On-the-fly scanning: Improvements in the EPICS *motor* and *mca* modules

Mark Rivers

University of Chicago

Motivation

- Traditional step scanning overheads:
  - Start and stop motors & mechanical systems
  - Arm and read detectors.
- Typically much more efficient to collect data on-the-fly as the motors are moving.
  - Better detectors, more flux with APS Upgrade, don't waste photons and time
- However, on-the-fly scanning with EPICS has been subject to limitations.
  - Typically limited to simple single-motor scans because EPICS did not provide a way to program motor controllers to perform synchronized motion of multi-axis controllers.
    - Exception to this was the Trajectory Scanning software I wrote for the Newport XPS and MM4005 motor controllers.
    - Not a clean solution: SNL program talking directly to the controller, "behind the back" of the motor record driver.
    - PMAC does have motion control programs – good for fixed, simple geometry
  - New solution that incorporates a coordinated motion API in the motor driver layer
- SIS3801 and SIS3820 multi-channel scalers are typically used for collecting pulse counting data in on-the-fly scans.
  - Drivers were previously not able to handle very large arrays (1,000,000 elements or more) without unacceptable slowing down the IOC response.
  - Drivers have been rewritten to eliminate this problem, and to add a number of significant new features.

EPICS Motor Support

- Top-level object is the EPICS **motor record**
  - Lots of code has been written to this object:
    - spec, IDL and Python classes, etc.
- Next layer is EPICS **device support**
  - Knows about the motor record, talks to the driver
- Lowest layer is EPICS **driver**
  - Knows nothing about motor record, talks to the hardware
- 3 models of device and driver support have developed over time

Model 1

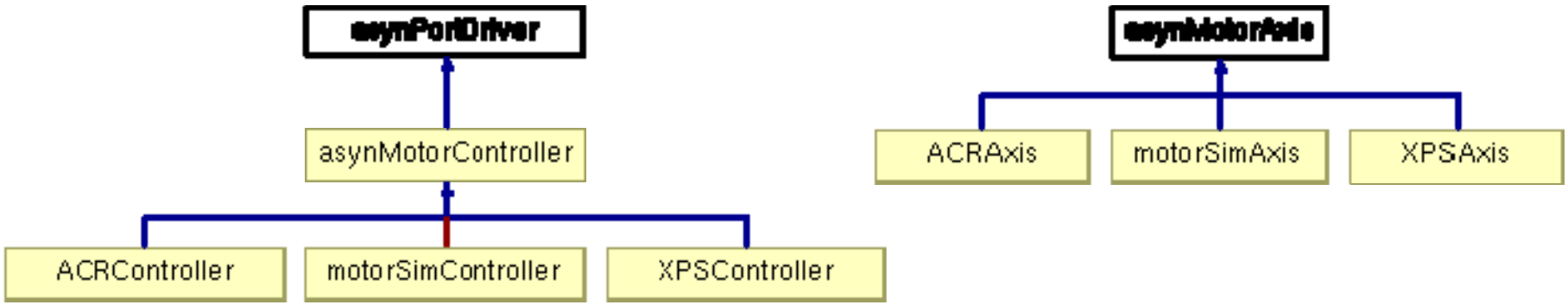
- Device-dependent device support and driver support for each controller type
- Communication between device support and driver is custom for motor code and very limited
- Cannot use other records to talk to driver only motor record
- Cannot take advantage of controller-specific features not supported by motor record
- No provision for multi-axis coordination
- Most drivers in the motor package are written this way for historical reasons

Model 2

- Developed 2006 by APS and DLS
- Uses standard *asyn* interfaces to communicate between device support and driver
- A *single device-independent* device support file
- A *single device-independent* driver support file for asyn interfaces
- Device dependent driver file below the asyn one
- *Can* use other records to talk to driver via asyn interfaces
- Not as easy as it should be to do so
- No implementation of multi-axis coordination
- A few drivers in the motor package are written with Model 2
  - Simulation driver, Newport XPS, Newport MM4000, ProDex MAXnet, Attocube ANC150, Aerotech Ensemble

Model 3

- New C++ model introduced here.
- Two base classes, asynMotorController and asynMotorAxis.
- Base classes provide much functionality, only need to write device-specific implementations.
- Easy to support controller-specific features
- Don’t have to use motor record.
- Direct support for coordinated profile moves in the driver API.



asynMotorController

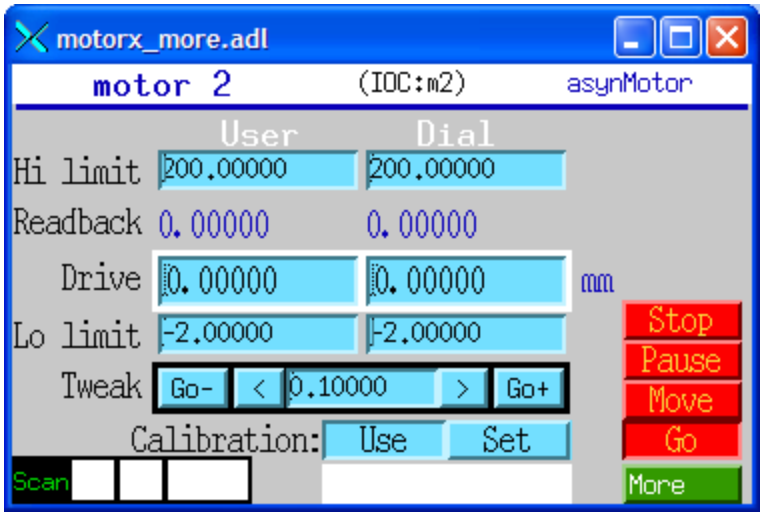
```
class epicsShareFunc asynMotorController : public asynPortDriver {
  asynMotorController(const char *portName, int numAxes, int numParams,
                      int interfaceMask, int interruptMask,
                      int asynFlags, int autoConnect, int priority, int stackSize);

  virtual asynMotorAxis* getAxis(asynUser *pasynUser);
  virtual asynMotorAxis* getAxis(int axisNo);

  virtual asynStatus startPoller(double movingPollPeriod, double
     idlePollPeriod, int forcedFastPolls);
  virtual asynStatus wakeupPoller();
  virtual asynStatus poll();
  void asynMotorPoller();

  virtual asynStatus initializeProfile(size_t maxPoints);
  virtual asynStatus buildProfile();
  virtual asynStatus executeProfile();
  virtual asynStatus abortProfile();
  virtual asynStatus readbackProfile();
```

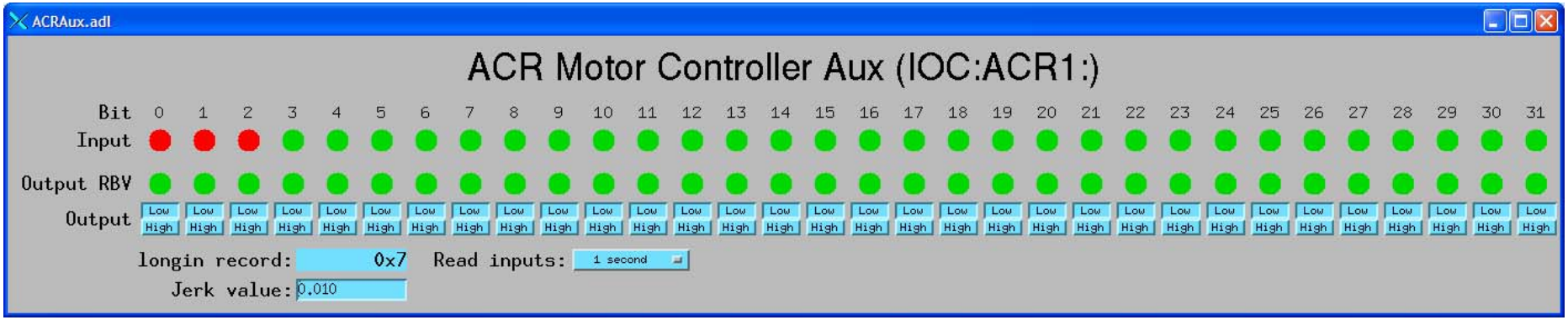
asynMotorAxis

```
class epicsShareFunc asynMotorAxis {
  virtual asynStatus move(double position, int relative,
            double minVelocity, double maxVelocity, double acceleration);
  virtual asynStatus moveVelocity(double minVelocity, double maxVelocity,
            double acceleration);
  virtual asynStatus home(double minVelocity, double maxVelocity,
            double acceleration, int forwards);
  virtual asynStatus stop(double acceleration);
  virtual asynStatus poll(bool *moving);
  virtual asynStatus setPosition(double position);

 virtual asynStatus initializeProfile(size_t maxPoints);
  virtual asynStatus defineProfile(double *positions, size_t numPoints);
  virtual asynStatus buildProfile();
  virtual asynStatus executeProfile();
  virtual asynStatus abortProfile();
  virtual asynStatus readbackProfile();
```

Parker ACR Model 3 Driver

- 650 lines of code total (ACRMotorDriver.h, ACRMotorDriver.cpp)
- Supports Jerk parameter, and bi, bo, longin records via standard asynUInt32Digital interface for TTL I/O lines
- Less than 100 lines of code for digital I/O support

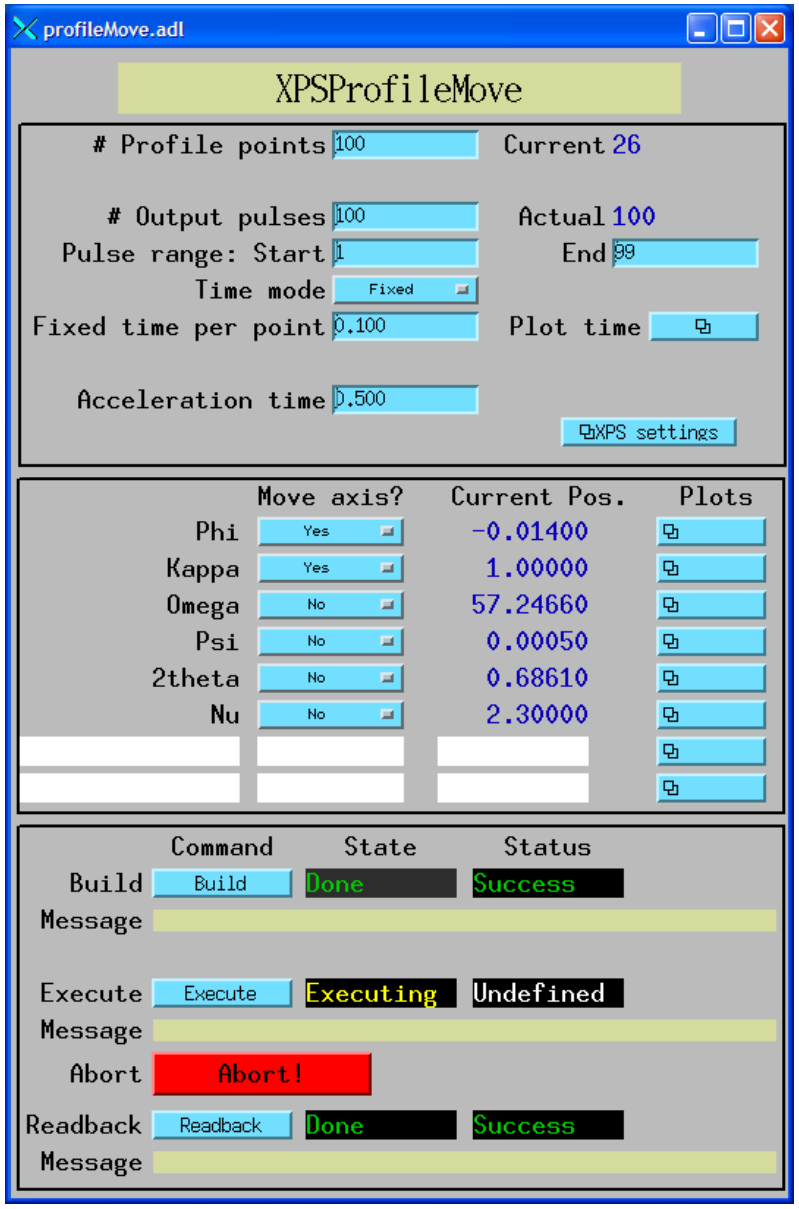




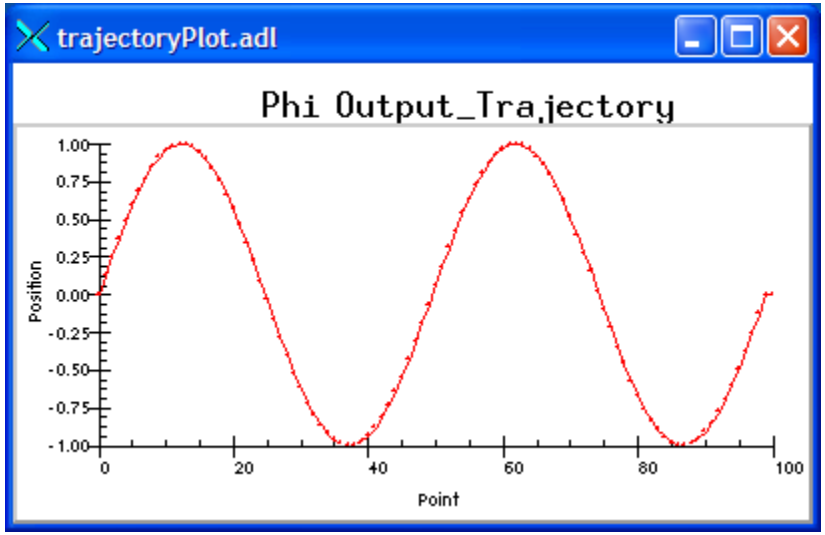
Coordinated Motion
(Formerly called Trajectory Scanning)

- Create arrays of target locations of each motor in a multi-axis controller
- Create array of time per element
- Define times/locations to output synchronization pulses to trigger detectors
- Execute coordinated move
- Optionally read back encoder positions when each synchronization pulse was output
- Many controllers have this capability
  - Newport MM4000, Newport XPS, Delta Tau PMAC, Prodex MAXv, Parker ACR series, etc.
- We now have an API to take use this capability in a consistent way
- Now done in same driver as for motor record, eliminates conflicts

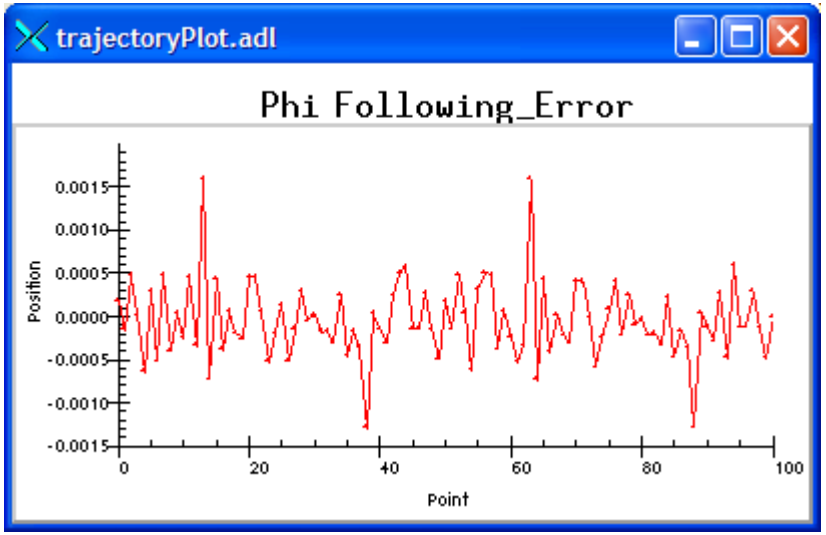
Coordinated Motion



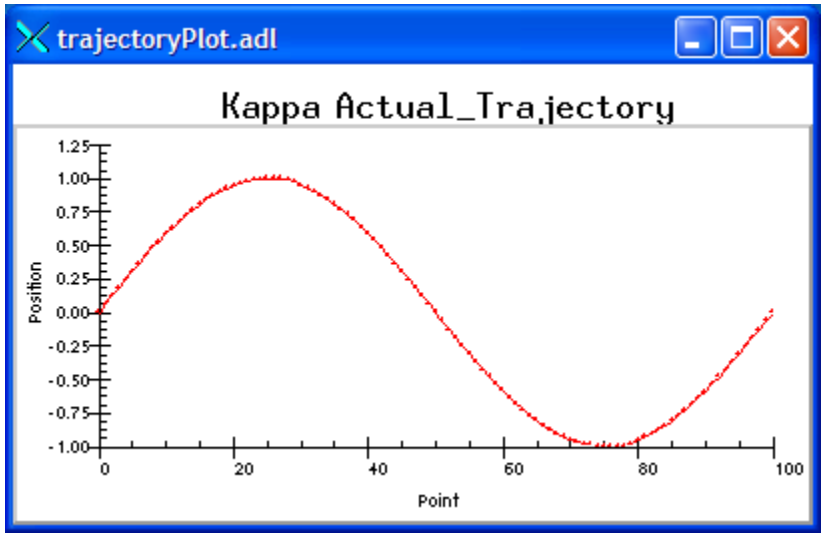

```
; IDL program to build, execute and readback profile
; This program builds a profile and executes it.

nelements = 100 ; 100 elements in the profile.
npulses = 100 ; 100 pulses in the profile.
naxes=2       ; We will move the first 2 motors (Phi and Kappa)

; Define array of positions
positions = dblarr(nelements, naxes)

; The Phi profile is a sin wave with two complete periods and an
; amplitude of +-1 degrees
positions[*,0] = 1.*sin(findgen(nelements)/(nelements-1.)*4.*!pi)

; The Kappa profile is a sin wave with one complete period and an
; amplitude of +-1 degrees
positions[*,1] = 1.*sin(findgen(nelements)/(nelements-1.)*2.*!pi)
profile = 'IOC:Prof1:'
group = 'GROUP1'

time = 0.1      ; Fixed time per profile point
status = profile_move(profile, positions, group=group, maxAxes=6, $
              build=1, execute=1, readback=1, time=time, $
              npulses=npulses, actual=actual, errors=errors)
end
```



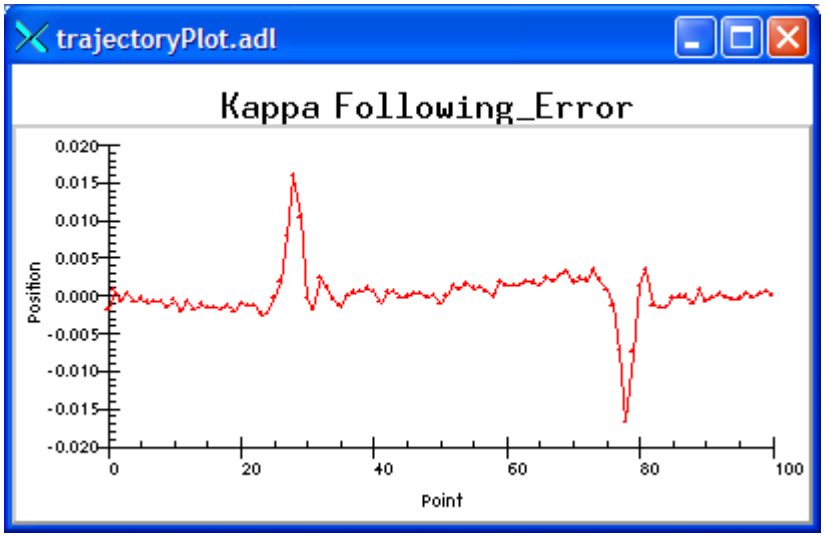
Phi output profile



Phi following error



Kappa readback profile



Kappa following error

SPEC Interface

- SPEC macros have been written to allow SPEC to utilize profile moves via EPICS interface
- Low level interface, all of SPEC's standard scans can be done "on-the-fly" with profile move software. Replacement macros for:
  - _ascan # Used by all ascan and dscan macros
  - Mesh
  - hklscan # Used by hscan, kscan and lscan
  - _hklmesh
  - _hkllline # Used by hkcircle, hlcircle, klcircle, hkradial, hlradial and klradial
  - _scanabort resume
  - _loop

SPEC Interface

- traj_index # Converts a SPEC motor index to an MM4005 motor index
- traj_build # Builds a trajectory
- traj_exec # Executes a trajectory
- traj_read_counts # Reads the data from the multi-channel scaler
- traj_read_actual # Reads back the actual MM4005 motor positions
- traj_scans_on # Enables trajectory scanning
- traj_scans_off # Disables trajectory scanning, uses step scanning
- Global variable TRAJ_USE_ACTUAL=(0,1)

Future Work

- Convert MAXv to Model 3
  - Tim Mooney already has MAXv trajectory scanning running in SNL program
- Support other controllers
  - PMAC, etc.
- Add additional features
- Work on replacement for motor record?

SIS 3801/3820 Software Improvements

- Joerger VSC8/16 no longer available, SIS3820 attractive alternative
- 32 inputs, on-board FIFO buffer memory (64MB/512 MB)
- Works in 2 modes
  - Conventional scaler (like Joerger) with scaler record
  - Multi-channel scaler with MCA record
- Ideal for on-the-fly scanning
  - Internal channel advance, 1 microsecond to many seconds
  - External channel advance with divide-by-N
    - Use encoder or motor pulses as channel advance
- SIS3801 is older model, much less memory, fewer features.
  - No preset capability, I emulate it in software when reading FIFO
  - Before that there was the nearly identical STR7201.

Problems with previous releases

- In previous releases the SIS3801 and SIS3820 drivers were "synchronous“:
- MCA records blocked while arrays were read from the FIFO on the VME modules.
- FIFO reads were done with VME program I/O (~4MB/s).
  - OK for small arrays (e.g. 2048 channels) because it only took a few ms to read each array.
- For very large arrays (e.g. 2,000,000 channels) it took several seconds to read each array, blocking execution of other EPICS records and tasks, leading to channel access disconnects and other bad effects.

Improvements in new version (mca R7-0)

- FIFO is read by background thread into memory buffer
  - Uses DMA on 3820, much higher performance (> 40MB/s)
  - MCA or waveform records just read from memory buffer, no VME I/O
- Can use either MCA record or standard EPICS waveform record.
  - Performance better with waveform record for very large arrays, less overhead
- New PVs for
  - Software channel advance.
  - Initial software channel advance when acquisition is started
  - Configuration modes: 4 input control signals and the 4 output control signals.
  - Channel 1 source: internal 25/50 MHz clock or external pulse input.
  - control the user-defined LED and user-defined output control line on 3820
  - Select which input is routed via the MUX output control signal on the 3820.
  - Current acquisition mode (MCS or Scaler), model number and firmware version.
  - Current channel number being acquired.
- Replaced logic in databases (e.g. Struck32.db) with simple SNL program
  - A single database can now be used. Previously needed different databases depending how many inputs (2, 8, 16, 32, etc.) were being used

Improvements in new version (mca R7-0)

- drvSIS38XX base class derived from asynPortDriver
- drvSIS3801 and drvSIS3820 derived from drvSIS38XX.
- With improvements and new features in the drivers the amount of code has **decreased** in R7-0 compared to previous releases

Size of driver source code			
R6-12-5		**R7-0**	
File	**Lines**	**File**	**Lines**
devStr7201.c	264	SIS38XX_SNL.st	116
devScalerST7201.c	326	apsLib.h	58
drvMcaSIS3820Asyn.c	1689	drvSIS3801.cpp	731
drvMcaSIS3820Asyn.h	285	drvSIS3801.h	210
drvSTR7201.c	812	drvSIS3820.cpp	1017
drvSTR7201.h	38	drvSIS3820.h	225
mcaSIS3820Register.c	62	drvSIS38XX.cpp	542
mcaSSTR7201Register.c	49	drvSIS38XX.h	164
sis3820.h	240	sis3820.h	240
N.A.	-	vmeDMA.h	53
TOTAL	**3765**	**TOTAL**	**3356**

Size of dataBases			
Struck32.db	623	SIS38XX.template	303
Struck8.db	215	SIS38XX_waveform.template	9
STR7201scaler.db	241	N.A.	-
SIS3820.db	7	N.A.	-
TOTAL	**1086**	**TOTAL**	**311**

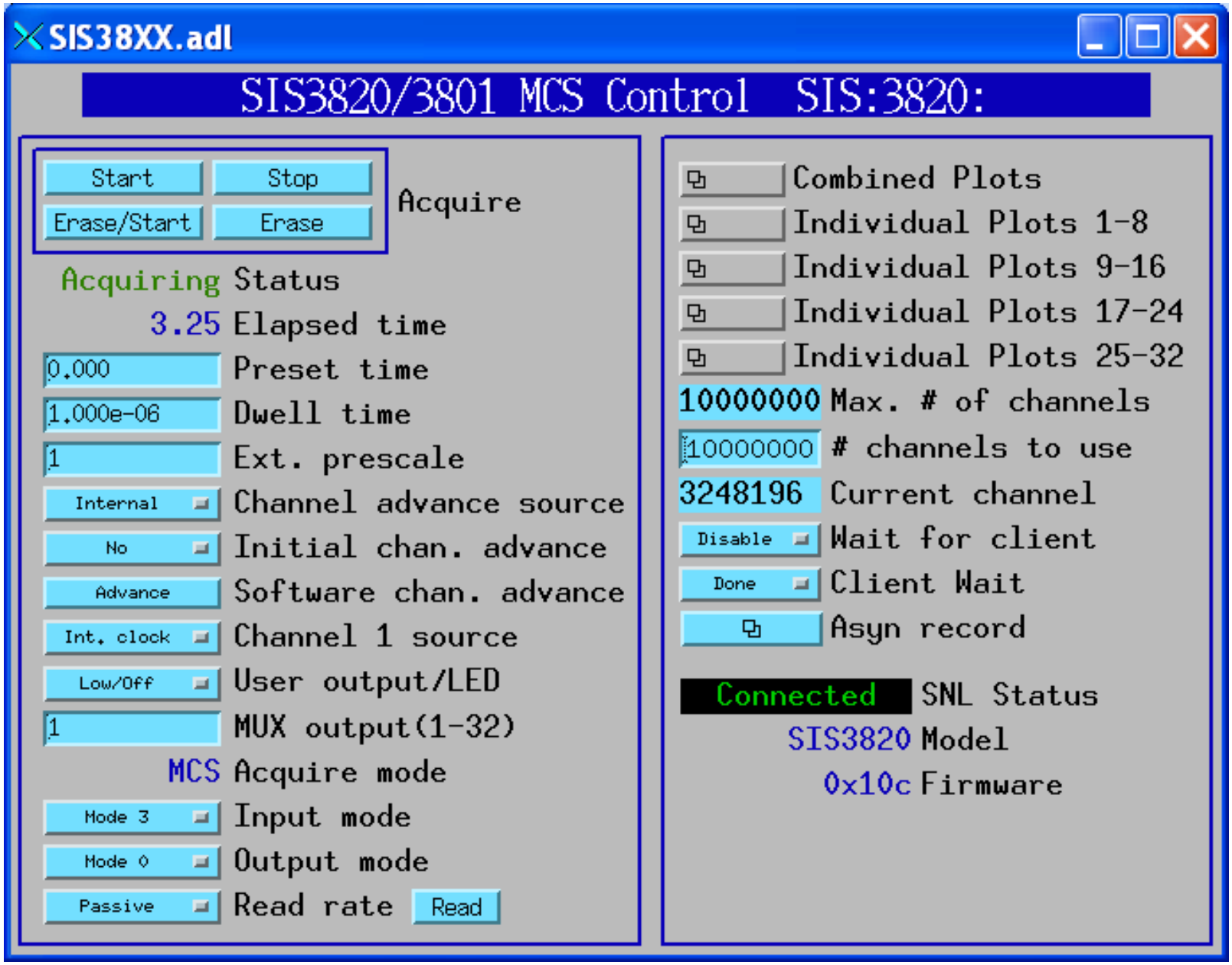
Performance with very large arrays

- Test configuration (simulating how XPCS might want to run):
  - SIS3820 module with 512MB of FIFO memory
  - MVME 5100 CPU with 512MB of memory
  - vxWorks 5.4.2 with Andrew Johnson's BSP with DMA support
  - 2 active inputs
  - 10,000,000 channels per waveform
  - 2 waveform records
  - Internal channel advance
  - 1 microsecond dwell time
  - ReadAll.SCAN = Passive
- Theoretical time for this acquisition to complete is 10.0 seconds. Output from the EPICS "camonitor" program on a Linux client looking at the Acquiring PV, and the first 2 channels of each waveform record:

```
> camonitor -tc -#2 SIS:3820:mca1 SIS:3820:mca2 SIS:3820:Acquiring
SIS:3820:Acquiring      (2011-05-01 11:19:35.050808) Acquiring
SIS:3820:mca1           (2011-05-01 11:19:46.017134) 2 50 50
SIS:3820:mca2           (2011-05-01 11:19:46.017223) 2 0 0
SIS:3820:Acquiring      (2011-05-01 11:19:46.017264) Done
```

- Time between when acquisition started and when the client received the data and the Acquiring=Done status was 10.97 seconds.
  - Less than 1 second of overhead in collecting 2 waveforms of 10,000,000 elements each
  - = 80MB of data

SIS38XX.adl


SIS38XX.adl
SIS3820/3801 MCS Control SIS:3820:
Start
Stop
Erase/Start
Erase
Acquire
Acquiring Status
3.25 Elapsed time
0.000 Preset time
1.000e-06 Dwell time
1 Ext. prescale
Internal Channel advance source
No Initial chan. advance
Advance Software chan. advance
Int. clock Channel 1 source
Low/Off User output/LED
1 MUX output(1-32)
MCS Acquire mode
Mode 3 Input mode
Mode 0 Output mode
Passive Read rate
Read
Combined Plots
Individual Plots 1-8
Individual Plots 9-16
Individual Plots 17-24
Individual Plots 25-32
10000000 Max. # of channels
10000000 # channels to use
3248196 Current channel
Disable Wait for client
Done Client Wait
Asyn record
Connected SNL Status
SIS3820 Model
0x10c Firmware

scaler_more.adl