SPEAR3 Insertion Device controls

Harvey Rarback
March 2012

Introduction

Difference between wigglers and undulators

Documentation

In addition to this document, the following insertion device (ID) documentation exists:

- http://ssrl.slac.stanford.edu/ets/slaconly/documentcontrol/en/m600.pdf
  describes the method for ID control by either operators or beamline users as of 2009 (before the row phase/linear phase mode of the BL13 EPU was implemented).

- http://www.slac.stanford.edu/~spear/epics/app/ID/bl5_epu_031412.pdf
  describes the BL5 EPU hardware which will be installed during the 2012 shutdown.

- http://www.slac.stanford.edu/~spear/epics/app/ID/SSRL_future_beamlines.pdf
  describes the plan for future IDs

- http://www.slac.stanford.edu/~spear/epics/
  is the portal for much of the SPEAR3 controls documentation.

ID history

In order to make sense of the current state of SPEAR3 insertion device controls, it helps to understand some of the history of the ID hardware and software. There is a lot of legacy that has wormed its way into the existing software: Some hardware like multiundulator control, motor brakes, ancillary filters, auxiliary motors and LVDT readouts have been removed but still persist in the code. There is no longer a need for fast wiggler trim tracking or trim table or orbit calculations but those early requirements have made the existing code more complicated that it need be. What follows is a description of the past, present and future of ID software and hardware at SPEAR.

• Hardware and software of the past

From the early 80's when the first IDs were installed in SPEAR all the devices were controlled through CAMAC interfaces accessible only from computers running OpenVMS. The motors were controlled through fairly dumb SMC CAMAC motor drivers and the trims and electromagnetic wigglers were driven by CAMAC DACs. The software had to fit into the framework of the PEP I controls and was written in FORTAN IV with a different program for each ID. The operator interface was written in FORTH and emulated some earlier touch panels. The feedforward tables for the ID trim compensation could only read the motor positions at about 2Hz so moving the IDs during stored beam operation would often lead to a beam dump. If the beamline user wanted to change the gap of an ID, he or she had to request the change through the Duty Operator who would in turn request the change be made by the SPEAR operator. Early in this century, we added an EPICS system channel access server on top of the SPEAR2 control system. This allowed us to use some of the EPICS client tools with the legacy software. In 2001, we changed the control language from FORTRAN to IDL which allowed for easier software development. In particular, all of the IDs were controlled from a single codebase. IDL's object oriented programming environment allowed for easier modularity and software inheritance; for example the motor class had subclasses of CAMAC and (later) VME motors, while the insertion device class was subclassed with permanent magnet and electromagnet "motors", etc.

The opening of SPEAR3 changed both the software and hardware environment for IDs. Gone were the electromagnetic wigglers and auxiliary motors for some of the IDs. The trim magnets were now controlled through MCOR-30s, much more flexible than the DACs they replaced. Gradually all the undulators on BL5 except the 10-period device were removed. When the EPU built at ADC and designed to be installed in BL13 failed to meet field quality standards, the old EPU at BL5 was installed on a jury-rigged frame and moved to BL13. This device still moves very unreliably because of its poor mechanical construction and will eventually be replaced.

More sophisticated accelerator physics Matlab codes along with better beam position monitoring could calculate trim tables much faster and more accurately than the older algorithms used in the IDL code. Code that talked directly to the CAMAC crates without using the legacy shared-memory based SPEAR2 control system sped up the acquisition of motor status so that 10Hz operation of the trim tables was possible. This became less important with the later introduction of fast orbit feedback running at 4kHz which controlled all of the ring correctors to correct the orbit when the IDs were being moved. A protocol to share control of the IDs with the beamline users was implemented. The routine control of the IDs was removed from the standalone IDL program and distributed to EDM displays and beamline control software.

- ## Hardware and software of the present

The table below summarizes the existing (and soon to be installed) ID hardware.

Insertion Device	Type	Interface	Motor Control	Encoder Readout	Dedicated Trims
BL4	Wiggler	CAMAC	SMC24B	QIR (16 bit)	2
BL5 (old)	Undulator	CAMAC	SMC24B	QIR (16 bit)	none
BL5 (new)	EPU	VME	MAXv 8000	SSI (6 axes)	4 ?
BL6	Wiggler	CAMAC	SMC-L	QIR (16 bit)	1
BL7	Wiggler	CAMAC	SMC24B	QIR (16 bit)	3
BL9	Wiggler	CAMAC	SMC-L	QIR (16 bit)	1
BL10	Wiggler	CAMAC	SMC-L	QIR (16 bit)	2
BL11	Wiggler	CAMAC	SMC-L	QIR (16 bit)	2
BL12-2	In-vacuum undulator	VME	MAXv 4000	IP470 (19 bit)	none
BL13	EPU	VME	MAXv 8000	SSI (6 axes)	1 skew quad

Insertion Device – The IDs are named for the main beamline they feed. The BL12-2 device is in a long straight section (where a Nobel Prize was won for the discovery of charm) which can support another (upstream) ID. The names of the beamlines are historical and unfortunately not related to their location.

Type – Wigglers are permanent magnet devices with a continuous spectral output. They are normally parked at their minimum gap and except for Accelerator Physics shifts are not moved. Undulators are also permanent magnet devices that produce quasi-monochromatic light output. Their gaps are often changed and even scanned during operations. EPUs are Elliptically Polarizing Undulators which have a movable gap as well as longitudinally moving magnet rows to change their "row phase" to produce light with a variable polarization. Both gap and row phrase are often changed during operations. All these devices are outside the vacuum chamber except for the Neomax-built excellent BL12-2 undulator which is located inside the vacuum chamber and consequently must be specially protected by interlocks and software to prevent the electron beam from striking its magnetic material.

Interface – Most of the present day IDs are controlled and read out through old CAMAC modules. There are three ways to address our CAMAC modules:

- Through the legacy shared-memory software
- Through the specialized direct-access CAMAC routines
- Though the portable channel access server on SPEAR3.

The server and the other software need to run on VMS because our CAMAC controllers currently only communicate via DECnet. All the realtime CAMAC ID software now runs on the VMS Alpha computer called SPEAR3. Our IDL code has only been licensed for the VMS Alpha platform. The existing software uses the direct-access CAMAC routines for speed and the legacy (slower) routines when they are adequate, e.g. when moving all the IDs at once. In the future if one wanted to run non-VMS control software without changing the hardware interface, one could imagine communicating with the (slow) portable channel access server.

The newer IDs use VME controls on IOCs running RTEMS, a realtime operating system. The plan is to convert all the CAMAC interfaces to VME when the resources to do so are available. In particular the BL6 CAMAC crate is physically close to the new BL5 VME crate and could be fairly easily converted to VME using spare channels of the BL5 motor controller and digital I/O. Similarly, the BL4 and BL11 controls could use the VME crate for BL13 operations.

Motor control – The Joerger SMC-L CAMAC motor controller controls a single stepper motor. The speed and acceleration time are controlled with onboard pots. The SMC24B has programmable speed and acceleration. These modules are no longer made. Our Pro-Dex MAXv VME motor controllers can control 4 (MAXv-4000) or 8 (MAXv-8000) motors. The EPICS motor record supports the MAXv controllers.

Encoder readout – SSRL has always used absolute linear encoders wherever possible which greatly simplifies control and reliability. The Joerger QIR CAMAC quad input register reads the 16-bit output of the absolute linear encoders without a strobe bit, but seems to give reliable data. The Acromag IP470A VME IP digital I/O module is also unstrobed but seems to give reliable data for the BL12-2 high resolution absolute linear encoders. The Kramert ECM-505/F VME SSI encoder readout module comes in the bizarre increment of 5 readouts but works fine compared to all the trouble we had with the earlier Kramert 505 hardware. The SSI encoders are also absolute linear encoders.

Dedicated trims – Some of the IDs were delivered or retrofitted with dedicated trim coils designed to compensate for first and second integral errors. Much effort had been expended trying to create feedforward trim tables to correct the orbit errors, but the big breakthrough occurred with the implementation of accurate fast orbit feedback. All the wigglers now have dedicated trim coils energized with non-feedback MCOR-30 controllers. The three undulators have no dedicated trims, but rely on fast orbit feedback for orbit correction. If fast orbit feedback is not running, the ID software uses two ring correctors on either side of the device to compensate the orbit. This doesn't work all that well. The BL13 EPU introduces a significant skew quad error when the gap is changed. This error manifests itself in beam size changes which affect the beam lifetime. To compensate for the skew quad errors, a dedicated skew quad corrector was fixed on the BL13 vacuum chamber. The software implements a feedforward table for this skew quad when the gap (but not the row phase) is changed.

- ## Limits

All the motions have hardware limit switches installed wired directly to the motor controllers which prevent any motions into the limits. The CAMAC controllers monitor the state of the switches all the time, but the MAXv controllers only check the state of the switches when a motion command is executed in the direction of the switches. This is unfortunate because conditions like a broken wire or radiation-damaged electronic limit switch can only be found when a motion command is issued. The VME motors (virtual or real) have PVs to indicate the state of the limit switches (0 for clear, 1 for engaged):

Motion	Inner (lower) limit PV	Outer (upper) limit PV
BL12-2 gap	BL12-2:GapMotor.LLS	BL12-2:GapMotor.HLS
BL13 gap	BL13:GapMotorLLS	BL13:GapMotorHLS
BL13 row phase	BL13:EPUMotorLLS	BL13:EPUMotorHLS

All the motions have software limits, which should prevent the devices from ever engaging a hardware limit switch by not allowing the move at all. The software limits for CAMAC motors are set in the initialization file IDL_USER:[INSERTIONS]CAMAC_MOTOR__INIT.PRO. The software limits for VME motors are set using a calculation in IDL_USER:[INSERTIONS]VME_MOTOR__DEFINE.PRO from this data:

- the .DLLM/.DHLM dial limits fields of the motor records set in EPICS databases
- the .BDST backlash field of the motor records set in EPICS databases
- allowed limits slop set in the initialization file VME_MOTOR__INIT.PRO.

It is sometimes necessary to override the software limits, especially when checking out the ID limit switches and hard stops. Currently, the only convenient way to override the software limits for a CAMAC motor is with the standalone IDL program: Enter a position (or field or encoder value) beyond the software limit in the boxes above the slider on the "mover window" and accept the popup warning that appears. The only convenient way to override the software limits for a VME motor is by changing the software limits in the diagnostic Motor Settings EDM display and then using the Tweak control in the diagnostic Motor Drive EDM display. (You can't use the slider control because they have built in limits.)

- ## Permits

Before an ID motor can be moved, certain hardware constraints ("permits") must be satisfied. The permit methods are listed in the initialization files CAMAC_MOTOR__INIT.PRO and VME_MOTOR__INIT.PRO. For VME motors, the EPICS databases check the hardware conditions and disable the motor records if the condition is unsafe and reenable the records when the hardware condition is removed. The move routines camac_motor::set_position() and vme_motor::set_position() execute the permit methods in CAMAC_MOTOR__DEFINE.PRO and VME_MOTOR__DEFINE.PRO and return without moving the device when permit is not satisfied. The permits required for the IDs are:

- **BL4** – remote control, no motor fault, no encoder fault, no emergency switch hit
- **BL5 (old)** – horizontal position is ok
- **BL6** – no permit methods
- **BL7** – transmit/receive chassis ok, remote control, no motor fault, no encoder fault, no emergency switch hit
- **BL9** – remote control, no motor fault
- **BL10** – no permit methods
- **BL11**– remote control, no motor fault, no encoder fault, no emergency switch hit
- **BL12-2** – remote control, no motor fault, no encoder has a fault, no emergency switch hit, no torque limiter fault, no excessive gap taper
- **BL13 gap** - remote control, no motor driver has a fault,
- **BL13 row phase** - remote control, no emergency switch hit

Presumably, the permits for the **new BL5 ID** will be similar to the existing BL13 permits.

Trim tables

As described above, some of the IDs including the **new BL5 ID** have dedicated trim coils. The initialization file INSERTION_DEVICE__INIT.PRO defines the following information for each ID:

- **trim_track** – 0 means don't use trim tables, 1 means use trim table, 2 means use fast orbit feedback (FOFB) when it is running
- **trim_device –** PV device name(s) of MCOR power supplies used in trim table
- **trim_table_ptr** – IDL pointer to full trim table structure defined in INSERTION_DEVICE__DEFINE.PRO
- **trim_table_up, trim_table_down –** when it has been determined that there is significant hysteresis in the trim magnets, separate tables of trim currents in amperes are used for increasing and decreasing gaps in mm
- **skew_track** – 0 means don't use skew quad tables, 1 means use them
- **skew_device** – PV device name of MCOR power supply used in skew quad table
- **skew_table** – skew quad current in amperes vs gap in mm
- **SOFB**.**disable** – 1 means disable slow orbit feedback (SOFB) when it is running instead of FOFB and use nearby corrector magnets for orbit correction, 0 means ignore SOFB

The trim tracking calculated in the ID_SERVER code is used when the ID gaps are moved. The breakpoint tables stored in the EPICS dbd files are currently copies of the data defined in the IDL initialization file and are currently used for EDM plots of the trim tables only. This should probably be changed.

At the conclusion of any ID move, the IDL code currently writes the trim table currents to <trim_device>:CurrReference which is an alias to <trim_device>:CurrSetptDes. It is this software desired value which is used to set the alarm state <trim_device>:State. Alarms can be set based on a discrepancy between the desired and actual values as described in V:\AD\SPEAR3\machine\spear3\ids\ID_trim_current_alarms.ppt. The desired trim currents are preserved across IOC reboots.

Field tables

The initialization file INSERTION_DEVICE__INIT.PRO defines the **field_table** for each ID. This is a table of the maximum B field in kilogauss for each gap in mm. This data is used in the IDL standalone display and control code. The data is copied to EPICS dbd files which are used for IOC calculations of device K factors and photon power used in EDM displays and the SPEAR home web page.

Ownership

In order to facilitate ID control by SPEAR operators as well as beamline users, the concept of ID ownership was introduced. The ID_SERVER listens to requests from only the current owner of the device. Ownership can be granted to beamline users by the SPEAR operator through an EDM panel. At any time the operator can take ownership effectively locking out the beamline user. Ownership (and the ID_SERVER) can be bypassed in controlling VME motors by using the EDM diagnostic panels. Trim tracking and logging will then also be bypassed.

The ownership logic is implemented in a combination of EPICS databases (beamlineGapControl.db and beamlineEPU.db) and the ID server software (INSERTION_SERVER__DEFINE.PRO and MOVER__DEFINE.PRO). The owner state PVs BLxx::GapOwnerState and BLxx:EPUOwnerState can have the following values:

String	enum	Initiated by	Managed by
No owner	0	nobody	nobody
SPEAR ops	1	operator EDM panel	ID_SERVER
Remote	2	remote beamline user	ID_SERVER
Local	3	IDL standalone app	IDL standalone app
Enter safe mode	4	ID_SERVER (BL12-2 injection)	ID_SERVER
Move back	5	ID_SERVER (restore BL12-2 gap)	ID_SERVER
Leave safe mode	6	ID_SERVER (restore BL12 owner)	ID_SERVER

In state 0, the ID_SERVER does not listen for any requests from that motor. This is the default state which the databases revert to when all ownership is relinquished. In state 1, ID_SERVER listens only for operator requests and in state 2 listens only for beamline user ("remote") requests. State 3 means that the motor is owned by the IDL standalone operator application which manages the motor through its GUI. (Its string is named "Local" for historical reasons, as it was the implemented before the ID_SERVER was built.) State 4 is only entered when the external CURMON (current monitor process running in a VMS process) senses a beam dump and asks the ID_SERVER to open the ID gap ("safe mode") so that the device is safe for a subsequent injection from zero. It is now owned by "Safe Mode". State 5 is entered when CURMON senses that the stored current after the beam dump is sufficient to allow the gap to safely return to the value it had before the beam dump. Finally, when CURMON senses that the operator has put the ring into BEAMS mode, state 6 restores the ownership and ownership state of the device. Currently, states 4-6 are only needed for the BL12-2 in-vacuum undulator but the mechanism will work for any ID that needs a safe mode.

- EPU touchup
- Radiation damage

V:\AD\SPEAR3\machine\spear3\ids\EPU_encoder_failures.ppt

IOC software

The application top directory for the motor software (IDs and SPEAR scrapers) is $EPICS_APP/Motor/prod/motorApp (/afs/slac/g/spear/epics/app/Motor/prod/motorApp).

Some relevant files in the listed directories are:

- **motorApp/src**
  - **bpt<corrector trim>.dbd** – breakpoint table for feedforward table for corrector magnet current (A) as a function of ID gap (mm).
  - **bptBL<id>.dbd** – breakpoint table for ID maximum field (kG) as a function of ID gap.
  - **menuConvert.dbd** – index of breakpoint tables.
  - **subBL.c** – source for beamline and ID subroutines.

- **motorApp /src/O.RTEMS-svgm**
  - **motor.obj** – image file.
    Currently, the application needs to run in the SVGM CPU because of motor record incompatibility with our preferred RTEMS MVME6100 CPU.

- **motorApp/Db/*.db** – EPICS database files
  - **beamline.db** – bending magnet and ID beamline properties
  - **beamlineAll.db** – bend magnet beamline magnetic field calculation
  - **beamlineEPU.db –** BL13 EPU phase motor ownership and requests
  - **beamlineEPUTest.db** - BL13 EPU phase motor testing PVs
  - **beamlineGap.db** – bending magnet and ID gap status
  - **beamlineGapControl.db** – ID gap motor ownership and requests
  - **beamlineI0.db** – beamline Izero signals (unrelated to IDs)
  - **beamlineIDTrim.db** - calculate ID corrector or trim magnet current from gap. The current <$(COR):CurrSetptTrim> is calculated from the breakpoint tables and is currently only used for EDM trim table plots
  - **beamlineIDTrimTable.db –** set trim table arrays from breakpoint tables
  - **beamlineMode.db** – BL5 mode motor ownership and requests. Will be obsolete when new BL5 EPU is installed.
  - **beamlineTest.db** – ID gap motor testing PVs
  - **bl12-2.db** – BL12-2 VME motor status and control
  - **bl13epu_motors.db –** BL13 EPU compound phase motor status and control
  - **bl13epu_realmotors.db** – BL13 EPU VME phase motors status and control
  - **bl13gap_motors.db** – BL13 gap compound motor status and control
  - **bl13epu_realmotors.db** – BL13 gap VME motors status and control
  - **scraper.db –** SPEAR scrapers VME motor status and control

- **motorApp/Db/*.substitutions** – EPICS template file sources
  - **beamlineProd.substitutions –** beamline status and control instantiation
  - **beamlineTestAll.substitutions** – ID gap motor testing PVs for 3 IDs

    - **bl12-2Prod.substitutions** - BL12-2 ID instantiation
    - **bl13Prod.substitutions** - BL13 ID instantiation
    - **beamlineEPUTestAll.substitutions –** BL13 EPU phase motor testing PVs
    - **scraperProd.substitutions –** SPEAR scraper instantiation

- **iocBoot/b116-iocmotor/st.cmd** – startup for BL12
  /afs/slac/g/spear/epics/app/Motor/prod/iocBoot/b116-iocmotor
    - loads **bl12-2Prod.db, beamlineProd.db, beamlineTestAll.db**

- **iocBoot/b131-iocmotor1/st.cmd** – startup for BL13
  /afs/slac/g/spear/epics/app/Motor/prod/iocBoot/b131-iocmotor1
    - loads **bl13Prod.db, beamlineEPUTestAll.db**

- **iocBoot/b132-iocmotor/st.cmd** – startup for SPEAR scraper
  /afs/slac/g/spear/epics/app/Motor/prod/iocBoot/b132-iocmotor
    - loads **scraperProd.db**

IDL software

IDL is used in two ways for insertion device control software:

- ID_SERVER, a daemon that listens for channel access requests and manages ownership, gap and phase changes. This process is required for ID control (except for direct control of the VME motor records through the diagnostic EDM panels).
- A standalone application which was the original way that IDL was used. This application currently has some unique functionality, but is not needed for simple ID gap and phase changes.

• ID_SERVER overview

ID_SERVER runs on CTRLBATCH3, an execution batch queue running on the OpenVMS computer SPEAR3, and performs these functions:

- Allocates control of the devices to SPEAR operations or authorized beamline users as appropriate. It listens for requests from the operators and remote users by monitoring PVs that are set by either the operator displays or other CA clients maintained by the beamline users. It keeps track of the current owner of all the IDs and will only allow the current owner to move the device. Ownership of a device does NOT automatically revert to the SPEAR operator when the SPEAR state drops down to AP or Down.
- Controls ID trim magnets and skew correctors using feedforward tables. It monitors the state of the Fast Orbit Feedback and modifies its trim tracking and MCOR initialization accordingly. It assures that the ID trims and correctors involved in trim tracking save their state so that they can be restored or standardized to a known configuration.
- Logs ID operations to the SPEAR event log and the OpenVMS SPEAR3 logfile SPEAR_LOG:INSERTIONS.LOG.
- Deals with many possible fault conditions.

• IDL standalone diagnostic application overview

The standalone application run on the VMS host and can be accessed from the red "X" on the SPEAR Windows console PCs. This display provides a similar interface to that of the EDM displays and also includes the following seldom used functions:

- View the log entries in a message widget on the Main Panel or mover window.
- View or control the ID by encoder ticks or peak magnetic field.
- View the motor permit state.
- View or change the trim track state.
- View the trim and corrector readbacks and alarm state.
- Control all the ID trim magnets simultaneously to ramp them to zero, to the trim table, or to standardize them.
- Control all the ID gaps simultaneously to move them out or in to their software limits, or back to their starting gaps.

- ### IDL files

Starting VMS IDL controls

- ### Production controls

  The following command starts the ID server on batch queue CTRLBATCH3:
  @SPEAR_ROOT:[INIT]ID_SERVER

  This command is automatically executed as part of SPEAR3 startup and should not normally need to be entered. The symbol ID_SERVER can also execute this command.

  The following command starts the standalone ID diagnostic display on SPEAR3:
  @IDL_USER:[INSERTIONS]LAUNCH_INSERTIONS

  There is an icon on the SPEAR Windows Consoles labeled "Insertion Device Control" which makes an ssh connection to SPEAR3 and executes this command.

- ### Debugging and development controls

  The following command starts an ID test server on batch queue CTRLBATCH3:
  @SPEAR_ROOT:[INIT]ID_SERVER SUBMIT TEST

  This command starts a debugging version of the ID standalone display on SPEAR3:
  @IDL_USER:[INSERTIONS.DEVELOPMENT]LAUNCH_INSERTIONS

  This software uses test PV's and doesn't allow control of real hardware. It is useful for debugging server and client logic without interfering with production software.

- ### Initial software parameters

Parameter	Production software	Development software
Server name	ID_SERVER	ID_SERVER_TEST
Default directory	IDL_USER:[INSERTIONS]	IDL_USER:[INSERTIONS.DEVELOPMENT]
Logfile locations	SPEAR_LOG:	IDL_USER:[INSERTIONS]
Server logfile name	ID_SERVER.LOG	
Client logfile name	INSERTIONS.LOG	

Hardware and software of the future

http://www.slac.stanford.edu/~spear/epics/app/ID/SSRL_future_beamlines.pdf

Eliminate CAMAC controls

Use Pro-Dex VME controller for motors and digital IP inputs

Eliminate IDL

Functionality needed:

Deal with all trims and each trim individually

Deal with all gaps

Display and move gaps by field strength and encoder

Ramp tables for quads to reduce emittance coupling

Save/restore all gaps and owners when enter AP mode/return to beams mode

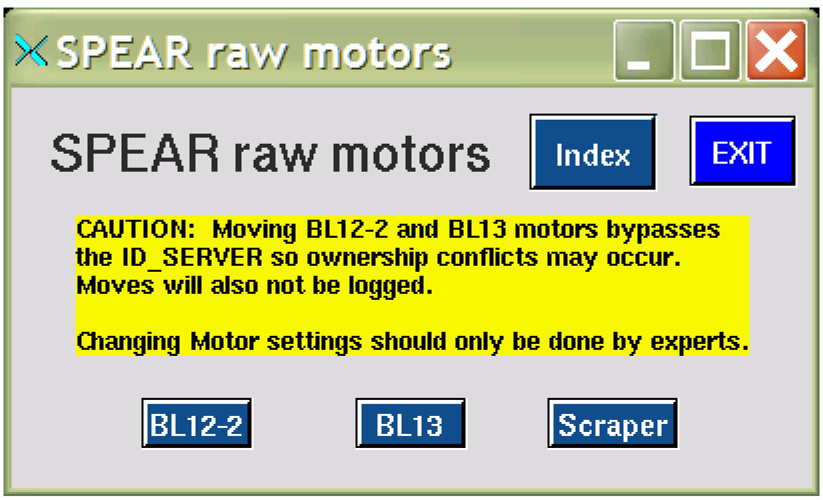
Displays

GUIs for insertion device status and control come in three general flavors:

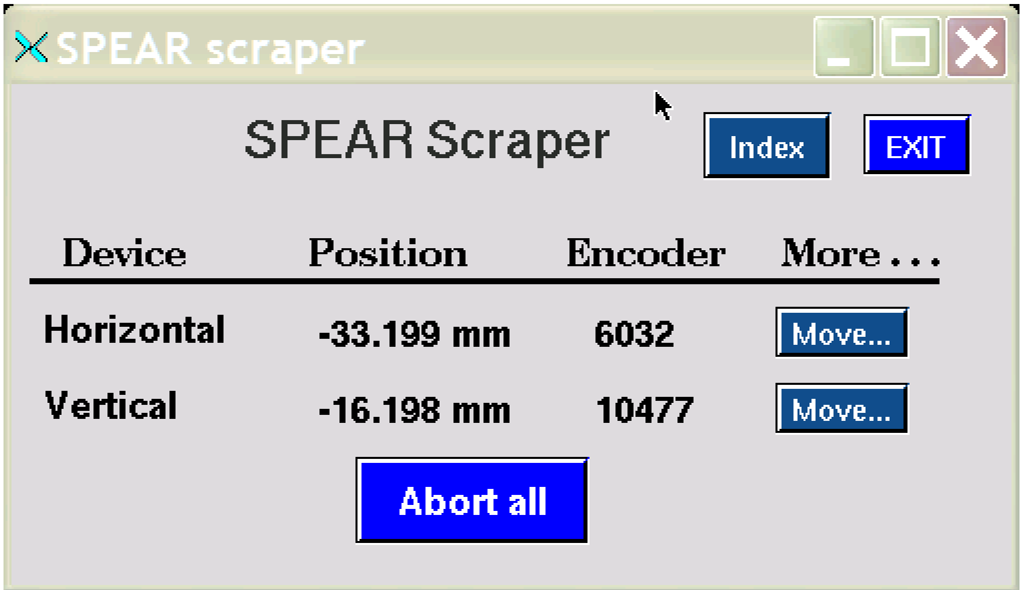
- Engineering EDM displays meant for experts to aid in debugging
- Operational EDM displays meant for SPEAR operators to control IDs
- Diagnostic displays written in IDL for functionality beyond the EDM displays
    - The functionality of these displays should be ported to EDM or some more maintainable GUI

Engineering displays

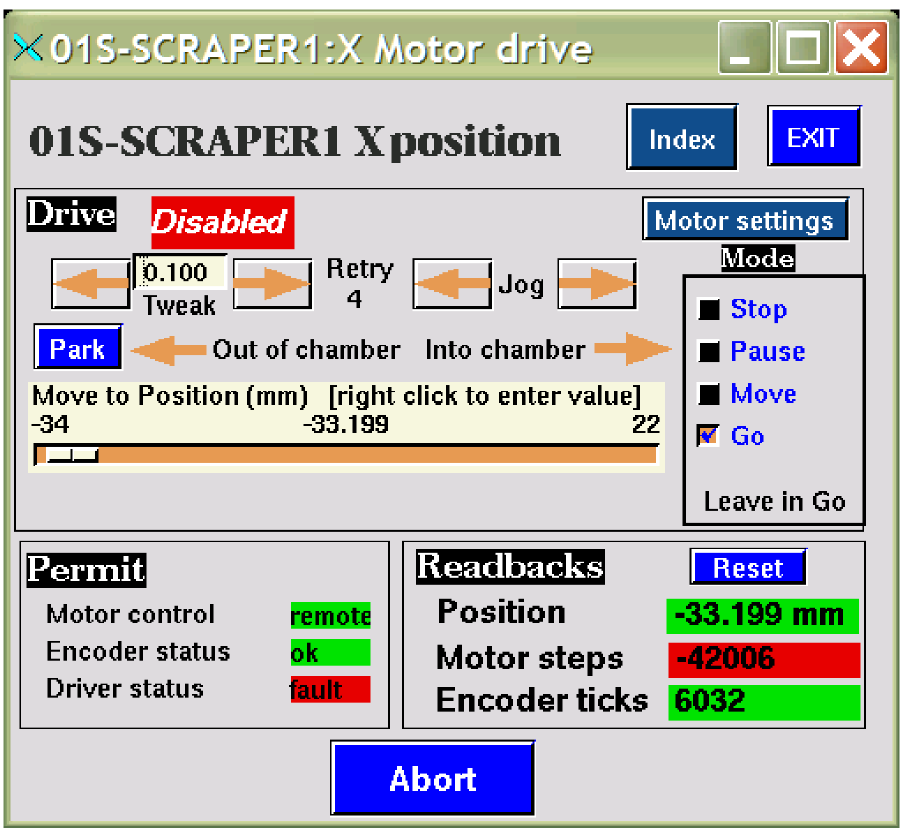
$EDM/motors.edl – portal for motor displays that talk directly to the EPICS motor record:



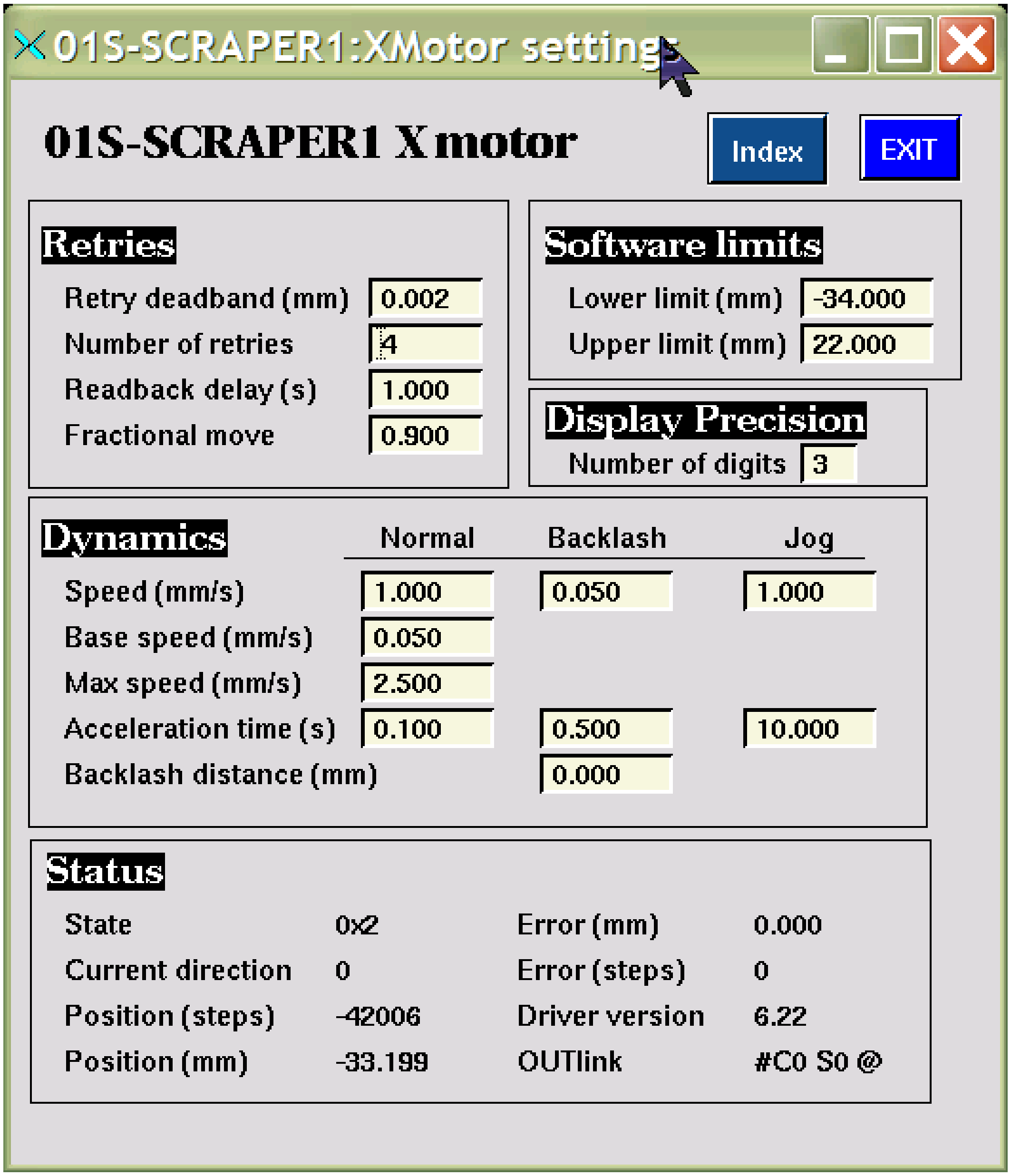
$EDM/scraper.edl – overview for SPEAR scraper motor control:



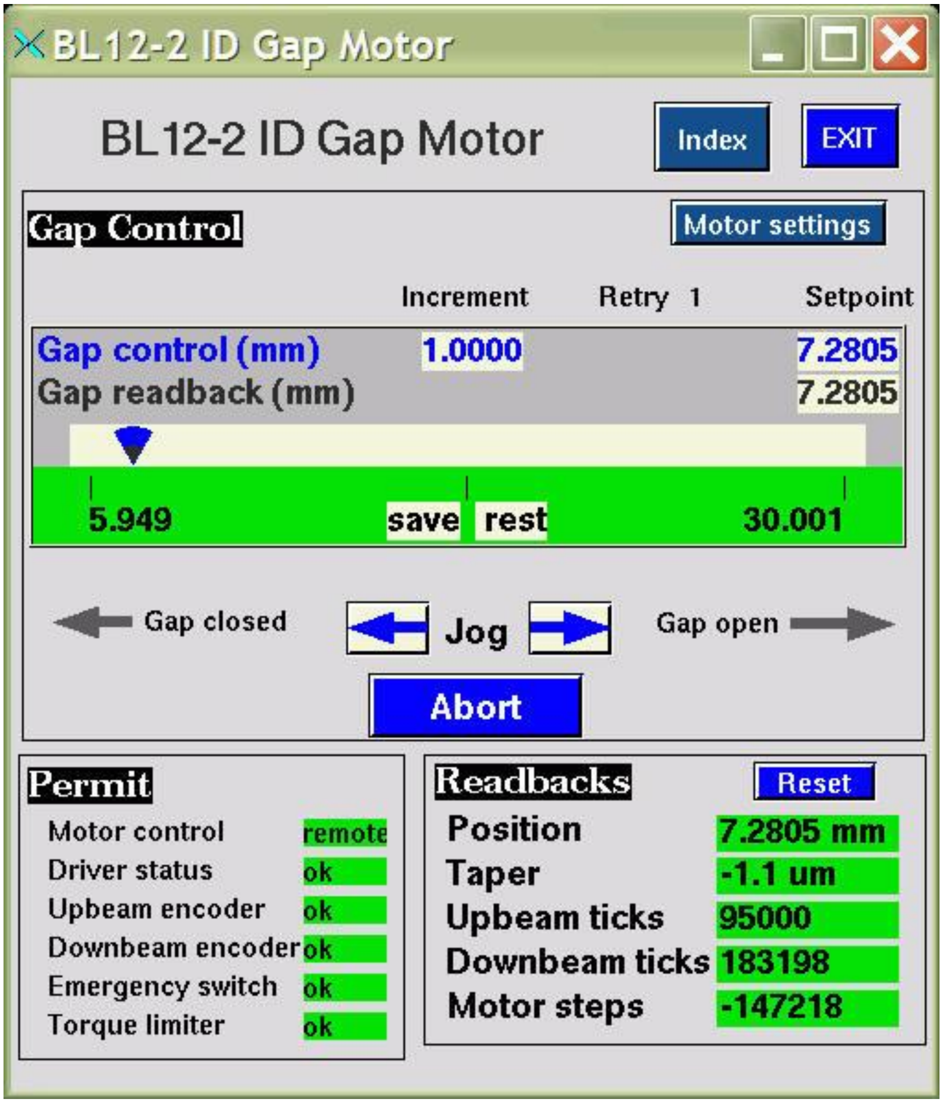
$EDM/motor_drive.edl - motor control for a scraper axis:



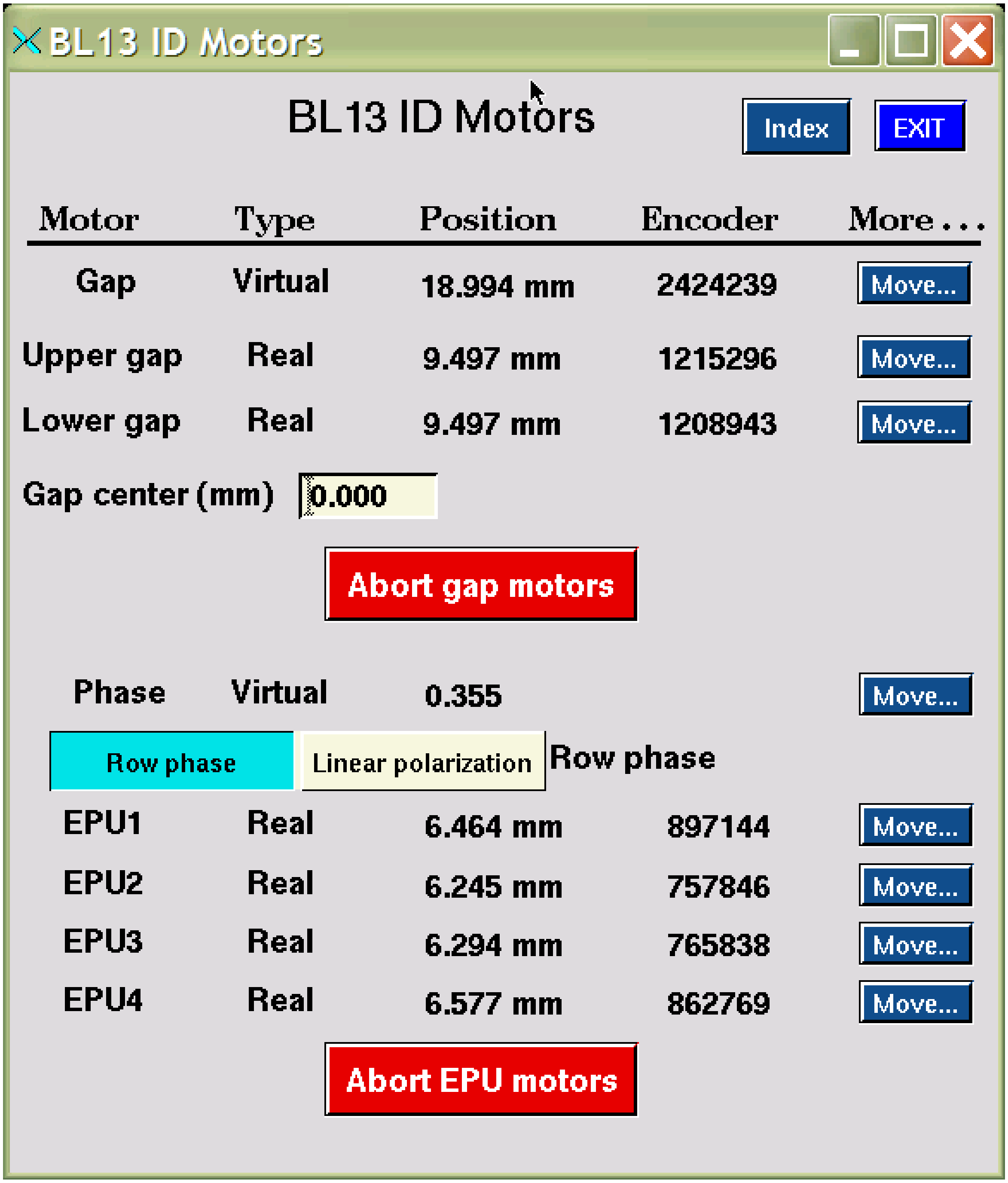
$EDM/motor_settings.edl - access for some motor record fields:



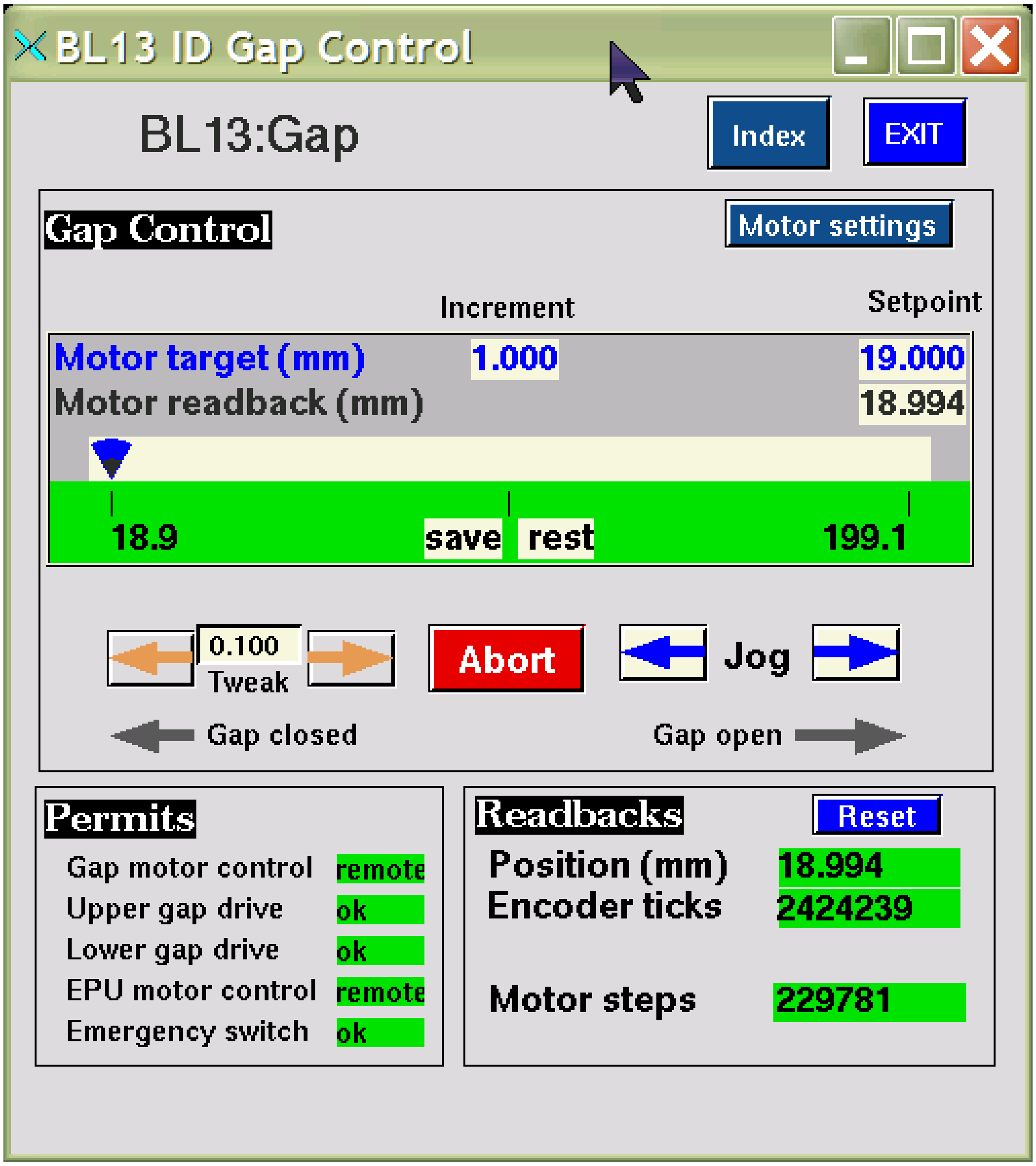
$EDM/bl12-2.edl – control of BL12-2 ID gap:



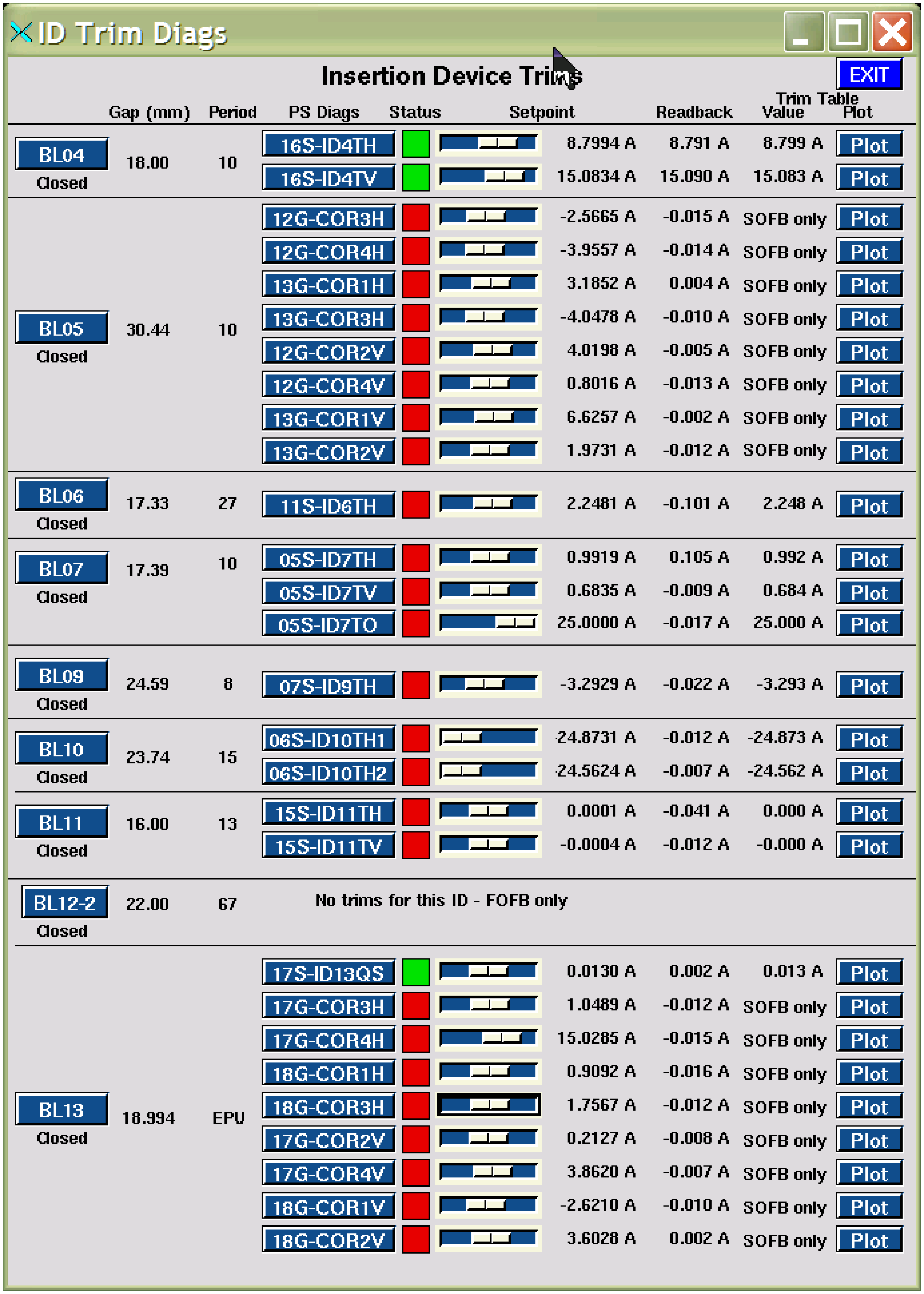
$EDM/bl13.edl – overview of BL13 motors (virtual and real):



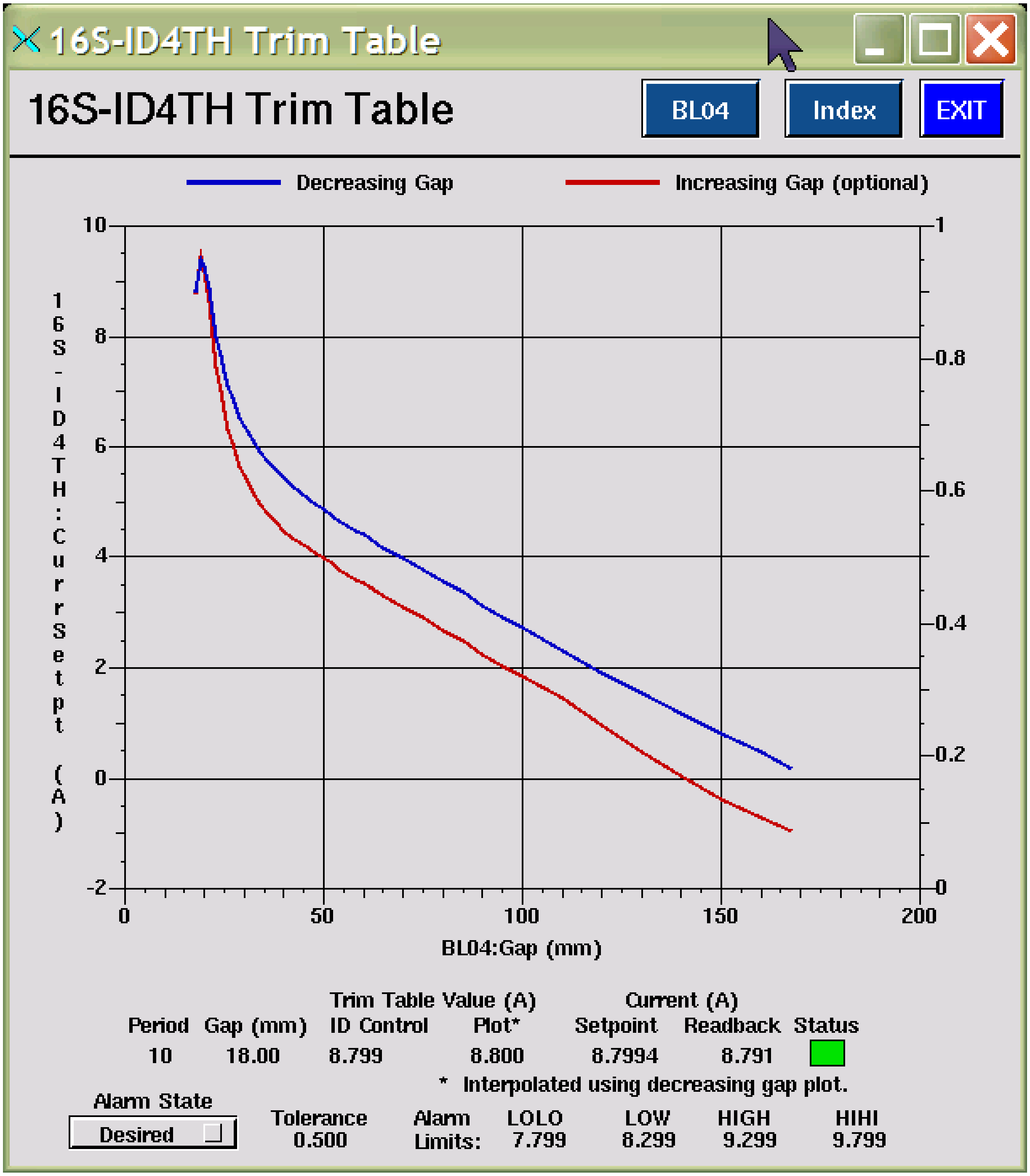
$EDM/bl13gapmotors.edl – control of BL13 gap virtual motor:



$EDM/diagnostics/id_diagnostics.edl - ID trims control (not through ID_SERVER):

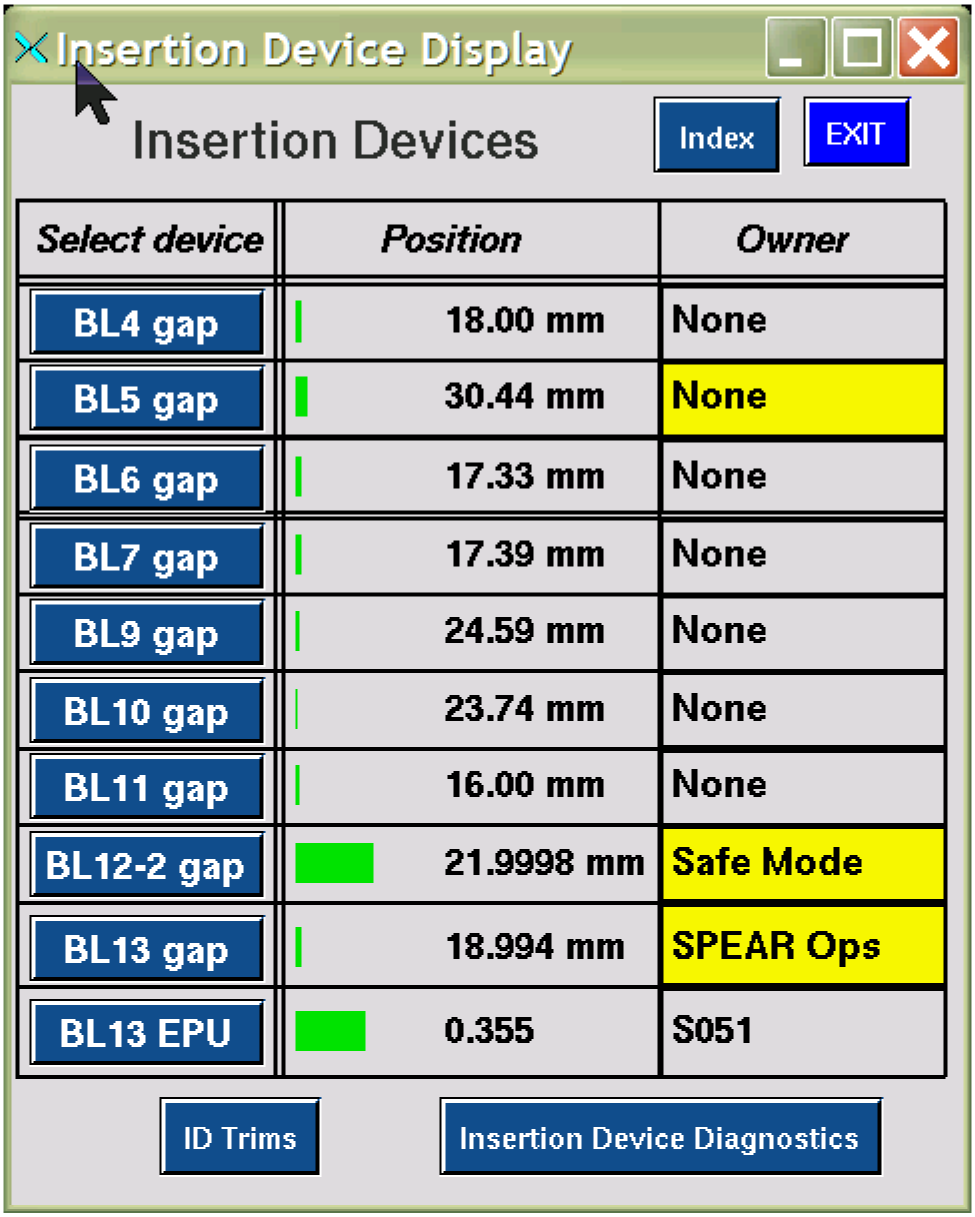


$EDM/diagnostics/id_trim_table.edl – example of an ID trim table display (not through ID_SERVER):

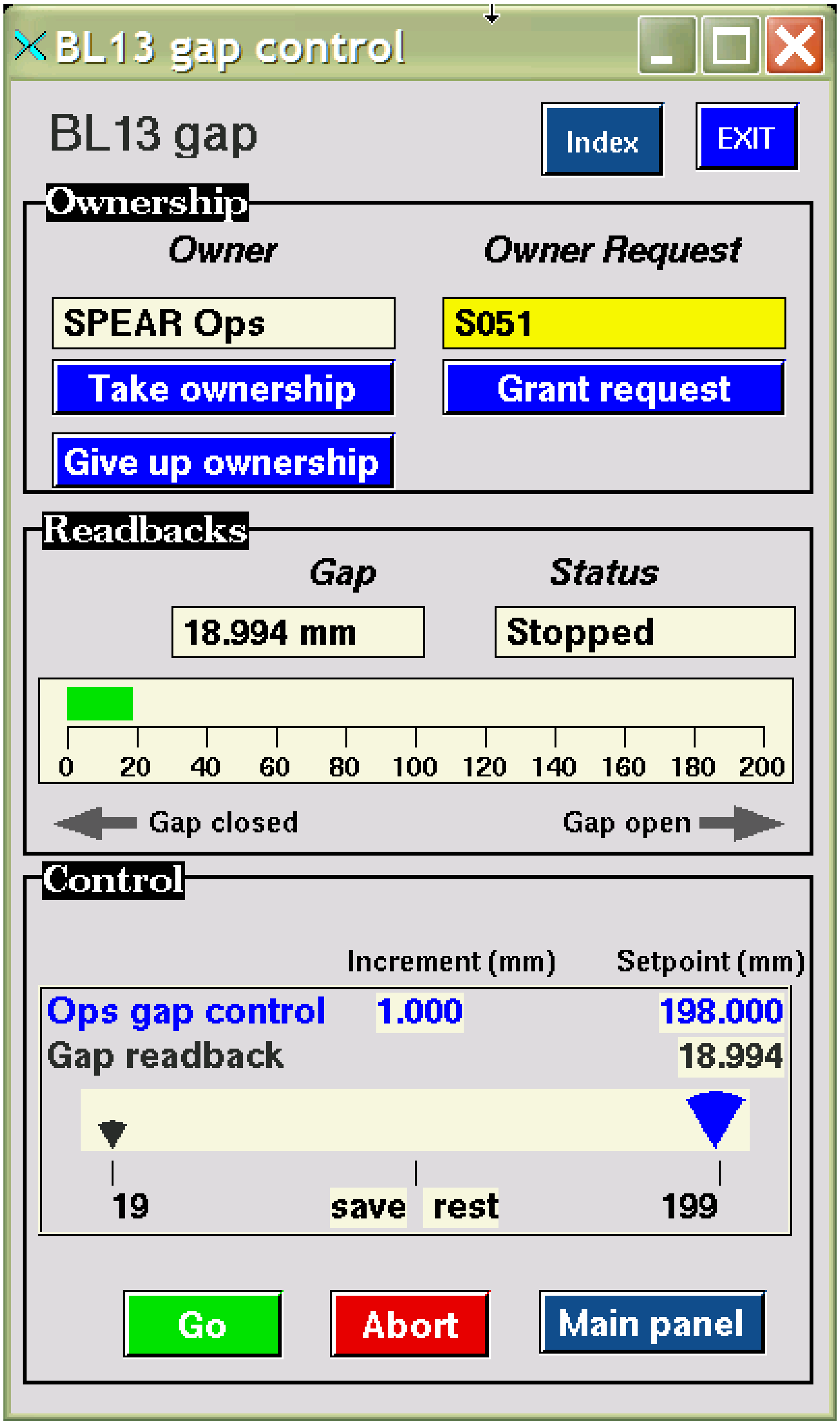


- Operational EDM displays

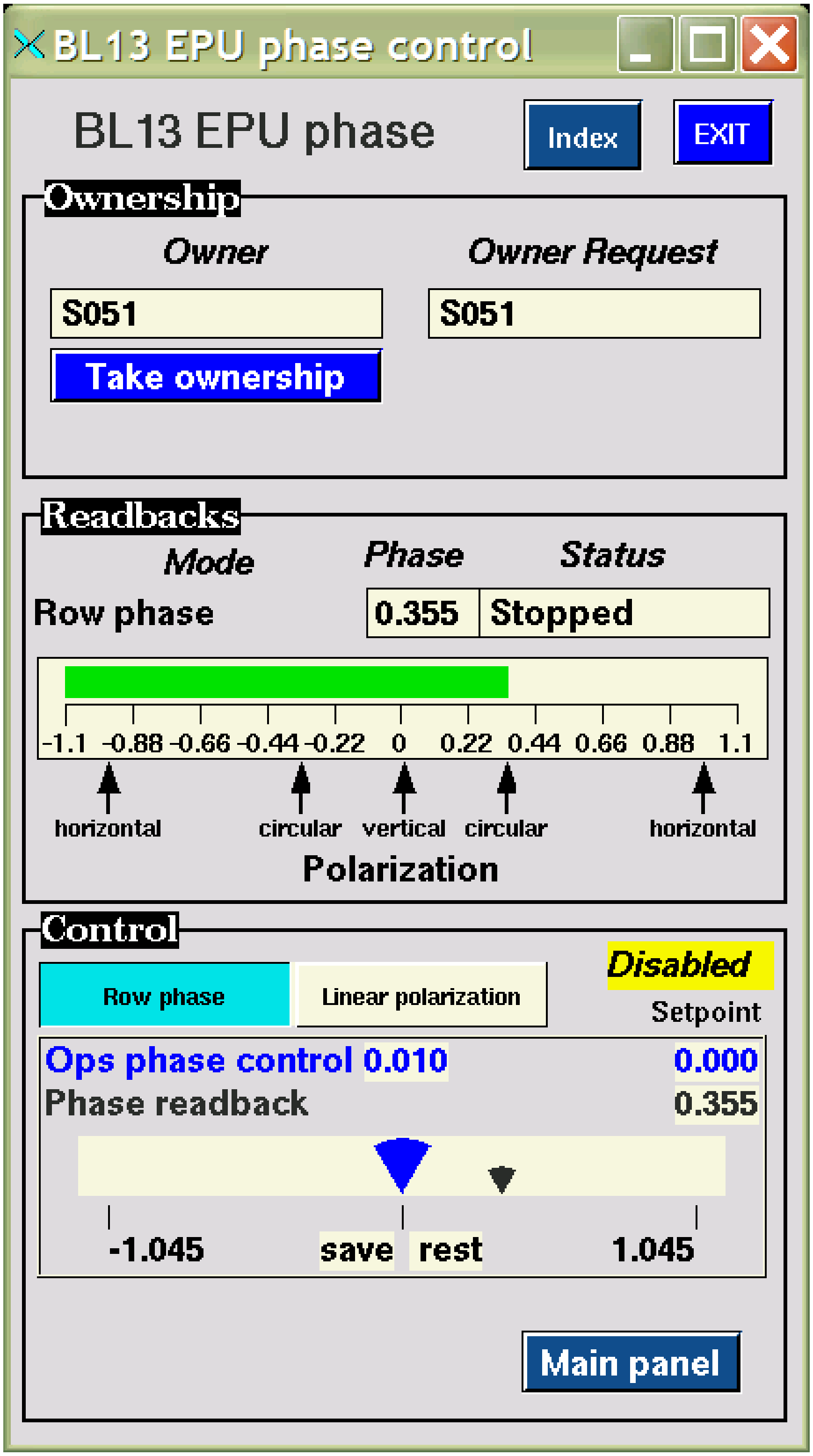
$EDM/id_display.edl – “main panel” portal for motor displays that access ID_SERVER has Related Displays which launch individual “mover windows”. If a remote user has made an ownership request which has not yet been granted, the background of the Owner cell will be yellow, the Minor Alarm color.



$EDM/id_gap_control.edl – "mover window" operator interface to ID gap control. The "Go" button must be hit before the device will actually move.

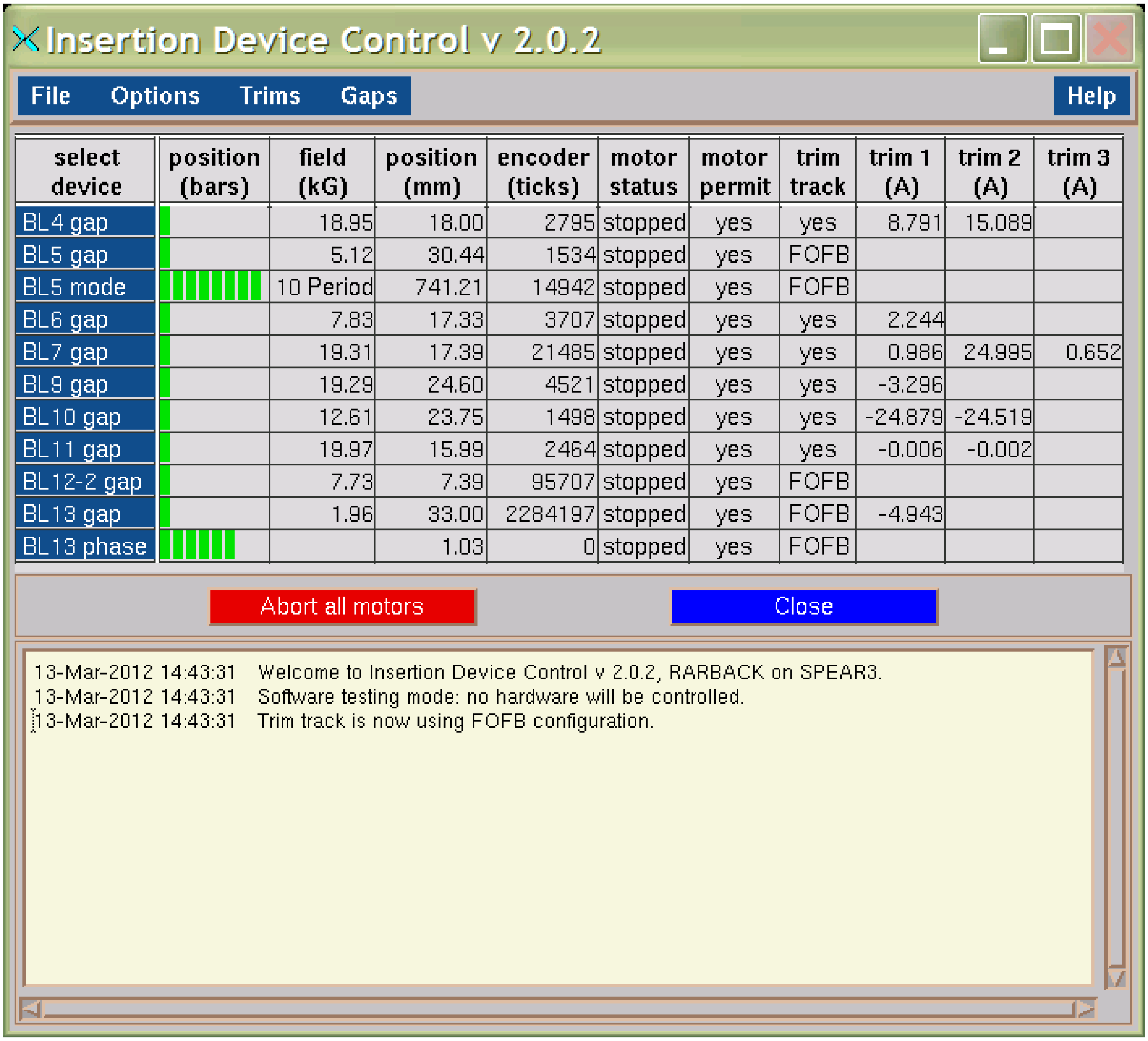


$EDM/id_phase_control.edl - operator interface to EPU phase control:



- Diagnostic displays written in IDL

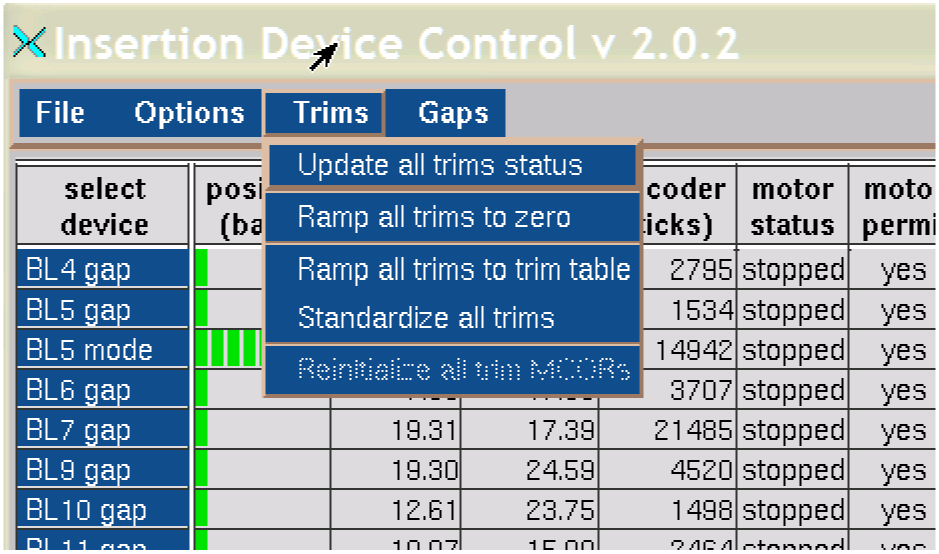
Main Panel for ID control with standalone application:



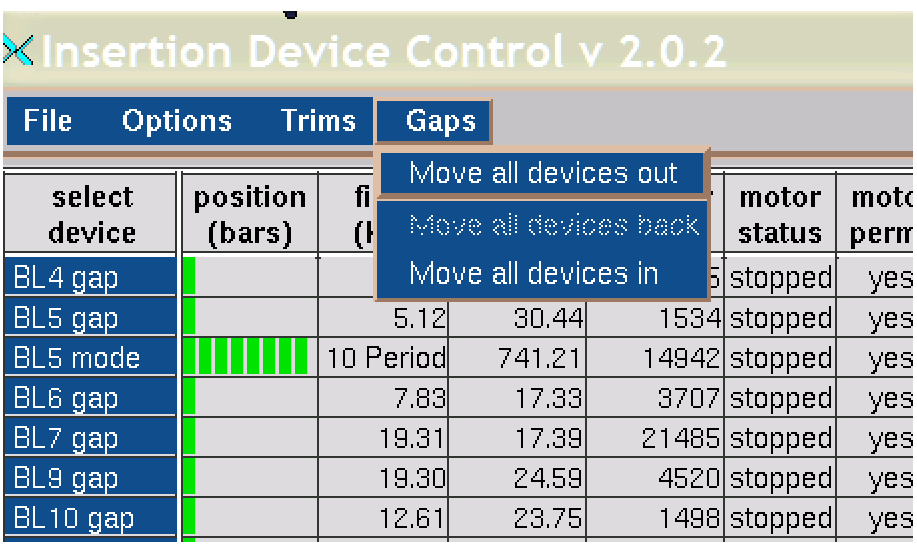
select device	position (bars)	field (kG)	position (mm)	encoder (ticks)	motor status	motor permit	trim track	trim 1 (A)	trim 2 (A)	trim 3 (A)
BL4 gap		18.95	18.00	2795	stopped	yes	yes	8.791	15.089	
BL5 gap		5.12	30.44	1534	stopped	yes	FOFB			
BL5 mode		10 Period	741.21	14942	stopped	yes	FOFB			
BL6 gap		7.83	17.33	3707	stopped	yes	yes	2.244		
BL7 gap		19.31	17.39	21485	stopped	yes	yes	0.986	24.995	0.652
BL9 gap		19.29	24.60	4521	stopped	yes	yes	-3.296		
BL10 gap		12.61	23.75	1498	stopped	yes	yes	-24.879	-24.519	
BL11 gap		19.97	15.99	2464	stopped	yes	yes	-0.006	-0.002	
BL12-2 gap		7.73	7.39	95707	stopped	yes	FOFB			
BL13 gap		1.96	33.00	2284197	stopped	yes	FOFB	-4.943		
BL13 phase			1.03	0	stopped	yes	FOFB			

-

ID trims control for all trims with standalone application:



ID gap control for all gaps with standalone application:



ID gap control with standalone application: