SR DSP configuration of SRanger (MK2/3Pro) inertial mover driver wave generate engine.

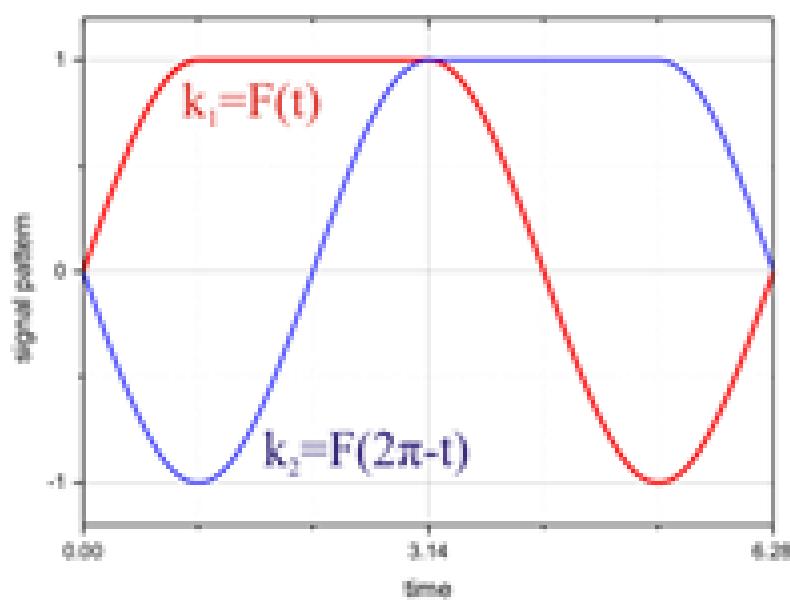


Figure 23.59.: Wave form used to operate a KOALA drive STM.

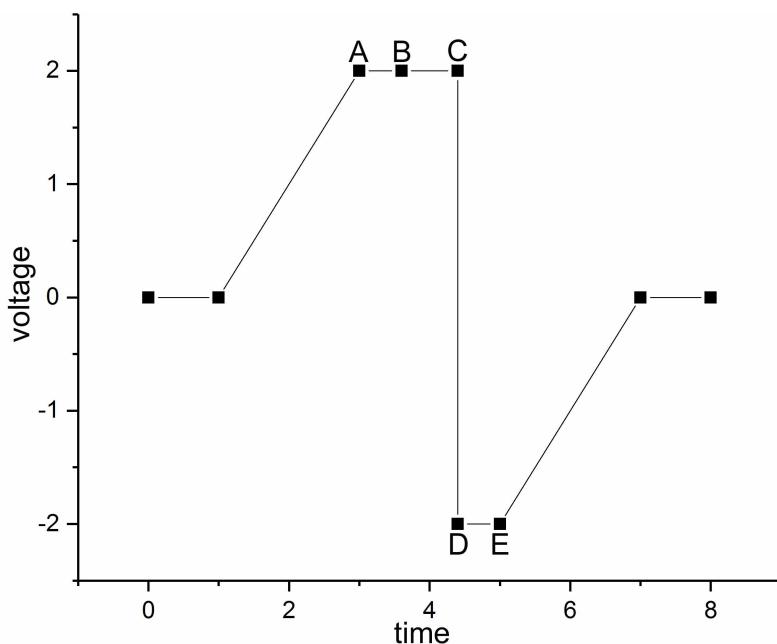


Figure 23.60.: Basic wave form used for Besocke drive STM.

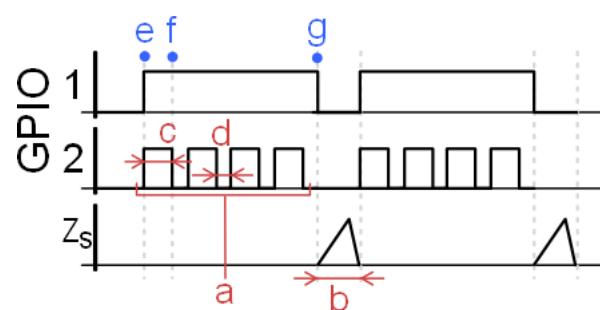


Figure 23.61.: Autoapproach signals. Z_s is scanning signal in Z axis.

Usage

Set the *GXSM preferences dialog, folder Hardware/Card* to "SRangerMK2:SPM".

The MK2 has to be programmed (Flashed) with the DSP software



/SRanger/TiCC-project-files/MK2-A810_FB_spmcontrol/FB_spmcontrol.out using the manufacturer supplied tools before it can be used with GXSM-3.0. Power cycle the MK2 after flashing and before plugging into the Linux system.

23.9.5. Setup – Gxsm Configurations for the MK2-A810.

Quick sample reference for the MK2-A810 specific GXSM-3.0 preferences.

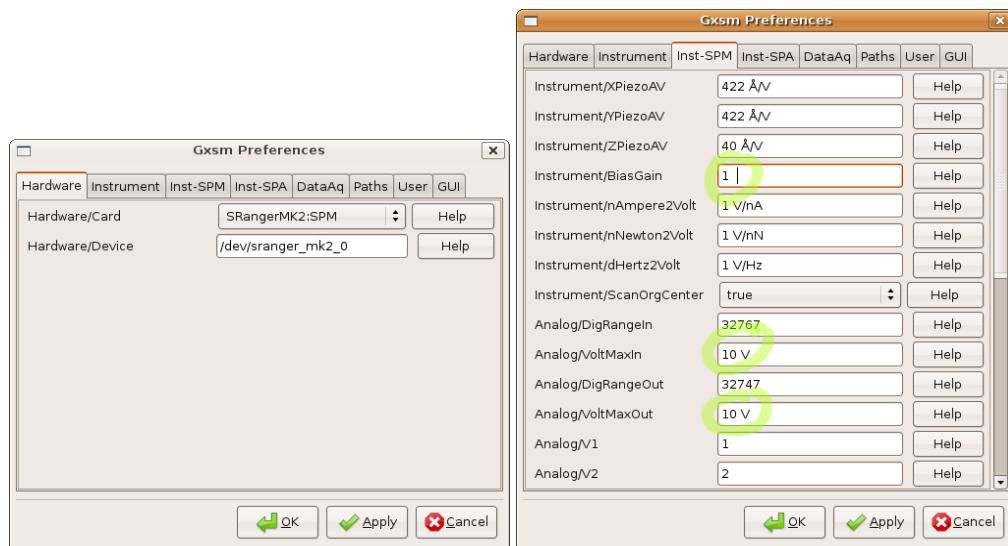


Figure 23.62.: left: MK2-A810 Hardware selection and device path setup., right: Instrument configuration

23. Plug-Ins: hard

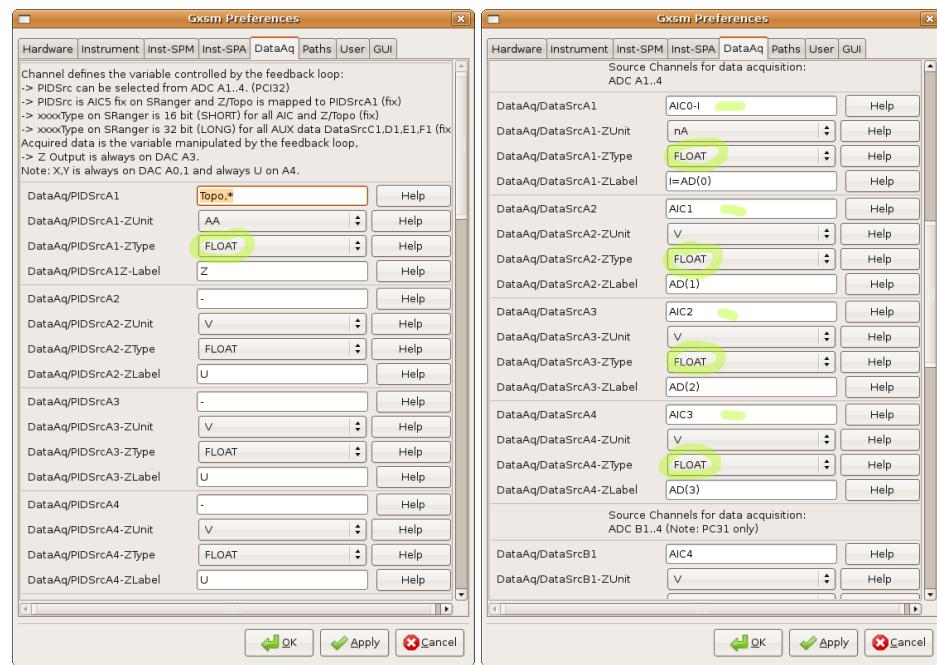


Figure 23.63.: left: Data Acquisition Channel configuration: Data Sources, right:

23.9. Signal Ranger MK2/3-A810 Hardware Interface

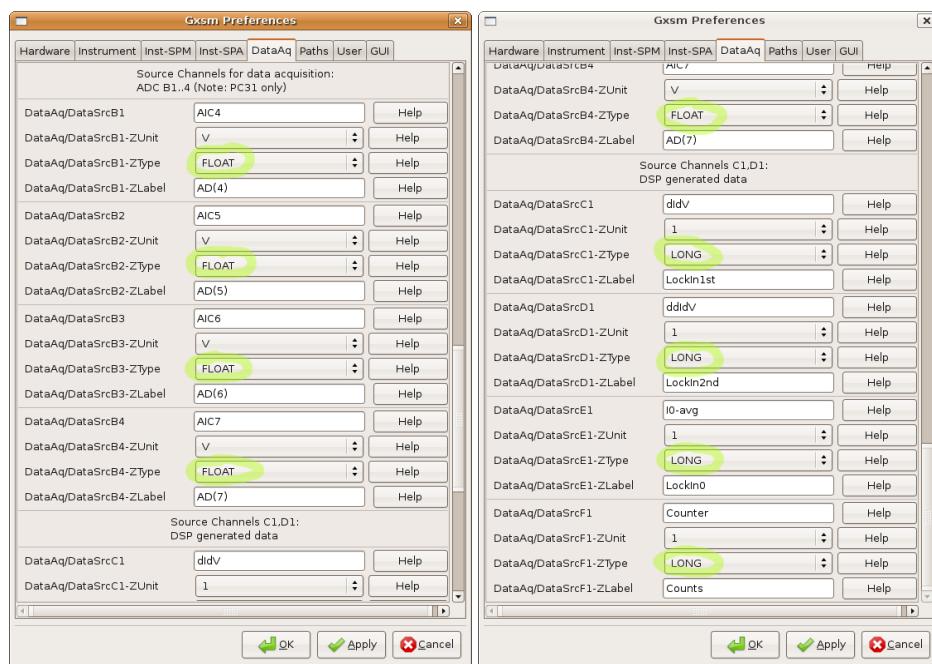


Figure 23.64.: left: Data Acquisition Channel configuration, more., right:

23.9.6. Sample STS IV,dIdV

Quick screen shot for a STS setup for taking IV and dIdV data at a point (current tip position). Example taken with home build Besocke style STM at room temperature, real data here!

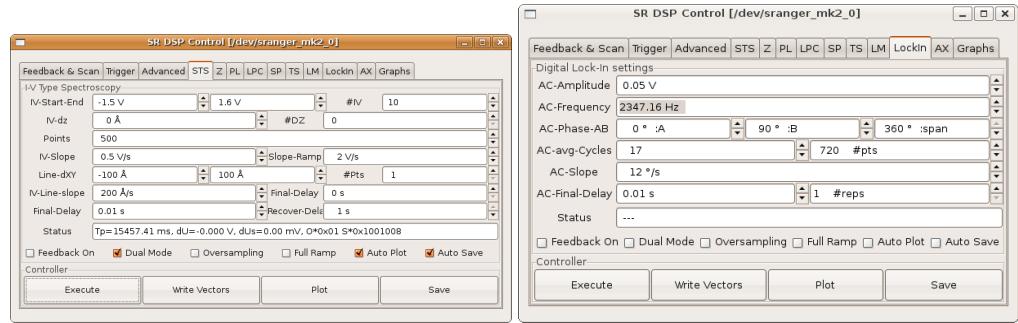


Figure 23.65.: left: STS Setup for IV and dIdV, right: Lock-In configuration used

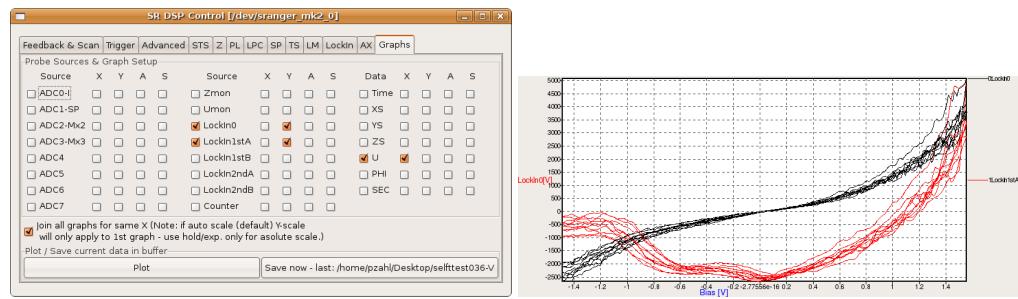


Figure 23.66.: left: Data channel selection and graphing configuration for this STS probe, right: Plot of the resulting data taking.

23.9.7. Sample STS along line demo setup

Quick screen shot for a STS setup along a line. Gxsm in self test configuration.

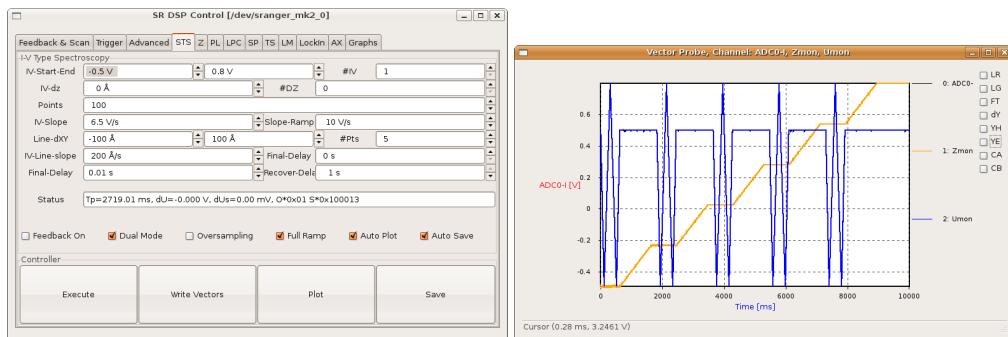


Figure 23.67.: left: STS Setup, right: Plotted Graphs

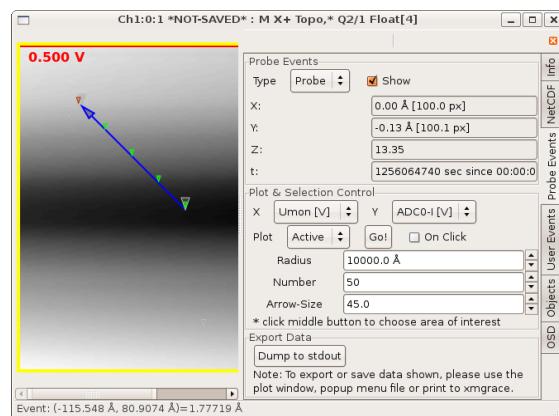


Figure 23.68.: Scan with markers

23.9.8. Signal Ranger MK3-Pro-A810/PLL Additional Features Guide

Introduction to a Open Signal Scheme

Adding more real time flexibility and the need to access even more data channels/signals – to manage a huge expansion of the “signal space”. See also the development background story about all this up front in the GXSM-3.0 preface .

Signals on the DSP are simply speaking just variables. Therefore, it is easy to manipulate them by connecting them to different source channels (i.e. ADC0-ADC7) or output channels (i.e. DAC0-DAC7). In this view, they are “hot plug-able” like a physical wire in your rack. They can be pulled from one connector and plug into another. But be aware, you have to know what you do. Secure the tip before trying new connections! Got me?

A typical STM configuration will look like the mostly default configuration as shown in Fig. 23.69. This live signal visualization is provided via the configuration application. Use the xdot viewer (the configuration application attempts to launch xdot for you automatically, if not present nothing will happen and only a signal_graph.dot and .svg output files are created. of better viewing, zooming and signal highlighting.

⚠ Warning

Please acknowledge and understand this message as you work with the “Signal Monitor” and “Patch Rack” tools: Due to the nature of a parallel execution of programs or multitasking on the host side and the graceful management of multiple processes talking to the DSP (MK3-Pro here) at the same time there can be “race” conditions occur under certain circumstances accessing the same DSP resources from two or more processes at the same time. While it’s guaranteed every process is receiving one completed data read/write access package of any size, so it can not be guaranteed that per process consecutive requests are consecutive for the DSP as any other processes request may be served at any time in between. This poses normally and per design no problem as long as different processes request/operate on independent DSP resources or tasks.

Now you must understand in this special case of the signal management the actual signal configuration of any signal can be requested and in a consecutive request read back as result. Both, GXSM-3.0 and the SPM configuration script are making use of this at times and it’s as of now the users responsibility to not run the script’s “Signal Monitor” or “Patch Rack” at the same time as GXSM-3.0 will make signal configuration read back requests. You must terminate (or at least temporary stop) any other signal requesting script/process at the following times when GXSM-3.0 makes signal read back requests: When GXSM-3.0 start up, Scan Start and Probe Start/Graphs Plot Refreshes. In general don’t run any not necessary scripts, but for sure you may want to run the script to make adjustments at initial system configuration/setup or tests or just to monitor signals via the Recorder (Signal Oscilloscope) or run a frequency/PLL tune – this

all is perfectly fine. It is not sufficient to close the “Patch Rack” or “Monitor” Window as this is only hiding the window, not destroying the tasks behind it. The “background task” requesting signals are installed at 1st time opening the window for the “Monitor” or “Patch Rack”. In future this problem may be controlled via spin-locks, but as of now this is a not essential overhead of USB communications.

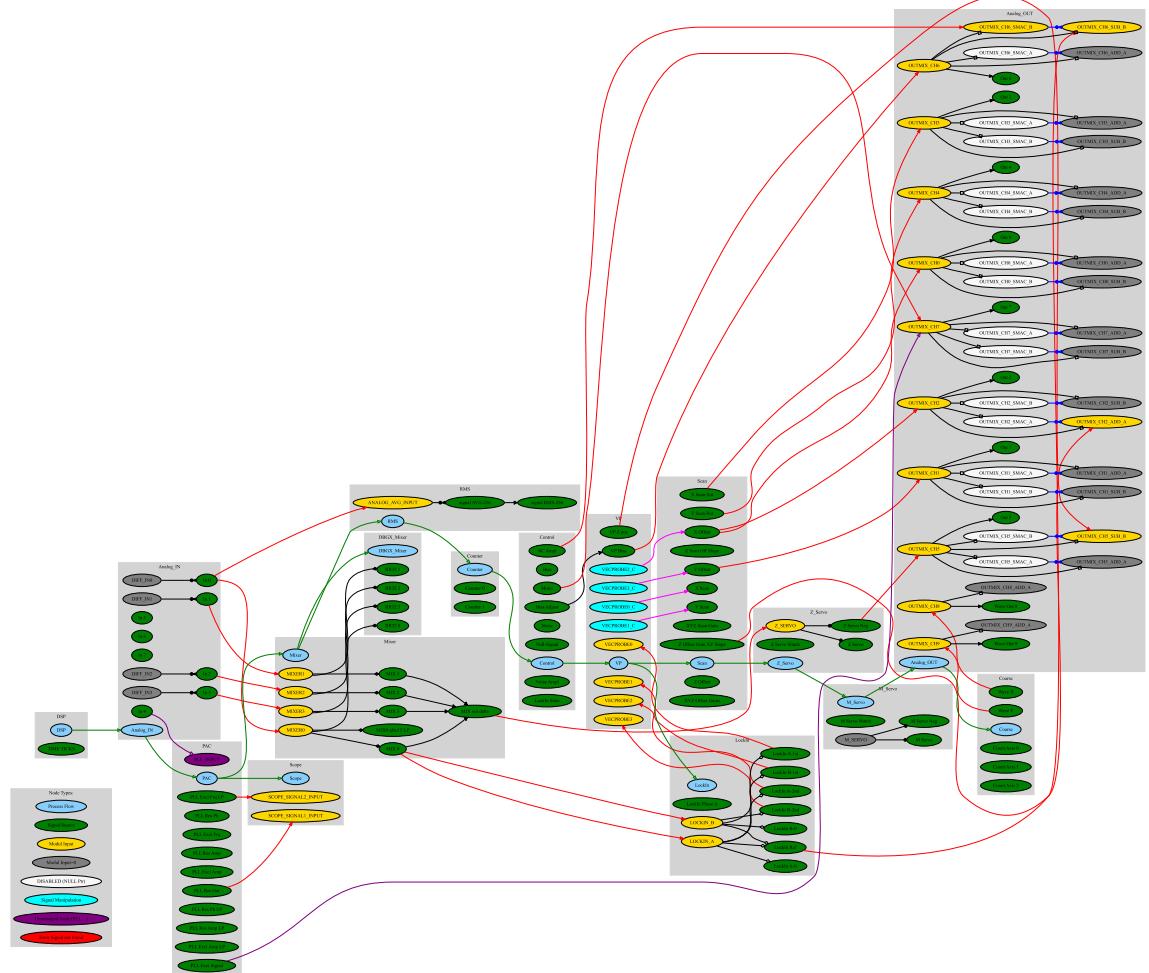


Figure 23.69.: Signal Graph for typical STM operation, as created by the tool and graphviz from the actual life DSP configuration.

Getting Started

Prerequisites: Having build and installed the latest code including a updated DSP MK3. Locate the configuration application (python script) in the GXSM-3.0 source code tree. Also you may want to have the dot file viewer “xdot” installed for convenient automatic viewing of the signal configuration with important signal highlighting ability. As those graphs may have many possibly overlapping signals.

Signal Monitoring and Configuration

The python application suite shown in Fig. 23.70

`plug-ins/hard/MK3-A810_spmcontrol/python_scripts/mk3_spm_configurator.py`
provides monitoring and configuration tools including the PLL tune application and step response test tools not directly provided via the GXSM-3.0 GUI itself.



Figure 23.70.: SPM Configurator Python Apps

The actual readings of the two banks of each 8 AD and DA converters provided by the A810 expansion board can be monitored via the A810 AD/DA Monitor App as shown in Fig. 23.71.

A subset of all signals can be configured via the Signal-Monitor for watching. In general for testing and debugging any signal can be selected at any monitor slot as shown in Fig. 23.72. But for normal operation GXSM-3.0 and als the SmartPiezoDrive (see 23.9.9) are automatically assigning Signal for efficient monitoring use and without in depth knowledge those shall not be altered after GXSM-3.0 start up.

23. Plug-Ins: hard

Type	Dim	DSP Variable	Signal Name	Unit	Conversion Factor	Module	Signal Description
SI32	1	analog_in[0]	In 0	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT1
SI32	1	analog_in[1]	In 1	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT2
SI32	1	analog_in[2]	In 2	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT3
SI32	1	analog_in[3]	In 3	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT4
SI32	1	analog_in[4]	In 4	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT5
SI32	1	analog_in[5]	In 5	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT6
SI32	1	analog_in[6]	In 6	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT7
SI32	1	analog_in[7]	In 7	V	DSP32Qs15dot16TO_Volt	Analog_IN	ADC INPUT8
SI32	1	analog_counter[0]	Counter 0	CNT	1	FPGA based Counter Channel 1	
SI32	1	analog_counter[1]	Counter 1	CNT	1	FPGA based Counter Channel 2	
SI32	1	probe.LockIn_0A	LockIn A-0	*V	DSP32Qs15dot16TO_Volt	LockIn	LockIn A 0 (average over full periods)
SI32	1	probe.LockIn_1stA	LockIn A-1st	*dV	DSP32Qs15dot16TO_Volt	LockIn	LockIn A 1st order
SI32	1	probe.LockIn_2ndA	LockIn A-2nd	*dV	DSP32Qs15dot16TO_Volt	LockIn	LockIn A 2nd order
SI32	1	probe.LockIn_B-0B	LockIn B-0	*dV	DSP32Qs15dot16TO_Volt	LockIn	LockIn B 0 (average over full periods)
SI32	1	probe.LockIn_B-1stB	LockIn B-1st	*dV	DSP32Qs15dot16TO_Volt	LockIn	LockIn B 1st order
SI32	1	probe.LockIn_B-2ndB	LockIn B-2nd	*dV	DSP32Qs15dot16TO_Volt	LockIn	LockIn B 2nd order
SI32	1	probe.LockIn_Ref	LockIn Ref	V	DSP32Qs15dot16TO_Volt	LockIn	LockIn Reference Sinewave (Modulation) (Internal Reference Signal)
SI32	1	probe.PRB_ACPHaseA32	LockIn Phase A	deg	180./((29/13)*CPN(16))	PAC	DSP internal Lockin PhaseA32 watch
SI32	1	InputFiltered	PLL_Res_Out	V	10./CPN(22)	PAC	Resonator Output Signal
SI32	1	SineOut	PLL_Exci_Signal	V	10./CPN(22)	PAC	Excitation Signal
SI32	1	phase	PLL_Res_Ph	deg	(180./((CPN(22)*M_PI))	PAC	Resonator Phase (no LP filter)
SI32	1	PL_Phase_Out	PLL_Exci_Freq	Hz	(10./CPN(22))	PAC	Excitation Freq. (no LP filter)
SI32	1	amp_estimation	PLL_Res_Amp	V	10./CPN(22)	PAC	Resonator Amp. (no LP filter)
SI32	1	volumSine	PLL_Exci_Amp	V	10./CPN(22)	PAC	Excitation Amp. (no LP filter)
SI32	1	Filter64Qout[0]	PLL_Exci_Freq_LP	Hz	(50000./((CPN(29)*2.*M_PI))	PAC	Excitation Freq. (with LP filter)
SI32	1	Filter64Qout[1]	PLL_Res_Ph_LP	V	(180./((CPN(29)*M_PI))	PAC	Resonator Phase (with LP filter)
SI32	1	Filter64Qout[2]	PLL_Res_Amp_LP	V	(0./((CPN(29)))	PAC	Excitation Amp. (with LP filter)
SI32	1	Filter64Qout[3]	PLL_Exci_Amp_LP	V	(0./((CPN(29))))	PAC	Mixer Channel 0 processed input signal
SI32	1	feedback_mixerFB_IN_processed[0]	DSP32Qs23dot08TO_Volt	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 0 processed input signal
SI32	1	feedback_mixerFB_IN_processed[1]	MIX_IN_1	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 1 processed input signal
SI32	1	feedback_mixerFB_IN_processed[2]	MIX_IN_2	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 2 processed input signal
SI32	1	feedback_mixerFB_IN_processed[3]	MIX_IN_3	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 3 processed input signal
SI32	1	feedback_mixerchannel[0]	MIX_OUT0	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 0 output signal
SI32	1	feedback_mixerchannel[1]	MIX_OUT1	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 1 output signal
SI32	1	feedback_mixerchannel[2]	MIX_OUT2	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 2 output signal
SI32	1	feedback_mixerchannel[3]	MIX_OUT3	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Channel 3 output signal
SI32	1	feedback_mixerdelta	MIX_out_delta	V	DSP32Qs23dot08TO_Volt	Mixer	Mixer Processed Summed Error Signal (Delta)/Z-Servo In normally)
SI32	1	feedback_mixerq_factor15	MIXO_qfac15_LP	Q	((CPN(15)))	Mixer	Mixer Channel 0 actual I/IIR cutoff watch; q LP fg, fin
SI32	1	analog_avg_signal	signal_AVG-256	V	DSP32Qs15dot00TO_Volt/256.	RMS	Hz via: (-log (q/f15) / 32767.) / (2.*M_PI*75000.)
SI32	1	analog rms signal	signal RMS-256	V2	(DSP32Qs15dot00TO_Volt/256.)	RMS	Averaged signal from Analog AVG module
SI32	1	Z_servo.control	Z_Servo	V	DSP32Qs15dot00TO_Volt	Z_Servo	RMS signal from Analog AVG module
SI32	1	Z_servo.neg_control	Z_Servo_Neg	V	DSP32Qs15dot00TO_Volt	Z_Servo	Z_Servo output
SI32	1	Z_servo.watch	Z_Servo_Watch	B	DSP32Qs15dot00TO_Volt	Z_Servo	Z_Servo status(boolean)
SI32	1	m_servo.control	M_Servo	V	DSP32Qs15dot00TO_Volt	M_Servo	M_Servo output
SI32	1	m_servo.watch	M_Servo_Watch	B	1	M_Servo status(boolean)	

Table 23.1.: MK3 DSP Signal Description

23.9. Signal Ranger MK2/3-A810 Hardware Interface

Type	Dim	DSP Variable	Signal Name	Unit	Conversion Factor	Module	Signal Description
SI32	1	probe.Upos	VP Bias	V	DSPI32Qs15dot16TO_Volt	VP	Bias after VP manipulations
SI32	1	probe.Zpos	VP Zpos	V	DSPI32Qs15dot16TO_Volt	VP	temp Z offset generated by VP program
SI32	1	scan.xyz_vec[i,X]	X Scan	V	DSPI32Qs15dot16TO_Volt	Scan	X-Scan signal
SI32	1	scan.xyz_vec[i,Y]	Y Scan	V	DSPI32Qs15dot16TO_Volt	Scan	Scan generator: Y-Scan signal
SI32	1	scan.xyz_vec[i,Z]	Z Scan	V	DSPI32Qs15dot16TO_Volt	Scan	Scan generator: Z-Scan signal (**unused)
SI32	1	scan.xy_r_vec[i,X]	X Scan Rot	V	DSPI32Qs15dot16TO_Volt	Scan	final X-Scan signal in rotated coordinates
SI32	1	scan.xy_r_vec[i,Y]	Y Scan Rot	V	DSPI32Qs15dot16TO_Volt	Scan	final Y-Scan signal in rotated coordinates
SI32	1	scan.z_offset_xy_slope	Z Offset from XY Slope	V	DSPI32Qs15dot16TO_Volt	Scan	Scan generator: Z-offset generated by slope compensation calculation (integrative)
SI32	1	scan.xyz_gain	XYZ Scan Gain	X	1	XYTZ_Scan_Gains: bitcoded -78/88 (0..255)x - 10x all: 0x000a0a0a	Scan
SI32	1	move.xyz_vec[i,X]	X Offset	V	DSPI32Qs15dot16TO_Volt	Scan	Offset Move generator: X-Offset signal
SI32	1	move.xyz_vec[i,Y]	Y Offset	V	DSPI32Qs15dot16TO_Volt	Scan	Offset Move generator: Y-Offset signal
SI32	1	move.xyz_vec[i,Z]	Z Offset	V	DSPI32Qs15dot16TO_Volt	Scan	XYTZ_Offset_Gains: bitcoded -78/88 (0..255)x - not yet used and fixed set to 10x (0x000ada0a)
SI32	1	move.xyz_gain	XYZ Offset Gains	X	1	XYTZ_Offset_Gains: bitcoded -78/88 (0..255)x - not yet used and fixed set to 10x (0x000ada0a)	Scan
SI32	1	analog_wave[0]	Wave X	V	DSPI32Qs15dot0TO_Volt	Coarse	Wave generator: Wave-X (coarse motions)
SI32	1	analog_wave[1]	Wave Y	V	DSPI32Qs15dot0TO_Volt	Coarse	Wave generator: Wave-Y (coarse motions)
SI32	1	autoapp.count_axis[0]	Count Axis 0	I	1	Coarse Step Counter Axis 0 (X)	Coarse
SI32	1	autoapp.count_axis[1]	Count Axis 1	I	1	Coarse Step Counter Axis 1 (Y)	Coarse
SI32	1	autoapp.count_axis[2]	Count Axis 2	I	1	Coarse Step Counter Axis 2 (Z)	Coarse
SI32	1	analog_bias	Bias	V	DSPI32Qs15dot16TO_Volt	DSP	Bias Voltage reference following smoothly the Bias Adjuster
SI32	1	analog_bias_adjust	Bias Adjust	V	DSPI32Qs15dot16TO_Volt	Control	Bias Adjuster (Bias Voltage) setpoint given by user interface
SI32	1	analog_motor	Motor	V	DSPI32Qs15dot16TO_Volt	Control	Motor Voltage (auxiliary output shared with P.I.L. excitation if PAC processing is enabled!)
SI32	1	analog_noise	Noise	I	1	White Noise Generator	Control
SI32	1	analog_vnull	Null-Signal	0	1	Null Signal used to fix any module input at Zero	Control
SI32	1	probe.AC_amp	AC Amp	V	DSPI32Qs15dot0TO_Volt	AC	AC Amplitude Control for Bias modulation
SI32	1	probe.AC_amp_aux	AC Amp Aux	V	DSPI32Qs15dot0TO_Volt	AC	AC Amplitude Control (auxiliary channel or as default used for Z modulation)
SI32	1	probe.noise_amp	Noise Ampl	V	DSPI32Qs15dot0TO_Volt	Noise	Noise Amplitude Control
SI32	1	probe.state	LockIn State	X	1	LockIn Status watch	Control
SI32	1	state.DSP_time	TIME TICKS	S	1/150000.	DSP	DSP TIME TICKS; real time DSP time based on 150kHz data processing loop for one tick, 32bit free running
SI32	1	feedback_mixer_ir.signal[0]	IIR32.0	V	DSPI32Qs15dot16TO_Volt	DBGX_Mixer	Mixer processed input tap 1 12bit
SI32	1	feedback_mixer_ir.signal[1]	IIR32.1	V	DSPI32Qs15dot16TO_Volt	DBGX_Mixer	Mixer processed input tap 2 12bit
SI32	1	feedback_mixer_ir.signal[2]	IIR32.2	V	DSPI32Qs15dot16TO_Volt	DBGX_Mixer	Mixer processed input tap 3 12bit
SI32	1	feedback_mixer_ir.signal[3]	IIR32.3	V	DSPI32Qs15dot16TO_Volt	DBGX_Mixer	Mixer processed input tap 3 32bit

Table 23.2.: MK3 DSP Signal Description continued

23. Plug-Ins: hard

Type	Dim	DSP Variable	Signal Name	Unit	Conversion Factor	Module	Signal Description
SI32	1	analog_out[0].s	Out 0	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 1
SI32	1	analog_out[1].s	Out 1	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 2
SI32	1	analog_out[2].s	Out 2	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 3
SI32	1	analog_out[3].s	Out 3	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 4
SI32	1	analog_out[4].s	Out 4	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 5
SI32	1	analog_out[5].s	Out 5	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 6
SI32	1	analog_out[6].s	Out 6	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 7
SI32	1	analog_out[7].s	Out 7	V	DSP32Qs15dot16TO_Volt	Analog_OUT	DAC OUTPUT 8
SI32	1	analog_out[8].s	Wave Out 8	V	DSP32Qs15dot07TO_Volt	Analog_OUT	VIRTUAL OUTPUT 8 (Wave X default)
SI32	1	analog_out[9].s	Wave Out 9	V	DSP32Qs15dot07TO_Volt	Analog_OUT	VIRTUAL OUTPUT 9 (Wave Y default)
SI32	1	state.mode	X	-	-	Process_Control	DSP statemachine status
SI32	1	move.pflg	X	-	-	Process_Control	Officer Move generator process flag
SI32	1	scan.pflg	X	-	-	Process_Control	Scan generator process flag
SI32	1	probe.pflg	X	-	-	Process_Control	Vector Probe (VP) process flag
SI32	1	autoapp.pflg	X	-	-	Process_Control	Auto Approach process flag
SI32	1	CR_generic_io.pflg	X	-	-	Process_Control	Generic IO process flag
SI32	1	CR_out_pulse.pflg	X	-	-	Process_Control	IO Pulse generator process flag
SI32	1	probe.gpio_data	X	-	-	Process_Control	GPIO data-in is read via VP if GPIO READ option is enabled
VI32	64	VP_sec_end_buffer[0]	"VP Sec V"	"XV"	DSP32Qs15dot16TO_Volt	"VP"	"VP" section data transfer buffer vector [64 = 8X Sec + 8CH matrix]

Table 23.3.: MK3 DSP Signal Description continued. With

DSP32Qs15dot07TO_Volt := 10/32767,

DSP32Qs15dot16TO_Volt := 10/(32767 * 2⁸),

CPN(N) := 2^N – 1

23.9. Signal Ranger MK2/3-A810 Hardware Interface

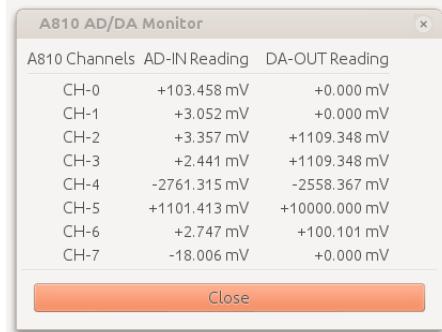


Figure 23.71.: A810 AD/DA Monitor

Table 23.4.: Module Input Definitions.

Input name and Block Start ID	Type	Input Description
DSP_SIGNAL_BASE_BLOCK_A_ID = 0x1000	Servo, Mixer, Inputstage, Scanmap	
DSP_SIGNAL_Z_SERVO_INPUT_ID	SI32	Z Servo Delta
DSP_SIGNAL_M_SERVO_INPUT_ID	SI32	Motor Servo Delta
DSP_SIGNAL_MIXERO_INPUT_ID	SI32	Mixer CH 0
DSP_SIGNAL_MIXER1_INPUT_ID	SI32	Mixer CH 1
DSP_SIGNAL_MIXER2_INPUT_ID	SI32	Mixer CH 2
DSP_SIGNAL_MIXER3_INPUT_ID	SI32	Mixer CH 3
DSP_SIGNAL_DIFF_IN0_ID	SI32	Differential IN 0
DSP_SIGNAL_DIFF_IN1_ID	SI32	Differential IN 1
DSP_SIGNAL_DIFF_IN2_ID	SI32	Differential IN 2
DSP_SIGNAL_DIFF_IN3_ID	SI32	Differential IN 3
DSP_SIGNAL_SCAN_CHANNEL_MAP0_ID	SI32	Scan Channel Map 0
DSP_SIGNAL_SCAN_CHANNEL_MAP1_ID	SI32	Scan Channel Map 1
DSP_SIGNAL_SCAN_CHANNEL_MAP2_ID	SI32	Scan Channel Map 2
DSP_SIGNAL_SCAN_CHANNEL_MAP3_ID	SI32	Scan Channel Map 3
DSP_SIGNAL_BASE_BLOCK_B_ID = 0x2000	Vector Probe/LockIn Module	
DSP_SIGNAL_LOCKIN_A_INPUT_ID	SI32	Lock-In Ch-A
DSP_SIGNAL_LOCKIN_B_INPUT_ID	SI32	Lock-In Ch-B
DSP_SIGNAL_VECPROBE0_INPUT_ID	SI32	VP Source Signal, 32bit, no oversampling
DSP_SIGNAL_VECPROBE1_INPUT_ID	SI32	VP Source Signal, 32bit, no oversampling
DSP_SIGNAL_VECPROBE2_INPUT_ID	SI32	VP Source Signal, 32bit, no oversampling
DSP_SIGNAL_VECPROBE3_INPUT_ID	SI32	VP Source Signal, 32bit, no oversampling
DSP_SIGNAL_VECPROBE0_CONTROL_ID	SI32	VP Control Signal (modified by VP)
DSP_SIGNAL_VECPROBE1_CONTROL_ID	SI32	VP Control Signal (modified by VP)
DSP_SIGNAL_VECPROBE2_CONTROL_ID	SI32	VP Control Signal (modified by VP)
DSP_SIGNAL_VECPROBE3_CONTROL_ID	SI32	VP Control Signal (modified by VP)

Continued on next page

DSP_SIGNAL_VECPROBE_TRIGGER_SIGNAL_ID	SI32	VP Trigger Control Signal Input
DSP_SIGNAL_BASE_BLOCK_C_ID = 0x3000		Output Signal Mixer Stage 8x + 2 Virtual
DSP_SIGNAL_OUTMIX_CH0...7_INPUT_ID	SI32	OUTPUT CH0...7 Source
DSP_SIGNAL_OUTMIX_CH0...7_ADD_A_INPUT_ID	SI32	A Signal to add
DSP_SIGNAL_OUTMIX_CH0...7_SUB_B_INPUT_ID	SI32	B Signal to subtract
DSP_SIGNAL_OUTMIX_CH0...7_SMAC_A_INPUT_ID	SI32	if set: gain for A Signal
DSP_SIGNAL_OUTMIX_CH0...7_SMAC_B_INPUT_ID	SI32	if set: gain for B Signal to add
DSP_SIGNAL_OUTMIX_CH8_INPUT_ID	SI32	Coarse Wave Signal selection
DSP_SIGNAL_OUTMIX_CH8_ADD_A_INPUT_ID	SI32	Signal to add
DSP_SIGNAL_OUTMIX_CH9_INPUT_ID	SI32	Coarse Wave Signal selection
DSP_SIGNAL_OUTMIX_CH9_ADD_A_INPUT_ID	SI32	Signal to add
DSP_SIGNAL_BASE_BLOCK_D_ID = 0x4000		Recorder and AVG/RMS Modules
DSP_SIGNAL_ANALOG_AVG_INPUT_ID	SI32	Source for AVG/RMS Module
DSP_SIGNAL_SCOPE_SIGNAL1_INPUT_ID	SI32	Recorder/Scope Input 1
DSP_SIGNAL_SCOPE_SIGNAL2_INPUT_ID	SI32	Recorder/Scop Input 2

 **Warning**

Reassigning Monitoring signals while normal Gxsm operation may result in wrong signal reading in the PanView and may corrupt the Gxsm-SmartPiezoDrive Link.

The Signal Monitor also allows to visualize Signals in a Galvanometer style for a better visual on changing readings as shown in Figs. 23.73 and 23.74.

 **Note**

The more gizmos you watch via Galvos or the scope, the more CPU time and USB bandwidth is utilized due to rapid refresh rates. This is normally no problem unless a high data throughput is requested for scanning or spectroscopy. It is advisable for normal operation to minimize or best not run any of the configurator scripts while taking data.

23.9. Signal Ranger MK2/3-A810 Hardware Interface

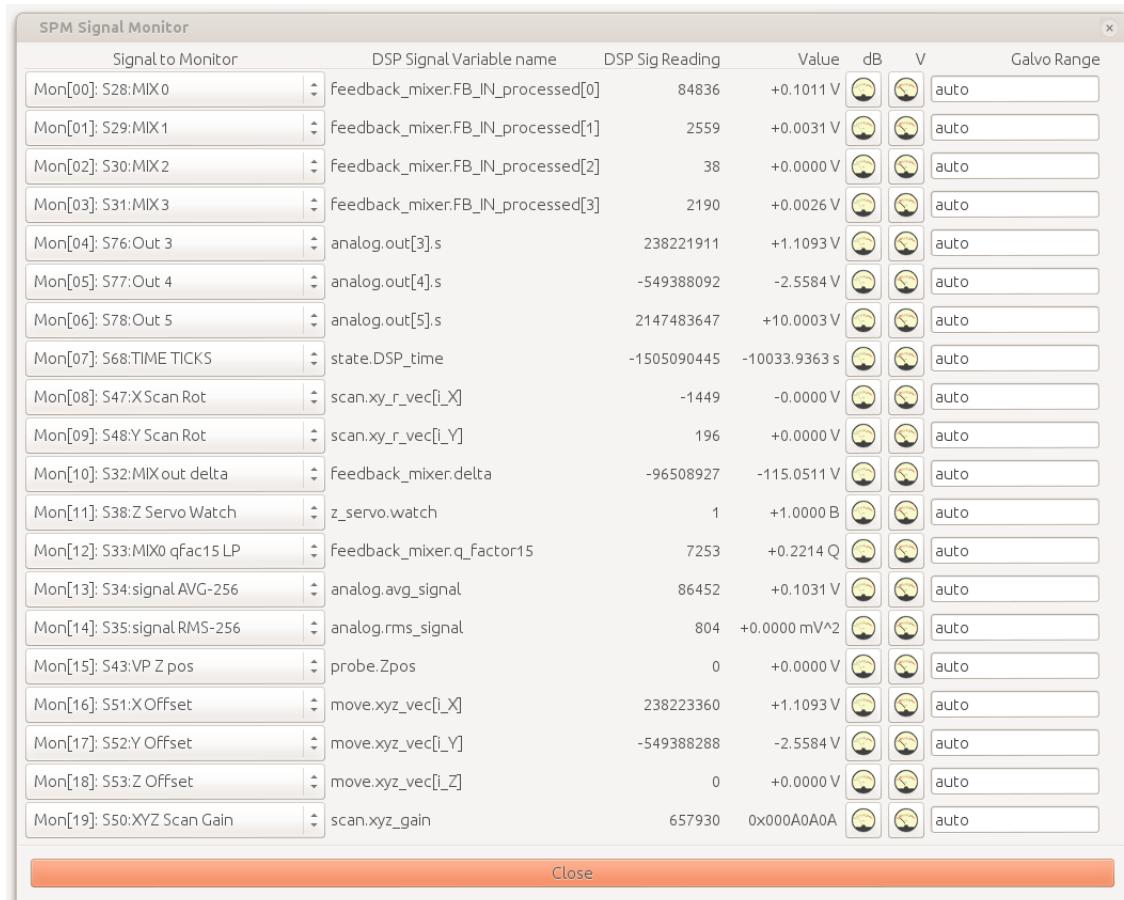


Figure 23.72.: Signal Monitor

23. Plug-Ins: hard

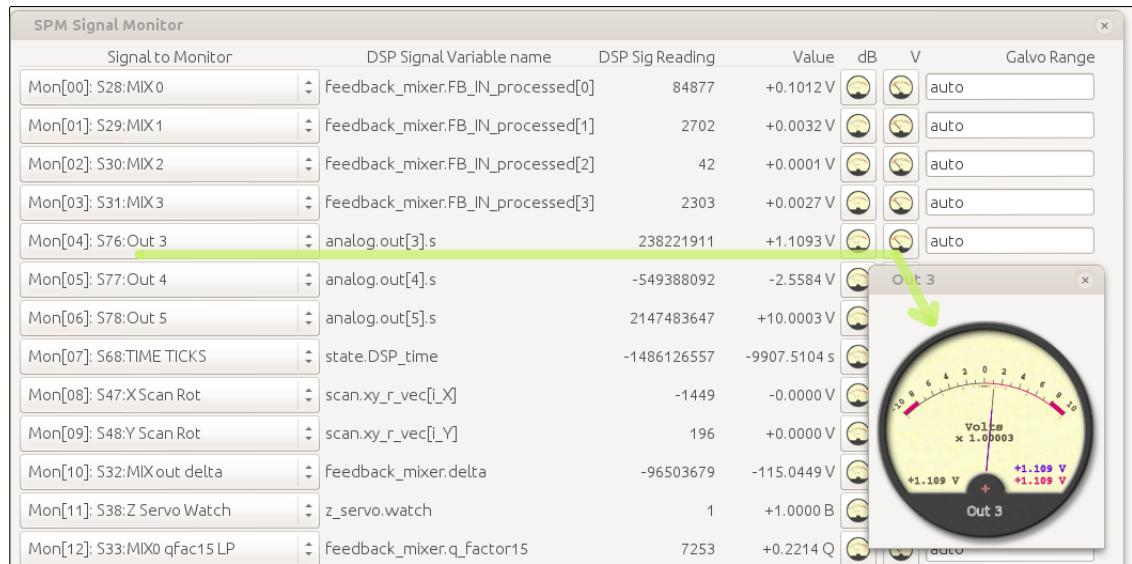


Figure 23.73.: Signal Monitor with Meter

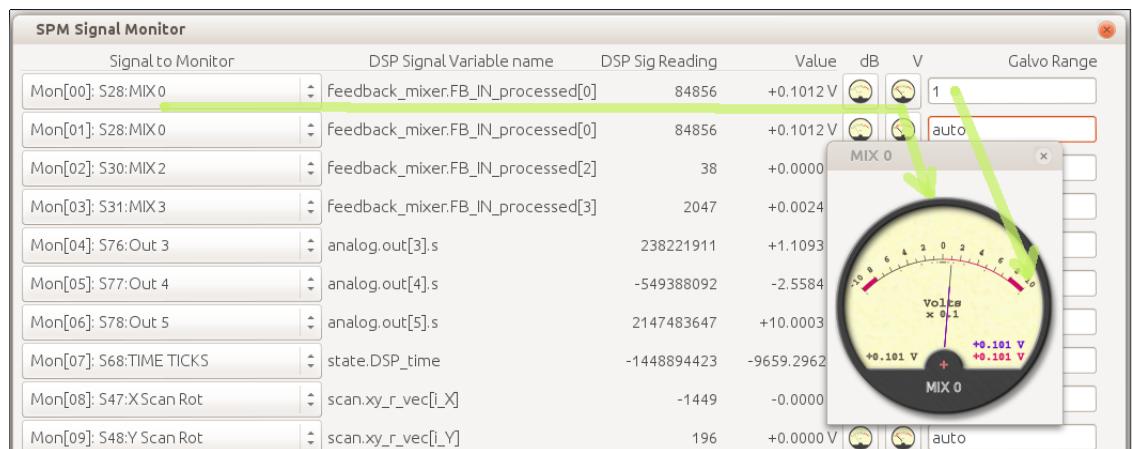


Figure 23.74.: Signal Monitor with Meter and full scale range override

Signal Patching / Configuration

To configure your own very custom SPM you may completely reroute any of the “red” signals (or make use of generic unutilized signals and inputs). For this purpose use the Patch Rack application shown in Figs. 23.75, 23.76, and 23.77.

Here you are the master and have to make sure thinks add up right. The Signal Monitor will be of great use to verify the functions.

Note to author: Need to add notes on signal precisions, etc.

 Note

SIGNAL	---	variable --->	MODULE INPUT
Sig: S00:In 0	== analog.in[0]==>		ANALOG_AVG_INPUT
Sig: S28:MIX 0	== Feedback_mixer.FB_IN_processed[0]==>	LOCKIN_A	
Sig: S28:MIX 0	== feedback_mixer.FB_IN_processed[0]==>	LOCKIN_B	
Sig: S64:Null-Signal	== analog.vnull==>		DIFF_IN0
Sig: S64:Null-Signal	== analog.vnull==>		DIFF_IN1
Sig: S64:Null-Signal	== analog.vnull==>		DIFF_IN2
Sig: S64:Null-Signal	== analog.vnull==>		DIFF_IN3
Sig: S32:MIX out delta	== feedback_mixer.delta==>		Z_SERVO
Sig: S00:In 0	== analog.in[0]==>		MIXER0
Sig: S01:In 1	== analog.in[1]==>		MIXER1
Sig: S02:In 2	== analog.in[2]==>		MIXER2
Sig: S03:In 3	== analog.in[3]==>		MIXER3
Sig: S14:LockIn B-1st	== probe.LockIn_1stB==>		VECPROBE0
Sig: S12:LockIn A-2nd	== probe.LockIn_2ndA==>		VECPROBE1
Sig: S12:LockIn A-2nd	== probe.LockIn_2ndA==>		VECPROBE2
Sig: S08:Counter 0	== analog.counter[0]==>		VECPROBE3
Sig: S44:X Scan	== scan.xyz_vec[i_X]==>		VECPROBE0_C
Sig: S45:Y Scan	== scan.xyz_vec[i_Y]==>		VECPROBE1_C
Sig: S51:X Offset	== move.xyz_vec[i_X]==>		VECPROBE2_C
Sig: S52:Y Offset	== move.xyz_vec[i_Y]==>		VECPROBE3_C
Sig: S18:PLL Res Out	== InputFiltered==>		SCOPE_SIGNAL1_INPUT
Sig: S24:PLL Exc Frq LP	== Filter64Out[0]==>		SCOPE_SIGNAL2_INPUT
Sig: S64:Null-Signal	== analog.vnull==>		M_SERVO

Figure 23.75.: Signal Patchrack

When satisfied with the signal configuration the Signal Manager application (Fig. 23.78) allows to store the current life configuration to the MK3’s FLASH memory. It will be automatically reloaded on any later DSP power-up. However, DSP software upgrades may include a modified signal table, in this case always the “Gxsm-MK2-like” build in default initialization will be loaded and the (old) FLASH configuration will be ignored as the configuration table includes a signal revision control mechanism to prevent future

23. Plug-Ins: hard

Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH0_ADD_A
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH0_SUB_B
DISABLED	== no signal ==>	OUTMIX_CH0_SMAC_A
DISABLED	== no signal ==>	OUTMIX_CH0_SMAC_B
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH1
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH1_ADD_A
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH1_SUB_B
DISABLED	== no signal ==>	OUTMIX_CH1_SMAC_A
DISABLED	== no signal ==>	OUTMIX_CH1_SMAC_B
Sig: S51:XOffset	== move.xyz_vec[i_X] ==>	OUTMIX_CH2
Sig: S49:Z Offset from XY Slope	== scan.z_offset_xyslope ==>	OUTMIX_CH2_ADD_A
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH2_SUB_B
DISABLED	== no signal ==>	OUTMIX_CH2_SMAC_A
DISABLED	== no signal ==>	OUTMIX_CH2_SMAC_B
Sig: S47:XScan Rot	== scan.xy_r_vec[i_X] ==>	OUTMIX_CH3
Sig: S51:XOffset	== move.xyz_vec[i_X] ==>	OUTMIX_CH3_ADD_A
Sig: S64:Null-Signal	== analog.vnull ==>	OUTMIX_CH3_SUB_B
DISABLED	== no signal ==>	OUTMIX_CH3_SMAC_A
DISABLED	== no signal ==>	OUTMIX_CH3_SMAC_B

Figure 23.76.: Signal Patchrack

problems with new or eventually non existent signals. The “Gxsm-MK2-like” default configuration can be reloaded at any time:

REVERT TO POWER UP DEFAULTS This actually loads the pre defined “Gxsm-MK2-like” build in defaults configuration.

STORE TO FLASH This stores the current signal configuration to flash.

RESTORE FROM FLASH This attempts to restore a signal configuration from flash if a valid flash table exists and the revision is compatible with the DSP code revision. Reflashing the DSP code does NOT alter the signal configuration table and depending on what is new the configuration will be in most cases still be usable. Only if the signal or input descriptions have changed it will fail – nothing will be changed in this case. You can just watch the configuration via the Patch Rack or compare the Signal Graphs.

ERASE (INVALIDATE) FLASH TABLE This simply deletes any stored signal configuration and thus invalidates the table. It does not change the current configuration, but the next DSP power cycle will obviously end up with the default configuration.



Advise: keep a human readable printout of your configuration. The list on the terminal,

23.9. Signal Ranger MK2/3-A810 Hardware Interface

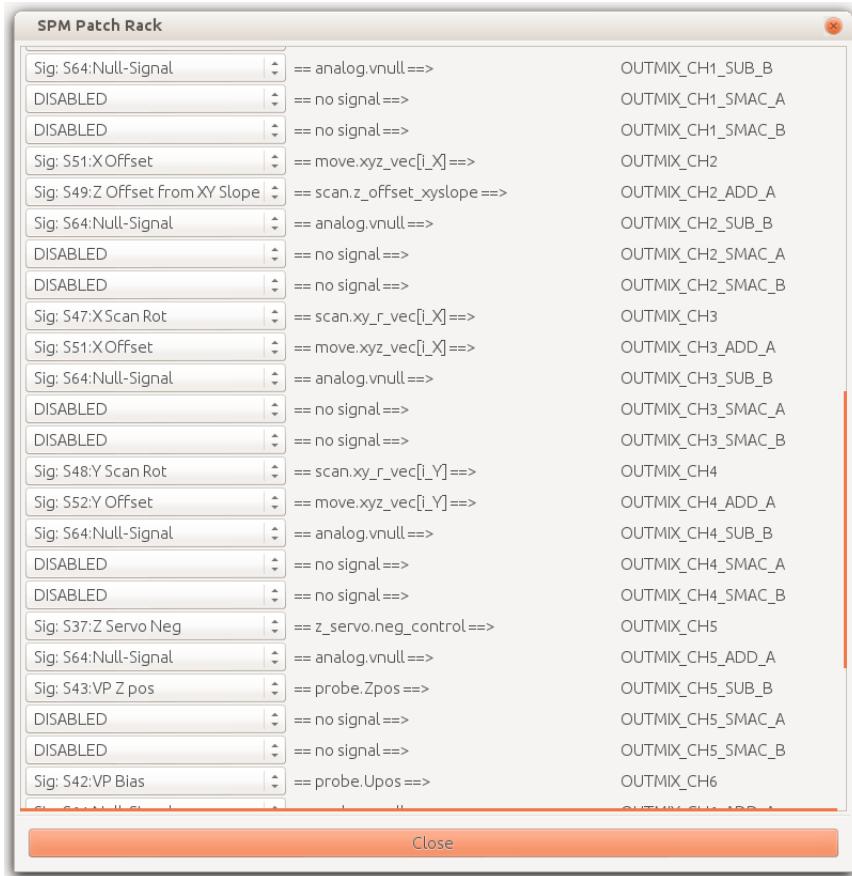


Figure 23.77.: Signal Patchrack

a few screen shots as shown here and may be also the signal graph.

Notes to author: It is surely possible to save configuration to a file on disk for later restore, but this is not yet implemented. ⚠ Note

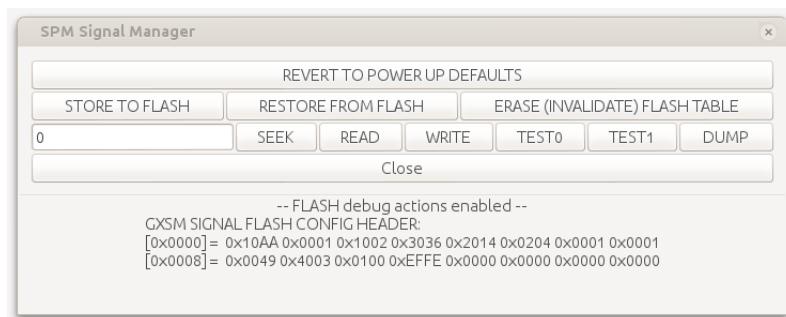


Figure 23.78.: Signal Manager/Flash Configurator. Please note, here are additional FLASH test/debugging controls visible which will be excluded per default soon.

Vizualization of the current Signal Configuration

The “Create Signal Graph” application inspects the current DSP configuration, checks for errors and generates via the GraphViz tool a dot file and a svg presentation of the configuration as shown in Fig. 23.69. This app writes a dot and svg file. It attempts to launch xdot for viewing. This fails if xdot is not installed.

Signal Oszilloscope

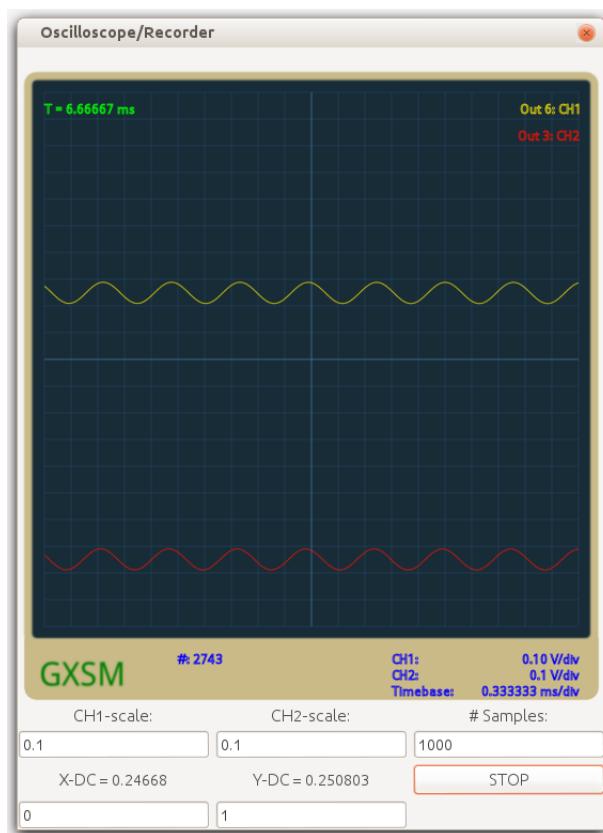


Figure 23.79.: Signal Oszilloscope

GXSM Signal Master Evolved and beyond

Configuring and selecting signals for acquisition within GXSM is not highly flexible. Fig. 23.80 highlights the dependencies from where signals are taken and how they are dynamically set for given slots in the channelselector source and VPsig source pull down menus.

– drafting –

Now full signal propagation and management via Gxsm including proper unit and scaling support derived from signal definition table. All live, actual settings read back from DSP at Gxsm startup and refresh at scan start (in case of external manipulations). Mixer channel signal selections are now reflected by signal name in Channelselector and also automatically used as Scan title/file name. New are also up to 4 fully free assignable signals (32bit res.) as scan-source signal and can be selected via the Channelselector, also automatic label propagation and automatic resources updates for channel assignment including units and scaling. Along with this also new is a on pixel/sub-grid level real time (fast) VP execute with "end-of-VP-section" data assignment to a special 8x8 signal matrix "VP Sec Vec64". More later about this. In general you can manipulate for example the bias while FB hold or on, etc... in pixel level and acquire at end of every VP section a value and assign this to a scan channel as data source.

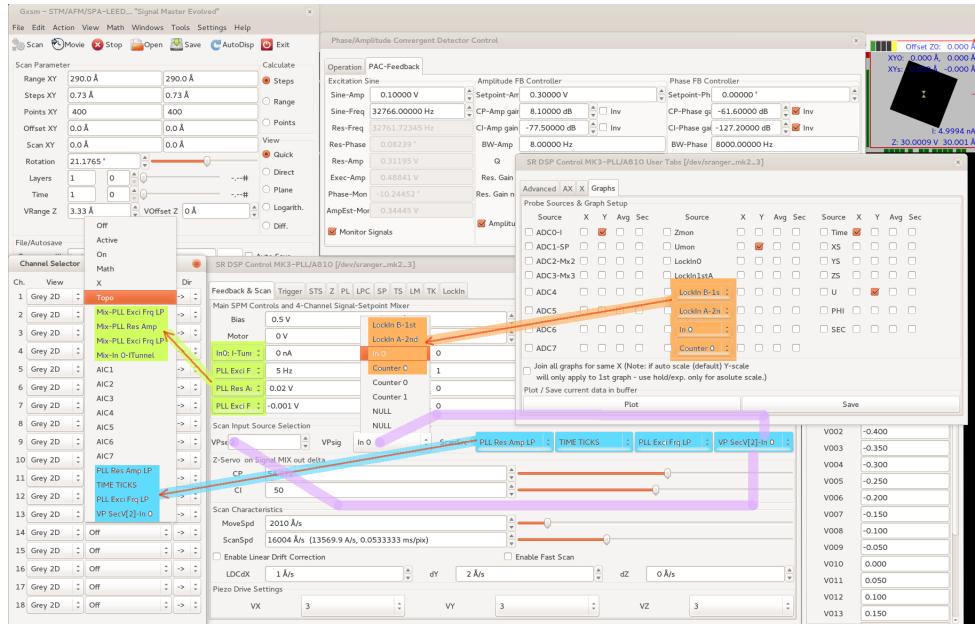


Figure 23.80.: Signal Master Evolved

PAC/PLL Support and Tuning with Gxsm – Startup guidelines

Please start reading and understanding first the PLL principle of operation, find the PLL user manual here: http://www.softdb.com/_files/_dsp_division/SPM_PLL_UsersManual.pdf

The GXSM-3.0 GUI is pretty much identical beyond a little different look and arrangement of same control elements. And the operation is identical as Gxsm uses this SoftdB PAC-library and their DSP level application interface.

For Gxsm as of now the PAC/PLL is started on channels¹ IN-4 (signal input) and OUT-7 (excitation) via the library call “StartPLL(4,7);”. This implies for all further operations that channel OUT-7 is not any more available for any other uses, but only when the PAC processing is enabled.



About the MK3-Pro/A810-PLL and the by SoftdB provided active miniature test oscillator PCB with two identical BNC's for input and output: The excitation “exec. signal input to the quartz” side (goes to A810 OUT-7) can be identified looking at the PCB and finding the BNC what is connected only via R10 to the quartz (big black block) or the BNC just next to the ON/OFF switch. The other end connect to IN-4.



Simple alternative to the active oscillator: you can actually just get a simple passive “watch” quartz with 32.768 kHz and hook it it directly inbetween OUT-7 and IN-4. Hint: just mount it inside a convenient small BNC-BNC “filter” box. It will work nice with just 0.5V excitaion!

1. For PAC/PLL signal monitoring select the two PLL related signals of interest (for example the Resonator Amplitude “PLL ResAmp” and Phase (Frequency equivalent) “PLL ResPh”) to watch with the scope via the Patch Rack and assign to “SCOPE_SIGNAL1/2_INPUT”.
2. Then select the phase detector time constant “TauPAC” – set it to 20 μ s or faster.
Referring to the PLL manual “4.1 Phase Detector Time Cst (s)”: This control adjusts the time constant of the phase detector. We suggest keeping time constant to 20 μ s (fast set-up), which allows a bandwidth of about 8 kHz. Note that the auto-adjustment functions for the PI gains of both controllers (amplitude and phase) automatically² set the time constant to 20us. This way, the bandwidth of the controller is only limited by the PI gains and the LP filter.

Any further tweaking of the time constant will prevent the auto-set via “Q” from resulting in stable control loop conditions, so you have to manually tweak those.

¹Channel count convention is here 0 ... 7.

²This is **not** the case for the Gxsm control and to use the auto gain adjustments you must set TauPAC manually to 20 μ s.

23.9. Signal Ranger MK2/3-A810 Hardware Interface

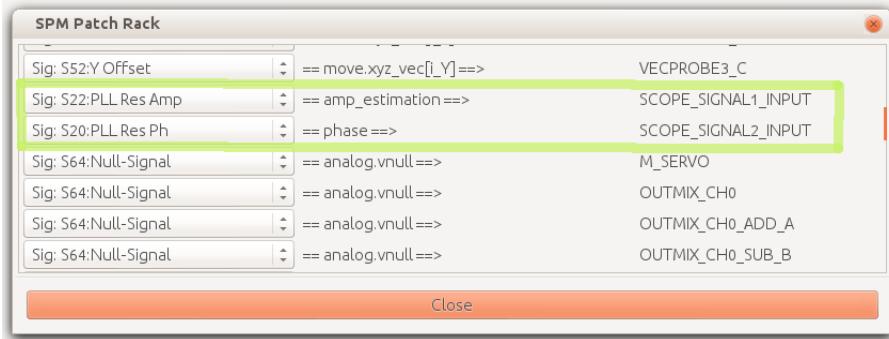


Figure 23.81.: Example Signal Patch Configuration for PLL view

3. Then turn on the PAC Processing by activating the check box. Or toggle to restart with changed time constant!
4. Operational ranges setup. This defines the range mapping and actual sensitivities for excitation frequency, resonator phase, excitation and resonator amplitudes. Set the phase range to $360^{\circ} (\pm 180^{\circ})$ for tuning (see Fig. 23.82). Select reasonable ranges for amplitudes matching your system needs.

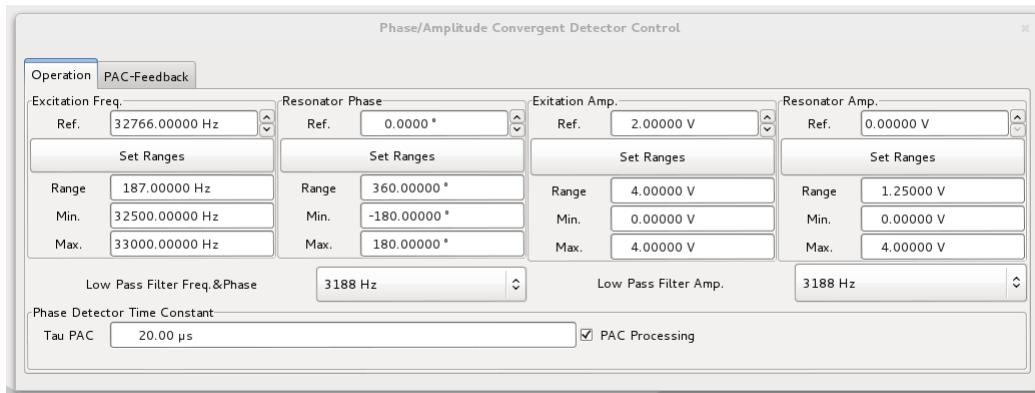


Figure 23.82.: PAC/PLL initial operation boundary setup example. Time constant set to $20 \mu\text{s}$ here, PAC processing enabled.

5. Check signals, manually set a estimated frequency, play manually and check for any initial response using the monitoring option to just read the values. Leave both FB Controllers off for now. You should see some with frequency changing resonator response amplitude, make sure ranges are working for you, if not, readjust.

23. Plug-Ins: hard

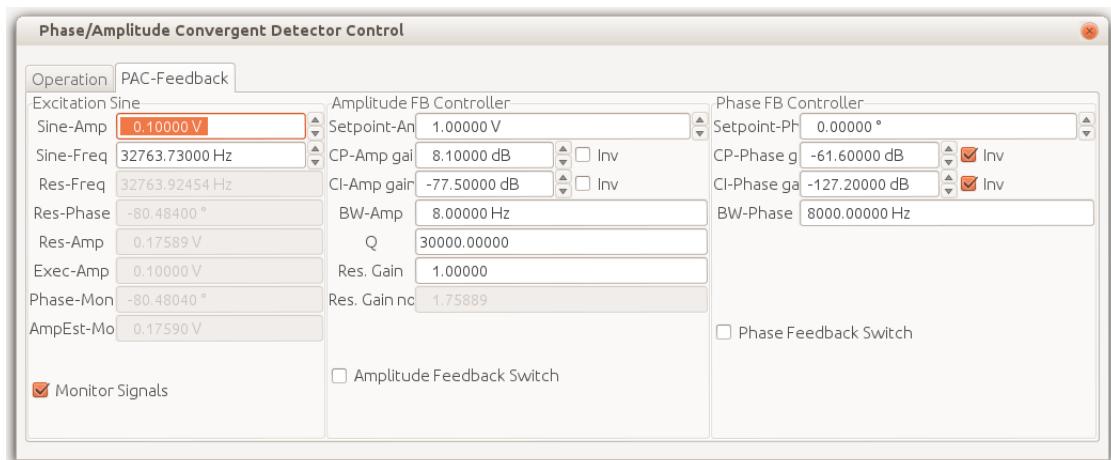


Figure 23.83.: PAC/PLL – initial manual pre-check of signals and responses.

23.9. Signal Ranger MK2/3-A810 Hardware Interface

- Now open the tune App to initiate a default sweep.

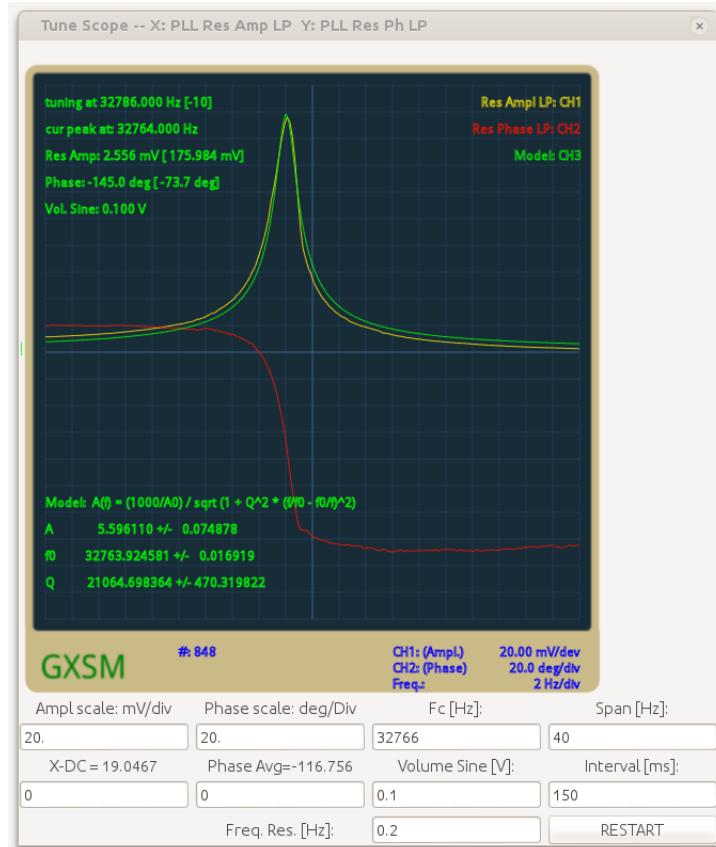


Figure 23.84.: PAC/PLL Tune App – initial run with phase reference set to zero. Here a passive watch quartz was used.

23. Plug-Ins: hard

7. You may now already adjust the “Setpoint-Phase” to the value the tune app detected phase at resonance. This is important as the Phase controller works always around phase zero. And you have to compensate via this Setpoint the phase offset. This done a consecutive tune sweep should align the phase accordingly.

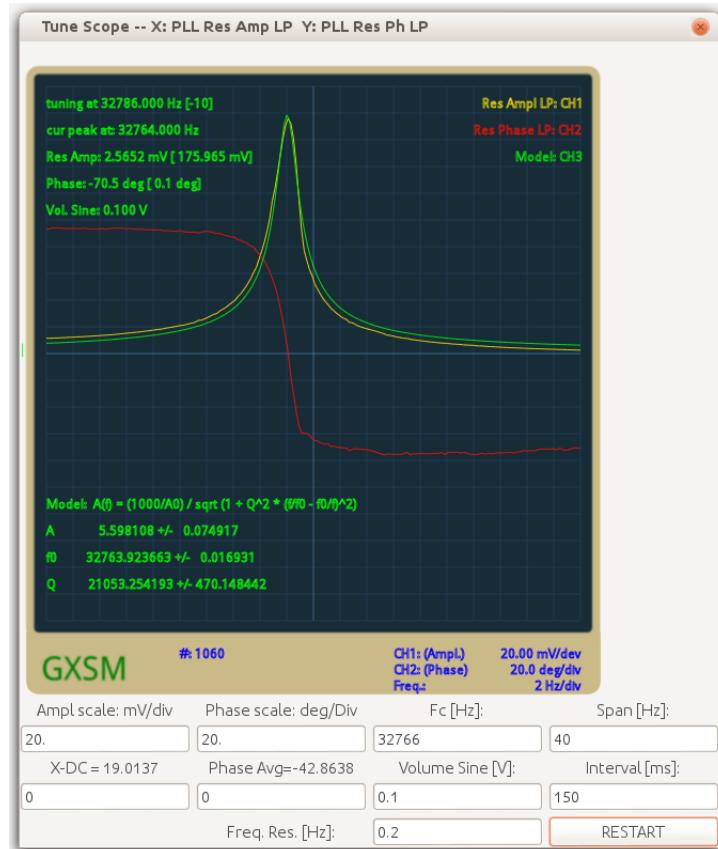


Figure 23.85.: PAC/PLL Tune App – frequency sweep with phase reference value was set to initial phase at resonance as needed for phase locked loop (PLL is always operating around phase 0). Here a passive watch quartz was used

23.9. Signal Ranger MK2/3-A810 Hardware Interface

8. Use the oscilloscope app to inspect the signals for further fine tuning.³

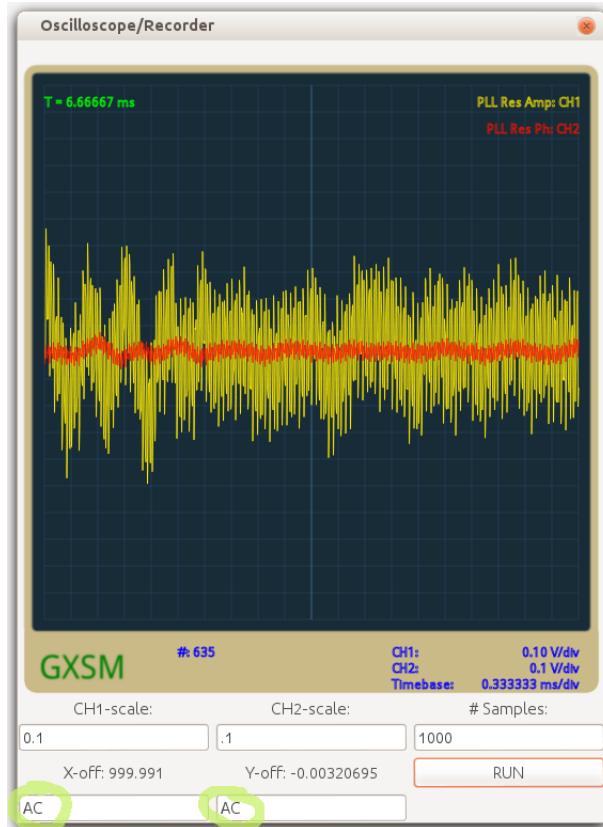


Figure 23.86.: Oscilloscope app for viewing PLL operation. Use AC coupling for both channels so that you can enlarge the signals in a convenient way.

9. Re run the tune tool with phase set to initial detected value (relative to the initial reference set to zero! Keep in mind any further adjustments are relative.).

³To be resolved issue, currently the scope max block size/count is about 2000. Hi-level Python issue. The DSP has storage for up to 1e6-1 value pairs.

23. Plug-Ins: hard

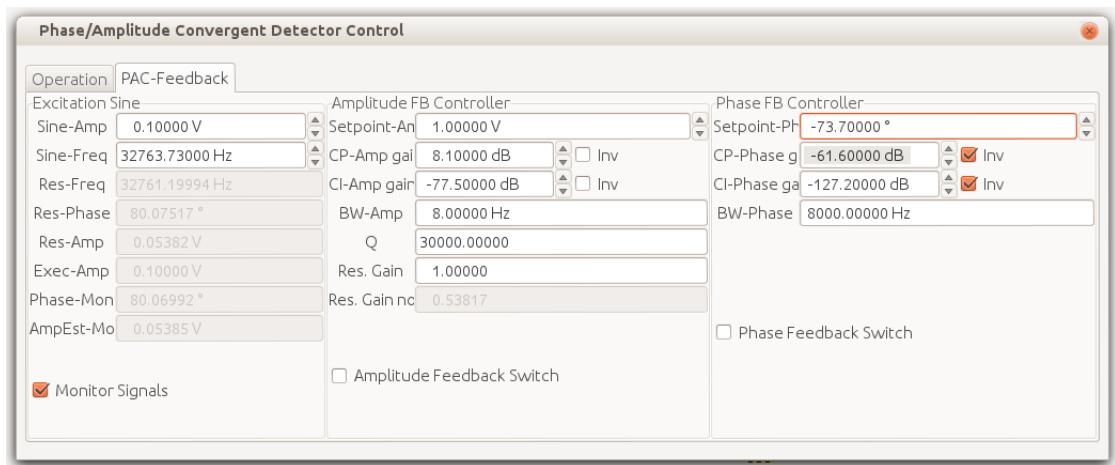


Figure 23.87.: PAC/PLL Feedback settings with phase reference set to initial phase at resonance (here -73.7°).

- Enter the Q value computed by the tune app fit here to auto-set the amplitude controller gains. Usually a good starting point. Remember, those are only good for a phase detector time constant of $20\mu\text{s}$ or faster as otherwise you limit the bandwidth before the controllers! Select a reasonable amplitude setpoint, make sure the excitation range is sufficient and turn on the amplitude controller.

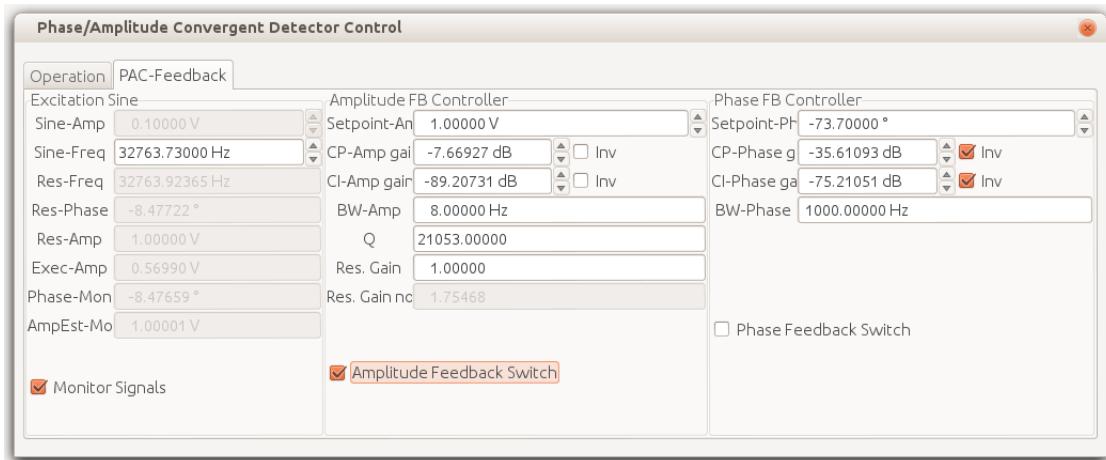


Figure 23.88.: PAC/PLL – Amplitude FB on

- Assuming the amplitude controller works you may proceed and engage the phase controller to finally close the phase locked loop. Just select the desired bandwidth. Then engage the controller check box.
- Step Response Test Tool – not yet completed – need to fix related issue with the python problem with big buffer transfers.

23. Plug-Ins: hard

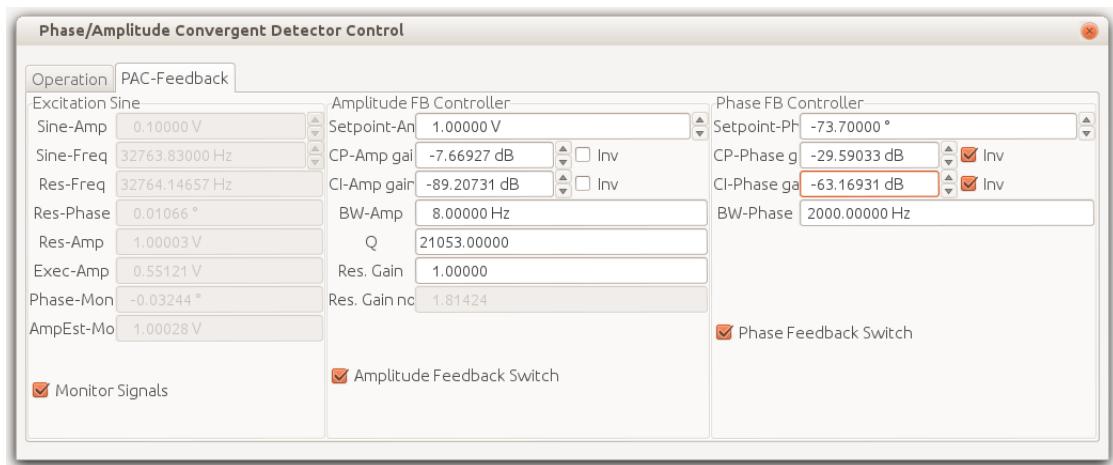


Figure 23.89.: PAC/PLL fully operational with both controller engaged. Amplitude stabilized. Phase fixed.

23.9. Signal Ranger MK2/3-A810 Hardware Interface

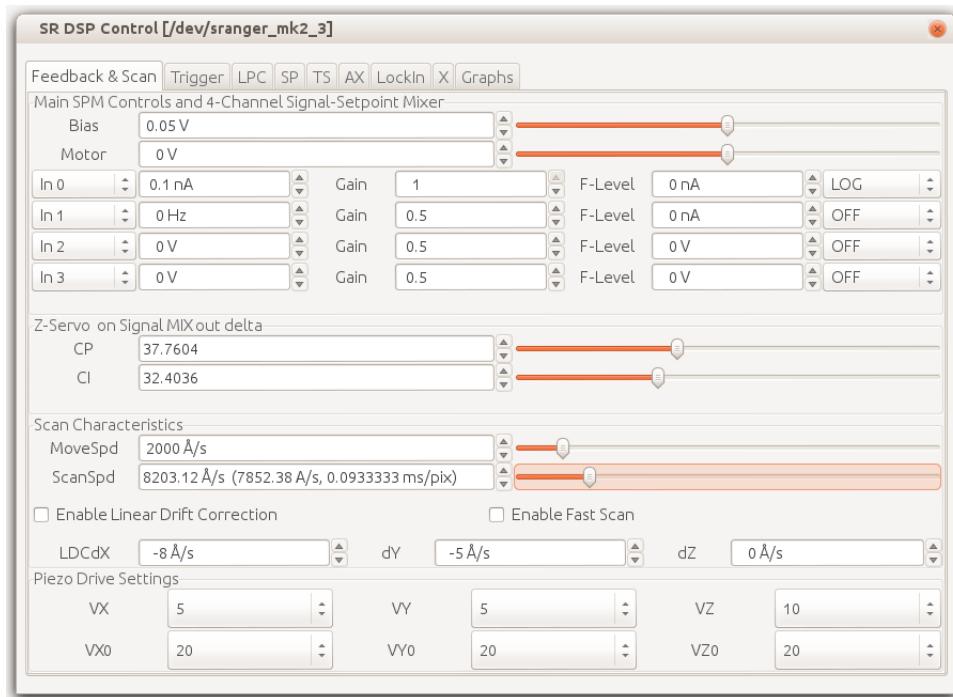


Figure 23.90.: DSP Control – Mixer, Servo

23. Plug-Ins: hard

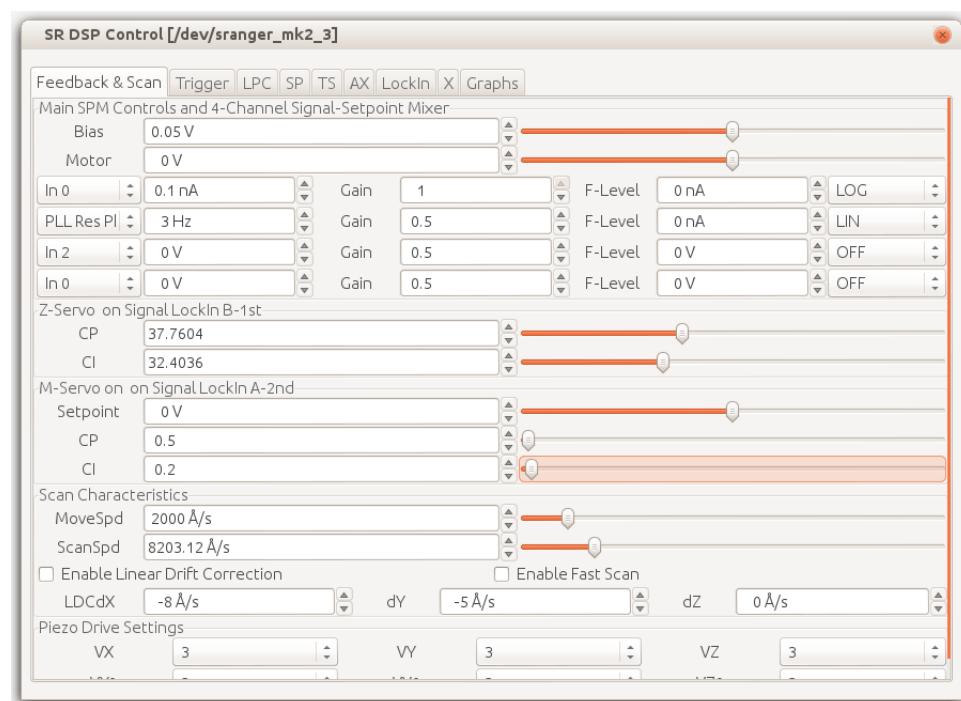


Figure 23.91.: DSP Control with M-Servo input configured

Few MK3-Pro related notes on LockIn updates:

- Reference Sinus table length is 512 samples, MK3 LockIn operated on full rate of 150 kHz. Available frequencies are: $\times 1, 2, 4, \dots 32$ (all $512 \dots \frac{512}{32}$ samples per period) $\rightarrow 292.968$ Hz, 585.938 Hz, 1171.875 Hz, 2343.75 Hz, 4687.5 Hz, 9375.0 Hz.
- Do not run LockIn in “run free” mode for Phase adjusting sweep.
- Amplitude setting is respected in real time.
- To change frequency in “run free” mode you must toggle it on/off after chaning to re initialize. Touch frq. entry to update on actual set freqency.

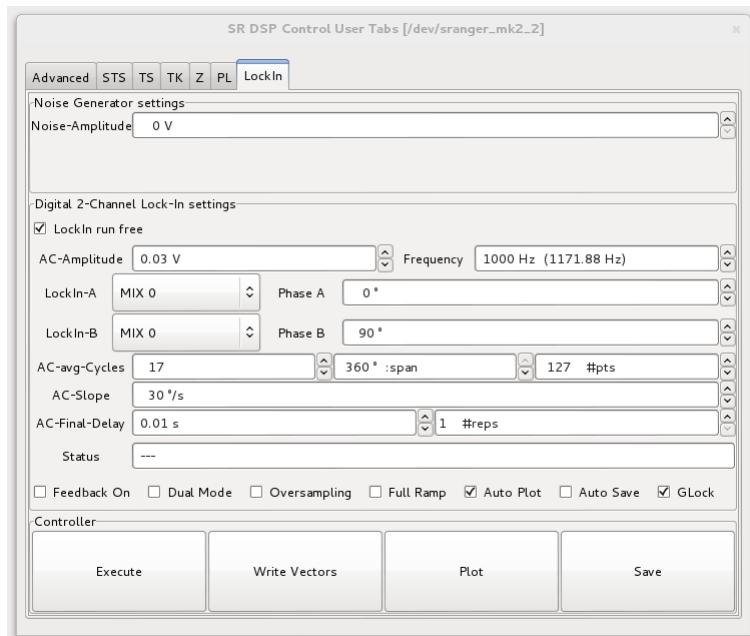


Figure 23.92.: DSP Control – LockIn: Free run mode enabled.

23. Plug-Ins: hard

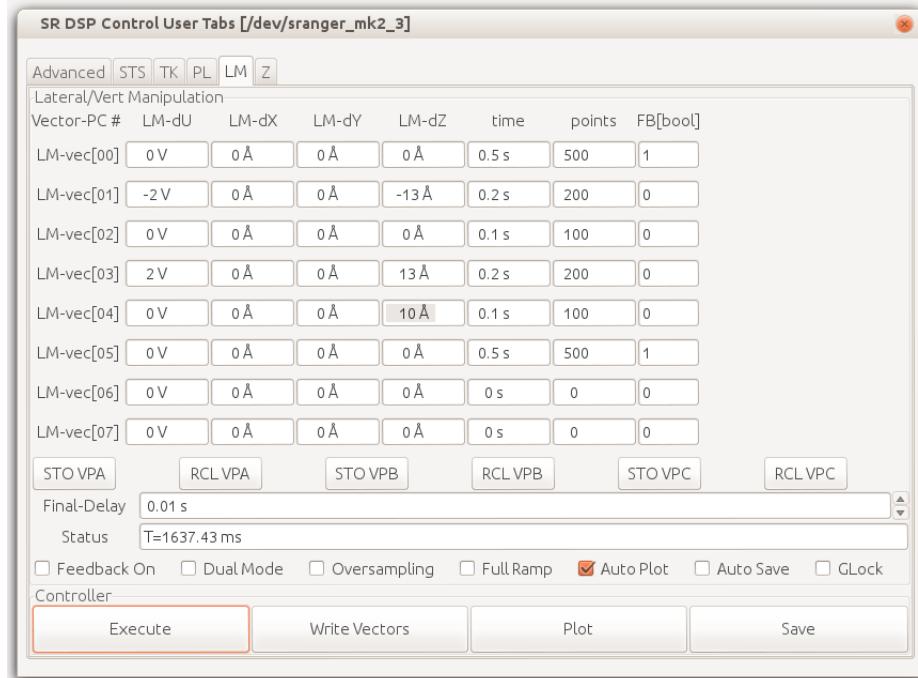


Figure 23.93.: DSP VP-LM – tip forming example procedure

23.9. Signal Ranger MK2/3-A810 Hardware Interface

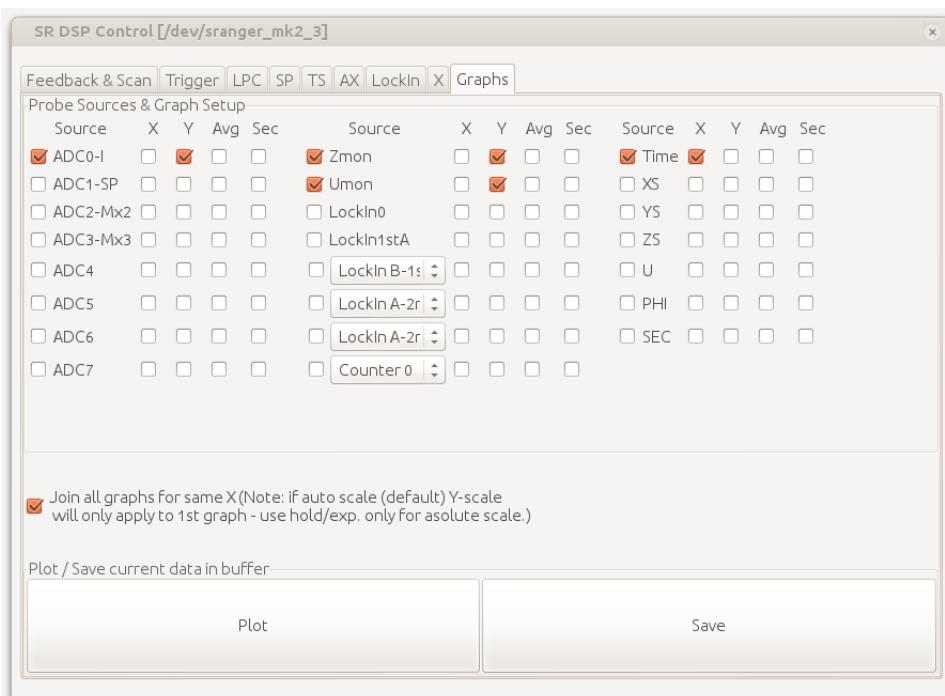


Figure 23.94.: Graphs example for VP-LM

23.9.9. Optional MK3-Pro-SmartPiezoDrive with Gxsm Link

The Smart Piezo Drive Linux Frontend (Fig. 23.95) with Gxsm-DSP Offset-Link capability is provided via this python application:

```
plug-ins/hard/MK3-A810_spmcontrol/python_scripts/mk3_spd_control.
```

It communicates with the SPM control DSP and periodically reads and updated it's offset settings of the Link is enabled. For this the last 4 Signal Monitor slots are utilized and these signals should not be changed when the Link is up. It automatically is setting up the Link and Signals at startup and expects those not to be changed later.



Beware: Later reassigning any of the last 4 Signal Monitor slots will result in wrong offset or gain settings.



Figure 23.95.: MK3 based Smart Piezo Drive (HV Amplifier) Control Panel with Gxsm-Link for digital Offset control.

Special features and behaviors to be documented here!

Info for Plug-In: Tools/SR-DSP Control

Plug-In name: sranger_mk2_hwi

File: hard/sranger_mk2_hwi.C

Author: Percy Zahl

Email: zahl@users.sf.net

24. Plug-Ins: hard/MK3- A810_spmcontrol/experimental_python_scripts

25. Plug-Ins: hard/MK3- A810_spmcontrol/python3_scripts

26. Using plug-ins

The GXSM-3.0 core provides only basic data handling and visualization functions. All more advanced image manipulation and analysis functions are implemented in form of plug-ins. A plug-in is basically a small piece of software designed for a usually very limited purpose, e.g. a low pass filter. Plug-ins, however, don't work as standalone programs but are dynamically linked to the GXSM-3.0 core. GXSM-3.0 plug-ins usually register themselves to menus within GXSM-3.0. Therefore, if everything works fine, the user doesn't notice at all that she/he is using a plug-in function and not a function hard linked to the GXSM-3.0 core.

The plug-ins are automatically loaded during the start of GXSM-3.0. Instead of simply loading all available plug-ins, GXSM-3.0 loads only the plug-ins used for the current *Instrument type* (see Sec. 10.2). For instance, SPA-LEED related plug-ins are not loaded if GXSM-3.0 is configured for STM as *Instrument type*. During runtime, GXSM-3.0 can be forced to reload the plug-ins by *main menu/Tools/Reload Plugins*. If the *Load All* button is pressed in the upcoming pop-up window, all available plug-ins are loaded regardless of the *Instrument type*.

The plug-in `listplugins` (see Sec. 22.8) is a “plug-in plug-in”. This plug-in can be called using *main menu/Tools/Plugin Details* and lists all currently loaded plug-ins in a window with detailed information on each of them. Highlighting a plug-in in the list and pressing the *Details* button displays all available information on it, such as the path to the plug-in, a short description, the name of its author, etc. Furthermore, the *About* and more important the *Configure* functions of the plug-in can be started from here (it is further referenced as *PlugIn configurator*).

The core function *main menu/Tools/Plugin Info* displays some information on all loaded plug-ins on `stdout`. The output is html formatted and currently used to provide a list of all plug-ins on the GXSM-3.0 website.

Plug-ins are the easiest method to extend the functionality of GXSM-3.0. For the most common problem of writing data manipulation functions a shell script included in the GXSM-3.0 source code can be used to automatically generate the source code for a new plug-in¹. Only the code for the actual manipulation of the data must be hand-written. The newly generated `PlugIn` provides already a simple math routine to

¹Run the shell script `./generate_math_plugin.sh` at the command prompt from the `Gxsm/plug-ins` directory to generate a new math-type plug-in.

demonstrate the basics of source and destination data access methods provided by the GXSM-3.0 core. For those curious hacker guys a plain plug-in template and step by step instructionset are available too ([20.7](#)).

For the second most common task, the import or export of data from or to, respectively, files generated by other programs a well commented example code exists and can be used as template for writing new import/export plug-ins.

For all other special tasks like instrument control a deeper knowlegde of the GXSM-3.0 core and hardware abstraction methods is required and a study of some existing PlugIns and/or the core classes is mandatory. But feel free to use the discussion forum located at the GXSM-3.0 project Web pages to get more help on this – you are welcome by the GXSM-3.0 team.

Part III.

Digital signal processing subsystem

Part IV.

Appendix

A. Terms and conditions

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

A. Terms and conditions

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate

A. Terms and conditions

automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the

code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source

A. Terms and conditions

fixed on a durable physical medium customarily used for software interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, re-

gardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

A. Terms and conditions

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

A. Terms and conditions

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server

or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

A. Terms and conditions

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <text><year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

A. Terms and conditions

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
```

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

References

- [1] Percy Zahl et al. “The flexible and modern open source scanning probe microscopy software package GXSM”. In: *Review of Scientific Instruments* 74 (2003), pp. 1222–1227. DOI: [10.1063/1.1540718](https://doi.org/10.1063/1.1540718).
- [2] Percy Zahl et al. “Open source scanning probe microscopy control software package GXSM”. In: *J. Vac. Sci. Technol. B*. Vol. 28. 2010, C4E39–C4E47. DOI: [10.1116/1.3374719](https://doi.org/10.1116/1.3374719).
- [3] Percy Zahl and Thorsten Wagner. “GXSM - Smart & Customizable SPM Control”. In: *Imaging & Microscopy* 17 (2015), pp. 38–41.

Part V.

Index

Index

1D-profile, *see* Channel View

3D Scene Setup, 66

autogen.sh, 40

C-PlugIn

 common/editnc.C, 215
 common/extra_scan_info.C, 254
 common/listplugins.C, 255
 common/mkicons.C, 256
 common/PanView.C, 218
 common/printer.C, 254
 common/ProbeIndicator.C, 217
 common/pyremote.C, 253
 common/queryDSPInfo.C, 255
 control/DSPPeakFind.C, 99
 control/inet_json_external_scandata.C,
 95
 control/nano_manipulator.C, 100
 control/NanoPlott.C, 98
 control/rhk_scancontrol.C, 103
 control/SpaLeedControl.C, 96
 control/spm_scancontrol.C, 102
 hard/demo_hwi.C, 281
 hard/grab_v4l.C, 282
 hard/innovative_DSP_hwi.C, 281
 hard/kmdsp_hwi.C, 280
 hard/LAN_rhk_hwi.C, 279
 hard/spm_simulator_hwi.C, 257
 hard/sranger_hwi.C, 277
 hard/sranger_mk2_hwi.C, 359

hard/tc211_ccd.C, 280
math/arithmetic/abs_scan.C, 213
math/arithmetic/add_scan.C, 209
math/arithmetic/div_scan.C, 211
math/arithmetic/invert_z.C, 210
math/arithmetic/log_z.C, 214
math/arithmetic/max.C, 208
math/arithmetic/mul_scan.C, 207
math/arithmetic/sub_scan.C, 211
math/arithmetic/Z_limiter.C, 212
math/arithmetic/Z_rescale.C, 213
math/arithmetic/Z_usr_rescale.C, 209
math/background/bg_z_drift.C, 136
math/background/bggamma.C, 125
math/background/DocOnlyEregress.C,
 134
math/background/DocOnlyLineRegress.C,
 127
math/background/lineregression.C, 126
math/background/mulconst.C, 130
math/background/parabolregress.C, 129
math/background/pass_cc.C, 135
math/background/plane3pkt.C, 133
math/background/plane_max_prop.C,
 128
math/background/plane_regression.C,
 126
math/background/removelineshifts.C,
 129
math/background/stop_ccr.C, 131
math/background/subconst.C, 135

math/background/template.C, 132
math/background/timescalefft.C, 133
math/background/waterlevel.C, 130
math/convert/make_test.C, 139
math/convert/short_to_short.C, 140
math/convert/to_byte.C, 137
math/convert/to_complex.C, 139
math/convert/to_double.C, 142
math/convert/to_float.C, 142
math/convert/to_long.C, 141
math/convert/to_short.C, 138
math/convert/uto_float.C, 140
math/filter1d/despike1d.C, 146
math/filter1d/diff.C, 148
math/filter1d/ft1d.C, 150
math/filter1d/koehler.C, 148
math/filter1d/linear_stat_diff.C, 150
math/filter1d/repair_cs.C, 145
math/filter1d/template.C, 147
math/filter1d/timedomfftfilter.C, 149
math/filter2d/curvature.C, 154
math/filter2d/despike2d.C, 159
math/filter2d/DocOnlyLineInterpol.C,
 155
math/filter2d/edge.C, 161
math/filter2d/ft2d.C, 158
math/filter2d/ift2d.C, 159
math/filter2d/lineinterpol.C, 158
math/filter2d/local_height.C, 161
math/filter2d/normal_z.C, 162
math/filter2d/smallconvol.C, 157
math/filter2d/smooth.C, 156
math/filter2d/stat_diff.C, 157
math/filter2d/Tderive.C, 155
math/filter2d/template.C, 160
math/misc/findlocmax.C, 173
math/misc/layersmooth.C, 171
math/misc/make_volume.C, 170
math/misc/minzlayer.C, 169
math/misc/psdadd.C, 172
math/misc/shape.C, 172
math/misc/spectrocut.C, 170
math/misc/workfuncextract.C, 174
math/probe/afm_lj_mechanical_sim.C,
 168
math/probe/probe_image_extract.C,
 166
math/statistik/add_trail.C, 206
math/statistik/AngularAnalysis.C, 204
math/statistik/autocorrelation.C, 202
math/statistik/average_profile.C, 206
math/statistik/crosscorrelation.C, 200
math/statistik/histogram.C, 204
math/statistik/nndistribution.C, 194
math/statistik/OpenCV_match.C, 189
math/statistik/OpenCV_recenter.C, 196
math/statistik/polarhist.C, 193
math/statistik/slopeabs.C, 205
math/statistik/slopedir.C, 203
math/statistik/spasim.C, 195
math/statistik/spasimkz.C, 200
math/statistik/stepcount.C, 201
math/statistik/VacancyLineAnalysis.C,
 191
math/statistik/vorlage.C, 198
math/transform/affine.C, 186
math/transform/autoalign.C, 179
math/transform/flip_diagonal.C, 179
math/transform/mandriftfix.C, 182
math/transform/merge_h.C, 184
math/transform/merge_v.C, 187
math/transform/mirror_x.C, 180
math/transform/mirror_y.C, 187
math/transform/movieconcat.C, 178
math/transform/multi_dim_transpose.C,
 177
math/transform/OctoCorr.C, 182
math/transform/quenchscan.C, 176

- math/transform/reverse_layers.C, 184
 math/transform/rotate.C, 181
 math/transform/rotate90.C, 175
 math/transform/scalescan.C, 178
 math/transform/shear_x.C, 176
 math/transform/shear_y.C, 177
 math/transform/shiftarea.C, 185
 math/transform/unwrap.C, 183
 math/transform/volume_transform.C,
 183
 scan/ascii_data_im_export.C, 118
 scan/bin_import.C, 123
 scan/converter.C, 113
 scan/cube_import.C, 106
 scan/external_converter.C, 115
 scan/g_dat_im_export.C, 112
 scan/gmeyer_im_export.C, 119
 scan/nanoimport.C, 121
 scan/omicron_io.C, 120
 scan/png_im_export.C, 122
 scan/primitiveimexport.C, 117
 scan/PsiHDF_im_export.C, 109
 scan/quicktime_im_export.C, 109
 scan/rhk2000_import.C, 113
 scan/rhk_spm32_import.C, 115
 scan/sdfimport.C, 119
 scan/spa4_d2d_im_export.C, 123
 scan/UK2k_import.C, 114
 scan/uksoft2001_im_export.C, 107
 scan/v5d_export.C, 112
 scan/wip_im_export.C, 105
 scan/WSxM_io.C, 124
- Channel modes, 58
 Channel View
 Grey 2D, 62
 No, 62
 Profile 1D, 77
 Surface 3D, 65
- Channels, 57
- Command line parameters, 51
 compiling, 40
 Configuration, *see* Preferences
 configure, 40
 Copyright, 371
- Data display modes, *see* View
 Diff, *see* View
 Direct, *see* View
 Direct HiLit, *see* View
 Druids, *see* Preferences
- Event, 65
- GConf, *see* Preferences
 GNU General Public License, 371
 Grey 2D, *see* Channel View
 GXSM-Menu
 _Math/Background/Rm Line Shifts,
 129
 File/Im,Export/WXsM, 124
 File/Import/ASCII, 118
 File/Import/Binary, 123
 File/Import/Cube, 106
 File/Import/G-dat, 112
 File/Import/GMeyer Dat, 119
 File/Import/Nano Scope, 121
 File/Import/Omicron_SPM_Import,
 120
 File/Import/PNG, 122
 File/Import/Primitive Auto Import and
 File/Export/Primitive Auto Ex-
 port, 117
 File/Import/PsiHDF, 109
 File/Import/Quicktime, 109
 File/Import/RHK SPM32, 115
 File/Import/RHK STM-200 import
 , 113
 File/Import/SDF-import, 119
 File/Import/SPA4-d2d, 123

- File/Import/Uk2k import, 114
File/Import/UKSOFT, 107
File/Import/WIP, 105
File/Print, 254
Hardware/Innovative_DSP:SPM:SPA-HwI, 281
Hardware/LAN_RHK:SPM-HwI, 279
Hardware/TC211-CCD-HwI, 280
Hardware/video4linux-HwI, 282
math-filter2d-sectionFT 2D, 158
math-transformations-sectionMovie
 Concat, 178
math-transformations-sectionMulti Dim
 Transpose, 177
math-transformations-sectionReverse
 Layers, 184
math-transformations-sectionRotate
 90deg, 175
math-transformations-sectionVolume
 Transform, 183
Math/Arithmetic/Absoluet Value, 170, 213
Math/Arithmetic/Add X, 209
Math/Arithmetic/Div X, 211
Math/Arithmetic/Invert, 210
Math/Arithmetic/Log, 214
Math/Arithmetic/Max, 208
Math/Arithmetic/Mul X, 207, 213
Math/Arithmetic/Sub X, 211
Math/Arithmetic/Z Limiter, 212
Math/Arithmetic/Z usr rescale, 209
Math/Background/Despike1d, 146
Math/Background/Despike2d, 159
Math/Background/E Regression, 134
Math/Background/Gamma, 125
Math/Background/Line, 129
Math/Background/Line Regress, 127
Math/Background/Lineregression, 126
Math/Background/Multiply Const, 130
Math/Background/Pass CC, 135
Math/Background/Plane 3 Points, 133
Math/Background/Plane max prop, 128
Math/Background/Plane Regression, 126
Math/Background/Stop CC, 131
Math/Background/Sub Const, 135
Math/Background/Template, 132, 147, 160
Math/Background/Timescale FFT, 133
Math/Background/Waterlevel, 130
Math/Background/Z drift correct, 136
Math/Convert/make test, 139
Math/Convert/to byte, 137
Math/Convert/to complex, 139
Math/Convert/to double, 142
Math/Convert/to float, 140, 142
Math/Convert/to long, 141
Math/Convert/to short, 138
Math/Convert/to short fix, 140
Math/Filter 1D/Diff, 148
Math/Filter 1D/Ft1d, 150
Math/Filter 1D/Koehler, 148
Math/Filter 1D/Lin stat diff, 150
Math/Filter 1D/Repair, 145
Math/Filter 2D/Curvature, 154
Math/Filter 2D/Edge, 161
Math/Filter 2D/IFT 2D, 159
Math/Filter 2D/Line Interpol, 155
Math/Filter 2D/Lineinterpol, 158
Math/Filter 2D/Local height, 161
Math/Filter 2D/Normal Z, 162
Math/Filter 2D/Small Convol, 157
Math/Filter 2D/Smooth, 156
Math/Filter 2D/Stat Diff, 157
Math/Filter 2D/T derive, 155
Math/Filter 2D/t dom filter, 149
Math/Misc/Find Loc Max, 173

- Math/Misc/Layersmooth, 171
 Math/Misc/Minzlayer, 169
 Math/Misc/PSD Add, 172
 Math/Misc/Shape, 172
 Math/Misc/Spectrocut, 170
 Math/Misc/Vorlage, 198
 Math/Misc/Workfuncextract, 174
 Math/Probe/AFM_mechanical_sim, 168
 Math/Probe/Probe_Image_Extract, 166
 Math/Statistic/Auto Correlation, 202
 Math/Statistic/Cross Correlation, 200
 Math/Statistics/Add Trail, 206
 Math/Statistics/Angular Analysis, 204
 Math/Statistics/Average X Profile, 206
 Math/Statistics/Histogram, 204
 Math/Statistics/Nndistribution, 194
 Math/Statistics/Opencvmatch, 189
 Math/Statistics/Opencvrecenter, 196
 Math/Statistics/Polar Histogram, 193
 Math/Statistics/Slope Abs, 205
 Math/Statistics/Slope Dir, 203
 Math/Statistics/SPALED Simkz, 200
 Math/Statistics/stepcounter, 201
 Math/Statistics/Vacancy Line Analysis, 191
 Math/Statistik/SPALED Sim., 195
 Math/Transformation/Affine, 186
 Math/Transformation/Octo Corr, 182
 Math/Transformation/Rotate, 181
 Math/Transformation/Shear X, 176
 Math/Transformation/Shear Y, 177
 Math/Transformation/Shift Area, 185
 Math/Transformations/Auto Align, 179, 182
 Math/Transformations/Flip Diagonal, 179
 Math/Transformations/Merge H, 184
 Math/Transformations/Merge V, 187
 Math/Transformations/Mirror X, 180
 Math/Transformations/Mirror Y, 187
 Math/Transformations/Quench Scan, 176
 Math/Transformations/Scale Scan, 178
 Math/Transformations/Unwrap, 183
 Tools, 256
 Tools/Converter , 113
 Tools/external_converter , 115
 Tools/Hardware Info, 254, 255
 Tools/NetCDF-View, 215
 Tools/Pan View , 218
 Tools/Plugin Details, 255
 Tools/Probe Indicator, 217
 Tools/Pyremote Console, 253
 Tools/SPM SIM Control, 257
 Tools/SR-DSP Control, 277, 280, 281, 359
 windows-section Inet JSON Scan External Data, 95
 windows-sectionDSP PeakFind, 99
 windows-sectionNano Manipulator, 100
 windows-sectionNano Plotter, 98
 windows-sectionRHK Scan Control, 103
 windows-sectionSPA-LEED Ctrl, 96
 windows-sectionSPM Scan Control, 102
 History, 3
 Horizontal, *see* View
 Installation, 39, 40
 License, 371
 Logarith., *see* View
 Main Window, 53
 make, 40
 make install, 40

- MK3-Smart Piezo Drive, 358
- Periodic, *see* View
- PlaneSub, *see* View
- PlugIn
- (DocOnly) Egress, 134
 - (DocOnly) LineInterpol, 155
 - (DocOnly) LineRegress, 127
 - abs_scan, 213
 - add_scan, 209
 - add_trail, 206
 - affine, 186
 - AngularAnalysis, 204
 - ascii_data_im_export, 118
 - autoalign, 179
 - autocorrelation, 202
 - average_profile, 206
 - baseinfo, 201
 - bg_z_drift, 136
 - bggamma, 125
 - bin_import, 123
 - Converter, 113
 - crosscorrelation, 200
 - cube_import, 106
 - curvature, 154
 - demo_hwi, 281
 - despike1d, 146
 - despike2d, 159
 - diff, 148
 - div_scan, 211
 - DSPPeakFind, 99
 - edge, 161
 - editnc, 215
 - external_converter, 115
 - extra_scan_info, 254
 - findlocmax, 173
 - flip_diagonal, 179
 - ft1d, 150
 - ft2d, 158
 - G_dat_Im_Export, 112
- gmeyer_im_export, 119
- grab_v4l, 282
- histogram, 204
- ift2d, 159
- inet_json_external_scandata, 95
- innovative_dsp_hwi, 281
- invert_z, 210
- kmdsp_hwi, 280
- koehler, 148
- LAN_rhk_hwi, 279
- layersmooth, 171
- linear_stat_diff, 150
- lineinterpol, 158
- lineregression, 126
- listplugins, 255
- local_height, 161
- log_z, 214
- make_test, 139
- make_volume, 170
- mandriftfix, 182
- max, 208
- merge_h, 184
- merge_v, 187
- minzlayer, 169
- mirror_x, 180
- mirror_y, 187
- mkicons, 256
- movieconcat, 178
- mul_scan, 207
- mulconst, 130
- multi_dim_transpose, 177
- nano_manipulator, 100
- nanoimport, 121
- NanoPlott, 98
- nndistribution, 194
- normal_z, 162
- OctoCorr, 182
- Omicron_IO, 120
- opencvmatch, 189

opencvrecenter, 196
PanView, 218
parabolregress, 129
pass_cc, 135
plane3pkte, 133
plane_max_prop, 128
plane_regression, 126
png_Im_Export, 122
polarhist, 193
primitiveimexport, 117
printer, 254
probe_image_extract, 166
ProbeIndicator, 217
psdadd, 172
PsiHDF_im_export, 109
pyremote, 253
quenchscan, 176
queryDSPInfo, 255
quicktime_im_Export, 109
removelineshifts, 129
repair_cs, 145
reverse_layers, 184
RHK2000-Import, 113
rhk_scancontrol, 103
rhk_spm32_import, 115
rotate, 181
rotate90, 175
scalescan, 178
sdfimport, 119
shape, 172
shear_x, 176
shear_y, 177
shiftarea, 185
short_to_short, 140
SlopeAbs, 205
SlopeDir, 203
smallconvol, 157
smooth, 156
spa4_im_export, 123
SpaLeedControl, 96
spasim, 195
spasimkz, 200
spectrocut, 170
spm_scancontrol, 102
spm_simulator_hwi, 257
sranger_hwi, 277
sranger_mk2_hwi, 359
stat_diff, 157
stop_ccr, 131
sub_scan, 211
subconst, 135
tc211_ccd, 280
Tderive, 155
template, 132, 147, 160
timedomfftfilter, 149
timescalefft, 133
to_byte, 137
to_complex, 139
to_double, 142
to_float, 142
to_long, 141
to_short, 138
UK2k_import, 114
uksoft2001_im_Export, 107
Unnamed, 168
unwrap, 183
uto_float, 140
v5d_Export, 112
VacancyLineAnalysis, 191
volume_transform, 183
vorlage, 198
waterlevel, 130
wip_im_Export, 105
workfuncextract, 174
WSxM_io, 124
Z_limiter, 212
Z_rescale, 213
Z_usr_rescale, 209

- Preface, 3
- Preferences, 83
 - Adjustments, 91
 - DataAq, 89
 - GUI, 92
 - Hardware, 83
 - Instrument, 85
 - Instrument-SPA, 88
 - Instrument-SPM, 86
 - Paths, 91
 - Probe, 91
 - User, 91
- prefix, 41
- Probe-Event, 65
- Profile 1D, *see* Channel View
- Quick, *see* View
- Quickstart, 45
- Scaling, *see* View Range
- Scan-Event, 65
- Scene Setup, 66
- Settings
 - Reset, 45
 - Restore, 45
 - Store, 45
- Signal Ranger MK2 Feedback, 287
- Signal Ranger MK2 HwI, 282
- Signal Ranger MK2 Scan Control , 287
- Signal Ranger MK2/3 Auto Approach, 313
- Signal Ranger MK2/3 Coarse Motions, 313
- Signal Ranger MK2/3 HwI, 282
- Signal Ranger MK2/3 LockIn, 307
- Signal Ranger MK2/3 Manipulation, 290
- Signal Ranger MK2/3 Mover, 313
- Signal Ranger MK2/3 Spectroscopy, 290
- Signal Ranger MK2/3 Tracking, 307
- Signal Ranger MK2/3 Vector Probe, 290
- Signal Ranger MK3, 326
- Signal Ranger MK3 LockIn, 355
- Signal Ranger MK3 M-Servo, 353
- Signal Ranger MK3 Mixer, Z-Servo, 353
- Signal Ranger MK3 PAC/PLL Tuning, 344
- Signal Ranger MK3 Signal Scope, 342
- Signal Ranger MK3 Signals, 329
- Signal Ranger MK3-Pro/PLL, 326
- Signal Ranger MK3-Smart Piezo Drive, 358
- Signal Selection in GXSM, 343
- Smart Piezo Drive, 358
- SPA–LEED, 61
 - CPS high/low, 61
- SPM SIM HwI / Template Code, 257
- Starting GXSM-3, 51
- Surface 3D, *see* Channel View
- SVN
 - update, 41
- System requirements, 39
- Tipps and Tricks, 47
- troubleshooting, 41
- update, *see* SVN
- User-Event, 65
- View
 - Diff, 59
 - Direct, 59
 - Direct HiLit, 59
 - Horizontal, 59
 - Logarith., 59
 - Periodic, 59
 - PlaneSub, 59
 - Quick, 59
- View modes, *see* View
- View Range, 60
 - automatic, 60

manual, 60
tolerant, 60
VOffset Z, 60
VRange Z, 60
Visualisation, 59
Visualization modes, 62