KAME ver 8.0 2026/4/2　北川　健太郎

序

kameはRuby/Pythonスクリプトによる自動化に対応した、C++で書かれたマルチスレッドで（ドライバを書けば）どんな測定にも対応できる測定プログラムです。

当研究室のODMR/固体NMRを用いた物性研究に使ってますが、特にODMR/NMR専用という訳ではなく汎用です。対応している測定器の組み合わせならばプログラムの変更が必要ないようになっています。グラフ表示がOpenGLにより速いこと、測定をPython/Rubyスクリプトで自動化できるのが特徴です。クロスプラットフォーム(Mac OS X/Windows)です。Python及びJupyter対応はオプションです。Linuxビルドは現在メンテしていません。

ライセンスはオープンソースのGPL(GNU public license) version 2 or laterです。

特徴

一般

マルチスレッド（同期にソフトウェアトランザクショナルメモリ使用）

OpenGLによる高速なグラフ描画

スカラ量（温度、電圧等）の組み合わせは任意にグラフ化可能

ほぼすべての設定等の保存／読み込みが可能

Python3/Rubyスクリプトによりほぼすべて制御可能

通信エラー等は例外捕捉により安全に処理

装置からの取得データを全記録／後日再解析可能

NMR部分

National Instruments社のDAQデバイスをパルサー／オシロとすることができる

FFT/MUSIC/MEM等によるリアルタイムスペクトラム解析

緩和曲線(T_1, T_2, $T_{st.e.}$)もリアルタイムフィット

フーリエステップサムによる磁場／周波数スイープスペクトル測定

緩和・スペクトル測定において測定後でも窓関数（畳み込み）選択可能

必要ライブラリ

以下に必要なライブラリ等を列挙します。

実行時

kameはQtソフトウェアなので、Qtが必須です。

Python3（測定の自動化に使用できます（オプション）。ビルド時はpybind11を使用します。）

Jupyter (notebook/qtconsole) （IPythonを使用する場合）

Ruby (オブジェクト指向言語。測定の自動化と設定の保存／読み込みに使います。Macでは最初から入ってます。)
http://www.ruby-lang.org/ja/

GSL(GNU Scientific Library。非線形フィットと補完に使ってます。)
http://www.gnu.org/software/gsl/

FFTW(ver.3系列。FFTライブラリ。)
http://www.fftw.org/

linux-gpib(LinuxでGPIBを使用する場合に必要。)
http://linux-gpib.sourceforge.net/

National Instruments NI-488.2ドライバ(Windows x86(-64)でNIのGPIBドライバを使用する場合のみ必要。)

National Instruments DAQmx (NI DAQデバイスを使用する場合のみ必要。)
http://www.ni.com/dataacquisition/ja/nidaqmx.htm

libtool-ltdl (Linux/Macのみ。GNU libtool dynamic module loader。起動時にモジュールを読み込む為に必要。)

libusb (USB機器の制御に必要。)

libdc1394 (Macのみ。Firewire接続カメラを使用する場合。)

Euresys egrabber (grablink/coaxlink接続カメラを使用する場合。)

zlib (Rawデータの圧縮用)

インストール

kameプログラム本体等は、以下のURLでダウンロードできます。
https://kitag.issp.u-tokyo.ac.jp/%e8%87%aa%e5%8b%95%e5%8c%96%e5%af%be%e5%bf%9c%e6%b8%ac%e5%ae%9a%e3%83%97%e3%83%ad%e3%82%b0%e3%83%a9%e3%83%a0kame/

Mac OS Xへのインストール

ソースコードからインストールする

TohoeとSequiaとVenturyとQt 6.10+XCode+XCode commandline tools+MacPortsで確認しています。

MacPorts及び、pybind11, py3**-pybind11, py3**-numpy, fftw-3, gsl, libtool, zlib, libusb, eigen3, (libdc1394)をインストールして下さい。qmake用に付属の.proファイルは標準の/opt/local以下にMacPortsが存在すると仮定しています。ユニバーサルバイナリでビルドする場合はfftw-3以外を+universalバリアントで、fftw-3は+universal +clang13 -gfortranでインストールして下さい。

複数のpythonがインストールされている場合、最初にpybind11が見つかったpythonを利用します。

Qtの6.8以降を、MacPortsでなく本家のオープンソース版をインストールして下さい。インストール時にQt 5 Compatibility Moduleを加えてください。
Qtの6.8以前では、キー入力を時々受け付けなくなるバグがあります。

ThamwayのUSB機器を使用する際は、ビルド前にソースコード内のmodules/nmr/thamwayフォルダにfx2fw.bix, slow_dat.bin, fullspec_dat.binをコピーして下さい。

必要あればEuresys eGrabberをインストールして下さい。apple siliconではdext版で十分です。

必要あればjupyterとその他、例えばMacPortsからpy3**-notebook, (py3**-qtconsole, py3**-pyside/pyqt5), py3**-matplotlib, py3**-scipyをインストールして下さい。MacPortsでなくhomebrewならbrew install jupyterなどです。

Qt Creatorからkame.proを開き、ビルドして下さい。問題があれば、プロジェクトのビルドの構成でPATHを/opt/local/binにも通して下さい。

QMakeの最中にxcodebuildがどうこう言われる場合は、以下を参照：
http://stackoverflow.com/questions/33728905/qt-creator-project-error-xcode-not-set-up-properly-you-may-need-to-confirm-t

OS Xアップグレード後に問題が生じれば、XCodeを入れ直す必要があります。XCodeインストール後は、XCode.appを起動してライセンス確認と追加のフレームワークをインストールして下さい。commandline toolsが入っていない場合は、xcode-select —installでインストールできます。以前のビルドディレクトリを一旦削除する必要もあるでしょう。それでもだめならQtの新しいバージョンを入れてください。

ベンダーのkextを使用するドライバがある場合、セキュリティポリシーの変更をしてください。

Qt Creatorの「プロジェクトの実行設定」では、「DYDL_LIBRARY_PATH…追加する」のチェックボックスを外して下さい。そうしないと起動しません。pdb等を使用する場合は”ターミナルで実行”します。

NI USB-GPIB-HS(+)の動作にNI社のドライバは＊不要＊になりました(v 8.0)。

NI USB-B / USB-HS / USB-HS+ / KUSB-488A / MC USB-488 は modules/charinterface/usermode-linux-gpib/ のユーザーモードドライバで動作します（linux-gpib 4.3.6 カーネルドライバのユーザーモード移植版）。カーネルモジュール不要。Apple Silicon Mac での USB-GPIB の唯一の手段。カーネルモジュールやプロプライエタリドライバを一切インストールすることなく、NI USB-B、USB-HS、USB-HS+、KUSB-488A、MC USB-488アダプタをmacOS・Linux・Windowsで使用できます。macOSではApple Silicon上でUSB-GPIBを使う唯一の手段です。

互換ヘッダ（osx_compat.h / win_compat.h）により、Linuxカーネルのkmalloc・スピンロック・USB URBをPOSIX/libusb（またはWin32）相当の実装に置き換えています。

(Intel CPUの場合) NI488.2がインストールされたMacでは以下の対応が必要です。そうでないと、kame起動時のcharinterfaceモジュール読み込みでクラッシュします。
https://www.ni.com/ja-jp/support/documentation/bugs/22/ni-488-2-21-5-known-issues.html#

(apple siliconの場合) Qt Creatorは8以降のものがベターです。

Linux PCへのインストール

(注：現在はサポートしていません)動作は32/64bitのx86版Fedora 14及び15で確認しています。

ただし、National Instruments社のDAQmxドライバを利用する場合は32bit版のFedora 14での動作を確認しています。

準備

以下では、kameの動作だけでなくコンパイルも可能な環境の構築を説明します。

Fedoraのインストール

DVDからのインストールをお勧めします。「KDEソフトウェア開発環境」等を選択して下さい。「Fedora Eclipse」もあると便利です。

/boot,/,/homeパーティションを分離してかつ空き領域を残しておくと複数のLinuxを起動する際に便利です。例えば、別バージョンのFedoraをインストールする際に、空き領域を/にして、/bootと/homeを元のものと共有すれば良いからです。ただし、その前に/boot/grub/grub.confの内容を写しておきましょう。更に言うなら、/,/homeはLVMにしておいた方がいいでしょう。

Windowsを残しておく場合は、あらかじめWindows側で領域を縮小してからFedoraをインストールすればいいですが、その時に、Windowsパーティション（通常最初の２つ）を残して残り領域にインストールしないといけないです。

以下の作業の前に「ソフトウェアの更新」を行なって下さい。

OpenGLがきちんと動く環境を用意して下さい。(Fedora 15まで)nVidia製GPUの場合は、nVidiaのホームページからLinuxドライバをダウンロードしてインストールして下さい。その場合、kernelの更新ごとにインストールしなおす必要があります。

ライブラリのインストール

入っていなければ、「ソフトウェアの追加／削除」から以下のパッケージを選択して下さい。もしくは、コマンドライン(su)でyum installして下さい。

gsl-devel, fftw-devel, atlas-sse2-devel, libgfortran, ruby, ruby-devel, kdelibs-devel, kernel-devel, libtool-ltdl-devel, rpmdevtools

以下のパッケージもあると便利です。

ccache, redhat-rpm-config

コンパイルする人は、一般ユーザでrpmbuildコマンドを利用できるようにするために、コマンドライン（一般ユーザー）でrpmdev-setuptreeを実行しましょう。そうすると~/rpmbuildフォルダにRPMパッケージが作られます。

Linux GPIBドライバのインストール

Linuxでは、National Instruments社のNI-488.2ドライバも利用可能ですが、私は確認してません。

kameと同じサイトに、Fedora用のRPMパッケージ及びSRPMパッケージを置いてあります。

詳細は、本家のドキュメントを参照
http://linux-gpib.sourceforge.net/doc_html/index.html

インストール

バイナリをインストールする場合（簡単）

linux-gpibと、kmod-linux-gpibのRPMをインストールします。コマンドラインでは、suでrpm -ivh等を使います。アップグレード時はrpm -Uvhです。

kmod-linux-gpibでは、kernelのバージョン(uname -rで確認できます)と一致していなければ意味有りません。なので、kernelは安易に更新しないようにしましょう。

SRPMからコンパイルしてインストールする場合

こんな感じでRPMパッケージを作りましょう
rpmbuild --define="kversion `uname -r`" --rebuild linux-gpib-kmod-{ver}.src.rpm
rpmbuild --rebuild linux-gpib-{ver}.src.rpm

あとは上記参照。

設定

/etc/gpib.confの編集

suで編集しましょう。vi /etc/gpib.confとかで。
board_type = "ni_pci"
の行を、ni_pciの部分を使っているデバイスに合わせましょう。

例えば、National Instruments社のUSB-GPIB-HSならば、
board_type = "ni_usb_b"
になります。

PCIまたはPCI ExpressのGPIBボードを使う場合

例えばこんな感じの項目を/etc/modprobe.d/modprobe.conf.localに追加する必要があります。
alias char-major-160 gpib_common
alias gpib0 tnt4882
install tnt4882 PATH=/sbin:/usr/sbin:/usr/local/sbin:$PATH;modprobe --ignore-install tnt4882;sleep 3;gpib_config --minor 0

USBのGPIBデバイスを使う場合

余計な設定は不要ですが、調子が悪ければ、suで、
/usr/sbin/gpib_config --minor 0
を打つと良いかも。

（必要な場合のみ）National Instruments社のDAQmxドライバのインストール

DAQmx8.0.2のISOをダウンロードして、中身を適当な展開しましょう。

とりあえずインストールします。suで、
cd (NIDAQ802 directory)
chmod +x INSTALL
LANG=C ./INSTALL
/usr/local/bin/updateNIdrivers

最後のコマンドは、kernelを更新する度に実行する必要があります。

RTSIの設定 (複数のPCI/PCIe DAQデバイスを同期して使用する場合)

以下を参照。
http://zone.ni.com/devzone/cda/tut/p/id/4620

nidaqmxconfig --export daq.config
で設定内容を書き出します。そしてdaq.configに以下の内容を足します。
[DAQmxRTSICable RTSICable0]
RTSI.ConnDevs=Dev1,Dev2

設定内容を読み込みます。
nidaqmxconfig --import daq.config

上手く行っていれば、”Operation Successful”と出ます。

SE Linuxのポリシー設定

設定ユーティリティで許容（permissive）にしてしまうか、

allow_execstackをチェックしましょう。

GUIでは、「管理」→「SELinux Management」で行います。system-config-selinuxがインストールされている必要があります。

kernelコマンドラインの設定

/boot/grub/grub.confをsuで編集します。

kernelのラインの最後に、以下を足します。
 iommu=off vmalloc=256M
これは、IO仮想化のIOMMUをOFFにして、kernel用のメモリを増やします。

メモリを4GB超積んでいる場合は、さらに以下を足します。
 mem=4096M

再起動しましょう。

DAQmxドライバは本当に困ったもんですね。

kameのインストール

バイナリからインストールする場合（簡単）

suでrpmコマンドを使ってkameとkame-modules-standardをインストールします。kame-modules-nmrをインストールすれば、NMRモジュール、kame-modules-nidaqmxをインストールすれば、National Instruments社のDAQデバイスを使用するモジュールも使えるようになります。

kame-debuginfoをインストールすると、不運にもクラッシュした時にデバッガを用いて調査ができるようになります。

ソースコード(tarボール)からインストールする場合

RPMを作る場合

tools/mkrpm.shを参考にしてください。

DAQmxモジュールが不要であれば、rpmbuildコマンドに
--define="build_nidaqmx 0"

を追加して下さい。

パッケージを作らない場合

build用のディレクトリを作ります。

そのディレクトリにcdして、
cmake ../(sourceディレクトリ)
make
make install
でインストールできますが、パッケージと混ざるとややこしいので推奨できません。

シリアルポートとlinux-gpibの為に、udevの設定が必要です。

Windowsへのインストール

バイナリからインストールする場合（簡単）

x86 64bit版の場合

Qtの6.10以上(llvm-mingw64構成を含む物、できれば複数のバージョンを含まないこと)をQt5 compatibility supportを含めてインストールして下さい。kame.batから起動する場合はMinGW64bit構成もインストールしてください。
https://www.qt.io/jp/

複数バージョンのQtがあって起動に失敗する場合は、qtdir.txtの中で6.10以上のもののみを残して削除して下さい。

NIのGPIBを使用する場合は、
National Instrumentsの488.2ドライバを（ANSI Cサポートやダイレクトエントリを外さないで）インストール

DAQmxを使用する場合は、
National InstrumentsのDAQmxドライバを（ANSI Cサポートを外さないで）インストールして下さい。

kame-win32-llvm64***.zipを展開し、kame.batから起動して下さい。ただし、下記のkame-msyspython.batからの起動でないと、numpyやそれを利用したドライバが利用できません。

ThamwayのUSB機器を使用する際は、実行フォルダにfx2fw.bix, slow_dat.bin, fullspec_dat.binをコピーして下さい。cypressのcyusb3.sysをインストールする必要があります。代わりにzadig+libusbKでも動くかもしれません。

同梱ではなくMSYS2でインストールしたpythonを利用する場合は、同じバージョンのpythonを確認の上、kame-msyspython.batから起動して下さい。

Jupyter等を利用する場合、MSYS2等で必要に応じてインストールして下さい。

```
pacman -S mingw-w64-x86_64-python-qtconsole
pacman -S mingw-w64-x86_64-python-jupyter_notebook
pacman -S mingw-w64-x86_64-python-scipy
```

32bit版の場合（obsolete）

Qtの5.7(MinGW32bit構成を含む物、複数のバージョンを含まないこと)をまずインストールして下さい。
https://www.qt.io/jp/

複数バージョンのQtがあって起動に失敗する場合は、qtdir.txtの中で5.7移行の行のみを残して削除して下さい。

GPIBを使用する場合は、
National Instrumentsの488.2ドライバを（ANSI Cサポートやダイレクトエントリを外さないで）インストール

DAQmxを使用する場合は、
National InstrumentsのDAQmxドライバを（ANSI Cサポートを外さないで）インストールして下さい。

kame-4**-win32.zipを展開し、kame.batから起動して下さい。

OpenGL2.1に対応していない古いマシンではグラフが正常に表示できません。古いintel内蔵グラフィックスマシンでkameが起動時かグラフを開く時に落ちる場合は、グラフィックドライバを更新した上で、設定画面で「トリブルバッファ」をオフにして「深度バッファ」を16bitにして下さい。

ThamwayのUSB機器を使用する際は、実行フォルダにfx2fw.bix, slow_dat.bin, fullspec_dat.binをコピーして下さい。

ソースコード(zip)からインストールする場合

上記のバイナリが動作する状態にまずして下さい。

x86 64bit版の場合

次に、いくつかのライブラリを準備する必要があります。

msys2をデフォルトのc:/msys64にインストールしてください。

まず、アップデートします。

```
pacman -Syuu
```

その後、以下をインストールしてください。

```
pacman -S make
pacman -S mingw-w64-x86_64-zlib
pacman -S mingw-w64-x86_64-fftw
pacman -S mingw-w64-x86_64-gsl
pacman -S mingw-w64-x86_64-eigen3
pacman -S mingw-w64-x86_64-pybind11
pacman -S mingw-w64-x86_64-libusb
pacman -S mingw-w64-x86_64-python-numpy
pacman -S mingw-w64-x86_64-ruby
```

Jupyter-qtconsoleを利用している等でQt5が入ってしまっている場合は、ビルド時にアンイストールしておきましょう（ビルド後に-Sしてもよい）。

```
pacman -Rdd mingw-w64-x86_64-qt5-base
```

Qt Creatorからkame.proを開き、llvm-mingw64構成でビルドして下さい。gitからcloneする場合は、autoCRLF=falseにしてください(automatic line endingをオフ)。さもなくばpyファイル等で改行コードに起因したエラーが出ます。

Qt Creatorから実行するかデバッグを開始して下さい。Projects設定のRunから”Add build library search path to PATH”のチェックを外してください。pdb等を使用する場合は”ターミナルで実行”します。また、環境変数でPYTHONHOME=c:¥msys64¥mingw64などと設定してください。PATHをC:\msys64\usr\bin;C:\msys64\mingw64\bin;C:\msys64\mingw64\lib:%PATH%のように通してください。PATHを通さないのであれば、が、C:\msys64\mingw64\binの中にある、zlib.dll, llibgsl-28.dll, libgslcblas-0.dll libfttw3-3.dll, libpython-3.12.dll, libgmp-10.dll, libusb-1.0.dllとx86-msvc*-ruby*.dllを実行ファイルのあるフォルダ（通常はbuild/build-kame-***-Debug/Release）にコピーしてください。

kame-msyspython.batから起動する場合は、環境変数設定は不要です。

関西弁にするには、kame_ja.qmを実行ファイルのあるフォルダにコピーしなければいけません。また、実行ファイルのあるフォルダにResourcesという名前のフォルダを作り、kame/script/rubylineshell.rbとpythonlineshell.pyをコピーしてこなければ、ラインシェルが使えません。Jupyter-notebookのためにkame/script/notebookの中の２つのファイルもResourcesフォルダに入れてください。

32bit版の場合（obsolete）

次に、いくつかのライブラリを準備する必要があります。

fftw-3.3.4-dll32.zip（同様にkameのソースディレクトリの一つ上に、ディレクトリ名fftw3で展開)
http://www.fftw.org/download.html

Ruby及びmsys（http://www.mingw.org/）を適当なところからダウンロード、インストールし、rubyにパスを通して下さい。また、ruby-2.2.*のソースを一つ上のディレクトリにrubyの名前で解凍して下さい。その後、msys.batを起動し、rubyのソースディレクトリにおいてmingwにパスを通し(PATH=$PATH:/c/QT/Qt-(適宜)/Tools/mingw4(適宜)_32/binなど)、
CFLAGS=“-g0 ” ./configure —enable-shared —disable-rubygems
make
でRubyのDLLを作ります。途中でエラーが出てもDLLができていればOKです。

zlibに関しては、以下から”Compiled DLL”をダウンロードし、一つ上のディレクトリにzlibの名前で解凍して下さい。
http://www.zlib.net/

GSLに関しては新しいバージョンが必要なため、自身でコンパイルする必要があります。
http://www.gnu.org/software/gsl/
からgsl-2.4.tar.gzをダウンロードし、一つ上のディレクトリにgslの名前で解凍して下さい。上記Rubyの場合と同じように./configure;makeして下さい。その後、./libs/libgsl-23.dll及びcblas/.libs/libgsl-cblas-23.dllを実行フォルダにコピーして下さい。

Qt Creatorからkame.proを開き、mingw32構成でビルドして下さい。gitからcloneする場合は、autoCRLF=falseにしておいた方が無難です。

実行するには、zlib1.dll(C:\Program Files*\GnuWin32\binの中), llibgsl.dll, libgslcblas.dll（gslの中）, ibfttw3-3.dll（FFTW3の中）, msvc*-ruby*.dll(rubyの中)を実行ファイルのあるフォルダ（通常はbuild-kame-***-Debug）にコピーしてから、Qt Creatorから実行するかデバッグを開始して下さい。

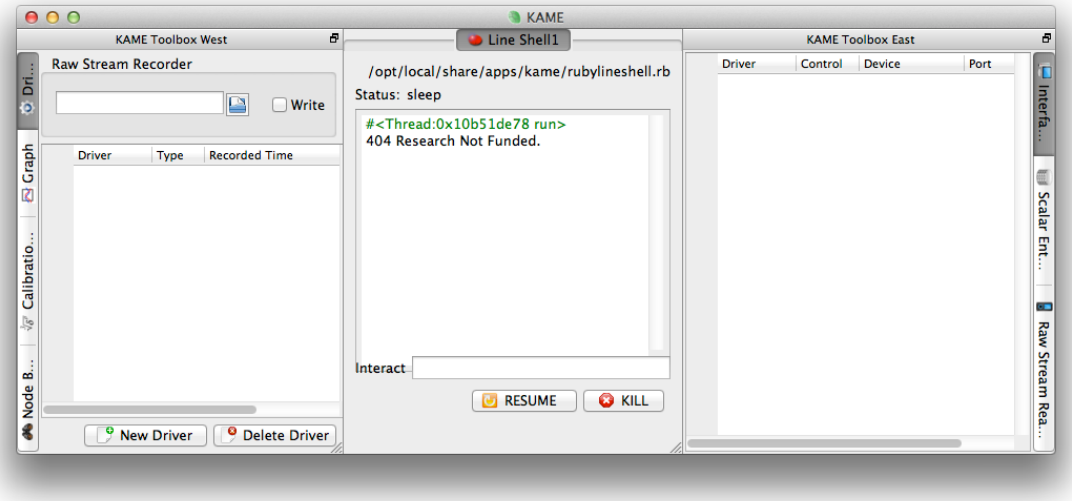
関西弁にするには、kame_ja.qmを実行ファイルのあるフォルダにコピーしなければいけません。また、実行ファイルのあるフォルダにResourcesという名前のフォルダを作り、kame/script/rubylineshell.rbをコピーしてこなければ、ラインシェルが使えません。

実行

通常は関西弁で、LANG=C kameとやると英語で起動します。

エラー等のログが/tmp/kame.log（windowsでは実行フォルダ）に残りますので、問題が起これば確認してみて下さい。

基本操作



（初回）「ドライバ」タブ中で測定器に対応するドライバを「新規」で追加。

（次回から）「ファイル」→「開く」で設定を保存した.kamファイルを読み込む。

「インターフェース」タブのなかでドライバを「開始」。GPIB/シリアルポート等であればアドレス(ポート)を指定しておく（以下の章を参照）。

「ドライバ」タブまたは「インターフェース」タブでドライバ名等をクリックするとドライバ固有の設定画面が開く。（一部のドライバには設定画面はありません。）

「スカラエントリ」。このタブは、時系列でデータファイルにいくつかの測定値を保存するのに使用する。例えば、温度と電圧とか。「デルタ」を正にすると、変化分がその値を超えたときに一行出力する。負にすると、新しい値が読み込まれる度に一行出力する。該当する値をクリックすると、横軸が時間のチャートが現れる。

「グラフ」。X-Y-(Z)のグラフを必要なだけ作れる。値は「スカラエントリ」から選ぶ。

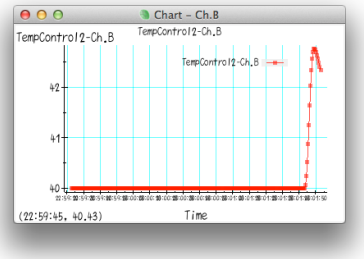
「ファイル」 →「保存」とすると、ほとんどの設定を.kamファイルに保存します。随時行いましょう。

テキストボックスの操作

打ち間違いや装置への悪影響を防ぐため、ほとんどの設定は、エンターキーを打つまで反映されません。編集中は文字が青色になります。間違った形式で打つと赤色になります。エンターキーを打たずに別の場所を選択すると、入力結果は破棄されて元の値を表示します。

殆どの場合、テキストボックス中の値は、Rubyスクリプトから操作可能です。

グラフ・チャートの操作



2次元グラフでも３次元空間に半透明で描画されています。

左ダブルクリックすると、詳細なダイアログが開きます。

右ダブルクリックすると、操作ヒントが出ます。

プロット中で左クリックしながら範囲を選択すると、オートスケールを止めて拡大します。

プロット中で右クリックすると、オートスケールに戻ります。

プロット中でホイール操作でズームします。

軸上で右クリックしながら範囲を選択するとその軸のオートスケールを止めて拡大します。

軸上で右クリックすると、その軸をオートスケールします。

軸上でホイール操作すると、３次元的に傾けます。

中ボタンを押しながら動かすと、３次元的にグリグリします。

中クリックすると、視点が元に戻ります。

保存可能なデータがある場合、上か下にファイルを示すバーがあります。フォルダアイコンをクリックすると、保存すべきファイルを選べます。ただし、その時点では保存されません。「書き出し」を押した瞬間のデータが追記で保存されます。一度書きこむとアイコンが変化します。

“Math”ボタンが備わっているグラフでは、解析ツールを利用出来ます。結果はスカラエントリに表示されますが、ドライバのレコード時刻とは同時ではなく少し遅れて結果が出ます。

コマンドラインオプション

普通は不要。

--logging

通信内容等もログに残します。メニューから「ログを取る」を選んでも一緒です。

—nooverpaint

グラフのフォント描画に２D描画でなく、renderText()を使います。

--moduledir <path>

モジュールを読み込むパスを追加します。ソースコードからインストールした場合に便利です。

メニュー項目

ファイル

開く

.kamファイルを読み込みます。中身はRubyスクリプトになっています。従って、スクリプト実行画面が中央部に開き、エラーが出れば赤く表示されます。

保存

現在開いている全ドライバや他の一部の設定を全て.kamファイルに保存します。内部的には、Rubyからコントロール可能なノードの内、保存属性が設定されているもののスナップショットが保存されます。

閉じる

全ドライバ、キャリブレーションテーブルが削除されます。誤って押さないようにしましょう。

ログを取る

測定器との通信ログ、ノードの追加／削除情報も/tmp/kame.log（Windowsでは実行フォルダのkame.log）に保存されます。ディスクを圧迫するので、不必要な時は止めましょう。この状態でドラ

イバを追加すると、インターフェースに”DUMMY”の選択肢があります。ダミーポートでは実際の通信をせずにファイルに送信内容を掃き出すことが出来ます。

終了

測定中はキャンセルされます。保存を忘れないようにしましょう。

測定

停止

全測定を停止させます。うっかり押さないようにしましょう。

スクリプト

開始

自動測定のためのPython/Rubyスクリプトを読み込みます。標準では.py/.seqファイルです。書き方は別章で説明します。標準（エラー）出力結果は、同フォルダの（ファイル名）.logに保存されます。

新規ラインシェル

Python/Rubyスクリプトを一行ごとに打ち込める画面を開きます。起動時に既に 1 つ開いています。

Launch Jupyter notebook

Jupyter notebookがインストールされていれば、kameの埋め込みIPythonのクライアントとして呼び出してデフォルトブラウザに表示します。クライアント側の停止ボタンは機能しませんので、セル内ではtime.sleep()を使用せずにsleep()を使用し、kame側のKILLボタンで停止してください。

表示

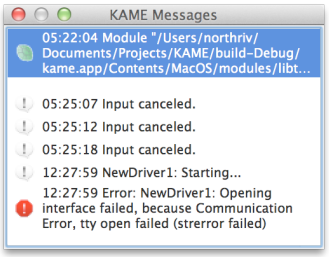
タブをクリックしても同じです。

ヘルプ

用意してません。このマニュアルを見て下さい。

メッセージウィンドウ

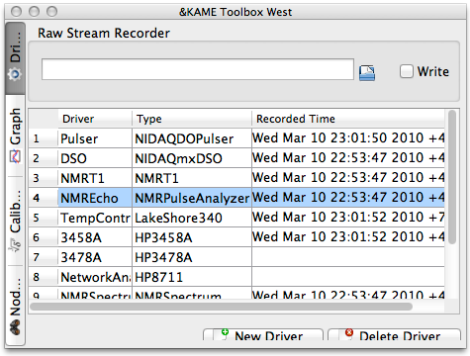
起動時から左端にいます。主要なメッセージ、警告やエラーが表示されます。もう少し細かい情報は



標準出力と/tmp/kame.log（Windowsでは実行フォルダのkame.log）に記録されます。

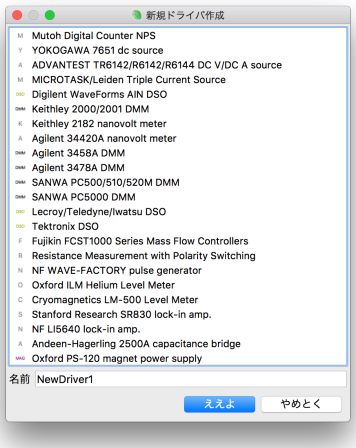
タブ

ドライバ



ドライバの追加・削除が出来ます。また、ドライバー覧上でクリックするとドライバ固有の設定ウィンドウが出ます。レコード時刻は、最後に測定器から記録に値するデータが来た時間を表示します。

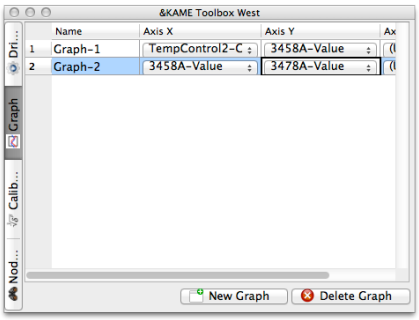
新規ドライバを押すと次のようなウィンドウが出て、必要なドライバに名前を付けて作成できます。名前は後から変更は難しいので、考えてつけて下さい。



Rawストリームレコーダ

生データを極力保存します。「書き込み」チェックボックスが入っている間、例えば、オシロスコープの波形が全て記録されます。GZIP形式で圧縮されますが、当然それなりのディスク容量を消費します。

グラフ

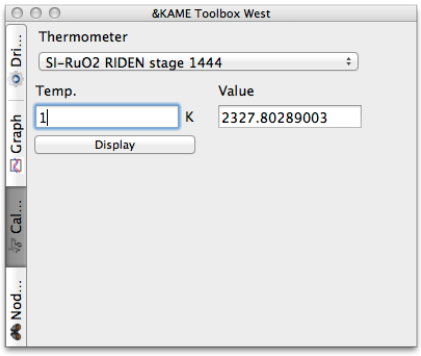


２次元／３次元グラフを生成します。X/Y/Z軸を「スカラエントリ」の中の項目から選んで下さい。

設定点数を超えた古いデータはグラフから消されます。

“Live”はメモリにレコードされた全データ、”Stored”は「テキストライタ」でディスクに記録されたデータを表示します。

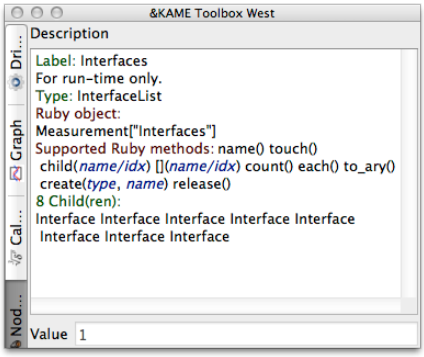
キャリブレーションテーブル



温度計の校正データを定義するのに使います。定義ファイルは今のところ、手動でRubyスクリプトを書く必要があります。サンプルがkame/Measurementsフォルダにあります。スプライン曲線による補完、またはチェビシェフ多項式による定義が可能です。

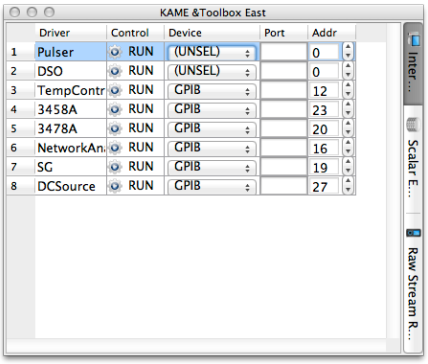
「表示」を押すと、校正曲線を表示します。

Node Browser



kameの中のデータは全てツリー構造に管理されています。Rubyスクリプトで制御する際、”Mesurement["Drivers"]["driver1"]["foo"].value=1.0という感じで行います。画面上の入力部がPython/Ruby上でどう扱われるか表示します。スクリプトのサンプルは、kame/Sequencesフォルダにあります。

インターフェース



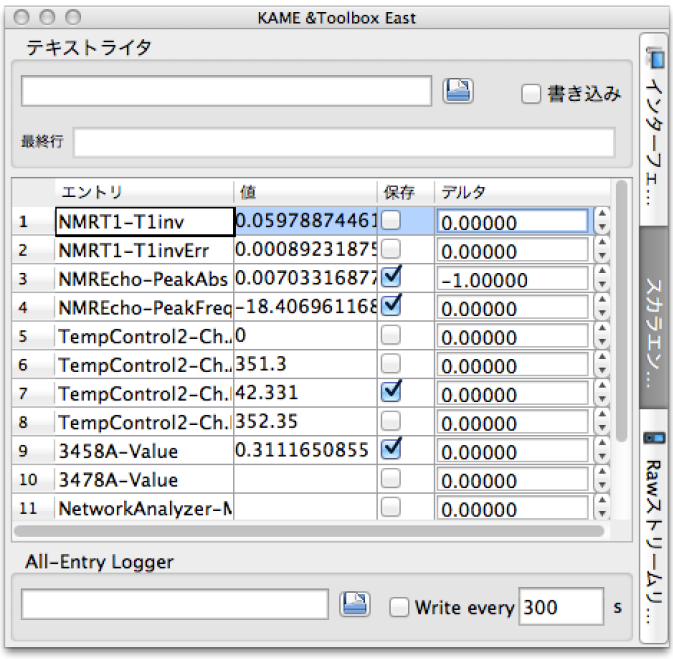
ドライバが測定器との通信経路を必要としている場合、ここに設定項目及び、起動／停止ボタンが出ます。通常は一ドライバに対し一つです。「デバイス」、「ポート」、「アドレス」をそれぞれ正しく設定して下さい。

シリアルポートの場合、 RS-232Cではアドレスは不要ですが、ポートは/dev/ttyUSB0(Linux),/dev/tty.usbserial(Mac)等のデバイスファイルを指定します。RS-485ではスレーブのアドレスを設定して下さい。ポートで右クリックすると候補を示します。

NI488 GPIBの場合、ボードが一つであれば、「ポート」は空白で構いません。Prologix USB-GPIBコンバーターのバア愛はシリアルポートと同様にポートを設定してください。

TCP/IPの場合、「ポート」に(IPアドレス):(ポート番号)を設定して下さい。

スカラエントリ



ドライバからレコードされた結果の数値が一覧に出ます。

該当する値をクリックすると、横軸が時間のチャートが現れる。チャートの操作法は、横軸をオートスケール出来ない以外はグラフと同じです。

テキストライタ

テキストファイルに一行ごとにスペース区切りで、「保存」を指定された値を出力します。改行はLFです。「書き込み」をチェックしないと出力しません。

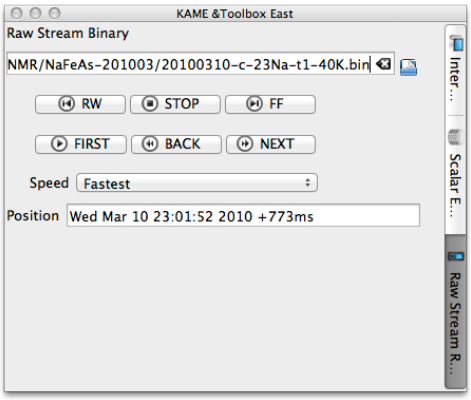
どのタイミングで一行が出力されるかは、「デルタ」で指定します。「デルタ」を正にすると、変化分がその値を超えたときに一行出力します。負にすると、その項目に新しい値が読み込まれる度に一行出力します。

項目の順番を変えるには、番号の上でCTRLキーを押しながらドラッグします。

ロガー

テキストファイルに一行ごとにスペース区切りで、指定した間隔で全ての値を出力します。

Rawストリームリーダ



上記のRawストリームレコーダで出力したファイルを読み込んで解析できます。巻戻し機能はバグっているので、何回も解析する場合は、「最初」に戻ってから「送り」で早送りして下さい。早送り速度は「速度」で指定します。「位置」はレコードが記録された時刻を表示しています。

ドライバ固有の設定

DCソース

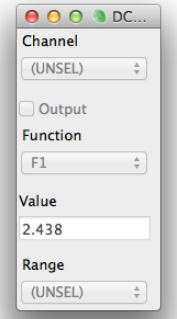
YOKOGAWA 7651 DC source (GPIB)

ADVANTEST TR6142/TR6144/R6142/R6144 DC source (GPIB)

MICROTASK/Leiden Triple Current Source (GPIB)

Optotune ICC4C-2000 lens current controller (Serial Port)

7651を使う場合は、他の装置よりも先に開始したほうが無難です。



デジタルマルチメータ(DMM)

Keithley 2000/2001 DMM (GPIB)

Keithley 2851A nanovolt meter (GPIB)

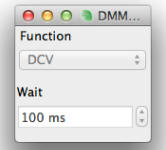
Keithley 2182 nanovolt meter (GPIB)

HP/Agilent 34420A nanovolt meter (GPIB)

HP/Agilent 3458A/3478A DMM (GPIB)

Keithley 6482 picoammeter (GPIB)

SANWA PC500/510/520M/PC5000 DMM (IR SerialPort)



「ウェイト」は、読み込み周期を設定します。

スカラエントリに値が送られます。

デジタルストレージオシロスコープ(DSO)

Tektronix DSO (GPIB)

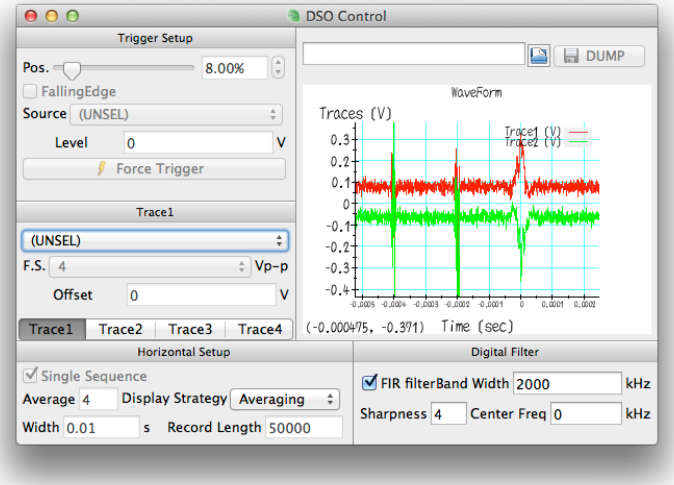
Lecroy/Iwatsu DSO (GPIB)

DSO on NI-DAQ M,S series (NI-DAQmx)

Thamway A/D conversion DV14U25 (USB)

Digilent WaveForms AIN DSO (USB)

Thamway PROT3 streaming DSO (TCP/IP, USB)



オシロスコープの/DIVでなく、縦軸はフルスケール「F.S.」で、水平軸は全長の「幅」で設定して下さい。

PCに連続取り込みするドライバでは、時間同期の取れているトリガ源（NMRパルサーなど）がソフトウェア的にトリガ位置を計算します。

「シングルシーケンス」にすると、「アベレージ」回数分積算してはクリアするの繰り返しになります。

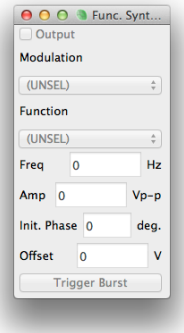
「表示の方法」を”Averaging”にすると適宜表示するので便利ですが、GPIBやCPUに大きな負担をかけます。「シングルシーケンス」モードで”Sequence”にすると指定積算回数に達した時にだけ表示します。

「デジタルフィルタ」、FIR(Finite Impulse Response)フィルタを掛けることができます。「中心周波数」(fc)から「バンド幅」(bw)で指定した帯域をそれぞれのチャンネルに掛けるので、fc-bw/2からfc+bw/2の周波数を通すようになります。FIRは畳込みで高速に処理されます。

ファンクションジェネレータ

NF WAVE-FACTORY (GPIB)

LXI 3390 arbitrary function generator (LAN)



液面計

Oxford ILM Helium levelmeter (GPIB, SerialPort)

Cryomagnetics LM-500 levelmeter (GPIB)

設定画面はありません。

スカラエントリに値が送られます。

ロックインアンプ／容量計

Stanford Resrearch SR830 lock-in amplifier (GPIB)

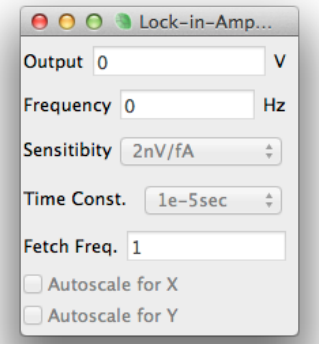
NF LI5640 lock-in amplifier (GPIB)

Signal Recovery 7265 lock-in amplifier (GPIB)

LakeShore M81-SSM synchronous source measure (USB)

Agilent/HP 4284A LCR meter (GPIB)

Andeen-Hagerling 2500A capacitance bridge (GPIB)



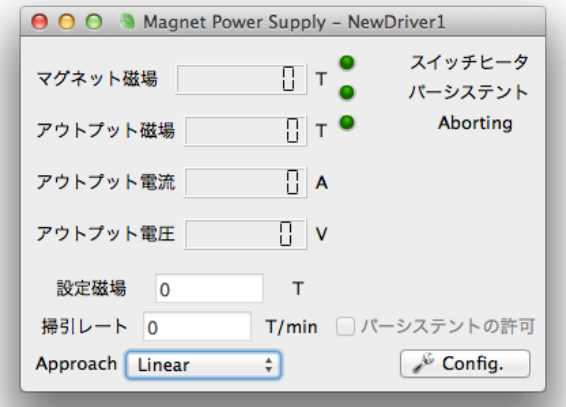
「取り込み頻度」はタイムコンスタントの逆数を基準にしています。

スカラエントリにX／Y値が送られます。

超伝導マグネット電源

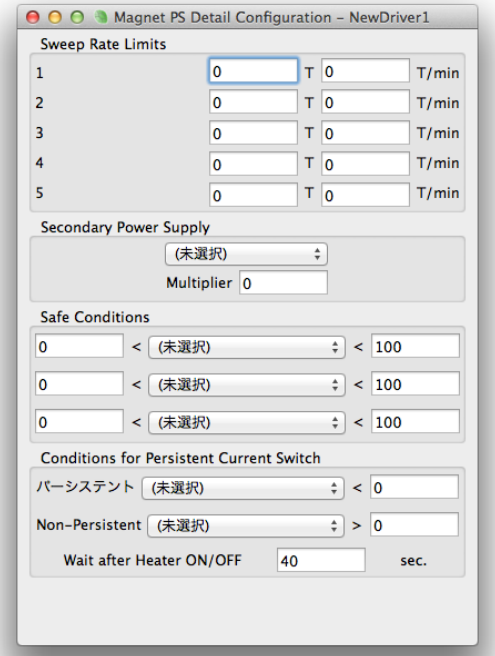
Oxford PS/IPS-120 magnet power supply (GPIB, SerialPort)

Cryogenic SMS10/30/120/150C magnet power supply (USB Serial Port)



パーシステントスイッチヒーターの制御は自動で行います。「パーシステントの許可」をチェックすると、磁場が設定値に到達した時にヒーターを切ります。”Approach”を”Oscillating”にすると、20%オーバーシュートしながら磁場をスイープします。

「設定」ボタンを押すと次の画面が開きます。



掃引レート及び、最高磁場に制限をかける場合は、”Sweep Rate Limits”に”最大磁場”-”レート”のペアを小さい順に必要な数だけ入れて下さい。

“Secondary Power Supply”はシムコイルの駆動に使います。

“Safe Conditions”は、スカラエントリの値を監視して、範囲から外れると磁場を0にします。

同様に、パーシステントスイッチ用の条件を付加することが出来ます。

“Wait after Heater ON/OFF”に、スイッチヒーターの待ち時間を入れて下さい。

標準信号発生器 (SG)

KENWOOD SG7130/SG7200 signal generator (GPIB)

DSTech. DPL-3.2XGF (SerialPort)

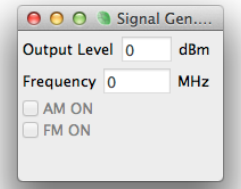
HP/Agilent 8643/8644/8648/8664/8665 signal generator (GPIB)

Rhode-Schwartz SML-01/02/03/SMV-03 signal generator (GPIB, SerialPort)

Thamway PROT NMR Control (TCP/IP, USB)

Keysight/Agilent E44xB signal generator (GPIB, LAN)

LibreVNA signal generator (USB)



PROTの場合は、ゲイン等のNMR設定が制御可能です。

ネットワークアナライザー

HP/Agilent 8711/8712/8713/8714 network analyzer (GPIB)

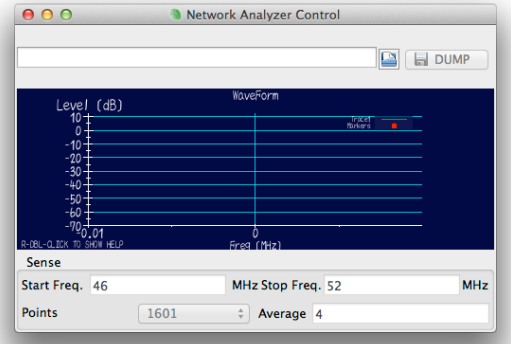
Agilent E5061/E5062 network analyzer (GPIB)

Copper Mountain TR1300/1,5048,4530 Network Analyzer (TCP/IP)

DG8SAQ VNWA3E w/ custom dll network analyzer (TCP/IP)

Thamway T300-1049A Impedance Analyzer (SerialPort)

LibreVNA network analyser (USB)



スカラエントリにマーカーの値を出力します。

温度計／温調器

Cryocon M32/M62 temperature controller (GPIB)

LakeShore 218 temperature monitor (GPIB)

LakeShore 340 temperature controller (GPIB)

LakeShore 350 temperature controller (GPIB)

LakeShore 370 temperature controller (GPIB)

LakeShore 372 temperature controller (GPIB)

Picowatt AVS-47 bridge (GPIB)

Oxford ITC-503 temperature controller (GPIB, SerialPort)

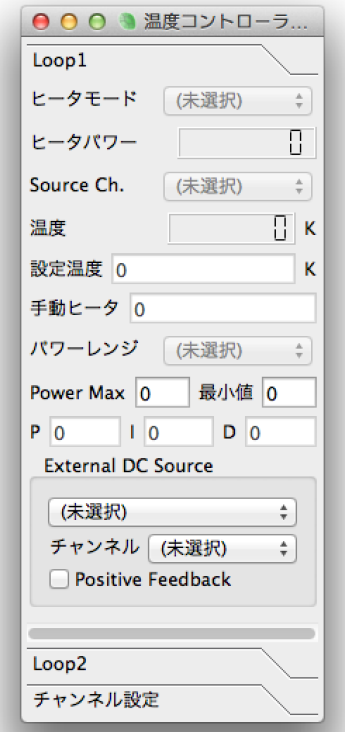
Neocera LTC-21 temperature controller (GPIB)

Keithley 2700 DMM w/ 7700 scanner (GPIB)

OMRON E5*C controller (Serial Port Modbus RTU 57600bps)

Scientific Instruments 9302/9304/9308 temperature controller (GPIB)

LinearResearch LR-700 resistance bridge (GPIB)



「温度計」を選択した場合、kameが抵抗値等から温度に変換します。温度計の校正曲線は「キャリブレーションテーブル」タブで確認できます。

「外部機器」を選ぶと、そのDCソースまたは流量制御弁を用いてPID制御を行います。

スカラエントリに値（抵抗値等と温度）が送られます。

表示はされていませんが、"Stabilized"というノードに「設定温度」からの「現在の温度」のエラーを一定時間平均した値が保存されています。

カウンター

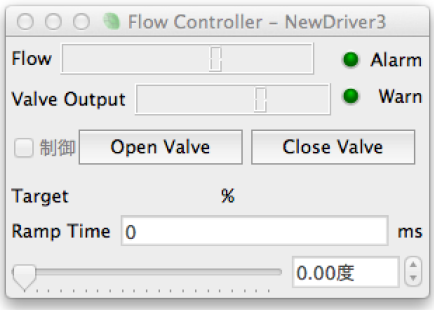
Mutoh NPS digital counter (SerialPort)

設定画面はありません。

スカラエントリに値が送られます。

流量制御

Fujikin FCST1000 Series Mass Flow Controllers (Serial Port 38400bps)



マスフローコントローラードライバです。

スカラエントリに値が送られます。

モーター制御

OrientalMotor FLEX CRK motor controller (Serial Port Modbus RTU 57600bps)

OrientalMotor FLEX AR/DG2 motor controller (Serial Port Modbus RTU 57600bps)

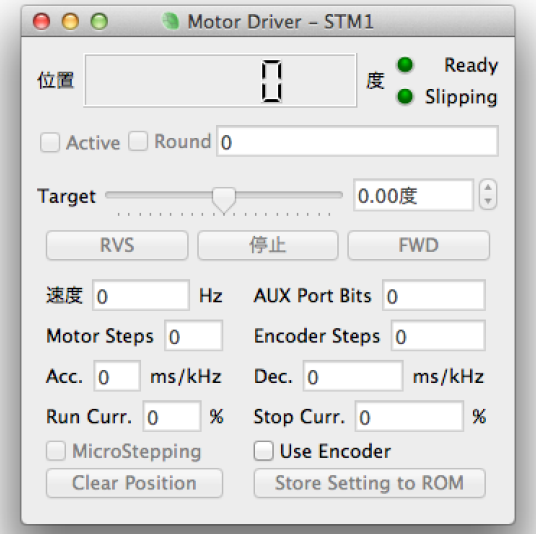
OrientalMotor CVD2B/CVD5B motor controller (Serial Port Modbus RTU 57600bps)

OrientalMotor EMP401 motor controller (Serial Port)

SigmaOptics PAMC-104 piezo-assisted positioner (Serial Port)

Micro CAM z/x/ϕ positioner

Two-axis sample rotator



ステッピングモータードライバです。

「Active」で励磁のON/OFFを設定します。

スカラエントリに値が送られます。

真空ゲージ

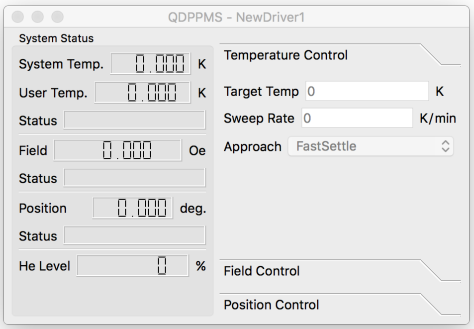
Pfeiffer TPG361/362 vacuum gauge (Serial Port)

ターボ分子ポンプ制御

Pfeiffer Turbo molecular pump controller TC110 (Serial Port)

PPMS制御

Quantum Design PPMS low-level interface (GPIB, Serial Port 9600bps)



QuantumDesign社のPhysical Property Measurement System 6000コントローラを制御します。

シリアル通信の場合には、MultiVuのGPIB制御と同時に通信出来るようです。本体で、9600bps, フロー制御なし、改行コードCRLF、ビット長,パリティ,ストップビットは”8,N,1”に設定して下さい。

スカラエントリに値が送られます。

カメラ／分光器

IEEE 1394 IIDC camera (Firewire)

Euresys eGrabber CoaXPress camera (PCIe)

Hamamatsu camera via Euresys Grablink CameraLink (PCIe)

JAI camera via Euresys Grablink CameraLink (PCIe)

OceanOptics/Insight USB spectrometer HR2000+/4000 (USB)

レーザーモジュール

Coherent Stingray laser diode driver (Serial Port)

Newport/ILX LDX-3200 laser diode driver (Serial Port)

Newport/ILX LDC-3700(C) laser diode controller (Serial Port)

NMRパルサー

NMR pulser on NI-DAQ M series (NI-DAQmx)

NMR handmade pulser on H8 (SerialPort)

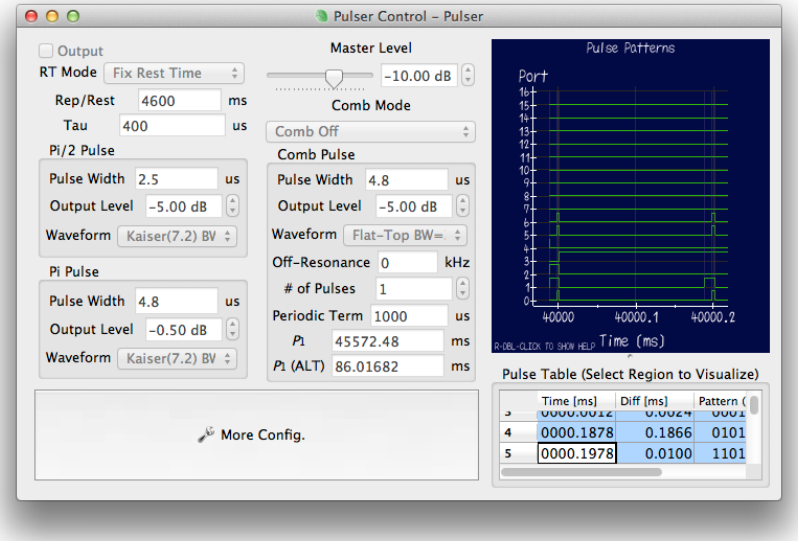
NMR handmade pulser on SH2 (SerialPort)

NMR pulser Thamway N210-1026 PG32U40 (USB)

NMR pulser Thamway PG027QAM (USB)

NMR pulser Thamway N210-1026S/T (GPIB, TCP/IP: experimental)

DAQmxについては付録も参照して下さい。



QAMに対応していない通常のパルサーの場合は、該当する部分（強度、波形等）は変更できません。

右側のグラフに、デジタル波形を表示します。表を選択すると選択範囲の波形を表示します。

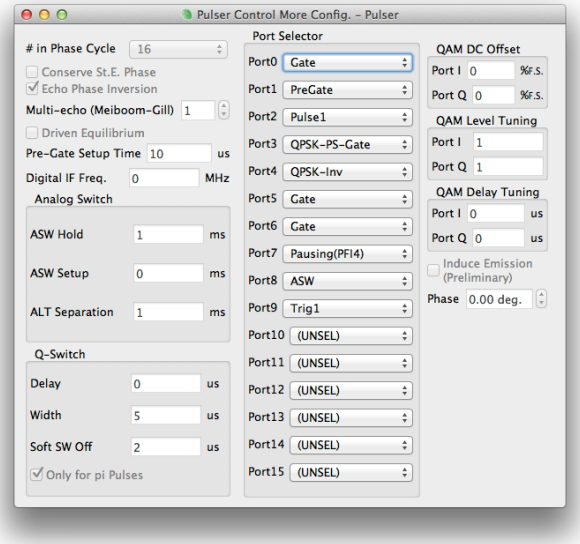
「RTモード」を”Fix Rep. Time”にすると、Saturation (Comb) Pulseまたは1st (Pi/2) Pulseの繰り返し時間をP_1によらず”Rep/Rest”時間にします。”Fix Rest Time”の時は、2τまたは最後のパルスの後から”Rep/Rest”時間待ちます。

パルスの時間間隔等（“Tau”, “P_1”など）の基準はパルスの中心です。ただし、「コムパルス」にかんしては、最後のサチュレーションパルスの中心が基準です。

「コムモード」を”P1 ALT”にすると、P_1の値を交互に変化させます。その際、T_1を測定するには、後述のFID/エコー測定ドライバが２つ必要になります。”ASW”の信号を使ってレシーバーのゲーティングをする必要があります。

同様に、”Comb ALT”にすると、コムパルスあり・なしを交互に打ちます。

「もっと設定」を押すと次のような画面が出ます。



「フェーズサイクル数」は、4までは通常のpi/2,piパルスに関するサイクルで、それ以上は、Combパルスの位相も変化させます。

「Conserve St.E. Phase」をチェックすると、Stimulated Echo測定用のフェーズサイクルになります。Combパルスを２個に設定して、Combのパルス間隔をτと同じに、pi/2パルスの幅を0に、piパルス幅をCombのパルス幅と同じに設定しましょう。

「マルチエコー」を２以上にすると、Carr-Purcell-Meiboom-Gill (CPMG)サイクルになります。CPMGでもフェーズサイクルします。FID/エコー測定のドライバでも適切な設定が必要です。

「ポートセレクタ」で、各ポートの機能を選んで下さい。
”Gate”はパルスのゲート。
”PreGate”はパワーアンプ用にセットアップ時間を設けたゲート。
”Trig1”は2τで立ち上がるトリガ用信号。
”Trig2”はpi/2パルスで立ち上がりpiパルスで立ち下がるトリガ用信号。
”ASW”は、レシーバー用のゲート。2τの前後の設定期間にHレベルになります。
”QSW”はパルス直後だけHレベルになる、Qスイッチ用の出力。
”Pulse1”はpi/2パルス。
”Pulse2”はpiパルス。
QPSK用のロジックは以下の通りです。

（位相）	“QPSK-A”	“QPSK-B”	“QPSK-NonInv”	“QPSK-Inv”	“QPSK-PS-Gate”
0°	L	L	H	L	L
90°	H	L	H	L	H
180°	H	H	L	H	L
270°	L	H	L	H	H

ちなみに、いわゆる西のNMRグループは、”QPSK-A”,”QPSK-B”を、東のNMRグループは、”QPSK-PS-Gate”,”QPSK-Inv”のロジックを使用しています。

電流反転抵抗測定

このドライバは、解析専用で、DCソースを反転させながら、DMMからの電圧データの差分を取ります。

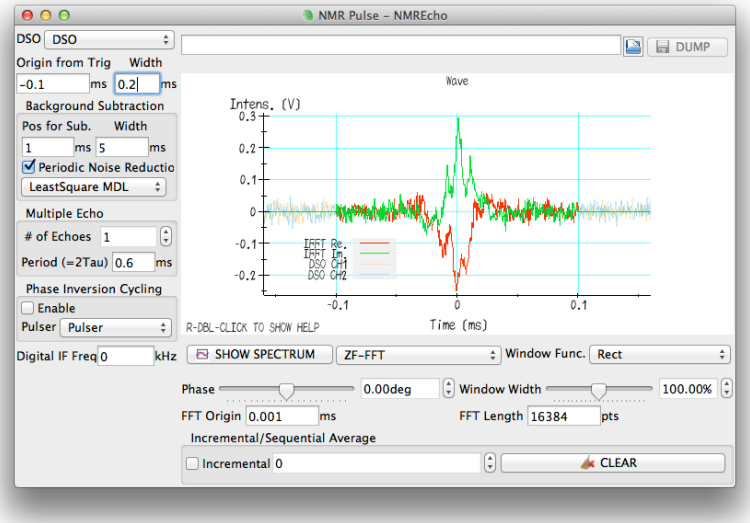
スカラエントリに値が送られます。

Pythonベース 4端子抗抗測定 Test4Res (シンプル) (pydrivers.py)

Pythonベース 4端子抗抗測定 Py4Res (多電流) (pydrivers.py)

NMR FID/エコー測定

このドライバは、解析専用で、DSOのデータをバックグランド除去し、切り出し、積算した後にスペクトラム解析します。



通常のエコーを観測する設定では、DSOのトリガーを 2τ(="Trig1")に設定して、DSOを「シングルシーケンス」モードにして下さい。

「トリガーからの位置」を始点にして「幅」でDSOのデータから解析対象を切り出します。ただし、「マルチエコー」の場合は、さらに「エコー数」分平均化して表示します。

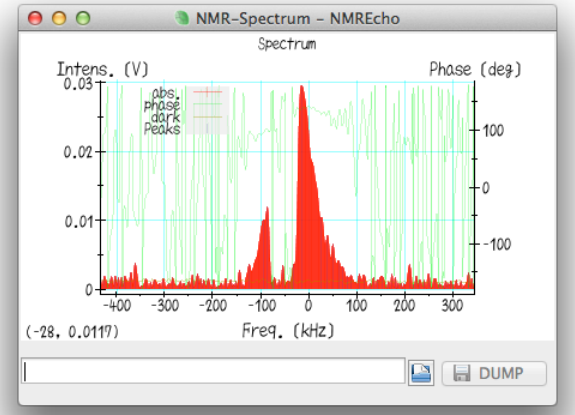
「バックグラウンド(BG)減算」の「幅」が0でない場合は、トリガーから「BG測定位置」を始点にして「幅」の平均レベルを引き算してデータを表示します。S/Nを損なわないために、減算用の「幅」は元データの「幅」よりも十分（数倍以上）長く取って下さい。
また、「周期性ノイズ除去」がチェックされている場合は、さらに、BG測定から卓越する周期性外部ノイズを幾つか推定し（最小二乗フィット）、その延長成分を減算します。ホワイトノイズかどうか判定する、情報量基準についてアルゴリズムを選択できます。この処理は、多数回のFFTを伴うの

で、CPUパワーを大きく必要とします。でもその価値はあります。周波数掃引の時は必ず使いましょう。

「フェーズ反転サイクル」をチェックすると、DSOで見てエコーの位相が反転する場合としない場合を交互に繰り返して差を取ります。時間依存するバックグラウンド成分を減算することができます。緩和率測定・スペクトラム測定には、偶数回数の平均の結果が引き渡されます。

「インクリメンタル」をチェックすると、一度画面がクリアーされ、「消去」を押すまで積算を続けます。積算回数が表示されます。信号を探したり、FTを取る時に使うと便利です。

「スペクトラム表示」をクリックすると次の画面が出ます。



通常は、”ZF-FFT”（Zero-Filling Fast Fourier Transform）の結果が表示されます。FFTの時間軸原点は「FFT原点」でトリガー位置を基準にして入れて下さい。FFTは「窓関数」を掛けることも出来ます。

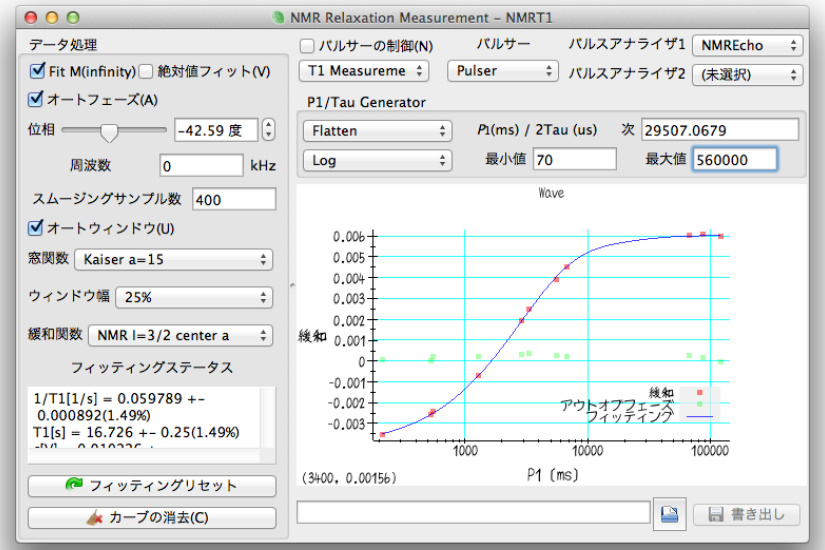
赤の部分”abs”が複素FFTでの絶対値です。FFT原点が合っていれば、”phase”はフラットになるはずです。”dark”はバックグラウンドから見積もられたノイズレベルです。”Peaks”は極大値を示します。

メイン画面の “IFFT Re.”/”IFFT Im.”には、スペクトラム解析の結果のIFFTが表示されています。つまり、ZF-FFTでは窓関数を掛けただけのものです。”DSO Re.”/”DSO Im.”にはDSOの波形からバックグラウンド処理をしたものが薄く表示されています。

スカラエントリには、最大値とその値をもつ周波数が送られます。

NMR緩和率測定

このドライバは、解析専用で、FID／エコー測定の結果から緩和率を測定します。パルサーの制御も行います。FID／エコーがインクリメンタル平均になっているときは使用できません。窓関数の設定は引き継がず、結果を見ながら合わすことができます。



「パルサーの制御」をチェックすると、一度結果が消去され、データがレコードされる度に次のP_1等の値がパルサーにセットされます。”Random”の場合、その値は完全に乱数で決められます。”Flatten”の場合は、空いている部分を重点的に割り当てます。「P_1分散」を”Log”にすると対数スケールで見て「最小値」と「最大値」の間から均等に探します。「次」に次回使用予定の値が表示されているので、必要に応じて変更できます。

「フィッティングステータス」には、「緩和関数」で最小二乗フィット結果が表示されます。

「オートフェーズ」をチェックすると、もっとも変化量が大きくなるような位相が選択されます。

「スムージングサンプル数」はフィッティングの前に近い横軸の値をまとめる分割数を指定します。

「オートウィンドウ」をチェックすると、最もS/Nが大きくなるような窓関数を選択しますが、周波数依存性が物理的に重要な場合（超伝導状態等）は、これをOFFにして下さい。

“Fit M(infinity)”をチェックすると、t→∞の値もフィッティングパラメータにします。通常のT_2測定ではOFFにします。

「周波数」は、FID／エコー信号の解析対象の中心周波数をセットします。この値を変更するとこれまでの測定結果はクリアーされます。

「フィッティングリセット」は、フィッティングが上手く行かないときに初期値を乱数で振りフィットをやり直すときに使います。

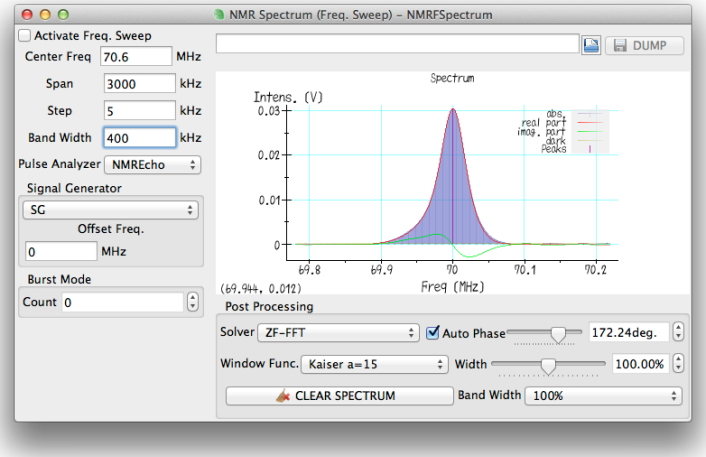
パルサーの設定が”P1 ALT”か”Comb ALT”の時はFID／エコー解析ドライバを２つセットする必要があります。

スカラエントリに値（$1/T_{1,2,st.e.}$と誤差）が送られます。

NMR周波数掃引測定

このドライバは、解析専用で、FID／エコー測定の結果から周波数掃引スペクトラムを測定します。SGの制御も行います。FID／エコーがインクリメンタル平均になっているときは使用できません。 窓関数の設定は引き継がず、結果を見ながら合わすことができます。

ZF-FFTの結果をずらしながら積算する、いわゆるFourier Step Sum(FSS)を行います。ただし、一度タイムドメインに戻すことで、窓関数の選択を後処理中で可能にしています。



「掃引開始」をチェックすると、結果をクリアーして、「中心周波数」-「スパン」/2から「ステップ」毎にSGの周波数を変更します。

「オートフェーズ」で実部が最大になるようにフェーズを選びますが、通常は周波数掃引では位相は意味がありません。

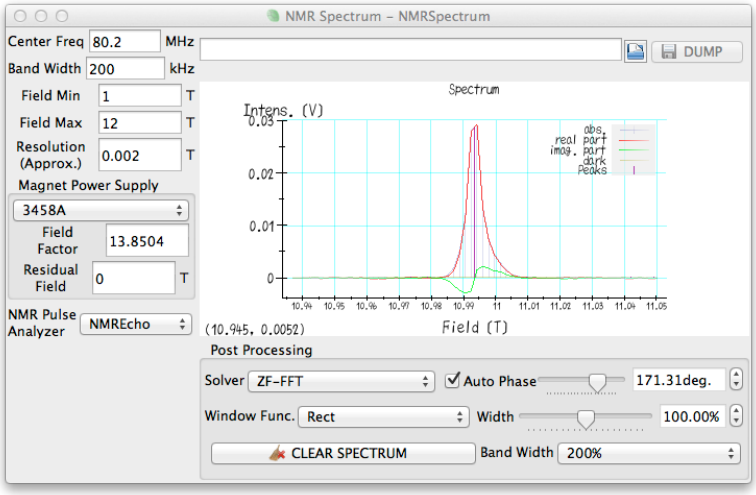
FSSの「バンド幅」は、測定開始前に数値で指定する他、後処理中でも”50%”,”100%”,”200%”の３種類から選択することが出来ます。

LC tuningでスイープ中に回路のチューニングをどうするか選択することが出来ます。”As is”ではなにもしません。”Auto Tune”ではオートチューナードライバを使用して”Step”毎ごとにチューニングを行います。”Await”では”Step”毎にパルスをOFFにしてユーザーがパルスを再度ONにするのを待ちます。

NMR磁場掃引測定

このドライバは、解析専用で、FID／エコー測定の結果から磁場掃引スペクトラムを測定します。磁場の値は、マグネット電源ドライバか、DMMの値から読み込みます。FID／エコーがインクリメンタル平均になっているときは使用できません。 窓関数の設定は引き継がず、結果を見ながら合わすことができます。

ZF-FFTの結果をずらしながら積算する、いわゆるFourier Step Sum(FSS)を行います。 ただし、一度タイムドメインに戻すことで、窓関数の選択を後処理中で可能にしています。



設定項目の説明については周波数掃引の項も参照して下さい。

内部動作としては、FSSの際に、周波数の原点を-log(H) x ”Center Freq”としています。これは、純粋なNMRの場合に磁場をずらしても正しく積算される仕掛けです。共鳴周波数が磁場に比例するか、”Center Freq”>>0.001x”Band Width”の場合に有効です。

磁場の値は、読み込んだ値に”Field Factor”を掛けて、”Residual Field”の値を足したものを使います。

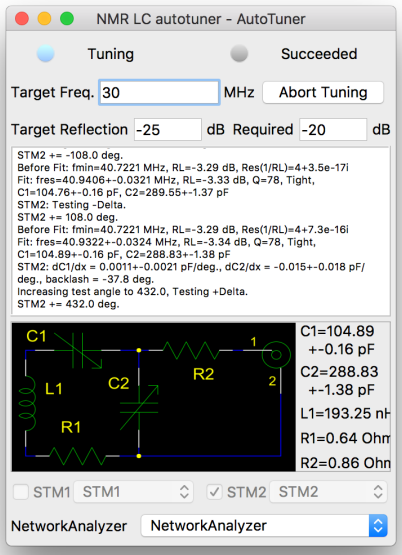
磁場のコントロールは、別のドライバ、または手動で行なって下さい。

“Resolution (Approx.)”は、横軸の精度を指定しますが、測定中に変更すると結果を一度クリアーする場合があります。

NMR自動LCチューナー

このドライバは、解析専用で、ベクトルネットワークアナライザをモニターさせながら、LC共振回路を調整します。LCR直列共振回路を仮定したフィッティングを行い、キャパシタ変位量とバックラッシを推定しながらチューニングします。調整の前後に、モータードライバの”AUX”のビットを反転さ

せるので、RFリレーの接続に利用できます。



スクリプトによる制御

Python(+ IPython)/Ruby言語による制御が可能です。kame内部では、Python/Ruby専用のスレッドが常時走っており、モニタープログラム(kame/script/pythonsupport.py, rubysupport.rb)がさらにPython/Rubyのスレッドを新しく作ってスクリプト(.py, .seq)を読み込み実行します。

標準入力と標準出力はリダイレクトされてkameの中央部に表示されます。文字コードはUTF-8です。

エラーが起これば、バックトレースが赤色で表示されます。

numpy as np/Mathモジュールはあらかじめ読み込まれています。

kameとの情報のやり取り／制御は、XNodeクラスのオブジェクトを介して行います。それらオブジェクトはツリー構造になっており、Measurementオブジェクトがルートです。

マウスポインタを制御対象に置くと “Node Browser”タブにPython/Rubyでのオブジェクト名とメソッド一覧及び現在の値が表示されます。

PythonではC++とほぼ同じ機能をサポートしていますが、modules/python/pydrivers.pyを参考にして下さい。help(hogehoge), inspect.getmembers(hogehoge)やJupyterのタブ補完も有用です。

Pythonでの制御

PythonからKAMEのノードツリーにアクセスするには、Root()オブジェクトを起点に操作します。値の読み取りはSnapshot、書き込みはTransactionを使います。

```
root = Root()                # ノードツリーのルート
```

```
# 値の読み取り（Snapshot）
shot = Snapshot(root)
print(shot[root])                # ルートノードのペイロード

# 子ノードへのアクセス
tempcontrol = root["tempcontrol"]
print(float(tempcontrol["temp"]))  # XDoubleNodeはfloatに変換可能

# 値の書き込み（Transaction）
for tr in Transaction(tempcontrol["setpoint"]):
    tr[tempcontrol["setpoint"]] = 4.2   # 競合時は自動リトライ
```

Pythonでドライバを記述する場合は、XPythonDriver<T>を継承したクラスをexportClass()で登録すると、コンパイル済みドライバと同様にフレームワークからインスタンス化されます。

```
class MyDriver(kame.XPythonCharDeviceDriverWithThread):
    def analyzeRaw(self, reader, payload):
        payload.local()["value"] = float(reader.pop_string())
    def visualize(self, shot):
        ...
MyDriver.exportClass("MyDriver", MyDriver, "My Instrument")
```

KAMEはIPythonカーネルを内蔵しており、IPythonが利用可能な場合はJupyterクライアントを実行中のプロセスに接続してインタラクティブな操作やライブプロットが行えます。メニューの「スクリプト → Launch Jupyter notebook」から起動できます。

以下はRubyに関してです。

XNodeクラスは、Arrayと似たようなメソッドをサポートしています。

例えば、Measurement["Drivers"]["NMRT1"]["P1Min"]で制御対象のオブジェクト（この場合はXDoubleNodeクラス）にアクセスできます。

XValueNodeクラス及びその子クラスは、value=メソッドで代入、valueメソッドで値の取得が可能です。

Rubyによる制御はわずかなタイムラグがありますので、連続して値を代入する場合は適当なスリープを入れて下さい。

Rubyでの例（各温度でスペクトラムを保存する）

```
def wait_within(node, val, min_wait, timeout, incr = 1)
 tstart = Time.now
 print "Wait for stabilize '#{node.name}' within #{val}"
 sleep min_wait
 while tstart + timeout > Time.now
  if node.get().abs <= val then
   print "OK. #{Time.now - tstart} sec. lost"
   return true
  end
  sleep incr
 end
 print "Time out #{Time.now - tstart} sec."
 false
end

tempctl = Measurement["Drivers"]["TempControl2"]
pulser = Measurement["Drivers"]["Pulser"]
pulse = Measurement["Drivers"]["NMREcho"]

for temp,rept in [[10,100],[20,50],[40,25]]
        print "T=#{temp}K\n"
        pulser["RT"].value= rept
        pulser["Output"].value= true
        tempctl["TargetTemp"].value=temp

        pulse["ExAvgIncr"].value= false
        sleep(3)
        pulse["ExAvgIncr"].value= true
        pulse["Spectrum"]["FileName"].set("hogehoge-#{temp}K-ft.dat")
        slp = [300.0, rept*100/1000.0].max()
        sleep(slp)
        pulse["Spectrum"]["Dump"].touch()

        wait_within(tempctl["Stabilized"], temp/100.0,30,36000)
end
```

ソフトウェアトランザクショナルメモリ（STM）

KAMEのコアデータモデルは、ロックフリーのスナップショットベースSTM（Software Transactional Memory）です。すべての測定器データはNode<XN>オブジェクトのツリーに格納され、読み書きはロックではなくスナップショットとトランザクションで行います。

読み取り（O(1)スナップショット）：

```
        Snapshot<NodeA> shot(node);        // アトミックロード、ロック不要
```

```
double x = shot[node].m_x;
```

書き込み（楽観的トランザクション、自動リトライ）：

```
node.iterate_commit([](Transaction<NodeA> &tr) {
    tr[node].m_x += 1;          // 初回アクセス時にコピーオンライト
});                             // 競合時は自動リトライ
```

このSTMはマルチスレッド環境でデッドロックなく動作し、UIスレッドがハードウェアI/Oを止めることはありません。PythonおよびRubyスクリプトも同じトランザクションAPIを通じてノードツリーにアクセスします。

ノードツリー構造

測定器データはNode<XN>オブジェクトのツリーに格納されます：

```
Node<XN>
 └─ Linkage  ──atomic_shared_ptr──▶ PacketWrapper
                                      └─ Packet
                                          ├─ Payload   (ユーザーデータ)
                                          └─ PacketList (子パケット)
```

コミットの仕組み

1. Transaction は構築時に m_oldpacket を保存します。

2. operator[] は最初の書き込み時にペイロードをクローン（コピーオンライト）し、一意のシリアル番号を付与します。

3. commit() は Linkage に対して単一のCAS操作を行い、packet ≠ m_oldpacket の場合は競合と判定してリトライします。

4. リスナーへのイベント通知はコミット成功後にのみ行われます（途中状態は外部から不可視）。

マルチノードの整合性は「バンドリング（bundling）」プロトコルで保証されます。親パケットがマルチフェーズCASプロトコルで子パケットを吸収し、サブツリー全体を単一アトミックポインタ配下で一貫させます。m_missingフラグが古い子パケットをマークし、必要に応じて再バンドリングを行います。

衝突バックオフ：Linkage::negotiate() は m_transaction_started_time タイムスタンプを使い、競合検出時に比例的な待機を課してライブロックを防ぎます。

iterate_commit_while(lambda) はラムダから false を返すことでリトライループを中断でき、条件付きトランザクションに使えます。

ロックフリーatomic_shared_ptr

上記のO(1)スナップショット読み取りとCASベースのコミットには、それ自体がロックフリーなスマートポインタが必要です。atomic_shared_ptr（kame/atomic_smart_ptr.h、2006年の2.0-beta3リライトで導入）がこれを提供します。C++20の std::atomic<shared_ptr> の独自実装です。

基本的な仕組みとして、参照制御ブロックへのポインタの下位ビットにローカル参照カウンタを埋め込みます（アロケータのアライメントで保証されたゼロビットを利用）。reserve_scan_() がCASでこのローカルカウンタをインクリメントしてポインタを安全に読み取り、leave_scan_() でデクリメントします。この間、他スレッドがポインタを差し替えても、ローカルカウントが非ゼロなのでオブジェクトは解放されません。グローバル参照カウンタが長期的な所有権を追跡し、差し替え時にローカルカウントをグローバルに転送することで、leave_scan_() がグローバルカウンタのデクリメントにフォールバックできます。

atomic_countable を継承するクラス（特にPayload）では、グローバル参照カウンタをオブジェクト自体のメンバとして持つ侵入的カウンティングを使い、スマートポインタごとのヒープ割り当てを削減します。

標準ライブラリ実装との比較（2024年末時点）：libstdc++（GCC）はスピンロック方式で優先度逆転に脆弱、MSVCはロックビット＋WaitOnAddressで競合時にブロッキング、libc++（Clang）は未実装。KAMEの実装（2006年〜）はタグ付きポインタCASによる真のロックフリーです。

【注意】ハードリンク（2つの親を持つ子）を含むツリーでトランザクション内にネストしたSnapshotを取ると整合性が崩れる可能性があります。その場合は nested Snapshot の代わりに tr[*node] を使ってください。

（以下はClaudeがソースコード解析に基づいて記述した内容です）

他のSTM設計との比較

一般的なSTM（Haskell TVar、Clojure dosync、ScalaSTM）は「フラット型」で、独立したトランザクショナル変数の集合を単位とします。KAMEのSTMは「ツリー構造型」であり、測定器ノードツリー全体が共有状態で、スナップショットは常にサブツリー単位で一貫します。

主な設計の違い：

競合の粒度：フラットSTM＝変数単位 / KAME＝パケット（サブツリールート）単位

読み取りモデル：フラットSTM＝readTVar/deref / KAME＝Snapshot（外側）またはtr[*node]（内側）

整合性スコープ：フラットSTM＝明示指定の変数群 / KAME＝バンドリングによるサブツリー全体

コミットログ：フラットSTM＝Redoログ or 書き込みセット / KAME＝コピーオンライト＋単一CAS

リトライ：フラットSTM＝retry/orElse / KAME＝iterate_commit/iterate_commit_while

ブロッキング：フラットSTM＝readセット変化で中断 / KAME＝なし（タイムスタンプバックオフ）

メモリ管理：フラットSTM＝GC / KAME＝ロックフリーatomic_shared_ptr（参照カウント）

Intel TSX/RTM（ハードウェアトランザクショナルメモリ）との比較：HTMはキャッシュライン競合で論理的に独立した操作でも中断し、容量制限も厳しいです。KAMEのSTMはパケット同一性の変化（意味的競合）のみで中断し、大きな読み取りセットを許容し、グローバルロックへのフォールバックではなくソフトウェアバックオフで劣化します。

TinySTM/NOrec（CライブラリSTM）との比較：これらはグローバルバージョンクロックとオブジェクトごとのバージョンスタンプ、トランザクションごとの読み書きログを使います。KAMEは読み取りログを持たず、Snapshotは不変ポインタにすぎないためトランザクション外の読み取りは真にゼロオーバーヘッドです。代わりに書き込みパスでペイロードをあらかじめクローンします。

KAMEのSTMが実験室ソフトウェアに有効な理由：

① デッドロックなし：ハードウェアI/OやUI再描画をまたいでロックを保持しません。低速のUIスレッドが高速データ取得スレッドを止めることはありません。

② マルチ測定器の一貫性：任意のサブツリーのSnapshotは常に内部一貫します。複数のドライバが同時に更新しても、UIが見るデータは常に整合した状態です。

③ Python/Rubyスクリプトからの安全なアクセス：スクリプトもC++と同じトランザクションAPIでノードツリーを操作するため、実行タイミングによらずスクリプトが測定器状態を破壊することはありません。

AI連携による実験自動化（MCP）

KAME 8.0では、Model Context Protocol（MCP）サーバーを内蔵しました。MCPはAnthropicが策定したオープンプロトコルで、AIアシスタント（Claude等）が外部ツールと連携するための標準規格です。

KAMEのMCPサーバーは、内蔵IPythonカーネルにjupyter_client経由で接続し、AIアシスタントがKAME内のPythonインタープリタで直接コードを実行できるようにします。Root()、Snapshot()、Transaction()など、Jupyterノートブックと同じ環境がAI側から利用可能です。

主な機能

ツール名	機能
kame_api	Python APIリファレンスの取得（AI用）
execute_code	KAMEインタープリタ内でPythonコードを実行
read_node	ノードパス指定で値を読み取り
read_scalar	数値ノードの値をJSON形式で取得
list_children	子ノード一覧（型・値付き）を取得

list_scalars	全スカラーエントリの値一覧を取得
kame_status	KAME動作状態とドライバー一覧の確認

利用例

AIアシスタントに自然言語で指示するだけで、計測器の操作・データ取得が可能です：

・「LakeShore1の現在温度を読んで」

・「磁場を0〜5 Tまで0.1 T刻みで掃引し、各点でNMR信号を記録して」

・「直近100回のDMM読み値をプロットして」

セットアップ

1. 必要パッケージのインストール：

pip install mcp jupyter_client

2. KAMEを起動し、Script → Launch Jupyter Notebook でノートブックを起動します。

3. KAMEが自動的に .mcp.json をノートブック作業ディレクトリに生成します。

4. 同じディレクトリでClaude Codeを開くと、MCPサーバーが自動認識されます。

5. KAME終了時に .mcp.json は自動削除されます。

技術的特徴

・KAMEの内蔵IPythonカーネルにZMQ経由で接続

・stdio トランスポートによるプロセス間通信

・kame_python_api.md をAIが自動参照し、試行錯誤を削減

・計測ソフトウェアへのMCPサーバー組み込みは世界初の試みです

FAQ

GPIBで通信エラーが起こるときは？

よくあるのは、アドレスが間違っているとか、ケーブルが長すぎるとかですが、まれに機器間の相性問題があります。古いOxford社機器とYOKOGAWAの7651、cryoconは、IEEE488.2規格に準拠していないので、非常に沢山の機器をつないだ場合に問題が起こることがあります。装置を減らすか、一部をシリアル通信に切り替える等しましょう。

シリアルポートの設定は？

上記に特に指定がない限り、9600bpsノンパリティです。

kameで対応していない機器を使いたい時は？

ドライバを作成する必要があります。ソースコード中のmodules以下のフォルダを参考にして作ることできますが、難しければ北川にコンタクトしてみて下さい。デジボルの機種を追加する程度であれば僅かな作業で出来ます。pythonでドライバを作製する場合は、modules/pythonを参考にしてください。

付録（obsolete）

NI製DAQmxデバイスを用いたNMRシステム(パルサー及びオシロスコープ／アベレージャ)

私は、SシリーズのPCI-6111（2ch12bit5MSps）をA-D変換器として利用しています。変調にQAMを使用する場合は、D-A変換部分も使います。上位機種のPCI-6115も利用できます。

内部動作としては、リアルタイムに全てリングバッファに取り込んで、ソフトウェアで積算しています。従って、いかなる繰り返し時間でも、取り込み範囲が重ならない限りは取りこぼしは有りません。また、積算回数に制限はありませんが、DSOドライバでは32bitで積算しているので65536を超える積算は、NMRエコードライバとの組合せで行なって下さい。

パルサーとしては、MシリーズのPCIe-6251(DIO 8ch10MHz)を使います。PCI-6220も遅いですが、使えます。PCIe-6251はディスコンになりますが、後継機種PCIe-6351についてはまだ動作確認していません。

その他必要な外部端子等としては、

PCI-6111用に
SH68-68-EP　1個
BNC-2110　1個

PCIe-6251用に
CB-68LPR　1個
SHC68-68EPM　1個
CA-1000 Box 1個
CA-1000 ラックマウント 1セット
CA-1000 BNCパネル 4個
CA-1000 カバー 5個

を使っています。

また、２つのボード間のクロック・トリガー同期用にRTSIケーブルの接続が必要です。

CA-1000 BNCパネルは絶縁BNCを使っていますが、これを非絶縁BNCに交換しないと、グリッチがNMRエコーに混じって観測されてしまいます。

Sシリーズデバイス

AI0,1(2ch)またはAI0-3(4ch)がA-D変換用の入力端子です。NMRではレシーバーのPSD出力に繋ぎます。0度と90度のポートが逆になってレシーバーもあるので注意。

ドライバで”National Instruments DAQ as DSO”を選択してください。

モジュレーターにQAMを使用する場合、パルサードライバとして、”NMR pulser NI-DAQ M Series with S Series”を選択して、2つ目のインターフェースにSシリーズを選んで下さい。AO0,1をQAMに繋ぎます。

Mシリーズデバイスのパルサー用接続

QAMを使用しない場合、パルサードライバとして、”NMR pulser NI-DAQ digital output only”を選んで下さい。

SシリーズをDSOとして使用する場合、DSOの「トリガー設定」で「ソース」にパルサーの適切なラインを選んで下さい。これはソフトウェアでトリガーが掛かるので、物理的に存在しないポートでも構いません。お勧めは、ポート8を”Trig1”に設定して、「ソース」を”Pulser名/line8”にすれば、2τの位置にトリガーが掛かります。ソフトウェアでトリガーをかけている都合上、DSOの縦横軸の設定を変更した後は、パルサーをOFF/ONしなおさないとトリガーがかかりません。

どんなパルサーにも言えますが、出力中に電源OFFした機器に繋いだり50Ω終端するのは止めましょう。ショートしているのとほぼ同じ事になります。

OFF時にはオープンステートにしますので、GATEとPRE-GATEのポートは数kΩ程度でGNDに落として下さい。

以下の写真を参考に接続して下さい。

ポート7は、内部動作用に予約しています（設定で変更できますが）。パルサーに信号変化がないときにカウントを停止させてボードを休ませる用途に使っています。その際、P0.7(J48)とPFI4(J41)を接続して下さい（写真中青い線）。また、NMRパルサーの設定画面の「もっと設定」内でポート7を”Pausing(PFI4)”に設定して下さい。

接続表は以下の通りです。
（ポート）（DAQポート名）（端子番号）
ポート0 P0.0 J52
ポート1 P0.1 J17
ポート2 P0.2 J49
ポート3 P0.3 J47
ポート4 P0.4 J19

ポート5 P0.5 J51
ポート6 P0.6 J16
ポート7 P0.7 J48 / PFI4 J41
GND D-GND 4,7,9,12,13,15,18,35,36,44,50,53のいずれか