The GXSM manual

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

April 4, 2014

Preface to the GXSM manual

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 Schutzhaube 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 einzuhauchen ...

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 vonL.Anderson und wurde von H.Bethge zum Messen eingesetzt.

Gleichzeitig produzierte R.Kumpe einige Assemblerroutinen zur Ansteuerung seiner Datatranslation-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 Höchtstleistung 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 gebrantmarktem DOS/Win3.X Programmierer als völlig ungeeinete Platform, so hatte ich doch die Vorzüge 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 Versuchte 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 Aufwandt ein STM nachrüsten zu können.

Das Resultat war ein Programm namens xspa, welches unter Verwendung der xforms Libary 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 PM-STM 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 Streulichapparatur (SARLS).

Die Computertechnologie schreitet unaufhaltsam vorann 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 Bilddarstellungsroutinen, 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 Aanwendungsbezogenen 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.
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 occured, 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 there 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-2 and Signal Ranger in daily data production!

It's a rainy Sunday, and the GXSM manual is going to be revised to the GXSM-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-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 an 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-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-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 some thing 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 engenieer 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 it's 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 preform 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, syncronized with sampling
- Evaluation of software based resolution enhancements, Z and X/Y HR-mode
- Real time magnitude dependent bandwith adjusting via FIR for current input
- Transition to Float in Gxsm in conjunction with tarnsfer of full statistical data (32-bit resolution summed values from DSP + normalization count)
- FIFO data transfer optimazation: 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-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 easy 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 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 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 is hardware speeds are moving up we will be able follow up the bandwidth! DSP code porting in 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 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 "Arosa Express" is out – get ready for more remote and freely via Python programmable SPM actions! Now VP probing and Mover/Autoapproach can by 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 an 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-siganls, counters, control values (liek 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 moudule 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 a input stage with the signal input and two additional adding inputs (modulation, etc.) with

Figure 0.1: Idea Sketch: Modules and signals.

optional gain.

To handle all this a new cleaned up python basic based support class and a configurator script with monitoring goodies and a signal grph 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 mroe but allow to choose form 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 nex 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 **??**.

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

Figure 0.2: Idea Sketch: Example SPM configuration.

Contents

1 Front Matter

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

Copyright (C) 1999, 2000, 2001, 2002, 2003 Percy Zahl and Andreas Klust
Email: zahl@users.sourceforge.net, klust@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: **??**

2 Introduction

GXSM[1] – Gnome X Scanning Microscopy – is a powerful graphical interface for any kind of 2D and 3D (multilayered 2D mode) data acquisition methods, especially designed for scanning probe microscopy (SPM).

GXSM provides various methods for 2D data (of various types: byte, short, long, double) visualization and manipulation. We are currently using it for scanning tunneling microscopy (STM), atomic force microscopy (AFM), scanning angle resolved light scattering (SARLS), and spot profile analysis of low-energy electron diffraction (SPA-LEED). The project was founded at the Institute for Solid State Physics [2] Leading developer of GXSM is Percy Zahl[3]. The program was developed for Linux using the Gtk+/Gnome libraries for the graphical interface.

And best of all: it's free: GXSM is licensed under the terms of the GNU General Public License (GPL, see **??**). Therefore, everyone can copy, use, and modify GXSM 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:

- Support for STM, AFM, SPA-LEED, SARLS, and any other 2D (2D layered and multi-channel) data-acquisition method can be supported by GXSM.
- The GXSM 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
  - "PlaneSub": on-the-fly 3-point defined plane subtraction
  - "Logarith.": logarithmic scaling mode, almost used by diffraction methods
  - "Horizontal": automatic line average subtraction
  - "Diff.": 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.

[1] The Project can be found in the Internet at **http://gxsm.sourceforge.net**

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

[3] e-mail: zahl@users.sourceforge.net

- 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 MesaGL) which now offers a sophisticated scene setup.
- The "high-level" scan controller is now separated from the GXSM 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:
    - Math operations on 2D data:
    - 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! 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 **??**
    - Digital Instruments/Veeco Metrology Group, NanoScope (import) (**http://veeco.com**, see **??**)
    - Omicron NanoTechnology, Scala (**http://www.omicron.de**, see **??**)
    - WSxM/Nanotec Electronica SPM, WSxM (**http://www.nanotec.es**, see **??**)
    - SDF (Surface Data Format, see **??**)
    - UK2000 v3.4, see **??**
    - G. Meyer STMAFM, see **??**
    - Park Scientific (AFM, basic import support, see **??**)
    - "d2d" data format (SPA–LEED), used by "spa4" and "xspa", see **??**
    - any NetCDF file containing a 2D data array
    - Grey Images in .pgm format (P5 type), see **??**
    - PNG image format, see **??**

    - Targa (.tga) export, in 8/16/24bits color depth, **??**
  - 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)
    - SPA-LEED: Peak Finder, Focus-Tool, Rate-Meter, Peak Intensity Monitor (used with SPA-LEED test module)
    - SPA-LEED Control (used with SPA-LEED test module)
    - CCD control (very special, just for gurus)
  - 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 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, change the scan speed[4] 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!

[4] depends on hardware solution used, e.g. PC31, PCI32, SRanger

- 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 scanning progress is fully scriptable using Python. This is extremely useful to program tricky and or long SPA-LEED scans. 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 available[5].
- GXSM 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.
- Included Add Ons:
- Linux COFF loader to launch the DSP program on the SRanger hardware (old: PC31, PCI32 cards are not any longer GXSM main stream). See the SRanger Linux project[6].
- the required kernel modules
- A universal SRanger DSP binary (precompiled, in COFF format).
- A set of external utilities written in Python for monitoring and configuring the DSP board.

The GXSM software can be divided into three parts: First, the GXSM 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 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 only for data analysis purposes.

[5] **http://gxsm4w.sf.net**

[6] included in the SRanger project related to Gxsm, refer to **http://sranger.sourceforge.net**

3 GXSM Project Installation

3.1 System requirements

GXSM needs a reasonable up-to-date machine running a recent version of almost any Linux variant. By the time reading year 2007, I'd recommend the Debian based Ubuntu. System memory requirements are ranging from little to several Gigabytes if you want to deal with big scans, movies or multidimensional data sets ☺.

GXSM also runs on a Power-PC/G4/G5 platform running Ubuntu-PPC or OSX (no hardware support for OSX).

The GXSM4Windows port supports SRanger based hardware.

To compile and run GXSM the full Gtk+/Gnome libraries, development packages and some additional libraries (FFTW, NetCDF, libQuicktime) are required, but all are available as packages for Debian and Ubuntu.

GXSM was build on a Debian, Ubuntu, Fedora, RedHat, Suse, Mandrake Linux. The recent Ubuntu (autumn 2012: 12.04 LTS) is the GXSM reference system and is used for development. For the reference system up to date debian packages are released via launchpad PPA[1] A ready to use Ubuntu remix is available for download[2] including not only GXSM but also the required libraries and a few useful add-ons. For more Linux distribution dependent installation guides and hints please refer to the on-line "GXSM Installation Knowledge" forum:
http://sourceforge.net/forum/forum.php?forum_id=297458

3.2 Getting GXSM-2 from CVS

Go to the GXSM Project's home page at **http://sourceforge.net/projects/gxsm** and follow the link almost at the bottom "CVS Repository" and have a look at the instructions there. The "module" is "Gxsm-2.0", also the module "Gxsm-Manual" can be checked out.

3.2.1 Configuration

Change into the Gxsm-2.0 directory and type

[1] You have to add the following sources to your APT-system:
deb **http://ppa.launchpad.net/totto/gxsm/ubuntu** YOUR_UBUNTU_VERSION_HERE main
deb-src **http://ppa.launchpad.net/totto/gxsm/ubuntu** YOUR_UBUNTU_VERSION_HERE main.

[2] see **http://www.ventiotec.de/linux/GXSM-Linux.iso**

$ ./autogen.sh

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

3.2.2 Compilation

Type make in the Gxsm-2.0 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-2.0 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:

$ make install (root privileges needed[3])

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 finger out what's missing, refer to **??**.

♪♫ *For a minimal Gxsm only installation just run* **$ make install** *in the following sub-directories: src, plug-ins, tips*

☺ For automatic thumb-nailing within Nautilus and setting up other goodies, check out the GXSM *Tips and tricks* forum:
http://sourceforge.net/forum/forum.php?forum_id=302195

3.2.4 Alternative installation using "stow"

If you have the "stow" utility installed and like to sort your local software, check out GXSM as described above. Then use the following commands to set up the source code, compile it and install the binaries:

$ ./autogen.sh −−prefix=/usr/local/stow/gxsm2

online **$ make**

$ stow gxsm2

[3] root: However, this depends on the access rights of the directory given as "prefix" before.

The use of "stow" might be convenient to test different version of GXSM. To sort them out you set up the source code with a new prefix as follows

$ ./autogen.sh −−prefix=/usr/local/stow/gxsm2-test

Then compile the code in the usual way and install it via

$ stow newprefix

To remove the old installation use

$ stow -D oldprefix

For further details on the "stow" tools please call the man pages via **$ man stow**!

3.2.5 Alternative installation using debian packages

If you want the package manager to take care our your GXSM-installation you can also use **$ [sudo] checkinstall make install** to copy the binary files to the installation path defined by your proper prefix (default: /usr/local). Then you can remove GXSM by **$ [sudo] dpkg purge/deinstall gxsm**. Please note that the debian-packages produced by "checkinstall" do not include the correct dependencies so that you should not distribute these packages.

Please be aware that by using "checkinstall" and having set up the path to the GXSM-PPA on launchpad your installation might be overwritten by the updating your whole system.

You can also do it the full debian way and make your full debian package: The folder /Gxsm-2.0/debian hold the full set to maintain the debian packages stored in the PPA on launchpad. To use it you have to set up a proper debuild environment.

3.2.6 Updating

For later updates just run

$ cvs update

in the Gxsm-2.0 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/gxsm2-test ; make; make install

The update process can be automated by running a python script ☺
$ /Gxsm-2.0/tools/update_GXSM_SRanger.py. The script can also be used to update via APT but please do not mix up both ways of installation because they use different pathes to store the GXSM-binaries.

3.2.7 Trouble shooting

If your configuration or compilation fails, most likely something is missing and needs to be installed before trying again. The CVS source tree is kept in a sane state which should always

allow a error free compilation. This is the golden-rule of CVS usage and all developers are strongly encouraged to adhere this – and as far as I remember the occurrences of bad checked in changes 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, rather than in the current development environment. A knoonlinewn issue is the fact, that the current GXSM-2 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 should 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!

Second, have look if the files, that are noted missing in your error messages are really not there or just installed in any unusual place.

compilation/linking If this fails, something is messed up with you 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.

call gxsm2 Please do not mix up different installations of GXSM. The installation from the CVS and the one from the APT repository usually use different folders to store the plug-ins and so on. So if you intend to use the latest version from the CVS repository deinstall your deb-packages

$ sudo apt-get purge gxsm sranger

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

3.3 Getting GXSM via APT

If you want to contribute to the GXSM-project we strongly recommend to use the latest source code available via the CVS repositry on sourceforge.net. In this case you can actively commit new code. But for just using GXSM (on an Ubuntu based system) there is maybe an easier way by using your package manager APT.

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

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

Then refresh you local database:

$ [sudo] apt-get update

and install the GXSMpackage via

$ [sudo] apt-get install gxsm [sranger sranger-modules-std-dkms sranger-modules-mk23-dkms]

The last command will not only install gxsm (and the optional sranger packagesin the [] brackets) but also fullfil the required dependencies.

Once you have installed GXSM you will be noticed by your package manager (in the lastest Ubuntu version this is called Ubuntu Software Center) about updates. You can also use the script **$ update_GXSM_Sranger.py** to do the job.

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

Part I

The GXSM core

4 Program Start

This chapter describes the very basics of GXSM [1] usage: the start of the program. Some of its behavior can be modified using the command line parameters described in section **??**.

4.1 Command line parameters

Usually just type – no special options at all:
$ gxsm2
Or for temporary hardware disabling, very handy for data analysis on your system configured for hardware usage by default:
$ gxsm2 -h no

It is recommended to use this as default handler of *.nc Mime-File types, i.e. used by Nautilus to launch GXSM while clicking a data file. ☺

GXSM uses the Gnome/Gtk+/Glib libraries. This offers a simplified and flexible use of command line parameters. Type
$ gxsm2 – –help
to get a list of all available options. Only the last section (GXSM specific options) being relevant for GXSM configuration:

```
pzahl@charon:~/Gxsm-2.0$ gxsm2 --help
Usage: gxsm2 [OPTION...]

Help options
  -?, --help                  Show this help message
  --usage                     Display brief usage message

Application options
  -h, --Hardware-Card=default Hardware Card: no | ... (depends on
                              available HwI plugins)
  -u, --User-Unit=default     XYZ Unit: AA | nm | um | mm | BZ |
                              sec | V | 1
  --disable-plugins           Disable default plugin loading on
                              startup
```

[1] GXSM mainstream versions are “gxsm” and the newer “gxsm2” (more recent and Gnome-2/gcc3.x compilant version)

```
--force-configure           Force to reconfigure Gxsm on startup
--force-rebuild-gconf-defaults Forces to restore all GXSM gconf
                            defaults on startup
--debug-level=INT           Set Gxsm debug level. 0: no debug
                            output on console, 1: normal, 2: more
                            verbose, ...5 increased verbosity
--pi-debug-level=INT        Set Gxsm-PlugIn debug level. 0: no
                            debug output on console, 1: normal,
                            2: more verbose, ...5 increased
                            verbosity
... more default options ...
```

Parameter	Description
-h, –Hardware-Card=*type*	set up type of hardware
no	do not use any hardware
...	depends on available HwIs
-u, –User-Unit=*unit*	XYZ unit
AA	Å, $1e-10$ m
nm	nm, $1e-9$ m
–disable-plugins	Disables the loading of plugins on startup mainly for debugging.
–force-configure	force to reconfigure Gxsm on startup (via Druids)

Table 4.1: 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 (**??????????????????????**) are autodetected.

4.2 First start of Gxsm

Starting with GXSM version 1.1.13 a new GXSM user is guided via a Configuration Druid to get the most essential settings right **??**. I expect this Druids to be well self-explaining. During the setup with the Druid you will not be able to set Hardware/Card, i.e. SRanger:SPM or SRangerMK2:SPM, because no HWI is loaded. Please select the right hardware the field below, i.e. /dev/sranger0 for SR-STD/SR-SP2 boards or /dev/sranger_mk2_0 for the SR-MK2/SR-MK3Pro boards. The later are used in the Open Source SPM controller. To set the Hardware/Card field, you have to restart GXSM and set it via Preferences.

[1]This is the default, the device location can be configured using the *GXSM preferences dialog, folder Hard-*

Figure 4.1: First two pages of Gxsm Configuration Druid.

ware/Hardware/Device

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

5 The main window

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

Figure 5.1: Gxsm Main Window, `SPM`-variant left, `SPALEED`-variant right.

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.

5.1 Understanding the main window's entries: SPM mode

This section explains the contents of the main window (see Fig. **??**) 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.

5.1.1 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. 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, 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 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 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. **??**.

5.1.2 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 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. Nethertheless, 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'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, but can be exported to ASCII, consider using `ncdump` and `ncgen` to create a nc-file from your data. Running ncdump on any GXSM-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. ☺

5.2 Understanding the main window's entries: SPA–LEED mode

The SPA–LEED version of the main window is very similar the the SPM variant. There are some changes and additional entries: The SPM *Range* entry is replaced by the SPA–LEED typical parameter scan *Lenght*. There is no *Steps* entry, it is always calculated from *Lenght*/*Points* and therefore the panel on right holds now two set of radio buttons to switch between Volts/%BZ for length inputs and eV/Phase (S) for to energy input.

In addition the SPA–LEED GUI adds inputs for *Gatetime* and *Energy*. For easy setting of the displayed (mapped) dynamic range there are entries for *CPS high* and *CPS low*.

The SPA-LEED GUI offers to the already from SPM known *Offset* an additional Offset *Offset (0,0)* which becomes handy, if you need an Offset to center the diffraction pattern, e.g. the (0,0) spot, to the center and want to zoom in some higher order spots, therfore you can use the regular *Offset* starting from (0,0). Both offset are always internally added. The units of *Offset (0,0)* is always Volt, while the regular *Offset* switches between Volt/BZ consisten with *Length*.

This especially becomes useful using remote control programming – see section **??**.

5.3 Printing large numbers of scans

main menu/Tools/Make Icons allows to print-out series of images for browsing and quick survey. You enter a regular expression in *Mask* and a path in *Path* and all files fitting into this expression are printed into a postscript-file *Icon File* for later (or immediate) printing. If the *Icon File* exists already, the new images are simply appended to it.

Gxsm can produce these icon pages in several layouts. You may select *2x3*, *4x6* or *6x9* images per page resulting in less and larger or more and bigger icons of your scans. The image resolution is scaled to printer-appropriate values to decrease render time for the printer and to minimize to risk of memory shortage for your page-printer.

A number of options can be selected from pull-down menus to improve your print-out. Play with the possibilities to find your favorite combination. The avaible options are described in Tab

option	value	description
Paper		Papersize
	A4/letter	Switch between DIN A4 and US letter
Resolution		Optimize the output for given printer
	300/600/1200dpi	Printer resolution
E Regression		Plane removal
	no	Off
	E 30/5%	The outer 30 or 5% of the scan are omitted for plane regression
L Regression		Background substraction
	no	Off
	lin. Reg.	Linear background from each line
View Mode		Same as in main window
Auto Scaling		Contrast and brightness calculation
	default	uses stored settings (as set at time of save)
	auto 5/20/30% Rand	Omitting outer scan area for calculation
Scaling		additional bitmap scaling
	min-max	scale to full 8 bits
	cps-lo-hi	use CPSHi/Low (SPA-LEED)

Table 5.1: Options for the *main menu/Tools/Make Icons* function.

??.

5.4 Drag and Drop

Gxsm accepts all loadable files via 'drag and drop', e.g. from the Gnome-midnight-commander, Nautilus, or Netscape Navigator. 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.

5.5 Keyboard-Accelerators

Using the keyboard for common tasks greatly improves productivity. To allow keyboard-shortcuts (called *accelerators* in Gnome) you have to modify an entry in the Gnome-Configuration. You can use the Gnome Configuration Editor gconf-editor[1] to change the entry

`/desktop/gnome/interface/can_change_accels`

[1]Debian package: gconf

to `enabled`.

If you run GXSM on a recent linux distribution (like Ubuntu 12.04 LTS) with Gnome-3.4 or higher you have to use the dconf-editor[2] using the key

`org/gnome/desktop/interface/can_change_accels`.

[2]Debian package: dconf-tools

6 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 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.

6.1 The channel dialog

Figure 6.1: Channel Selector Window

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 **??** for details:

1. No During data aquisition no data are displayed.
2. Grey 2D The data are displayed as a grayscaled image in an own window.
3. Surface 3D Three dimensional view of your data in an own window.

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 chosing $2>$ or <2 an *experimental* mode is activated in which each scanline is scanned twice, i.e. this mode is used for magnetic force measurements.

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

7 Visualization

As a multi purpose 2D/3D data visualization, acquisition and manipulation system GXSM provides a set of powerful graphical presentation modes. Available are 2D grey/false color, profile and 3D representations. The internal objective structure of GXSM 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.

7.1 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 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 diplay 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 in the GXSM main window.

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

PlaneSub 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 GXSM main window or automatically calculated to fit a selected rectangular area.

Logarith. 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:

$$Grey/Colornumber \propto \log\left(1 + |Data - Min| \times Contrast + Bright\right),$$

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.

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

Diff displays a averaged and weightened X-derivative, similar to the Koehler filter **??**, but using a smaller averagening range (16 pixels).

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.

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

7.1.1 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 s*ets the data Z-Range which should be mapped to full color space.

*VOffset Z entry s*ets 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 t*o 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 **??** is selected the selected area will be scaled automatically to full color space available. *VOffset Z entry i*s always set to zero by *toolbar/Autodisp*.

If your scan has some distorsions/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.

7.1.2 Using a palette: false color mapping

7.1.3 Custom and GXSM 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 *toolbar/Autodisp*.

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 1024 RGB-entries (R, G, B values in the range 0 ...255) just following the header. Use *gimp* to generate this conveniently. [1]

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

7.1.4 Additional Information

In particular during data acquisition it is convinient 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 active the display of additional meta-data of the image by *main menu/View/Side Pan*. That will show you on the different tabs the paramaters/varialbes of the nc-file, Probe Events, User Events, and a list of the Objects.[2].

7.2 Visualization modes

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

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

[2] The list of objects is currently not autoupdating (Gxsm-2 1.40.0 'Sandy Experiment')

7.2.1 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.

7.2.2 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:

Figure 7.1: Grey 2D view with all types of objects. The currently active object (colored nodes) is the Poly-Line object.

```
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).

Popup Menu

Tthe *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 **??**.

File load/save scan data in this channel,
load/save objects (**??**),
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).

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 **??**.

Objects set type of object to create, see **??**. *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 and Grid) 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 (**??**).

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 menue *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.

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

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).

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 off[4].

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"Scan Event" holds a position and possible more specific information like what happened. I.e a simple bias change (User-Event-Entry) or a full set of probe data (Prove-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 you personal hot-keys with the pop-up, just select an pop-up entry and press a key!

E.g: Set [=],[−] for *Grey 2D view popup window View/zoom in,out*!

7.2.3 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, see **??**. The popup offers some options for controlling the 3D view:

[4] Showing 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.

`GL Options` Quick access to the display options
Zero Plane (enable/disable showing of the "bottom box"),
Mesh (use meshing instead of solid rendering),
Smooth (use smooth color mode, looks better, but is slower),
Tickmarks (enable/disable 3D tick marks – sorry, not yet available).

All these options are available via the `Scene Setup` *too.* ♪♫

`3D Control` Select the function of the Mouse Wheel, e.g. select zooming (default), rotation around X/Y/Z axis or scaling of height.

`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

This dialog allows a sophisticated 3D rendering configuration, it's parameter names are close the the original GL-naming, which are effected. A GL knowledge is not necessary, but helps understanding the rendering process. The configuration is splitted into the following folders:

View Setting of the object (surface) orientation, using *Rotation X,Y,Z* and *Translation*.
FoV sets the field of view.
Dist sets the distance from observer to object, relative to width of surface.
PerspZfar sets the far limit of rendering, everything in greater distance (relative to surface width) is not shown!
PerspZnear sets the near limit of rendering, everything closer than this distance (relative to surface width) is not shown!
Hskl use this to scale the height of the surface, relative to width of surface.

Light Setup of light sources. Set a ambient light intensity/color. And switch on up to three lights at specified positions and with given properties, press the *Help button* to get help.

Use the color "Value" to set light intensity! ☺

Surface Material Setup ambient, diffuse and specular color reflectance of the surface material. The *Shininess* sets the amount of specular light reflected by the surface, depending on angle of incidence.

It is also possible to use different sources (data channels) as a color map:
ColorMode: select from *Material, Terrain, Palette* to set the color space to be used for mapping. The setting *Material* will use unicolor mapping, as defined above. *Palette* will use the currently activated Gxsm palette, as set in preferences.
ColorSrc: select from *Uniform, Height, X-Chan*.
Uniform will disable color mapping and uses (please set set *ColorMode* to *Material*) the material color.
Height will use the height (Z-value) for color indexing.
X-Chan will use the Z-value of the first channel set to Mode "X" for color indexing. In case the sizes are differing, Gxsm will automatically shrink/expand the color source to cover the full 3D surface.
Using *Color Contrast, Offset, Saturation* the color mapping can be adjusted for best results. Set Contrast to 1, Offset to 0 and Saturation to 1 for full range mapping.

Box Material Box material properties, similar to surface material.

Render Options Please refer to the provided *Help* buttons.

Tips

Gxsm uses a very efficient automatic method to reduce the number of points used (subsampling) for the 3D grid construction. The resolution depends on render window size and the *RenderOption/QuenchFactor* setting. Use a small window and/or a setting of 1 or higher for the Quench Factor for adjusting the perspective view and colors, once satisfied, decrease the Quench Factor to about 0.5 and maximize the render window to get a high quality 3D image. Use *The Gimp* (File/Acquire/Screenshot) for getting a screen show and save this!

Be careful setting QuenchFactor to zero if the source data set has a huge number of pixels – this could quickly end up with an out of memory after a while of swapping and will end up in a termination of Gxsm, killed by system! There are several float vectors (coordinates, normals, color, etc.) stored for each pixel!

7.2.4 The general purpose "Profile 1D" view

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

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 activated[5].

[5] There 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.

Figure 7.2: Profile as used by the Show-Line object.

Pop-up Menu

The popup menu of the *Profile View* offer the folowing 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.
- **Symbols** – not yet available –
- **Legend** – not yet available –
- **Tickmarks** Enable display of tickmarks
- **no Gridlines** Disable gridlines

Y Scaling Several Y scaling options[6]:

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

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.

☺ 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 –

Cursor This menu allows to enable up to two cross hair cursers (blue and green). Use the left mouse button to dragg 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 profie data can be saved in Ascii format. The file is preceeded 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
# 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. This 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-matplot* to show your profile. You can export to many bitmap formats from this. Python-matplot 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

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.

[7] Follow 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

8 Configuration

All your settings are saved using the gconf database[1].

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 the configuration wizard will pop up to create your settings from the defaults.

With Settings/Preferences (dt: Einstellungen/Einstellungen) GXSM 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.

8.1 Hardware folder

Figure 8.1: Screenshot of the GXSM preferences folder, hardware page. Here the connected hardware type and device is specified. Shown are the settings for the Signal Ranger.

Hardware/Card Choose hardware: see also section **??** about Hardware-Interface (HwI) plug-ins for specific details.)

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

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`, ...

8.2 Instrument folder

Figure 8.2: 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.

8.2.1 Inst-SPM folder

Figure 8.3: GXSM 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: see **??**, **??**

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): see **??**, **??**

Analog/VoltMaxIn AD Voltage corresponding to DigRangeIn: see **??**, **??**

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): see **??**, **??**

Analog/VoltMaxOut DA Voltage corresponding to DigRangeOut: see **??**, **??**

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.

8.2.2 Inst-SPA folder

Instrument/X,YCalibV SPALEED: X,Y calibration factor, 1V at DA $\rightarrow \pm 10$ V resp. 15 V at octopol front/back.

Instrument/EnergyCalibVeV factor*Volt = energy in eV

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

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.

8.3 DataAq folder

Figure 8.4: 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 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, but four input-lines are adressed to each of them via a 4x-multiplexer, which results in eight usable analog input lines. Name*

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

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 GXSMdata sources is defined as shown here:

Channel Descriptor	Name	Notes
DataAq/PIDSrcA1	Topo,*	Z: generated by the Feedback
DataAq/PIDSrcA2...4	—	not used
DataAq/DataSrcA1	AIC5	AIC5: I, Force, dFrq, ... what is used as feedback signal
DataAq/DataSrcA2	AIC0	AIC0 input (Fric)
DataAq/DataSrcA3	AIC1	AIC1 input (Damp., opt. FUZZY source)
DataAq/DataSrcA4	AIC2	AIC2 input
DataAq/DataSrcB1	AIC3	AIC3 input
DataAq/DataSrcB2	AIC4	AIC4 input
DataAq/DataSrcB3	AIC5	AIC5 input
DataAq/DataSrcB4	AIC6	AIC6 input
DataAq/DataSrcC1	dIdV	LockIn 1st of AIC5 (32bit)
DataAq/DataSrcD1	ddIdV	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 8.1: Signal Ranger board the hardware channel to GXSM 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.

8.4 Probe folder

– obsolete –

8.5 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.

8.6 Paths folder

Path/Logfiles

Path/Data

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

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

Path/Plugins Additional Plugin searchpath

8.7 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 Autodisplay, 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 ...

8.8 GUI folder

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

9 Programming via Remote-Control

The remote control feature of GXSM allows to automatically run scan procedures.

This version of "Remote-Control" is now obsolete and replaced by the Python-Remote mechanism, see **??**.

9.1 Using the fifo device for remote control

A fifo-device[1] is a dummy device through which you can send command sequences to X/GXSM. Default names are `remote` and `remoteecho`. `remoteecho` is for receiving information from the process.

The simplest way to use this feature is to use pipes to pass all commands and arguments straightaway into the fifo:
`cat mycommandfile > remote`
or if you want to enter your commands interactively:
`cat > remote`

You will find the file `rscript.perl` as an example in the GXSM source code. For testing purposes have a look at `remote.sh` (in `Gxsm/src`). The backchannel must be opened for reading (`cat remoteecho`). The scripts care for that. **Note:** Before interrupting the fifo you have to enter `cmd quit` or GXSM fails to notice its stop and waits forever.

The remote control can be called via Tools. Pressing Run blocks GXSM, until *both* fifos are opened. Interrupt the remote control with Stop and the remote command `cmd quit`.

Using remote control, individual channels can be accessed. The channelindex begins with 0. Channels are activated with the command `cmd chmodeA 0` (for the first channel) and can be autoscaled with `cmd autodisp`.

[1] Fifo=first in, first out, named pipe, see `man mkfifo`. You can create a fifo with `mkfifo remote remoteecho`.

9.2 Command set

The remote control understands the following commands:

Symbols:

X:	real Value
N:	int Value
S:	String

Commands:

Cmd	Arg.	Values		description
set	RangeX	X		set X-Range to Value X
set	RangeY	X		...
set	StepsX	X		
set	StepsY	X		
set	PointsX	N		
set	PointsY	N		
set	OffsetX	X		
set	OffsetY	X		
set	Rotation	X		
set	Display	X		
set	Contrast	X		
set	...	X		
set	LengthX	X		set X-Length to Value X (SPALEED)
set	Energy	X		set Energy to Value X (SPALEED)
set	...	X		
scan	start			Start Area-Scan
scan	init			Init Area-Scan only
scan	update			Update all Hardpars.
scan	line	N		Scan Line N
scan	stop			Stop Area-Scan
file	save			auto save to File **Note:** Channelindex starts with N=0!
file	saveas	S	N	channel N saved to File=SS
file	load	S	N	load to Channel N, File=SS
gnu	import	S	N	import to Channel N, GnuFile=SS
gnu	export	S	N	export from Channel N, GnuFile=SS

Commands (continued):

Cmd	Arg.	Values	Beschreibung
cmd	autodisp		autodisp on active channel
cmd	chmodeA	N	channel N to mode Active
cmd	chmodeX	N	channel N to mode entryX
cmd	chmodeM	N	channel N to mode Math
cmd	quick		Quick-View Active Ch
cmd	direct		Direct-View Active Ch
cmd	log		Log.-View Active Ch
cmd	sleepms	N	sleep for N ms
cmd	quit		close remote fifo
cmd	unitbz		set Length in %BZ
cmd	unitvolt		set Length in Volt
cmd	uniteV		set E in eV
cmd	unitS		set E as S (Phase)
cmd	echo	IamReady	Echo Meldung (Sync Script with GXSM)
cmd	logev	String	Write `String` to Log-File
analog	da0	X	set analog values [V]...
menupath	S		Activate Menupath=S
action	X		Action X, used by plug-ins see plug-in

DSP Peak Find Plugin Commandset for the SPA-LEED peak finder:
Commands Plugin `DSP Peak Find`:

Cmd	Arg.	Values	Beschreibung
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

You can also choose any action, that can be reached through menus by using the so-called menu path. This allows the execution of any math- or plug-in-operation.

```
menupath Math/Background/Line\ Regress
```

Spaces have to be escaped with a backslash.

The echo-command can be used to syncronize a scan with the script execution:

```
cmd echo Hallo
```

9.3 Sample script for ”‘Heringplots”’

More examples are in Gxsm/src/*.perl

This script offers several helpful functions, like R(), Set() or Echo():

```
#!/usr/bin/perl

# Example Script for programing Xxsm (C) PZ 1998-12

use Time::Local;
use IO::Handle;

# if you have no fifo, then
# create fifo before with "mkfifo remote"
$remotefifo = "remote";
$remotefifoecho = "remoteecho";

# now open the fifo used for remotecontrol
open(REMOTE, "> $remotefifo\0")
or the "sysopen $remotefifo: $!";
# use autoflush !!
REMOTE->autoflush(1);

# Open Echo-Fifo for status return
open(REMOTEECHO, "< $remotefifoecho\0")
or the "sysopen $remotefifoecho: $!";

# ------ Users program starts here --------
# setup scan
R("cmd chmodeA 0");
R("cmd log");
R("cmd unitS");
R("cmd unitbz");
$px=600;
Set("PointsX", $px);
Set("PointsY", 200);
$bzx=240;
Set("LengthX", $bzx);
Set("LengthY", 0);
$Xoff=0;
$Yoff=0;
```

```
Set("OffsetX",0);
Set("Offset00X",$Xoff);
Set("Offset00Y",$Yoff);
R("scan init");
$Gate=2;
Set("Gatetime",$Gate);

# now go from S=3.3 to S=7.2 !
for($l=0, $S=3.3; $S<7.2; $S+=0.02, $l++){
# Set Phase (Energy is in S-Mode)
   Set("Energy", $S);
# do some Offset correction...
   Set("Offset00Y",$Yoff+($S-3.3)*0.3/4);
# Set Length to force recalculation with new Energy
   Set("LengthX", $bzx);
   Set("LengthY", 0);
   Echo();
# wait until Energy has stabilized
   sleep(1);
# start scanline
   Scan("line", $l);
# wait for scanline done
   Echo();
}
# end scan mode
R("scan stop");

# ------- Users program ends here ---------

# say goodbye
R("cmd quit");
R("byebye");
# and close remote session
close(REMOTE);

# ---------- Program End ---------------

# some usefull subs

# Send Remote Commands in some ways...
# Syntax Example: Set("energy",72,0);
```

```
# Syntax Example: R("set energy 72");
sub R {
    REMOTE->printf("$_[0]\n");
    printf("$_[0]\n");
}

# Syntax Example: Set("energy",72);
sub Set{
    REMOTE->printf("set $_[0] %f\n",$_[1]);
    printf("set $_[0] %f\n",$_[1]);
}

# Syntax Example: Scan("line",133);
sub Scan {
    REMOTE->printf("scan $_[0] $_[1]\n");
    printf("scan $_[0] $_[1]\n");
}

# Syntax Example: Rx("file","name.nc",0);
sub Rx {
    REMOTE->printf("$_\n");
    printf("$_\n");
}

# wait for echo
sub Echo {
   $dummy="";
   REMOTE->printf("cmd echo ready-echo\n");
   printf("ping send !\n");
   REMOTEECHO->read($dummy, 11);
   printf("got echo !\n");
}
```

Part II

Plug-ins

10 Using plug-ins

The GXSM 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 core. GXSM plug-ins usually register themselfs to menus within GXSM. 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 core.

The plug-ins are automatically loaded during the start of GXSM. Instead of simply loading all avaible plug-ins, GXSM loads only the plug-ins used for the current *Instrument type* (see Sec. **??**). For instance, SPA-LEED related plug-ins are not loaded if GXSM is conbfigured for STM as *Instrument type*. During runtime, GXSM 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 avaible plug-ins are loaded regardless of the *Instrument type*.

The plug-in `listplugins` (see Sec. **??**) 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 d*isplays all avaible 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 website.

Plug-ins are the easiest method to extend the functionality of GXSM. For the most common problem of writing data manipulation functions a shell script included in the GXSM source code can be used to automatically generate the source code for a new plug-in [1]. 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 demonstrate the basics of source and destination data access methods provided by the GXSM core. For those curious hacker guys a plain plug-in template and step by step instructionset are available too (**??**).

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.

[1] Run 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.

For all other special tasks like instrument control a deeper knowlegde of the GXSM 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 project Web pages to get more help on this – you are welcome by the GXSM team.

11 Plug-Ins: control

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

11.1 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 impropper used!

Info for Plug-In: Windows/Nano Manipulator

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

11.2 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 11.1: The Nano-Plott window.

Usage

Set desired parameters for Pen Up and Pen Down, put in the path/file to your plottfile (.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.

The PCI32-DSP vector scan generator is only able to move in horizontal, vertical or diagonal direction. This limitation dose not apply to the SRanger DSP code. ♪♫

Use *hp2xx* for preview of your HPGL files, available for debian! Example file to draw a line from 0 Å, 0 Åto 0 Å, 5000 Å: ☺

```
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/Nano Plotter

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

11.3 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 controling a future SPA–LEED unit :-)

Usage

Open *main menu/Windows/SPA-LEED Ctrl* and play.

Figure 11.2: The SPA–LEED Control window.

Not intuitive?

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

Info for Plug-In: Windows/SPA-LEED Ctrl

Plug-In name:	SpaLeedControl	File:	control/SpaLeedControl.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/DSP 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/DSP PeakFind

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

11.5 RHK Scan Control (to be ported)

Description

Provides a Scan Control Window for the RHK STM 100 electronics. It connects to the GXSM 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 continous (movie) high speed (HS) frame capturing process, were 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 adquiring 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 the scanning controls, it does the whole job of data adquisitation 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/RHK Scan Control

Plug-In name:	rhk_scancontrol	File:	control/rhk_scancontrol.C
Author:	Farid El Gabaly, Juan de la Figuera	Email:	farid.elgabaly@uam.es, juan.delafi

11.6 SPM Scan Control

Description

Provides a SPM Scan Control Window and connects to the GXSM 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* (Botton 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 GXSMkeeps 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 acquisitation/sorting itself.

Usage

Used for advanced SPM data acquisitation control, open the control panel via *main menu/Windows/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).*

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 opposide 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/SPM Scan Control

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

12 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 standart 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 standart 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 **??**.

12.1 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

12.2 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	File:	hard/kmdsp_hwi.C
Author:	Marcello Carla'	Email:	carla@fi.infn.it

12.3 Demo Hardware Interface

Description

This provides a demonstartion 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_hwi	File:	hard/demo_hwi.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

12.4 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 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.

12.4.1 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 12.1: GXSM SR DSP Control window: Feedback and Scan Control 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 v*alue 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

Figure 12.2: GXSM SR DSP Control window: Advanced settings page

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 the system, tip and sample. In general CP can be about 150% of CI to gain more stability with same CI.

12.4.2 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 an 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 updated the trigger data table, toggle/enable the Trigger Enable check-box. 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 12.3: Trigger Control window

12.4.3 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 12.4: Advanced setup with Raster enabled

Figure 12.5: STS folder

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 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 f*or 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:

Tp 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:

$$N = tp * 22100Hz = 22100Hz * dU/(IV - Slope * Points)$$

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.

Figure 12.6: left: GXSM SR DSP Control window, left: Z manipulation, right: LM, lateral and Z 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 f*or 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.

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.

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.

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 f*or this). Another option is applying a charge pulse using this *PL entry d*ialog.

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.

Figure 12.7: GXSM SR DSP Control window: PL mode

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.

Figure 12.8: 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/per 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

$$\tau = \frac{\text{AC-Ave-Cycels} \cdot 8}{\text{Frq}}$$

$$\text{Frq} = \frac{22100\,\text{Hz}}{M = 128, 64, 32, 16}$$

.

There for the following discrete frequencies are available: 172.7 Hz, 345.3 Hz, 690.6 Hz, 1381.2 Hz.

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:

$$U_{\text{LockIn0}} = \sum_{i=0}^{N-1} U_{in,i}$$

$$U_{\text{LockIn1stA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i\frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn1stB}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i\frac{2\pi}{M} + \text{Phase-B})$$

$$U_{\text{LockIn2ndA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i\frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn2ndA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i\frac{2\pi}{M} + \text{Phase-B})$$

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 via FIFO. GXSM applies the normalization before plotting.

Informations about the check options can be found in STS.

Auxiliary Probe Control

Figure 12.9: 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.

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 12.10: 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 even if X or Y is checked.

12.4.4 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 aud tip approach. Set *GXSM preferences dialog, folder User/User/SliderControlType* to *mover entry t*o 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 f*older.

Figure 12.11: left: GXSM SR generic coarse mover controller, right: Auto approach controller

Figure 12.12: GXSM SR DSP configuration of SRanger inertial driver engine.

12.4.5 Extra Python SR-DSP Control and Configuration Scripts

The Python script SRanger/TiCC-project-files/FB_spmcontrol/python_scripts/sr_spm_control.py can be used for inspection and DSP/SPM software configuartion:

Figure 12.13: Main Menu

Figure 12.14: AIC Offset Control

Figure 12.15: CoolRunner Rate Meter

Figure 12.16: AIC Gain Control

Figure 12.17: SR-DSP/SPM software version info

Figure 12.18: SPM settings

Figure 12.19: 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!
Execute “loadusb” like this before starting GXSMor any other DSP tool:
besocke@thundera: /SRanger/loadusb$ ./loadusb ../TiCC-project-files/FB_spmcontrol/FB_spmcontrol.out

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

12.5 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_hwi	File:	hard/innovative_dsp_hwi.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

12.6 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.rhk-tech.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	File:	hard/LAN_rhk_hwi.C
Author:	Farid El Gabaly, Juan de la Figuera	Email:	johnnybegood@users.sourceforge.ne

12.7 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	File:	hard/tc211_ccd.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

12.8 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 (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 equiped with the Analog-810 (MK2-A810 SPM Controller), this is the new SPM dedicated analog module GXSM will supports best starting in 2009 (official introduction: Feb-2009) on. Main differenes/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.

The MK2-A810 AD/DA channel configuration is by now equivaltent to the old SRanger with excetion of the new 4-channel feedback soure mixer inputs. SR-AIC0..7-in/out correspondes to MK2-A810 AD/DA0..7 (See Appendix, SRanger DSP).
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 feedack source mixer, AD4..7: Auxilary.

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.

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 ADC_i are managed

Figure 12.20: 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.

in this automatic scan speed matching bandwidth mode:

$$\mathrm{V} = \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{ADC}_i(n), \quad N = \frac{75000\ \mathrm{Hz}}{\mathrm{Pixel/s}}$$

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 tp adjust the Gxsm preferences as indicated on Gxsm startup if not setup correctly. Refer also to the sample configuration shown in the appendix **??** 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 weighted in different ways via the gains G_i as illustrated in Fig. **??**. It also includes a special "FUZZY" mix mode for signal

Figure 12.21: 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.

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 ADC$_0$ a real time self adaptive Infinite Respose (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 ADC$_0$ input bandwidth in dependence of the signal magnitude $|\mathrm{I}_n|$ according to:

$$f_0(q) = -75000\,\mathrm{Hz}\frac{\ln(q)}{2\pi}, \quad q(\mathrm{I}_n) = 1 - \frac{\frac{f_{\min}}{f_{\max}}\mathrm{I}_c + |\mathrm{I}_n|}{\mathrm{I}_c + |\mathrm{I}_n|}$$

This real time computed q is limited to a $q_{\min}$ matching the given $f_{\max}$ before the bandwidth limited (IIR filter) current signal $\tilde{\mathrm{I}}_n$ which is recursively computed on the DSP according to:

$$\tilde{\mathrm{I}}_n = q\tilde{\mathrm{I}}_{n-1} + (1-q)\mathrm{I}_n$$

Fig. **??** 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 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.

Figure 12.22: Life IIR performance demonstration in Gxsm self-test configuration. A stair case like current input signal starting at 3.6 pA noise level assuming 1 V/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: $f_{\text{min}} = 50$ Hz, $f_{\text{max}} = 20$ kHz, $\text{I}_c = 500$ pA.

The IIR settings used for this demonstration are: $f_{\text{min}} = 50$ Hz, $f_{\text{max}} = 20$ kHz, $\text{I}_c = 500$ 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".

12.8.1 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.

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* value has to be set to 1.

Figure 12.23: GXSM MK2-A810 DSP Control window: Feedback and Scan Control page

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.

Mixer Gains For scaling of the error signal (SetPoint - Source), use about 0.5 at max. Bascially 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.

Figure 12.24: GXSM MK2-A810 DSP Control window: Advanced settings page

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 are 0.004 for both. Typically they can be increased up to 0.01, depending on the the system, tip and sample. 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, CI are set to one in linear mode.

12.8.2 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 an fixed speed of (92 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 updated the trigger data table, toggle/enable the Trigger Enable check-box. 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 12.25: Trigger Control window

12.8.3 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.

Frq-Ref must be set to 75000 kHz 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) occurrs 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 $(1 + 2 * PrePts/ScanPointsX)$. Use the Pan-View plug-in to prevent trouble with the maximum scan size.

STS

Figure 12.26: 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).

Figure 12.27: STS folder

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 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 f* or 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 coordinates, use the Line Object and read the dxy vector from the status line as indiceted 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 **??** 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:

Tp 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:

$$N = tp * 75000Hz = 75000Hz * dU/(IV - Slope * Points)$$

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.

Figure 12.28: left: GXSM SR DSP Control window, left: Z manipulation, right: LM, lateral and Z manipulation – unchanged

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 f*or 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.

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 12.29: GXSM SR DSP Control window: PL mode setup

Figure 12.30: GXSM SR DSP Control window: PL mode - demo setup with PanView showing Z adjusted to 4.7 Å for testing.

Figure 12.31: GXSM SR DSP Control window: PL mode - demo plot in self-test mode (Bias loop back to current)

Figure 12.32: GXSM SR DSP Control window: PL mode - demo Graphs configuration

Figure 12.33: GXSM SR DSP Control window: PL mode - demo for repetions

Figure 12.34: GXSM SR DSP Control window: PL mode - demo for repetitions

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),
Phase-A/2nd order (LockIn2ndA), Phase-B/2nd order (LockIn2ndB).

Please always select LockIn0 for STS.

Figure 12.35: 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/per 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: 585.9 Hz, 1171.9 Hz, 2343.7 Hz, 4687.5 Hz.

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:

$$U_{\text{LockIn0}} = \sum_{i=0}^{N-1} U_{in,i}$$

$$U_{\text{LockIn1stA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i\frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn1stB}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(i\frac{2\pi}{M} + \text{Phase-B})$$

$$U_{\text{LockIn2ndA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i\frac{2\pi}{M} + \text{Phase-A})$$

$$U_{\text{LockIn2ndA}} = \frac{2\pi}{N} \sum_{i=0}^{N-1} \text{AC-Amp} \cdot U_{in,i} \cdot \sin(2i\frac{2\pi}{M} + \text{Phase-B})$$

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 via FIFO. GXSM applies the normalization before plotting.

Information about the check options can be found in STS.

Auxiliary Probe Control

Figure 12.36: 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 12.37: 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.

12.8.4 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 aud 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 12.38: left: GXSM SR generic coarse mover controller, right: Auto approach controller

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 autoapproach is necessary to setup GPIO ports, which may control macro approach motor. In config tab is possible to set:

GPIO Pon Sets ports as logic 1 or 0 in point e - at the beginning of each pulse (picture **??**).

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.

Figure 12.39: GXSM SR DSP configuration of SRanger inertial driver engine.

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 measure 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 figure **??** marked with letter a. Length of each step (c) is determined by input field *Duration entry* .

Stepper motor tab

This tab is used for control macro approach, which can be realised by stepper motor. Approach motor is controlled by pulses on GPIO ports (in the fig. **??**)

Figure 12.40: Autoapproach signals. Z_s is scannig signal in Z axis.

There are used these three signals:

Enable Turn on the stepper motor. If is set to 0, no current is flowing (it can rotate freely).

Direction Sets direction of motor rotation. If is set to 1, it will rotate CCW.

Clock Each pulse means one step.

It is also possible, besides the manual mode, use the automatic approach. Typical values for GPIO settings are in the fig.**??**.

Beware:MK2-A810 operates with 3.3V logic! ☺

For easy stepper motor control via MK2-A810 use stepper motor controller L297. ♪♫

Figure 12.41: GXSM SR DSP configuration of SRanger inertial driver engine.

Figure 12.42: Pulses used for control of stepper motor.

Figure 12.43: Stepper motor control tab.

12.8.5 SR-DSP Phase/Amplitude Convergent Detector (PAC) Control

GXSM with the SRanger MK3-Pro Phase/Amplitude Convergent Detector.... (PLL).

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. Power cycle the MK2 after flashing and before plugging into the Linux system.*

12.8.6 Setup – Gxsm Configurations for the MK2-A810.

Quick sample reference for the MK2-A810 specific GXSMpreferences.

Figure 12.44: left: MK2-A810 Hardware selection and device path setup., right: Instrument configuration

Figure 12.45: left: Data Acquisition Channel configuration: Data Sources, right:

Figure 12.46: left: Data Acquisition Channel configuration, more., right:

12.8.7 Sample STS IV,dIdV

Quick screen shot for a STS setup for taking IV and dIdV data at a point (current ti position). Example taken with home build Besocke style STM at room temperature, real data here!

Figure 12.47: left: STS Setup for IV and dIdV, right: Lock-In configuration used

Figure 12.48: left: Data channel selection and graphing configuration for this STS probe, right: Plot of the resulting data taking.

12.8.8 Sample STS along line demo setup

Quick screen shot for a STS setup along a line. Gxsm in self test configuration.

Figure 12.49: left: STS Setup, right: Plotted Graphs

Figure 12.50: Scan with markers

12.8.9 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 preface **??**.

Signals are simply variables and connections are made simply via pointing the input to the source variable – as simple as this – "signals" are "hot plug-gable" – but beware, it is just like pulling a plug on your rack and plugging it in else where – so depending what you plug, better secure the tip or probe before doing so! Got me?

A typical STM configuration will look like the mostly default configuration as shown in Fig.**??**. This life 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.

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 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 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 makes signal read back requests: When GXSM 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 12.51: 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 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.**??**
`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 GUI itself.

Figure 12.52: 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**??**.

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.**??**. But for normal operation GXSM and als the SmartPiezoDrive (see **??**) are automatically assigning Signal for efficient monitoring use and without in depth knowledge those shall not be altered after GXSM start up.

Reassigning Monitoring signals while normal Gxsm operation may result in wrong signal read-

Figure 12.53: A810 AD/DA Monitor

ing 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. **??**, **??**.

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. ♪♫

SPM Signal Monitor

Signal to Monitor	DSP Signal Variable name	DSP Sig Reading	Value	dB	V	Galvo Range
Mon[00]: S28:MIX 0	feedback_mixer.FB_IN_processed[0]	84836	+0.1011 V			auto
Mon[01]: S29:MIX 1	feedback_mixer.FB_IN_processed[1]	2559	+0.0031 V			auto
Mon[02]: S30:MIX 2	feedback_mixer.FB_IN_processed[2]	38	+0.0000 V			auto
Mon[03]: S31:MIX 3	feedback_mixer.FB_IN_processed[3]	2190	+0.0026 V			auto
Mon[04]: S76:Out 3	analog.out[3].s	238221911	+1.1093 V			auto
Mon[05]: S77:Out 4	analog.out[4].s	-549388092	-2.5584 V			auto
Mon[06]: S78:Out 5	analog.out[5].s	2147483647	+10.0003 V			auto
Mon[07]: S68:TIME TICKS	state.DSP_time	-1505090445	-10033.9363 s			auto
Mon[08]: S47:X Scan Rot	scan.xy_r_vec[i_X]	-1449	-0.0000 V			auto
Mon[09]: S48:Y Scan Rot	scan.xy_r_vec[i_Y]	196	+0.0000 V			auto
Mon[10]: S32:MIX out delta	feedback_mixer.delta	-96508927	-115.0511 V			auto
Mon[11]: S38:Z Servo Watch	z_servo.watch	1	+1.0000 B			auto
Mon[12]: S33:MIX0 qfac15 LP	feedback_mixer.q_factor15	7253	+0.2214 Q			auto
Mon[13]: S34:signal AVG-256	analog.avg_signal	86452	+0.1031 V			auto
Mon[14]: S35:signal RMS-256	analog.rms_signal	804	+0.0000 mV^2			auto
Mon[15]: S43:VP Z pos	probe.Zpos	0	+0.0000 V			auto
Mon[16]: S51:X Offset	move.xyz_vec[i_X]	238223360	+1.1093 V			auto
Mon[17]: S52:Y Offset	move.xyz_vec[i_Y]	-549388288	-2.5584 V			auto
Mon[18]: S53:Z Offset	move.xyz_vec[i_Z]	0	+0.0000 V			auto
Mon[19]: S50:XYZ Scan Gain	scan.xyz_gain	657930	0x000A0A0A			auto

Close

Figure 12.54: Signal Monitor

Figure 12.55: Signal Monitor with Meter

Figure 12.56: 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. **??**, **??**, **??**.

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.

Figure 12.57: Signal Patchrack

When satisfied with the signal configuration the Signal Manager application (Fig.**??**) 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 problems with new or eventually non existent signals. The “Gxsm-MK2-like” default configuration can be reloaded at any time:

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:X Offset	== 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:X Scan Rot	== scan.xy_r_vec[i_X] ==>	OUTMIX_CH3
Sig: S51:X Offset	== 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 12.58: Signal Patchrack

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, 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. ♪♫

Figure 12.59: Signal Patchrack

Figure 12.60: 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.**??**. 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 12.61: Signal Oszilloscope

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 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 channels[1] 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".

Figure 12.62: Example Signal Patch Configuration for PLL view

[1] Channel count convention is here 0 . . . 7.

2. Then select the phase detector time constant "TauPAC" – set to 20 μs 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 20μs (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) automatically[2] 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.

3. Then turn on the PAC Processing, check 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 °(±180 °) for tuning. Select reasonable ranges for amplitudes matching your system needs.

Figure 12.63: PAC/PLL initial operation boundary setup example. Time constant set to 20 μs 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.

[2] This is **not** the case for the Gxsm control and to use the auto gain adjustments you must set TauPAC manually to 20 μs.

Figure 12.64: PAC/PLL – initial manual pre-check of signals and responses.

6. Now open the tune App to initiate a default sweep.

Figure 12.65: PAC/PLL Tune App – initial run with phase reference set to zero. Here a passive watch quartz was used.

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 12.66: 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

8. Use the oscilloscope app to inspect the signals for further fine tuning.[3]

Figure 12.67: Oscillocpe app for viewing PLL operation.

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.).

[3] To 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.

Figure 12.68: PAC/PLL Feedback settings with phase reference set to initial phase at resonance (here -73.7°).

10. 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 20μs 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 12.69: PAC/PLL – Amplitude FB on

11. 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.

12. Step Response Test Tool – not yet completed – need to fix related issue with the python problem with big buffer transfers.

Figure 12.70: PAC/PLL fully operational with both controller engaged. Amplitude stabilized. Phase fixed.

Figure 12.71: DSP Control – Mixer, Servo

Figure 12.72: 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, ... 32 (all 512 ... $\frac{512}{32}$ 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 frequincy.

Figure 12.73: DSP Control – LockIn: Free run mode enabled.

Figure 12.74: DSP VP-LM – tip forming example procedure

Figure 12.75: Graphs example for VP-LM

Figure 12.76: Mover Configuration

12.8.10 Optional MK3-Pro-SmartPiezoDrive with Gxsm Link

The Smart Piezo Drive Linux Frontend (Fig.**??**) with Gxsm-DSP Offset-Link capability is provided via this python application:

`plug-ins/hard/MK3-A810_spmcontrol/python_scripts/mk3_spd_control.py`

It communicates with the SPM contol 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 12.77: 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

13 Plug-Ins: hard/MK3-A810_spmcontrol/python_scripts

14 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.

14.1 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! ♪♫

Info for Plug-In: Tools/Hardware Info

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

14.2 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

14.3 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.

Figure 14.1: The PanView window.

Usage

Although this is a plugin it is opened automatically upon startup of GXSM. There is not interaction with the user.

Figure 14.2: The indicators of the PanView window.

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 feedback, scan in progress, vector proce in progress, mover in progress (coarse approach)

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)

♪♫

☺

Known Bugs

None

Info for Plug-In: Tools/Pan View

Plug-In name:	PanView	File:	common/PanView.C
Author:	Kristan Temme, Thorsten Wagner, Percy Zahl	Email:	stm@users.sf.net

14.4 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. It's content is added to the NetCDF file as variable named extra_scan_info.

Usage

Works automatic if loaded.

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

14.5 Python remote control

Description

This plugin is an interface to an embedded Python Interpreter.

Usage

Choose Pyremote from the *main menu/Tools* menu to execute your command script. In the appendix you will find a tutorial with examples and tips and tricks.

14.5.1 Reference

The following script shows you all commands that the emb-module supports:

```
import emb
print dir(emb)
```

The result will look like this:

```
['__doc__', '__name__', 'autodisplay', 'chmodea',
'chmodem', 'chmoden', 'chmodeno', 'chmodex',
'chview1d', 'chview2d', 'chview3d', 'da0', 'direct',
'echo', 'gnuexport', 'gnuimport', 'load', 'log',
```

```
'logev', 'menupath', 'quick', 'save', 'saveas',
'scaninit', 'scanline', 'scanupdate', 'scanylookup',
'set', 'sleep', 'startscan', 'stopscan', 'unitbz',
'unitev', 'units', 'unitvolt', 'waitscan', 'createscan']
```

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. `chview1d(2)`

'(X)' equals one Float argument. No example.

'(S)' equals a string. Often numbers are evaluated as strings first. Like in `set("RangeX", "234.12")`

'(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.
`gnuimport(S,N)`	Import file S to channel N.
`gnuexport(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.
`rtquery(S)`	Ask current HwI to run RTQuery with parameter S, return vector of th
`y_current()`	Ask current HwI to run RTQuery what shall return the actual scanline
`echo(S)`	Print S to console.
`logev(S)`	Print S to logfile.
`sleep(N)`	Sleep N/10 seconds.
`da0(X)`	Set Analog Output channel 0 to X Volt. (not implemented).
`menupath(S)`	Activate Menuentry 'S'

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()`.

```
import emb
print "OffsetX = ", emb.get("OffsetX")
emb.set("OffsetX", "12.0")
print "Now OffsetX = ", emb.get("OffsetX")

for i in emb.list():
    print i, " ", emb.get(i), " as string: ", emb.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 Â°
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 Numeric # for fromfunction
import math    # for sin
import emb     # gxsm embedded module

def dist(x,y):
   return ((Numeric.sin((x-50)/15.0) + Numeric.sin((y-50)/15.0))*100)

m = Numeric.fromfunction(dist, (100,100))
n = Numeric.ravel(m) # make 1-d
p = n.tolist()       # convert to list

examplearray = array.array('l', p) #
emb.createscan(100, 100, 10000, 10000, examplearray)
```

Figure 14.3: 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`-command. Several of them, however, open a dialog and ask for a specific parameter, e.g. the diff-PI in *main menu/Math/Filter1D*. 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:

```
import emb
print "Welcome to Python."
emb.logev('my logentry')
emb.startscan()
emb.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

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 depricated. All entry fields with hover-over entry id is now remote capable.

`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 exectutes 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.

Configuration

The plugin can be configured in the preferences. The script that will be executed must be 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 PYTONPATH.

The module with GXSM internal commands is called emb.

To find the Python-script remote.py, the environment-variable PYTHONPATH is evaluated. If it is not expliticly declared, GXSM will set PYTHONPATH to your current working directory. This is equivalent to the following call:

```
$export PYTHONPATH='.'
$gxsm2
```

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 you want to pause script execution, use the embedded sleep command `emb.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

Plug-In name:	pyremote	File:	common/pyremote-threadded.C
Author:	Stefan Schröder	Email:	stefan_fkp@users.sf.net

14.6 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	File:	common/mkicons.C
Author:	Stefan Schröder	Email:	stefan_fkp@users.sf.net

14.7 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

14.8 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

14.9 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.

14.9.1 Reference

The following script shows you all commands that the emb-module supports:

```
import emb
print dir(emb)
```

The result will look like this:

```
['__doc__', '__name__', 'autodisplay', 'chmodea',
'chmodem', 'chmoden', 'chmodeno', 'chmodex',
'chview1d', 'chview2d', 'chview3d', 'da0', 'direct',
'echo', 'gnuexport', 'gnuimport', 'load', 'log',
'logev', 'menupath', 'quick', 'save', 'saveas',
'scaninit', 'scanline', 'scanupdate', 'scanylookup',
'set', 'sleep', 'startscan', 'stopscan', 'unitbz',
'unitev', 'units', 'unitvolt', 'waitscan', 'createscan']
```

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. `chview1d(2)`

'(X)' equals one Float argument. No example.

'(S)' equals a string. Often numbers are evaluated as strings first. Like in `set("RangeX", "234.12")`

'(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.
`gnuimport(S,N)`	Import file S to channel N.
`gnuexport(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.
`rtquery(S)`	Ask current HwI to run RTQuery with parameter S, return vector of th
`y_current()`	Ask current HwI to run RTQuery what shall return the actual scanline
`echo(S)`	Print S to console.
`logev(S)`	Print S to logfile.
`sleep(N)`	Sleep N/10 seconds.
`da0(X)`	Set Analog Output channel 0 to X Volt. (not implemented).
`menupath(S)`	Activate Menuentry 'S'

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()`.

```
import emb
print "OffsetX = ", emb.get("OffsetX")
emb.set("OffsetX", "12.0")
print "Now OffsetX = ", emb.get("OffsetX")

for i in emb.list():
    print i, " ", emb.get(i), " as string: ", emb.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 Â°
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 Numeric # for fromfunction
import math    # for sin
import emb     # gxsm embedded module

def dist(x,y):
   return ((Numeric.sin((x-50)/15.0) + Numeric.sin((y-50)/15.0))*100)

m = Numeric.fromfunction(dist, (100,100))
n = Numeric.ravel(m) # make 1-d
p = n.tolist()       # convert to list

examplearray = array.array('l', p) #
emb.createscan(100, 100, 10000, 10000, examplearray)
```

Figure 14.4: 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`-command. Several of them, however, open a dialog and ask for a specific parameter, e.g. the diff-PI in *main menu/Math/Filter1D*. 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:

```
import emb
print "Welcome to Python."
emb.logev('my logentry')
emb.startscan()
emb.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

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 depricated. All entry fields with hover-over entry id is now remote capable.

`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 exectutes 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.

Configuration

The plugin can be configured in the preferences. The default script that will be loaded when the console is enetred 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 emb.

To find the Python-script remote.py, the environment-variable PYTHONPATH is evaluated. If it is not expliticly declared, GXSM will set PYTHONPATH to your current working directory. This is equivalent to the following call:

```
$export PYTHONPATH='.'
$gxsm2
```

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 you want to pause script execution, use the embedded sleep command `emb.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

15 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.

15.1 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.

Figure 15.1: QT Export File Dialog

There seam 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 seams to depend on the system and libquicktime version used, so please try it out.

Figure 15.2: left: QT Export Setup step one., right: QT Export Setup step two.

Info for Plug-In: File/Import/Quicktime

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

15.2 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	File:	scan/PsiHDF_im_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.3 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	File:	scan/UK2k_import.C
Author:	Juan de la Figuera	Email:	juan.delafiguera@uam.es

15.4 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	File:	scan/v5d_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.5 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

15.6 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	File:	scan/wip_im_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.7 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

15.8 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. 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	File:	scan/omicron_io.C
Author:	Andreas Klust	Email:	klust@users.sf.net

15.9 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	File:	scan/external_converter.C
Author:	Thorsten Wagner	Email:	stm@users.sf.net

15.10 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	File:	scan/converter.C
Author:	Kristan Temme and Thorsten Wagner	Email:	stm@users.sf.net

15.11 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 GXSMto control it :-).

Usage

The plug-in is called by *main menu/File/Import/RHK SPM32*. It is also automatically invoked by GXSMwhen 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	File:	scan/rhk_spm32_import.C
Author:	Andreas klust	Email:	klust@users.sf.net

15.12 UKSOFT/U-view Import

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.:

```
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.

Figure 15.3: left: UKSOFT/U-view import file dialog, right: setup for multi dimensional image processing

Info for Plug-In: File/Import/UKSOFT

Plug-In name:	uksoft2001_im_Export	File:	scan/uksoft2001_im_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.13 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.

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

15.14 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	File:	scan/rhk2000_import.C
Author:	Peter Wahl	Email:	wahl@fkf.mpg.de

15.15 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
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 -2.0684 -2.0498 -2.037
-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 -2.0645 -2.0464 -2.033
-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 -2.0601 -2.0420 -2.029
-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 -2.0557 -2.0391 -2.026
-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 -2.0537 -2.0327 -2.025
-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 -2.0420 -2.0278 -2.014
-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 -2.0405 -2.0220 -2.014
-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 -2.0337 -2.0171 -2.009
-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 -2.0312 -2.0146 -2.002
-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 -2.0264 -2.0083 -1.999
-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 -2.0249 -2.0054 -1.994
-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 -2.0171 -1.9990 -1.991
-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 -2.0142 -1.9971 -1.986
-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 -2.0142 -1.9941 -1.979
-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 -2.0054 -1.9868 -1.974
-1.5229 -1.5225 -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 -2.0010 -1.9819 -1.971
-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 -1.9976 -1.9761 -1.969
-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 -1.9941 -1.9717 -1.962
-1.4497 -1.4687 -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 -1.9897 -1.9683 -1.959
-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 -1.9854 -1.9653 -1.959
-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 -1.9790 -1.9634 -1.948
-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 -1.9741 -1.9590 -1.945
-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 -1.9663 -1.9531 -1.940
-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 -1.9683 -1.9487 -1.936
-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 -1.9609 -1.9453 -1.929
```

Usage

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

Suffix	description
.dat	(old) STM data format, used by e.g. PMSTM, dicontinued.
.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 15.1: file formats supported by the *primitiveimexport* plug-in.

Info for Plug-In: File/Import/ASCII

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

15.16 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. Nethertheless, the support of these formats allows to exchange data between GXSM 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. **??**. The ".dat" format refers to the old, STM specific data format used by the OS/2 software PMSTM and the Linux based Xxsm, both predecessors of GXSM. 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 project provides an extensive set of small tools included in the GXSM 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 myimage.pgm results in a greymap image in the binary Portable Greymap (pgm) format. For other supported formats and their correspondent suffixes see Tab. **??**.

☺

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

15.17 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	File:	scan/gmeyer_im_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.18 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.

DnD and URL drops are not tested.

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

Plug-In name:	WSxM_io	File:	scan/WSxM_io.C
Author:	Thorsten Wagner	Email:	stm@users.sf.net

15.19 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	File:	scan/g_dat_im_export.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

15.20 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.

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

16 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.

16.1 Max of two sources

Description

This filter merges two (same sized and aligned) scans by using the max Z value of source one (actice) 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

16.2 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:

$$Z_{\text{math}} = \log\left(Z_{\text{active}} - \text{minval}\left(Z_{\text{active}}\right) + 1\right)$$

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

16.3 Z Limiter

Description

The Z Limiter limites 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 rectange (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	File:	math/arithmetic/Z_limiter.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

16.4 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

16.5 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

16.6 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 coeficients 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

16.7 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

16.8 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	File:	math/arithmetic/sub_scan.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

16.9 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. Thers is an $\epsilon = 10^{-8}$ defined as minimal divisor, if the absolute value of the divisor is smaller than ϵ the original divident 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

16.10 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

17 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…

17.1 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. ♪♫

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

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 comnponents (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 badn 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	File:	math/filter1d/timedomfftfilter.C
Author:	Percy Zahl	Email:	zahl@users.sourceforge.net

17.2 Koehler filter

Description

The Koehler filter differentiates the image data line by line (1D). It uses a local weightened differentation 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:

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

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

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

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

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

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

17.3 Linear stationary differentation

Description

Edge enhancement via differentation 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

17.4 Repair filter

Description

This filter is obsolete for all Gxsm only users. It just fixes a high/low words order bug – occured 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

17.5 1dim differentation

Description

One dimensional differentation...

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

18 Plug-Ins: math/convert

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

18.1 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

18.2 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	File:	math/convert/short_to_short.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

18.3 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 fom 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

18.4 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	File:	math/convert/to_long.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

18.5 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

18.6 Generate test data

Description

Makes a test data set – for test and demostration 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

18.7 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

18.8 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

18.9 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:

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

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

19 Plug-Ins: math/transform

The *math/transformation* plugins are all type of math doing geometric transformations with the image like rotation, scaling, etc.

19.1 Flip diagonal

Description

Flips an image along it's 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	File:	math/transform/flip_diagonal.C
Author:	A. Klust, P. Zahl	Email:	zahl@users.sf.net

19.2 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	File:	math/transform/mirror_x.C
Author:	A. Klust, P. Zahl	Email:	zahl@users.sf.net

19.3 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	File:	math/transform/merge_v.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

19.4 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

19.5 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 rottate the scan and apply the usual background correction functions.

Info for Plug-In: _Math/Transformations/Rotate 90deg

Plug-In name:	rotate90	File:	math/transform/rotate90.C
Author:	Andreas Klust	Email:	klust@users.sf.net

19.6 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, ridig 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. Thevénaz, 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	File:	math/transform/autoalign.C
Author:	P. Thev enaz, U.E. Ruttimann, M. Unser	Email:	zahl@users.sf.net

Percy Zahl

19.7 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	File:	math/transform/shear_y.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

19.8 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	File:	math/transform/mirror_y.C
Author:	A. Klust, P. Zahl	Email:	zahl@users.sf.net

19.9 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

19.10 SPA–LEED octopole distorsion correction

Description

This is a experimental filter to reduce the SPA–LEED typical octopole cuishion distorsion using a ”ab-initio” like approch. This plugin applies the inverse of this distorsion 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 (distorsion 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 distorsion 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

19.11 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/Volume Transform

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

19.12 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

19.13 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/Multi Dim Transpose

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

19.14 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 a 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

19.15 Multi Dimensional Layer Reverse

Description

Reverse Layer Order.

Usage

Activate a channel and run it.

Info for Plug-In: _Math/_Transformations/Reverse Layers

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

19.16 Affine Transformation

Description

In case you want to undo a linear distorsion of an image (e.g. slow and contineous 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 lenght 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

19.17 Movie Concat

Description

Movie Concat allows to concatenate two movie data set in time with a choosen range in time and in layers of both sources. The number of layers choosen for each must match.

Usage

The active and X marked channels are used as data sources.

Info for Plug-In: _Math/_Transformations/Movie Concat

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

19.18 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

19.19 Horizontal merge

Description

Used to horizontal merge to 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

19.20 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	File:	math/transform/shiftarea.C
Author:	Stefan Schröder	Email:	stefan_fkp@users.sf.net

20 Plug-Ins: math/misc

The *math/misc* plugins are all types of math not fitting well into all other sections here.

20.1 Smoothing layers in 3D

Description

This filter applies a gausian 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 jet finished, the 3D smooth (accross layers) is not jet implemented. Please set Radius in Layer Dim to zero! ♪♫

Info for Plug-In: Math/Misc/Layersmooth

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

20.2 PSD add – SARLS

Description

Experimental filter for adding PSD-slope-signals.

Usage

It is only available form *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

20.3 Find local maxima – SARLS

Description

Finds local maximan 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. ♪♫

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

20.4 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

20.5 Cut spectra from 3D data

Description

This plug-in is made for choosing areas from a topograhic 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 spectoscopic 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	File:	math/misc/spectrocut.C
Author:	Andreas Klust	Email:	klust@users.sf.net

20.6 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.

Info for Plug-In: Math/Misc/Workfuncextract

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

21 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.

21.1 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

21.2 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

21.3 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 flatened.

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.

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

21.4 Gamma correction

Description

Applys a gamma correction as it is defined here:

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

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

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

21.5 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 $(2, 3, 4, \ldots 13)$ 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

21.6 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

21.7 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 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/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	File:	math/background/stop_ccr.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

21.8 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	File:	math/background/DocOnlyLineRegress.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

21.9 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 threashold 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 ajusted. 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 threashold 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

21.10 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 $z(x, y) <$ waterlevel then $z(x, y) =$ 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	File:	math/background/waterlevel.C
Author:	Andreas Klust	Email:	klust@users.sourceforge.net

21.11 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	File:	math/background/plane_m
Author:	Percy Zahl, L.Anderson, Greg P. Kochanski	Email:	zahl@users.sf.net

21.12 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

21.13 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

22 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:

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

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

22.1 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	File:	math/filter2d/Tderive.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

22.2 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. **??**).

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

22.3 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/Filter _2D/FT 2D

Plug-In name:	ft2d	File:	math/filter2d/ft2d.C

Author: Percy Zahl Email: zahl@users.sourceforge.net

22.4 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

22.5 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	File:	math/filter2d/curvature.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

22.6 Line Interpolation

Description

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

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

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

Usage

It works in two modes:

a) Assuming line 100 (in pixel corrdinates) 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 flatened 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 an threashold, which is used to determine if a line is distored, therefore it compares the average Z withing the marked X range of any line in the mared 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	File:	math/filter2d/DocOnlyLineInterpol.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

22.7 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

22.8 Stat Diff

Description

Stationary Differentation in scan direction using a convolution kernel:

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

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

22.9 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

23 Plug-Ins: math/statistik

The *math/statistik* plugins are all type of math doing some statistical analysis like a histogram creation.

23.1 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? ♪♫

Info for Plug-In: Math/Statistics/stepcounter

Plug-In name:	baseinfo	File:	math/statistik/stepcount.C
Author:	Stefan Schröder	Email:	stefan_fkp@users.sf.net

23.2 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 chanel 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

23.3 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 jet, 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

23.4 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 staring 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 doese 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.
$\longrightarrow$ 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

23.5 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 $2 \ldots 3$ 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 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 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	File:	math/statistik/VacancyLineAnalysis.C
Author:	J.S. Palmer	Email:	jspalmer@mines.edu

23.6 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:

$$e^{2\pi i S Z(x,y)}$$

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/SPALEED Simkz.* 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

23.7 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

23.8 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 happenes with more or less bins!

Info for Plug-In: Math/Statistics/Opencvmatch

Plug-In name:	opencvmatch	File:	math/statistik/OpenCV_match.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.9 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

23.10 Statistic base info

Description

This plugin calculates some basic statistical information about the active scan or a selected rectangle within the active scan:

Abs. Z min: Minimum Z-value and its coordinate

Abs. Z max: Maximum Z-value and its coordinate

$\sum Z$**:** Sum of all Z-values

$\sum Z^2$**:** Sum of all squared Z-values

Averaged Z**:** Average of all Z-values:

$$\overline{Z} = \frac{1}{A} \sum_{x,y} Z(x,y) \text{ mit } A := x \cdot y$$

RMS-roughness: root mean square roughness:

$$\begin{aligned}
\sigma^2 &= \frac{1}{A} \int_A d\vec{x}\, [(Z(x,y) - \overline{Z})^2] \\
&= \frac{1}{A} [\sum_{x,y} Z(x,y)^2 - 2\overline{Z} \sum_{x,y} Z(x,y) + \sum_{x,y} \overline{Z}^2] \\
&= \frac{1}{A} [\sum Z(x,y)^2 - 2\overline{Z} \sum Z + \overline{Z}^2 \sum 1] \\
&= \frac{1}{A} [\sum Z(x,y)^2 - 2\overline{Z} \sum Z + \overline{Z}^2 A]
\end{aligned}$$

Usage

Choose Baseinfo from the *main menu/Math/Statistics/Baseinfo* menu.

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

A dialog box appears, which contains the information in tabular form. You can press 'Dump to Stdout' to print the information on the console. Pressing 'Dump to comment' concatenates the calculated information to the comment field of this scan in the main window.

Configuration

None.

Known Bugs

During the calculation of rms-roughness or the sum overflows can occur even in medium sized scans. You better choose a smaller region then.

Calculated parameters are (with example values):

Area:	*(-35.0571 μm,-600.586 μm)-(-18.0728 μm,-656.711 μm)*
Area size:	*18450 Int*^{2}
Minimum:	*98 Int*
Min Pos:	*(-27.942 μm,-600.815 μm)*
Maximum:	*10438 Int*
Max Pos:	*(-31.6143 μm,-628.763 μm)*
Sum:	*4.95591e+07 Int*
Sum Sq:	*1.86421e+09 Int*
Average:	*2686.13 Int*
rms-roughness:	*nan Int*

Info for Plug-In: Math/Statistics/Baseinfo

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

23.11 Nndistribution

Description

The NN-distribution plugin calculates the nearest neigbour (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 happenes 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

23.12 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	File:	math/statistik/crosscorrelation.C
Author:	Erik Muller	Email:	emmuller@users.sourceforge.net

23.13 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 chanel 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	File:	math/statistik/slopeabs.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.14 Autocorrelation

Description

Computes the autocorrelation of an image.

$$Z' = |\mathrm{IFT}(\mathrm{FT}(Z))|$$

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	File:	math/statistik/autocorrelation.C
Author:	Erik Muller	Email:	emmuller@users.sourceforge.net

23.15 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

23.16 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	File:	math/statistik/AngularAnalysis.C
Author:	Percy Zahl	Email:	zahl@users.sf.net

23.17 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/SPALEED 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 you image before! Check also for correct step heights, if applicable.

Info for Plug-In: Math/Statistik/SPALEED Sim.

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

24 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, einzig was in mem2d.h von interesse
ist, ist die eine (und einizge) 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 (mit
        double get_position (double &x, double &y); // Position abfragen
        double distance (double *xy); // Abstand zu Punkt xy, ist array x
        guint get_event_count() ; // Anzahl Events in Liste

Data:
        GSList *event_list; // Liste von Daten-Events "hier" bei XYV,
                            // i.a. nur ein DatenEvent, type der Pointer


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 "Konevntion" Probe Events mit dem Namen "Probe"
versehen und User Events mit dem Namen "User", ist zum Identifikation.
Und Weiterhin sollen die Anfangsbuchstaben von Probe und User
eineindeutig sein, d.h. 'P' für Probe:

        gchar* description () { return name; }; // der volle Name
        gchar description_id () { return name[0]; }; // Identifikation vi


Dann gibt es den ProbeEntry  (etwas gestrippeder Code)  für alle Probe
artigen Daten:

class ProbeEntry : public EventEntry{
public:
        ProbeEntry (gchar *Name, time_t Time, GPtrArray *Labels, GPtrArra
*Unitssymbols, int Chunk_size); // Name, zur Zeit nur "Probe", time,
// dann ein Array von Pointern auf Labels und auf Unitssymbols, und Chunk

// Chunk_size ist die Anzahl der aufgezeichneten Channels des Probe, aus
// Performance gründen speichere ich den gazne 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.

24.1 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 andan chanel. 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	File:	math/probe/probe_image_extract.C
Author:	Bastian Weyers	Email:	weyers@users.sf.net

25 List of undocumented and ignored GXSM Plug-Ins

25.1 hard/MK3-A810_spmcontrol/python_scripts/test.C

This PlugIn is needs documentation!

25.2 scan/v5d_binio.C

This PlugIn is needs documentation!

25.3 scan/v5d_v5d.C

This PlugIn is needs documentation!

25.4 math/transform/autoalign_turbo_reg.C

This PlugIn is needs documentation!

25.5 math/transform/autoalign_stack_reg.C

This PlugIn is needs documentation!

25.6 math/statistik/histoHOP.C

This PlugIn is needs documentation!

25.7 math/statistik/islandlbl.C

This PlugIn is needs documentation!

25.8 Summary of PlugIn documentation scan

Number of scanned PlugIns/Files: 185
Number of documented PlugIns: 133
Number of undocumented PlugIns: 7
Number of ignored PlugIns: 0
Number of ignored subroutine Files: 45

Part III

Digital signal processing subsystem

26 Digital Signal Processing and GXSM

What means DSP? It stand for "Digital Signal Processor" and it almost means a bit more, it's usually a complete independent single or even multiple processor computer system. A DSP is designed to be best suited and highly efficient to stream and process data at high speed and in real time. It runs it's own software and can communicate with the external world via some special kind of interfaces.

In this section the user should get an idea how Gxsm and the DSP data acquisition subsystem are working together. From developers point of view it is of a big advantage and helps a lot to know how the engine works under the hood and as I'm writing this for your information I hope this helps you to get better used to the whole thing by knowing some details – so you are getting closer having an idea what fictional features would be possible to support in future! Let us know you great fictions about your future SPM!

26.1 Generic introduction to Gxsm and the DSP

Let's start with an overview to the whole thing looking at diagram **??**. The SPM user interacts with the SPM via the Gxsm software using the X11 window system. Gxsm itself sends and receives commands and data via a Unix device, which is managed by a hardware dependent Linux kernel module. This module has to assure, that the requested actions are performed by the connected hardware. This makes Gxsm independent of any hardware. The PC interface board (here: PCI32) is build around a *TMS320C32* DSP, and provides 4× Analog Digital Converters (ADC) and 4× Digital Analog Converters (DAC) with each 16 bits resolution and a digital IO (DIO) port. The DSP board runs a SPM specialized software. The SPM head/piezo tube(s) are connected to a piezo drive, which provides the needed symmetric $\pm X$, $\pm Y$ and Z voltages at various gains (1×, 2×, 5×, 10×, 15×), as needed. Analog signal sources (e.g. tunneling current, amplified via a current voltage converter (IVC) and auxiliary (AUX) signals) are connected to the analog input(s). Digital ports are available for a slider control or other optional purposes.

In case of the newer Signal Ranger the scenario is similar, but differs in detail, the USB bus is used for communication and data transport and also there are more AD/DA codecs (AIC's) available and a newer and faster 100 MHz low power *TMS320C5402* is used.

Lets look closer to the software running on the DSP machine, which dose the real hard work. In diagram **??** the three main working blocks are shown. The DSP program is designed like a state machine, this means that the system has always to be in one, or a allowed combination, of the defined states. The *Idle Loop* does all state controls and communications. The feedback

loop interrupt subroutine (ISR) is always running at a constant rate and is controlled via the feedback parameters and a feedback on/off state flag. The additional shared task ISR is used for all remaining tasks, which have to be timed precise. The main states (modes) are:

CMD Command Mode – it recognizes commands from outside. This mode can be temporary disabled, but is has to be assured it will switched back to command mode afterwards.

SCAN After having received a linescan command with corresponding parameter set this mode is entered. This will enable the scanning task, which restores to the previous mode if it is done and also signals that the linescan data are ready for transfer.

MOVE After having received a move command with corresponding parameter set this mode is entered and a move to a now position is executed via the move task.

PROBE After having received a probe command with corresponding parameter set this mode is entered. The probe task is activated. If done signals also data ready.

SLIDER (it is called AFMADJ in the source) This mode will be entered for mover or slider coarse positioning.

APPROCH (it is called TIPDN in the source) This mode will be entered for automatic tip approach. The tip approach task works itself like a separate state machine for the three phases of coars approach, piezo approach (via feedback) and piezo retract phases.

PID The feedback is enabled – the always running feedback tasks checks this state and stops Z-value corrections if disabled.

All tasks are expected to be “one shot” for each call and without any lengthy loops – so they have to do one step at a time and the next at next call.

That’s how it all works in principle, implementations for different DSP subsystems and DSP software generations may differ in detail.

Figure 26.1: Schematic diagram of a typical SPM control system using Gxsm

Figure 26.2: Schematic diagram of DSP subsystem (PCI32 design)

27 DSP subsystem PC31/PCI32

The PC31 and PCI32 are DSP boards [1] are PC cards with AD/DA some digital IO and a TMS320 type DSP. Both boards are working in a similar way and the communication with the host and GXSM is realized via a so called dual ported RAM (DPRAM). This project also provides generic Linux kernel modules for both cards.

- The (very old) ISA-bus card "PC31", still supported, but out of date:
  - 50 MHz DSP *TMS320C31*
  - 8 × 16 bit AD, differential inputs and 2 × 4 multiplexed, 4 × 16 bit DA (serial), 200kHz
  - 3 × 8 bits digital IO (via 8255)
  - 3 additional external timer/counters (via 8254)
- The (old) PCI-bus card "PCI32", still supported, but out of date:
  - 60 MHz DSP *TMS320C32*
  - 4 × 16 bit AD, differential inputs, all independent; 4 × 16 bit DA, 200kHz
  - 16 bits digital IO
- Refer to the original PC31 and PCI32 hardware documentation for a complete description: **http://sourceforge.net/project/showfiles.php?group_id=12992&release_id=16019**

27.1 Gxsm and the PC31/PCI32

As already described in **??** the GXSM system can communicate with this generation of DSP and it will also auto detect some special features build into the DSP software, which has been in development for some time and now exists in the third major generation of DSP code.

This boards are best tested and working with the older GXSM(-1) version but will also work with GXSM-2. However they are not any longer verified to work with every change, sorry so ♪♫

[1] PC31 and PCI32 are made by Innovate Integration (**http://innovative-dsp.com/**) but are not any longer in production and may not be available today. The GXSM project does still support these old boards, but no further active developments related to these boards is done. The Signal Ranger (see **??**) is the "up-to-date" board.

far...
FYI: The CVS managed source code archive allows to check out any old version.

The current DSP source code is available via CVS in the Gxsm module, browse it all here:
http://cvs.sourceforge.net/viewcvs.py/gxsm/Gxsm/pci32/dspC32/xsm/

The last (V1.38/2002-12-16) precompiled DSP binary for PCI32 is available here, section "DSPCode-xsm-STM NEW!!":
http://http://sourceforge.net/project/showfiles.php?group_id=12992&release_id=128590

There are actually two variants, one for cards with 128kWord memory and boards with 256kWord. The board with 256 kWord is preffered due to less data size restrictions.

There are some old DSP binaries for the PC31 board as well (project's file area:
http://sourceforge.net/projects/gxsm/*),*
but they are highly customized to fit best for the instrument. The customization is hard coded/configurable in the DSP source via a instrument describing header files...

The DSP binary (in COFF1 format) can be downloaded to the DSP board and started using the loadpci tool available by this project in the Gxsm module, browse it all here:
http://cvs.sourceforge.net/viewcvs.py/gxsm/Gxsm/pci32/load/

This load tool is build within the *Gxsm-tree* build process.
The so called loadpci is capable of detecting the DSP variant and of loading both, PC31 and PCI32 code! Check the command line options: loadpci –help

The DSP program has to be launched before GXSM is started, GXSM will connect to the DSP and will *not* reset or restart the DSP program!

27.2 Important GXSM preferences, PC31/PCI32 specific

To assure correct operation of the DSP software in conjunction with the GXSM system the following DSP-software-related preferences have to be set exactly as described in table **??**. These are almost default values.

27.3 PC31/PCI32 connections

In case of PC31 and also PCI32 this depends on the configuration of the DSP code. A the basic guide line how to setup connections to/from the PCI32 in an SPM environment is described in **??**. Please contact on the GXSM project experts for further details.

[1] This could be also /dev/pcdsp using old kernel 2.2.x modules!

Preference-key	Value
Hardware/Card	Innovative_DSP:SPM
Hardware/Device	/dev/pcdsps/tms320[1]
Instrument/Type	STM or AFM
Instrument/ScanOrgCenter	false
Analog/DigRangeIn	32768
Analog/VoltMaxIn	10 V
Analog/DigRangeOut	32768
Analog/VoltMaxOut	10 V

Table 27.1: Important GXSM preferences for usage of the PC31/PCI32 hardware.

27.4 DSP software for PC31/PCI32

27.4.1 Building the DSP binary

To build DSP binaries the Texas Instruments DSP TMS320C30 MS/DOS commandline compiler and runtime libraries are needed, sorry they cannot be included here due to copyrights...

Notes for remake of DSP software

This section is a very very draft version – I'm just hacking on some stuff and adding this notes on the fly...

1. Symlink/Copy the needed Ti-DSP-CC header files and libraries into the Gxsm/pci32/... subtree. It should look like:

```
$> ~/Gxsm/pci32$ ls . dspcc dspC32 dspC32/include dspC32/lib

.:
CVS  Client  dspC32  dspcc  load  modules

dspC32:
CVS  include  lib  server  spa  xsm

dspC32/include:
CVS        c3x.h      float.h     proto.h     stdlib.h
analog     ctype.h    icclbutl.h  safesp.h    string.h
analog.h   digital    limits.h    serprt30.h  time.h
assert.h   digital.h  math.h      setjmp.h    timer30.h
bus        dma30.h    misc        signal      trg_mbox.h
bus.h      dma32.h    misc.h      stdarg.h    trgstruc.h
bus30.h    dsp.h      periph.h    stddef.h    values.h
bus32.h    errno.h    periph31.h  stdio.h

dspC32/lib:
CVS          boot32.lib  digita31.lib  math.lib  stdio.lib
analog.lib   bus.lib     digital.lib   misc.lib
boot31.lib   bus31.lib   digital3.lib  rts.lib

dspcc:
AC30.EXE   CL30.EXE   LIMITS.H     PRTS30.SRC  STDARG.H
AR30.EXE   CLIST.EXE  LNK30.EXE    PRTS40.SRC  STDDEF.H
ASM30.EXE  CTYPE.H    MATH.H       README.1ST  STDLIB.H
ASSERT.H   CVS        MATHASM.SRC  RMINFO.EXE  STRING.H
```

```
C30.CMD    ERRNO.H    MK30.EXE     RTS.SRC     TIME.H
C40.CMD    FLOAT.H    OPT30.EXE    SAFESP.H    as.bat
CG30.EXE   HEX30.EXE  PMINFO.EXE   SETJMP.H    cc.bat
```

But I have to note the boot31.lib, analog.lib, boot32.lib, bus.lib and bus31.lib libraries are not *original*. I've applied my own patches to make the older PC31 stuff software compatible with the PCI32 in some way and I'm not really able to explain this all here in detail – sorry. The patched tree is available in the Institut für Festkörperphysik of Universität Hannover. Get in contact!

2. Start xdosemu -home and switch to the "DOS in a BOX" window.

3. cd into your Gxsm/pci32/dspC32/xsm directory, it is expected here: D:
   DSP
   DSPC32
   XSM – so make the proper symbolic link in your home directory, which is redirected to DOS drive D: in DOSEMU. E.g: DSP $-$ $>$ Gxsm/pci32

4. call doscfg – it sets some enviroment vars used by the tms320 C compiler:

```
D:\DSP\DSPC32\XSM>type doscfg.bat
echo off
echo ------------------------------------------------------------
echo Configure Dos to run w tms320 cc...
echo - Note: you need to enable DPMI in you dosemu.conf!!
echo make sure D:\DSP points to (symlink!!) your Gxsm/pci32 directory!
echo ------------------------------------------------------------
echo adding path to cc exe files
set path=%path%;d:\DSP\dspcc
echo setting cc options to find include and libs
SET C_OPTION=-mn -v30 -x2 -o2 -q
SET C_DIR=D:\DSP\dspC32\include;D:\DSP\dspC32\lib
SET A_DIR=D\DSP\dspC32\include;D:\DSP\dspC32\lib
echo done.
```

5. call make

6. That's it, you should now have a new xsm.out file in place!

Notes for testing the new version

1. Make sure the special device has been created correct (using mknod) and has assigned the right minor/major pair – should look like:

```
$> ~$ ls -ld /dev/pcdsps/
drwxrwxrwx    2 root  root      4096 May  1 11:32 /dev/pcdsps/
$> ~$ ls -ld /dev/pcdsps/tms320
crw-rw-rw-    1 root  root  240,   0 May  1 11:32 /dev/pcdsps/tms320
$> ~$
```

2. Make sure the pci32.o kernel module is loaded (using insmod) and has detected the PCI32 card:

```
$> ~$ /sbin/lsmod
Module                  Size  Used by    Tainted: P
pci32                   7528   0
```

3. Check (dmesg or /var/log/messages, should look like:

```
Jun 17 10:48:07 atomic kernel: pci32: init_module PCDSP
Jun 17 10:48:07 atomic kernel: pci32: Major number :240
Jun 17 10:48:07 atomic kernel: pci32: PCI32 card #0 found at
                               I/O:cc00 mem:cfffa000 IRQ 12
Jun 17 10:48:07 atomic kernel: pci32: ioremapping
                               0xcfffa000 to 0xe51ee000
Jun 17 10:48:07 atomic kernel: pci32: checking PCI_COMMAND
Jun 17 10:48:07 atomic kernel: pci32: PCI_CMD_MEMORY
Jun 17 10:48:07 atomic kernel: pci32: PCI_CMD_IO
Jun 17 10:48:07 atomic kernel: pci32: PCI_CMD_MASTER
Jun 17 10:48:07 atomic kernel: pci32: The PCI BIOS has not enabled
                               this device!
                               Updating PCI command 0107->0003.
```

4. Launch the the DSP software using loadpci:

```
$> ~/Gxsm/pci32/dspC32/xsm$ loadpci -d /dev/pcdsps/tms320 xsm.out

COFF Loader/Terminal Emulator PC31/PCI32 v1.10 for Linux by PZ
Module is PCI32
Talker Monitor up, Read: 0x0000001f
Talker Monitor revision: 0x00000200
MBOX Write SPEED Cmd: OK
MBOX Write SPEED start/stop timer: OK
MBOX Write SPEED Cmd: OK
```

```
MBOX Write SPEED start/stop timer: OK
Counts: n0=118297, n[1s]=14986506  => 40MHz
Starting Program... good luck !
Branch to entry point ...
```

5. You can check the new running version using the lcd program, it should show the current version number and some additional debugging info. And most important the “right LED” should be flashing with approx. 1 Hz. If so – great! Close the LCD.
6. Start GXSM and go further with testing all functions.

27.5 GXSM PC31/PCI32 Kernel Module Installation

Skip the section if you are not intending to use any data acquisition hardware. If you are going to use the Signal Ranger refer to section **??** for details and skip this section.

Only for PC31/PCI32 DSP card users!

27.5.1 2.4.x kernel and DevFs

The included kernel modules (PC31, PCI32) are tested with Linux kernel 2.2.14-18, support of older (2.0.X kernel) was removed. Most recent PCI32 modules are dependent on the 2.4.x kernel tree with DevFs support.

To compile the modules you will need to have the kernel properly installed in "/usr/src/linux" and you should have the symlinks to the actual kernel code from

/usr/include/linux $-$ > /usr/src/linux/include/linux

and

/usr/include/asm $-$ > /usr/src/linux/include/asm.

The running kernel should be consistent with your kernel source!

The Linux 2.4.x kernel generation is supported. The updated modules are now located in Gxsm/plug-ins/hard/modules, while the 2.2.x version modules are frozen and untouched residing in Gxsm/pci32/modules. The 2.4.x modules are designed to work with DevFs (Device File System), but it's not necessary to have it enabled.

27.5.2 Compiling 2.4.x kernel version modules for PC31/PCI32

Change to Gxsm-2.0/plug-ins/hard/modules

make

The modules pc31.o and pci32.o should be there now.

There is no need to make devices if you are using devfs, else you need to do it yourself!

The devices are automatically generated here (devfs): /dev/pcdsps/tms320

You need to do insmod/install the modules yourself.

27.5.3 some helping scripts

All file paths in the here provided scripts are hand hacked, so please adjust to you needs!

It's handy to install a small "rc.boot script" like:

```
percy@nano:~$ cat /etc/rc.boot/pci32

#!/bin/sh
echo Loading PCI32 module...
insmod pci32
```

```
echo Done.
echo Loading an starting SPM DSP program...
# replace the paths and xafm.out file locations by your needs
/usr/local/bin/loadpci -d /dev/pcdsps/tms320 \
    /usr/local/stow/gxsm/share/gxsm/pci32/tms320-htstm.out
echo Done.
```

And for (emergency) DSP restart a small script like may be handy:

```
percy@nano:~$ cat /usr/local/stow/gxsm/share/script/dsp-reset.sh

#!/bin/sh
sync
echo ----------------------------------------------------------------------
echo DSP emergency restart, be prepared!
echo ----------------------------------------------------------------------
echo Killing all possibly DSP using apps next.
echo Ready to go? I will kill all DSP Apps: goszi, gxsm and lcd!!!
echo ""
echo hit return to go
read
echo ----------------------------------------------------------------------
sync
echo killing goszi
killall -9 goszi
echo killing gxsm
killall -9 gxsm
echo killing lcd
killall -9 lcd
echo waiting 2s....
sleep 2
echo Done.
echo ----------------------------------------------------------------------
echo Syncing Disks for emergency
sync
echo ----------------------------------------------------------------------
echo "Resetting DSP, Loading an (re)starting SPM DSP program..."
echo ----------------------------------------------------------------------
sync
# adjust paths and .out file name to you needs!
loadpci -d /dev/pcdsps/tms320 /usr/local/stow/gxsm/share/gxsm/pci32/tms320-htstm
.out
echo Done.
echo ----------------------------------------------------------------------
echo Have fun...
echo ""
```

```
echo hit return
read
```

27.5.4 Installing

The required module can now be loaded using insmod.

The to access the driver a char device "/dev/pcdsp" is needed (only 2.2.x kernel):

$ mknod /dev/pcdsp c 254 0 -m 666 (as root)

will create this.

Check if everything is right: Have a look at "/var/log/messages" while doing insmod / rmmod:

$ tail -f /var/log/messages (as root)

The Major-Number reported at insmod should be consistent with the device major 254!

27.6 DSP: PC31, PCI32 Card and Tools

Get the Gxsm module from CVS!

The DSP Card needs to be feed with the DSP program, therefore we need some tools:

Change to "Gxsm/pci32/load"

do there a

$ xmkmf; make

28 BurrBrown Card (SPALEED)

GXSM supports a mode to operate a SPA–LEED using the non DSP using Burr-Brown AD/DA/counter cards of type BB_PCI2000XX. Therefor a special Linux kernel module is available, which emulates the DSP functionality and accesses the BB-card.

FYI: There is a real DSP card SPA–LEED software also, it's very customized and running on the PC31 DSP board. Also the Signal Ranger has the potential to operate a SPA–LEED, but no DSP soft is available yet – Contact the GXSM team for details. ♪♫

28.1 BurrBrown Module Installation

Compilation of the kernel modules:
In Gxsm/plug-ins/hard/modules/ ein make ausführen. (dazu ist eine Installierter Kernel 2.4.x/DevFs + Source/Header notwendig)
Then you will find:
Gxsm/plug-ins/hard/modules/:
Makefile

dspemu.c	→ basic DSP emulation
spaleed_emu.c	→ SPA-LEED simulations module
spaleed_bb.c	→ SPA-LEED BurrBrown module
	(Base Adr: 0xd0000 is preset, editor!!)
dspbbspa.o	→ loadable module with insmod for BB
dspspaemu.o	→ simulation module

28.2 Important GXSM preferences, SPA–LEED/BB

To assure correct operation of the SPA–LEED DSP emulation software/module in conjunction with the GXSM system the following SPA–LEED related preferences have to be set exactly as described in table **??**.

Preference-key	Value
Hardware/Card	Innovative_DSP:SPA
DataAq/PIDSrcA1	Counts, all other channels to "-"
Instrument/Type: Instrument/XCalib, YCalib Instrument/EnergyCalibVeV	SPALEED set as desired set as desired
User/Unit	V

Table 28.1: Important GXSM preferences for usage Burr-Brown hardware with a SPA–LEED and the DSP emulation module.

29 DSP subsystem Signal Ranger

The Signal Ranger DSP board[1] is powered by the *TMS320C5402* running at 100 MHz and uses a full speed USB interface for all communications with the host PC and GXSM.

29.1 GXSM-2 and the Signal Ranger DSP board

GXSM supports the Signal Ranger DSP board (SR-STD and SR-SP2 – I'll call it just SR in the next sections). To use the SR with GXSM the Linux SR support needs to be installed first, please refer to the Linux Signal Ranger Project and Documentation
http://sranger.sourceforge.net
and
http://sranger.sourceforge.net/sranger-main.pdf
for guidance.

The SR DSP SPM software is available via CVS from the SR-Linux Project as well in source and binary (COFF2). The SR-DSP binary for SPM control via GXSM can be found here in the SR-CVS:
SRanger/TiCC-project-files/FB_spmcontrol/FB_spmcontrol.out

There is a configuration tool to setup/alter some option at runtime (anytime) the SR-DSP SPM is running:
SRanger/TiCC-project-files/FB_spmcontrol/sr_spm_control.py

And there is a handy and eye-candy DSP-Watch tool to monitor some AIC data in real-time:
SRanger/TiCC-project-files/FB_spmcontrol/spm_watch

29.2 Important GXSM preferences, SRanger specific

To assure correct operation of the SR-DSP software in conjunction with the GXSM system the following SR-related preferences have to be set exactly as described in table **??**. These settings may not be your **not** GXSM defaults, so please adjust!

[1] The Signal Ranger DSP board is build by the Canadian Company Soft dB (**http://www.softdb.com**) and was designed by B. Paillard at University of Sheerbroke, Quebec

[2] this could be also /dev/sranger1, 2, . . . 15 in case of multiple connected SR boards.

Preference-key	Value
Hardware/Card or: Hardware/Card Hardware/Device	SRanger:SPM SRangerTest:SPM /dev/sranger0[2]
Instrument/Type	STM or AFM
Instrument/ScanOrgCenter	true
Analog/DigRangeIn Analog/VoltMaxIn Analog/DigRangeOut Analog/VoltMaxOut	32768 10 V 32746 2.06 V

Table 29.1: Important GXSM preferences for usage of the SR hardware.

29.3 SR connections

Here is an schematic example of the SR - STM/AFM interface. The recent DSP SPM software allows DSP-level offset adding for simplicity, this is the current default mode, but using the DSP configuration tool this option can be changed on-the-fly (**??**).

GXSM supports a Besocke-Type Mover control (see **??**), which is connecting to the SR. The special desigend ramp like wave forms (parabola shaped) are DSP generated and can be scaled in time and amplitude. The signal is available on X_0 and Y_0 (XY-motions, in case of rotation both are the same), the X_0 signal is mirrored on the "M" output with FIR disabled for higher frequency response.

Due to the to ±2.06 V limited AIC output voltage range you may need extra amplification to fit to usual 10 V range inputs.

Figure 29.1: SR connections. X_0, Y_0, Z_0 are X/Y/Z offset signals. X_S, Y_S are the X/Y scanning signals and Z_S is the Z, i.e. feedback coupled signal. I_t is the "tunneling current" voltage input connected to the output of the current-voltage converter (STM) or the force/deflection signal (AFM) or frequency shift signal (DFM). U_B is the bias voltage typically connected to the sample. M (experimental for "Motor"). AIC inputs 0...4 can be used for any signal to be recorded while scanning.

Figure 29.2: Piezo amplifier block diagram. Note: the offsets are currently suggested to be at the same gain as the scanning signals, but this will changed/be flexible in future. The gains of 1/2/5 or only suggestions and can be configured to arbitrary values in GXSM.

29.4 SR DSP soft and GXSM launch

I prefer to use this srboot-gxsm.sh script to startup all, please customize the directory path "xxxx":

```
#!/bin/sh

# remind the user...
gdialog --title "Attention before SR-boot" \
        --msgbox "Please zero/unconnect all SR AIC inputs\n\
                  for automatic offsetcompensation now, then click OK\n\
                  boot the SR-SPM program." 5 70

cd /home/xxxx/SRanger/loadusb
./loadusb ../TiCC-project-files/FB_spmcontrol/Debug/FB_spmcontrol.out

gdialog --title "SR-boot OK" --msgbox "Please reconnect all SR AIC inputs now!"

# just create a dir for today... this is the way I like it
cd /home/xxxx/MyData

mkdir `date --iso-8601`
cd `date --iso-8601`

# and start gxsm2 there!
gxsm2
```

In addition you may want to launch the "spm_watch" tools and check the SR-SPM settings. Launch the sr_spm_control.py script and click on "SR DSP SPM settings", this will pop up some options:

Figure 29.3: SR DSP SPM software configuration tool (Python script).

29.5 SR DSP code design overview

Let's start with an overview to the whole thing looking at diagram **??**. The SPM user interacts with the SPM via the GXSM software using the X11 window system. GXSM itself sends and receives commands and data via a Unix device, which is managed by a hardware dependent Linux kernel module. This module has to assure, that the requested actions are performed by the connected hardware. This makes Gxsm independent of any hardware. The Signal Ranger DSP board (SR-STD) is build around a TMS320C5402 DSP, and provides 8 channels of Analog to Digital Converters (ADC) and 8 channels Digital to Analog Converters (DAC) with each 16 (15) bits resolution and an optional digital IO expansion (CoolRunner). The DSP board runs a SPM specialized software. The SPM head/piezo tube(s) are connected to a piezo drive, which provides the needed symmetric $\pm X$, $\pm Y$ and Z voltages at various gains ($1\ldots100\times$), as needed. Analog signal sources (e.g. tunneling current, amplified via a current voltage converter (IVC) and auxiliary (AUX) signals) are connected to the analog input(s). Digital ports are available for a slider control or other optional purposes.

Lets look closer to the software running on the DSP machine, which dose the real hard work. In diagram **??** the three main working blocks are shown. The DSP program is designed like a state machine, this means that the system has always to be in one, or a allowed combination, of the defined states. The *Idle Loop* does all state controls and communications. The data processing interrupt subroutine (ISR) is always running at a constant rate of 22 kHz and run the feedback loop and any enabled additional shared task.

All tasks are expected to be "one shot" for each call and without any lengthy loops – so they have to do one step at a time and the next at next call.

Figure 29.4: Schematic diagram of a typical SPM control system using GXSM and the SR

Figure 29.5: Schematic diagram of SR-DSP subsystem

30 DSP subsystem Signal Ranger MK2-A810

The Signal Ranger Mark2 DSP with the A810 analog board[1] is powered by the *TMS320C5502* running at 300 MHz and uses a high speed USB-2.0 interface for all communications with the host PC and GXSM.

This project provides Linux support for the Signal Ranger Mark 2 (SR-Mk2) via **http://sranger.sourceforge.net**.
It is tested for all Linux Kernels 2.6.20 and later. A memory debug utility with still experimental experimental COFF loader functions is also available. However, as the SR-Mk2 is always self booting from flash at power up, so far best practice is to pre-program it using the manufacturer supplied tools (mini-debugger) and then all you need is to plug it in to the Linux box and you are all set.

The SR-Mk2 with MK2-A810 is the NEW (officially available by SoftdB from Feb/2009) and most recent and now full supported and recommended DSP platform for the Open Source SRanger-DSP and GXSM SPM control system. It also will be easier to handle, load DSP/FPGA code once to on-board FLASH using the manufacturer supplied tools, and it is ready after power up and USB plug-in! Only need to install the Linux module. SR-Mk2:

- 300 MHz TMS320C5502 DSP 16-bits fixed point
- XC3S400 FPGA 63 user-configurable I/Os
- High-Speed USB 2.0 interface to host PC (approx. 19 Mbps)
- 64 KBytes on-chip (DSP) dual-port-access RAM
- 4 MBytes external 75MHz SDRAM
- 2 MBytes external FLASH ROM
- Can be stand-alone unit or connected to PC (Self boot option and FPGA load from FLASH)
- With AIC Analog-16 expansion: 16-Channel 16-bit AD/DA (0...26 kSPS) (not recommended for SPM, but may sound/acoustic applications)

[1] The Signal Ranger DSP board is build by the Canadian Company Soft dB (**http://www.softdb.com**) and was designed by B. Paillard at University of Sheerbroke, Quebec

- With Analog-810 expansion MK2-A810:
  - 8-Channel 16-bit high performance AD up to 150kSPS, minimal loop delay (2 samples), providing high stability and precision
  - 8-Channel 16-bit high performance DA up to 150kSPS, minimal loop delay (2.5..3 samples), providing high stability and precision
  - Two Counter/Timer channels with precision hardware gating, synchronous to the AD/DA sampling (for example used for Channel Electron Multiplier Detector readouts)
  - General Purpose Digital IO

30.1 GXSM-2 and the MK2-A810 SPM Controller

GXSM supports now the latest (Feb/2009) Signal Ranger Mark2 DSP board with the SPM dedicated analog expansion A810 referred to as MK2-A810.

Linux Kernel Module and support via our Signal Ranger Linux Support Project:
http://sranger.sourceforge.net

All needed from here is the kernel module and the DSP binary – unless you want to start working on the also open source DSP code your self. The SR DSP SPM software is available via CVS from the SR-Linux Project as well in source and binary (COFF2). The MK2-A810 DSP binary for SPM control via GXSM can be found here in the SRanger-CVS:

SRanger/TiCC-project-files/MK2-A810_FB_spmcontrol/FB_spmcontrol.out

The also needed generic A810 support FPGA logic (DSP-A810 interface) from SoftdB is here:

SRanger/TiCC-project-files/MK2-A810_FB_spmcontrol/
SR2_PrepareFlashForSR2_Analog_810/SR2_Analog_810_V200.rbt

These two files are needed to program the Flash of the MK2 using the manufacturers tools for use with Gxsm. There is also a configuration and test tool to setup/alter some option at runtime (anytime) the SR-DSP SPM is running:

SRanger/TiCC-project-files/MK2-A810_FB_spmcontrol/python_scripts/mk2_spm_control.py

30.2 Important GXSM preferences, SRanger specific

To assure correct operation of the MK2-A810 DSP software in conjunction with the GXSM system the following SR-related preferences have to be set exactly as described in table **??**. These settings may not be your **not** GXSM defaults, so please adjust!

Preference-key	Value
Hardware/Card	SRangerMK2:SPM
Hardware/Device	/dev/sranger_mk2_0[2]
Instrument/Type	STM or AFM
Instrument/ScanOrgCenter	true
Analog/DigRangeIn	32768
Analog/VoltMaxIn	10.0 V
Analog/DigRangeOut	32746
Analog/VoltMaxOut	10.0 V

Table 30.1: Important GXSM preferences for usage of the SR hardware.

Further data input channels have to be set to type "FLOAT". Refer to the HwI secion for the MK2-A810.

30.3 SR connections

Here is an schematic example of the SR - STM/AFM interface. The recent DSP SPM software allows DSP-level offset adding for simplicity, this is the current default mode, but using the DSP configuration tool this option can be changed on-the-fly (**??**).

GXSM supports a Besocke-Type Mover control (see **??**), which is connecting to the SR. The special desigend ramp like wave forms (parabola shaped) are DSP generated and can be scaled in time and amplitude. The signal is available on X_0 and Y_0 (XY-motions, in case of rotation both are the same), the X_0 signal is mirrored on the "M" output with FIR disabled for higher frequency response.

30.4 SR DSP soft and GXSM install and launch

In brief:

1. Make sure the MK2-A810's flash is up-to date with the latest DSP code for Gxsm and (FPGA) logic matching the Gxsm version. (Best to do CVS checkout about the same time for the CVS modules Gxsm-2.0 (Gxsm project) and SRanger modules from SRanger project.)

2. Compile and install the mk2 kernel module if not yet done, follow the README instructions there. May adjust your "udev" permission settings to grant automatically read/write access to the device[3].

[2] this could be also /dev/sranger_mk2_1, 2, ... 15 in case of multiple connected MK2 boards.

Figure 30.1: MK2-A810 connections. The A810 ADC and DAC default configuration is shown in the picture. X_0, Y_0, Z_0 are X/Y/Z offset signals. X_S, Y_S are the X/Y scanning signals and Z_S is the Z, i.e. feedback coupled signal. I_t is the "tunneling current" voltage input connected to the output of the current-voltage converter (STM), further feedback sources like force/deflection signal (AFM) or frequency shift signal (DFM) can be connected simultaneously to ADC0-3. V_{Bias} is the bias voltage typically connected to the sample. M (experimental for "Motor"). ADC inputs 4-7 can be used for any signal to be recorded while scanning. Also two puls counting channels are available. Note: The shown Bias – Feedback-I connection is for self-test configuration only.

3. Compile and install Gxsm-2.0
4. Power cycle the MK2-A810 before pluggin into the Linux system the first time after programming, plug it in.
5. Start gxsm2 and configure it according to **??**, restart Gxsm for new settings to be used.

Example permissions configuration: Open /etc/udev/rules.d/40-permissions.rules and add at end

```
KERNEL=="sranger_mk2_[0-4]" MODE="0666"
```

[3]Edit your permission rules and add a line for the mk2 module with desired permissions.

Part IV

Appendix

A Using Python

A.1 Introduction

Sometimes a GUI is not the right thing to evaluate your data. If what you need is an interface to a programming language that allows you to play around with your NetCDF-files easily, then Python is for you.

Python is a script language which earned itself quite a bit of respect in the programmers community. For more information visit python.org.

There are several powerful Numeric and Scientific packages for Python. To make use of Pythons power, we need some of these modules, that allow us to access NetCDF files and do advanced numeric computation.

Here you can get the modules and read the documentation:

SciPy SciPy is an open source library of scientific tools for Python. SciPy supplements the popular Numeric module, gathering a variety of high level science and engineering modules together as a single package.

SciPy includes modules for graphics and plotting, optimization, integration, special functions, signal and image processing, genetic algorithms, ODE solvers, and others.
http://www.scipy.org

Scientific Python [1] ScientificPython is a collection of Python modules that are useful for scientific computing. In this collection you will find modules that cover basic geometry (vectors, tensors, transformations, vector and tensor fields), quaternions, automatic derivatives, (linear) interpolation, polynomials, elementary statistics, nonlinear least-squares fits, unit calculations, Fortran-compatible text formatting, 3D visualization via VRML, and two Tk widgets for simple line plots and 3D wireframe models. There are also interfaces to the netCDF library (portable structured binary files), to MPI (Message Passing Interface, message-based parallel programming), and to BSPlib (Bulk Synchronous Parallel programming).
http://starship.python.net/ hinsen/ScientificPython/

Numpy [2] Numerical Python adds a fast, compact, multidimensional array language facility to Python.
http://www.pfdubois.com/numpy/

[1] Debian: python-scientific
[2] python-numeric and python-numeric-ext

Gnuplot-py [3] Gnuplot.py is a Python package that interfaces to gnuplot, the popular open-source plotting program.
http://sourceforge.net/projects/gnuplot-py/

Matplotlib matplotlib is a Python-2D-plotting library which produces high quality figures. Gxsm2 uses it for plotting line profiles. http://http://matplotlib.sourceforge.net/

A.2 A first example

Now follows a sample program which demonstrates how to use Python to analyze a nc-file, created by Gxsm. If you do not want to type the program yourself, simply copy and paste it from the Gxsm-python.tex file in the Gxsm-CVS on Sourceforge.

This example was tested with Python 2.1 on Debian 3.0. Your modules may be named slightly different on other systems.

```
#filename: python-example1.py
from Numeric import *
from Scientific.IO.NetCDF import *

import Scientific
import Scientific.Statistics

import Gnuplot

file = NetCDFFile('demo.nc', 'r')

def wait(str=None, prompt='Press return '):
    if str is not None:
        print str
    raw_input(prompt)

foo = file.variables['H']
bar = file.dimensions['dimx']
#mac = file.variables[0]
foo_array = foo[:]
#foo_units = foo.units

print "--Variable Keys---------------------"
for varname in file.variables.keys():
        print varname

print "--First Row-------------------------"
print foo_array[0][0][0]
```

[3] Debian: python-gnuplot

```
print "--Second Row------------------------"
print foo_array[0][0][1]
print "--Full Array------------------------"
print foo_array[0][0]

print "--Length----------------------------"
#print foo_units[:]

print len(foo_array[0][0][0])
print "--Sum columnwise--------------------"
print sum(foo_array[0][0])

print "--Repeat----------------------------"
a=foo_array[0]
b=a[0]
c=b[0]
print "-- Full Array ----------------------"
print b
print "--First Row-------------------------"
print c

print "--Cross-correlation of first row----"
d=cross_correlate(c,c,2)
#www.pfdubois.com/numpy/html2/numpy-9.html#pgfId-67917
print d

g = Gnuplot.Gnuplot()
x = arange(16, typecode=Float)
y = arange(16, typecode=Float)

for i in range(16):
  y[i] = d[i]

e = Gnuplot.Data(x, y, with="lines")
g.plot(e)
wait()
file.close()
```

A.3 A second short example

If you want to extract single values from your nc-files, the easiest way is to use a pipe of `ncdump` and `grep`:

```
ncdump stefan002-Xp-Topo.nc | grep "dsp_utunnel"
```

But if you want to do some more calculations or manipulations, then you can write a short script like this:

```
#!/usr/bin/python
#filename nc-read.py
from Scientific.IO.NetCDF import *
import sys
file = NetCDFFile(sys.argv[1], 'r')
Bias = file.variables['dsp_utunnel']
print sys.argv[1], " Bias:", Bias.getValue(), getattr(Bias, 'unit')
file.close()
```

Use your favourite texteditor to write a wrapper:

```
#!/bin/sh
for file do
python nc-read.py $file
done
```

A.4 Some more fun with Python: Images

As mentioned before Python can be extended with modules, which greatly enhance its functionality. Here we would like to introduce another great package: PIL. It is the Python Image Library by Fredrick Lundh.

Scanned images are, when all is said and done, nothing more than 2D-pictures. Like those many file formats that are out there: PNG, GIF, JPEG, PPM and so on.

Python can read and manipulate these images with help by the PIL. In a simplified meta-language a program could look like this.

- Open a JPEG-File
- Smooth it.
- Rotate it 90 degree.
- Save it in the PNG file format.

In Python this program looks like this.

```
#filename: python-example2.py
import Image, ImageFilter
im = Image.open("myimage.jpg")
im = im.filter(ImageFilter.SMOOTH)
im = im.rotate(90)
im.save('myimage.png', 'PNG')
```

As you can see, this program can be simplified to create a simple image converter by removing the smoothing and the rotation.

Now why is this interesting for us? Wouldn't it be nice to write a program like this:

- Open a Gxsm-NetCDF-File
- Smooth it.
- Rotate it 90 degree.
- Save it in the PNG file format. Or GIF or whatever.

or in true Python:

```
#filename: python-example3.py
import Image, ImageFilter
im = Image.open("myimage.nc")
im = im.filter(ImageFilter.SMOOTH)
im = im.rotate(90)
im.save('myimage.png', 'PNG')
```

We now show you how to accomplish this and more. To be able to recognize an image format, the Image Library needs to have a *driver* for your file format. You have to install the PIL first of course. I will not help you there.

Find the directory where all the other drivers are stored, for example by searching the file GifImagePlugin.py. For me that is

/usr/lib/python2.1/site-packages/PIL/.

As you can see in this directory, a large number of file formats is already supported. We now write a driver for Gxsm-Netcdf-files.

Here it is: Copy it to the above mentioned directory or link it from where you put it in your home, if you want to modify it. I removed the empty lines here, so it will all fit on one page.

```
#filename: GxsmnetcdfImagePlugin.py
import Image, ImageFile
class Gxsmnetcdf(ImageFile.ImageFile):
  format = "nc"
  format_description = "Gxsm netcdf"
  def _open(self):
    self.i=1
    self.mode = "L"
    self.fp.seek(0)
    t=self.fp.read(3)
    if t != "CDF":
        raise SyntaxError, "not a CDF file"
    from Scientific.IO.NetCDF import *
```

```
    self.f= NetCDFFile(self.filename, 'r')
    self.size = self.f.dimensions['dimx'], self.f.dimensions['dimy']
  def load(self):
    #for some reason, load is executed twice ->dont need this
    self.i=self.i+1
    if self.i!=2:
        return
    from Numeric import *
    foo=self.f.variables['H']
    a=foo.getValue()
    b=a[0][0]   #We need only the data
    d=ravel(b)  # linearize
    d=d-min(d)  #  map to [0..256]
    d=d*256./max(d)  # we should catch div by 0
# but this happens only when the image has only one color
    self.im = Image.core.new(self.mode, self.size)
    self.im.putdata( d )
Image.register_open("nc", Gxsmnetcdf)
Image.register_extension("NC", ".nc")
```

I will not go through it line by line. Visit
http://www.pythonware.com/library/pil/handbook/decoder.htm
to find out more about writing an image driver. Only so much: We use the Scientific-module to decode the NetCDF-file. We use Numeric's array-functions to store the data of the image. We map the data to the interval 0 . . . 256 to get a grayscaled image. We create an image from scratch and put the array into it. That's it.

When you did everything correctly, the program python-example3.py runs without error.

A.4.1 pilprint and pilconvert

The PIL-driver which was developed in the previous chapter can be used with all the tools, that come with the library like pilconvert[4]:

```
pilconvert.py file.nc file.png
```

This way you can easily convert your nc-file to lots of different formats, like GIF, PNG, EPS, TIF, JPG and more.

You can also print nc-files directly with

```
pilprint.py file.nc
```

This will print a postscript-version of your file to stdout. Use the commandline-options to -p or -P to print to the printer directly.

The Gxsm-project also supplies a python-script that allows more elaborate printing options. In the scripts directory you find `gxsmprint.py` and `gtkgxsmprint.py`. The former is a

[4]Debian: python-imaging-doc

modified version of pilprint, that adds some information from the nc-file to your page. The latter is a graphical frontend, that allows easy usage for commandline-muffels.

For these programs we need the python-modules for imaging[5] and netcdf-access[6]. For N-up printing we use `psnup` and `ps2ps`. Be sure to have them. Furthermore we need the python-gtk2 modules[7] for the frontend.

Figure A.1: Screenshot of Sketch with STM-image imported.

A.4.2 sketch

Sketch (recently renamed to Skencil) is a vector-oriented drawing program for several platforms written in Python. Sketch uses the PIL for inclusion of images, thus, as soon as you installed the GxsmnetcdfPlugin.py, Sketch can include Nc-files, like any other graphics format. Repeat: You need no further conversion, but can work directly with your Nc-file.

Sketch has many features, runs stable is extensible with Plugins, has LaTeX support, knows gradient-fill, pattern fill, can export to PS and PDF. The development-branch in CVS also supports alpha-transparency. Be sure to have a look at it.

[5] Debian: python-imaging
[6] Debian: python-scientific
[7] Debian: python2.3-gtk2

A.4.3 Image-analysis

Next I will present another python-example, that shows some of the image manipulations and analysis features. The program contains the weblinks which direct you to the documentation.

```
#filename: python-example4.py
import Image,ImageFilter,ImageOps,ImageChops,ImageStat,ImageDraw
import Scientific.Statistics

# This is not really necessary but should speed up loading.
import PngImagePlugin
import GxsmnetcdfImagePlugin
Image._initialized = 1
#

OPT=1

somefiles=['110603009PTopo-p.nc']

for j in somefiles:
  print "Opening: ", j ,
  i = Image.open(j)
  out = i.copy()

  maindata = list(out.getdata())

  if OPT==0:
    print
    s=Scientific.Statistics
#www.esrf.fr/computing/bliss/python2/
#     Scientific/HTML/Scientific_35.html
    print "moment (2) is: ", s.moment(maindata,2)
    print "mean is: ", s.mean(maindata)
    print "variance is: ", s.variance(maindata)
    print "stddeviation is: ",s.standardDeviation(maindata)
    print "median is: ", s.median(maindata)
    print "mode is: ", s.mode(maindata)
    print "norm.mode (2) is: ",s.normalizedMoment(maindata, 2)
    print "skewness is: ", s.skewness(maindata)
    print "kurtosis is: ", s.kurtosis(maindata)
    l=ImageStat.Stat(i)
#www.pythonware.com/library/pil/handbook/imagestat.htm
    print "extrema are: ", l.extrema
    print "count is: ", l.count
    print "mean is: ", l.mean
    print "rms is: ", l.rms
```

```
  elif OPT==1: #play around with the image.
#www.pythonware.com/library/pil/handbook/imagefilter.htm
#Available filters are
# BLUR, CONTOUR, DETAIL,EDGE_ENHANCE,EDGE_ENHANCE_MORE,
# EMBOSS,FIND_EDGES,
# SMOOTH,SMOOTH_MORE,SHARPEN
    c = i.filter(ImageFilter.CONTOUR)
    c = c.filter(ImageFilter.EMBOSS)

#www.pythonware.com/library/pil/handbook/imagechops.htm
    c = ImageChops.add(c,c)
    c = ImageChops.invert(c)
    out = ImageChops.add(i,c)

  elif OPT==2: #draw a cross on the image
    draw = ImageDraw.Draw(out)
    draw.line((0, 0) + out.size, fill=255)
    draw.line((0, out.size[1], out.size[0], 0), fill=0)
    del draw

  elif OPT==3:
# This is a filter which modifys a number of STM-Images.
# It creates a sort of 3d look.
# It is a simplified Koehler-algorith.
    SIZEX, SIZEY = i.size
    from Numeric import *
    d=ravel(maindata)    # make 1d.
    d=d-min(d)    #map to [0..256]
    d=d*256/max(d) #if max()==0 -> oops
        # math-range error may occur here theoretically
    z2 = zeros( (SIZEX*SIZEY) )
    left = zeros( (SIZEX*SIZEY) )
    right = zeros( (SIZEX*SIZEY) )
    for i in range(1, SIZEX*SIZEY -1 ):
      if (i+1)%SIZEX==0:
        left[i]=0
      else:
        left[i] = d[i+1] - d[i]
        right[i] = d[i-1] - d[i]
    e = abs(left)+abs(right)
    e = e*10  #increase this one
    e = d+e
    out.putdata( e )

  print "  Now saving ", j, " as PNG.",
```

```
#To learn about other formats goto:
#effbot.org/zone/pil-formats.htm
  out.save(j+'.png', 'PNG')
  print " OK. "
```

You can modify the OPTS variable at the beginning of the file to go through the different options. The PIL is a stable and sophisticated library. With it you can modify brightness or contrast, add text and geometric shapes and even create Postscript.

Be sure to have a look at it

http://www.pythonware.com/library/pil/handbook/index.htm

B Pyremote

A Word of Warning! Python is a fully featured programming language. It can read and write files and thus erase them, too. Never trust a script from an unclear source. Never login as root-user when running scripts you do not understand or you may destroy important files or worse open your system to crackers.

B.1 Introduction

Pyremote is an interface to the programming language Python, that was already introduced in previous chapters. Pyremote allows you to write scripts in Python to control the behaviour of Gxsm. It is a replacement for the old remote control that is based on FIFOs. This old implementation lacked flexibility, was no plugin and often had problems to sync the script with the application.

Entering Python. When you first start Gxsm and press *main menu/Tools/Pyremote* you will get an error message. This is because Gxsm tries to find your script and must fail, because you did not write one or you set the environment variable `PYTHONPATH` wrongly (See installation instructions in the next section). Quit Gxsm or open another console and use your favorite text editor to create a file with the name remote.py in the current directory. (The name remote.py is hardwired into the plugin, more on this later.) Save the file with the following content:

```
print "Hello Gxsm."
```

Now restart Gxsm and press *main menu/Tools/Pyremote* again. Your console output should look like this.

```
[...]
AppBase
AppBase::WidgetInit: SPM Scan Control
spm_scancontrol_query:res
spm_scancontrol_query:done
AppBase
AppBase::WidgetInit: DSP Control
Hello Gxsm.
```

Congratulations. Gxsm now can execute Python scripts. But you cannot control Gxsm yet. Gxsm implements a Python module. This module is called emb. You can import it and access

Gxsm through the commands provided by it. Add the following to your Python script (you don't need to restart Gxsm after editing) and start *main menu/Pyremote* from the menu:

```
import emb
print "Hello Gxsm."
emb.logev('beware')
```

Now have look at the Gxsm logfile in the current directory. It is called Ev_Sep2003.log or similar. The last line should look like this

```
Tue Sep 23 16:35:56 2003 RemoteLogEv :   beware :0.000000   [...]
```

This was our first example to control Gxsm. Now we extend this example:

```
import emb
print "Hello Gxsm."
emb.logev('my automatic scan')
emb.startscan()
```

When you start this script, Python will order Gxsm to start a scan using all current parameters. You can change the settings with the set command:

```
import emb
print "Hello Gxsm."
emb.logev('my automatic scan')
emb.set("RangeX", "345")
emb.set("RangeY", "345")
emb.startscan()
```

B.2 Hijacking

'Hijacking' is to take over control of a process. When you start a remote-session, you might not yet know what you really want to do with it. To keep flexibility I designed the following little script. With this script as your remote.py you are able to feed commands into your running Gxsm session from remote even without a graphical console.

It works as follows: We check for the existence of a command-file (Here hijack.py). We check and trigger our exit-condition if the file size exceeds 20. We execute the file as Python-script and remove it afterwards.

```
import emb
import os
import time
import sys
FILENAME='hijack.py'
```

```
BOUND=20
while (1):
        if (os.access(FILENAME,os.F_OK)): #if file exists
                len_log = os.stat(FILENAME)[6] #check size
                if (len_log > BOUND):
                        break
                execfile(FILENAME)
                os.remove(FILENAME)
        time.sleep(5)
```

When you execute pyremote, this script will infinitely wait for input.

Now do a `echo "emb.startscan()" > hijack.py`. When the next cycle begins, this line will be executed and afterwards forgotten. You can enter all commands. If you want to execute longer pieces of code, you have to increase the file size for the exit condition.

With `echo "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" > hijack.py` or any other line, that has more bytes than your boundary, you can quit the script.

B.3 Another real world example

Taken from a correspondence in the GXSM forums on pyremote.

```
# Written by PZ. Updated by StS.
import emb
print "Hello GXSM-2 and Nanostencil!"

emb.logev ("Starting pyremote script.")

for i in range (12):
  emb.logev ("SetPoint -0.15 * 50Hz)")
  emb.set ("DSP_SetPoint", "-0.15")
  emb.logev ("Remote Scan Start")
  emb.startscan ()
  emb.logev ("SetPoint -0.05 (-2.5Hz)")
  emb.set ("DSP_SetPoint", "-0.05")
  emb.logev ("Remote Scan Start")
  emb.startscan ()
  emb.logev ("SetPoint 0.1 -- Tip away!")
  emb.set ("DSP_SetPoint", "0.1")
  emb.logev ("Sleeping 3600s --- Zzzzzz...")
  emb.sleep (36000) # sleep one hour

print "Done."
```

C Terms and conditions

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330
Boston, MA 02111-1307, USA.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) 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 this service 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 make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. 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.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute 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 and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute

the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source

along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries,

so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the 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 a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, 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.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE 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.

END OF TERMS AND CONDITIONS

Part V

Index