

SW Control of NICI mechanisms

1. Introduction

This document will discuss the software that controls and monitors the servo-mechanical devices that move filters, gratings, mirrors and other wheels. These mechanical elements are needed to control the light that finally arrives at the imaging device. This document describes these elements, the software modules involved and the control of those software modules.

2. Requirements

The NICI Statement of Work requires that each of these mechanisms be easily, safely and reliably controlled. It requires that each mechanism be brought into a known or “home” position. And finally, it requires that the position be remembered from moment to moment during operation.

3. Mechanisms not described in this document:

The deformable optics are controlled by the Adaptive Optics system which has its own mechanism structure, and is therefore not addressed in this document.

4. Safety and Interaction of mechanisms

There are no known safety issues involved with any of these mechanisms. All moving parts are internal to the external case. The light sensitive image elements are always behind some protective filtering position.

However, for efficient use of NICI, overexposing the Aladdin III array needs to be avoided, therefore, the use of the Tip/Tilt Steering Mirror and the Pupil Mask Wheel must be done in concert when used in a dithering mode. This interaction is explained here:

The steering mirror provides indirect control of the image by altering the position of the image as seen by the Adaptive optics. It can be thought of as being controlled by three separate inputs: The atmospheric wavelength correction, the displacement for dithering, and the correction for possible instrument flexure. The desired position of the steering mirror is the vector sum of these inputs. The displacement for dithering is not a simple input for the steering mirror only: it affects the position of the Pupil Mask Wheel (which must change position along with the star offset to avoid overexposing the image array)

A virtual mechanism, the Digital Offset (DOFF) will be created to affect this higher level usage. The DOFF is a command object but does not correspond to a single real mechanism. The DOFF will be described below, after the discussion of the real mechanism controllers.

5. Real Mechanism Control Objects

The NICI instrument has the following real mechanisms:

Internal name	NICI Mechanism
FOCS	Fiber Optic Calibration Source
TTSM	Tip/Tilt Steering Mirror
NDFW	Neutral Density Filter Wheel
FPMW	Focal Plane Mask Wheel
SMR	Spider Mask Rotator
PMW	Pupil Mask Wheel
DW	Dichroic (Beam Splitter) Wheel
C1FW	Channel 1 Filter Wheel
C2FW	Channel 2 Filter Wheel
PI	Pupil Imager

Table 1-- Command Objects for Real Mechanisms

The Tip/Tilt Steering Mirror (TTSM) will be controlled by a linear voice coil mechanism. The analog control voltages will be generated by an Ethernet controllable DAC. The position of the mirror will be also monitored by an Ethernet accessible ADC. From a software standpoint, the position is set by sending a properly formatted message to control the position of the mirror. The updates to the mirror are expected to be small and at a cadence of no faster than 10 seconds. The command object (software interface) for the TTSM will simply re-format the position, and pass it through to the Ethernet controlled DAC and read the resulting position of the mirror back through the ADC. It follows the Animatics interface (described below) so closely that it will appear to the user of NICI as simply another of the mechanisms described below.

The adaptive optics filter wheel (NDFW) will be an off-the-shelf device that responds to a BCD (binary coded decimal) position request. The interface will be through an Ethernet to digital output only device to cause the device to turn. There is no feedback on this wheel so software will not sense its position but will calculate the time of rotation and respond with the completion code to the user only after this time has elapsed. Otherwise it will appear to the user of NICI as simply another of the mechanisms described below.

The remaining mechanisms will be driven by Animatics Smartmotors. Smartmotor control for MKIR instruments is described in the appendix "Mechanism Control by use of Animatics Smartmotors in MKIR instruments." This note describes in generic terms the operation of four different kinds of mechanism: linear, rotary, continuous and discrete. Within this framework each real mechanism of the NICI instrument becomes a "command object" as described in that document.

The kinds of mechanisms and the abbreviations are:

- LC: Linear Continuous
- LD: Linear Discrete
- RC: Rotary Continuous
- RD: Rotary Discrete

Rev 1.5

The following table describes the Command Object name and the important attributes involved in software control of the instrument.

Table 2 -- Command Object for Controllable Mechanisms

NICI mechanism	Kind and positions	Position Names	Home Position
FOCS	LD:4	Single, grid, corner, open	Single
TTSM	LC: 2 axis	-100 to 100, -100 to 100	0,0
NDFW	RD:6	ND2,ND3,ND4,RED,OFF,OPEN	Off
FPMW	RC	0-360	0
SMR	RC	0-360	0
PMW	RD:8	Blank, Open,2,3,4,5,6,7	Blank
DW	RD:15	H/K, L/M, K-Methane, L-Methane, 50/50 Short, 50/50 Long, Hole, Mirror, Methane, [Fell], H2 1-0 s(1),Bracket Gamma	
C1FW	RD:22	OFF,J,H,K',L',M',H methane On, H Methane Off, K Methane On, K Methane Off, L Methane On, L Methane Off, J age Filter 1, J age filter 2, [Fell], [Fell]cont.,H2 1-0 s(1),Br gamma, NB Continuum, Grism 1, Grism 2	OFF
C2FW	RD:22	OFF,J,H,K',L',M',H methane on, H Methane Off, K Methane On, K Methane Off, L Methane On, L Methane Off, J age Filter 1, J age filter 2, [Fell], [Fell]cont.,H2 1-0 s(1),Br gamma, NB Continuum, Grism 1, Grism 2	OFF
PI	RD:3	Open, Blank, Pupil	Open
DOFF	Virtual	0 - 10	Null

Each kind of real mechanism is implemented by one of four generic mechanism drivers. The differences among any two discrete rotary mechanisms are summed up by the number of positions, the names of the positions, the home position and the hardware address of the physical connection to the CPU of the NICI controller. These differences are used as parameters to create instances (objects) from the generic classes that define the four mechanism types.

6. Command Summary:

This list summarizes the command syntax for each of the real mechanisms.

6.1. Test:

Every command object in the NICI system supports a "test" variant. This will parse the command for spelling and check if the command could be performed in the current state of the system. No movement or other state changes are done by the test variant.

Example:

```
FOCS TEST MOVE "Grism 1"
```

Would respond:

```
FOCS {TEST MOVE "Grism 1"} {No position "Grism 1"} reject
```

Rev 1.5

6.2. Status:

Every command object in the NICI system supports a "cget" variant. This will report on internal state. The command objects for mechanisms all have the `position` attribute. No movement or other state changes are done by the `cget` command. Example:

```
C1FW cget -position
Would reply
C1FW {cget -position} {K Methane Off} accept
```

6.3. Movement:

To cause the mechanism to position itself:

```
<CO> move <position>
Where <CO> is the command object and <position> is the new position requested.
```

```
Fiber Optic Calibration source
FOCS move grid
Would respond:
FOCS {move grid} accept
```

Further messages are possible, for example, if the mechanism moved through the "open" and "corner" positions, the FOCS would respond with intermediate positional information:

```
FOCS position open transient
```

Followed shortly by

```
FOCS position corner transient
```

And finally,

```
FOCS position grid stable
```

These status messages are broadcast to all user connections (ex. Engineering gui and thin-client). These status messages are similar for all of the mechanisms and will not be mentioned in the summaries below.

6.3.1. FOCS

```
FOCS move in
FOCS move out
FOCS move grid
FOCS move corner
```

The fiber optic calibration system will report its position as above.

6.3.2. TTSM

```
TTSM move X Y
```

Where X and Y are floating point values between -10 and 10. For example:

```
TTSM move 1.23 -3.545
```

The tip/tilt mechanism will report its position as above.

6.3.3. NDFW

```
NDFW move ND2
NDFW move ND3
NDFW move ND4
NDFW move RED
```

Rev 1.5

NDFW move OPEN

NDFW move OFF

The Neutral density Adaptive optics filter wheel will report the final position on completion.

6.3.4. FPMW

FPMW move <position>

Where position is an rotational position in degrees.

Example

FPMW move 45.2

The focal plane mask wheel will report the final position on completion.

6.3.5. SMR

SMR move <position>

Where position is an rotational position in degrees.

Example

SMR move 45.2

The focal plane mask wheel will report the final position on completion.

6.3.6. PMW

PMW move BLANK

PMW move OPEN

PMW move 2

PMW move 3

PMW move 4

PMW move 5

PMW move 6

PMW move 7

The pupil mask wheel will report intermediate positions as described above.

6.3.7. DW

DW move "<position>"

Where <position> can be any one of: H/K, L/M, K-Methane, L-Methane, 50/50 Short, 50/50 Long, Hole, Mirror, Methane, [FeII], H2 1-0 s(1), Bracket Gamma. For example:

DW move "50/50 Long"

The dichroic beam splitter wheel will report intermediate positions as described above.

6.3.8. C1FW

C1FW move "<position>"

Where <position> can be any one of: OFF, J, H, K', L', M', H methane On, H Methane Off, K Methane On, K Methane Off, L Methane On, L Methane Off, J age Filter 1, J age filter 2, [FeII], [FeII]cont., H2 1-0 s(1), Br gamma, NB Continuum, Grism 1, Grism 2. For example:

C1FW move "K Methane Off"

The channel 1 filter wheel will report intermediate positions as described above.

6.3.9. C2FW

C2FW move "<position>"

Where <position> can be any one of: OFF, J, H, K', L', M', H methane On, H Methane Off, K Methane On, K Methane Off, L Methane On, L Methane Off, J age Filter 1, J age filter 2, [FeII], [FeII]cont., H2 1-0 s(1), Br gamma, NB Continuum, Grism 1, Grism 2. For example:

C2FW move "K Methane Off"

The channel 2 filter wheel will report intermediate positions as described above.

6.3.10. DOFF -- Digital Offset Command Object

The virtual command object Digital Offset will be used during normal NICI operation. The Instrument sequencer (IS) will compute information about the position of the telescope, relative rotation of the sky, and position of the instrument as these data items are contained in a database directly accessible to the IS, but not to the other portions of the NICI instrument. The IS will pass needed corrections to the instrument controller (IC) as commands to the Digital Offset (DOFF) command object. The DOFF will contain variables corresponding to the dithering offset and the compensation which may be read back in the standard way by:

```
DOFF cget -offset  
And  
DOFF cget -compensation
```

If the sum of the compensation and the dithering offset would exceed the allowable range of the Tip/Tilt Steering Mirror, the command will be rejected with an appropriate message.

The command summary will be:

```
DOFF NULL
```

The current position of the Pupil Mask Wheel and tip/tilt mirror (and hence the star on the imaging array) will be considered to be in the primary position. This position will be the origin for dithering offsets.

```
DOFF moveto <position>
```

The TTSM and the pupil mask wheel PMW are moved together to move the image of the star to <position> relative to the NULL position of the image array. The position is a floating point number.

```
DOFF move <position>
```

The TTSM and the pupil mask wheel PMW are moved together to move the image of the star to <position> relative to its current position on the image array. The position is a floating point number.

```
DOFF compensate <x> <y>
```

The correction of the star position also depends on the atmospheric wavelength correction (since the AO steering is done by visible light and the imaging array is sensitive to infrared), as well as the instrument flexure. Both of these depend on information known to a database accessible to the IS. The IS will precompute the vector sum of the compensation required. That compensation will be added current dithering position (vector sum) and sent to the TTSM by the Digital Offset.

The intended use of the DOFF is as follows:

- Close off optical path:
- Move DOFF to zero.
- Turn off AO control of system.
- Slew telescope to star.
- Turn on AO control of system.
- Move Pupil Mask wheel to desired position.
- Apply positional correction with DOFF compensate command
- Null the star position with DOFF NULL
- Open the optical path

The TTSM should not be controlled directly by the IS during normal operation.

7. Appendix:

Mechanism control by use of Animatics Smartmotors in MKIR instruments

7.1. Animatics smartmotor characteristics

The Animatics company creates a universal, programmable servomotor with an embedded 8051 processor and serial (RS232) connections. The smartmotor not only senses its position internally, it responds to specific commands, and also allows the downloading of a sequence of commands to create a smartmotor program. This mechanism allows the smartmotor to sense its own environment and to act fully independently. This feature allows the designer to create moving systems with simple, robust and failure resistant controls. In MKIR systems, these smartmotors are attached independently to separate serial ports on a TIP (Terminal interface controller) that translates between the Internet protocol (IP) on a single Ethernet connector and multiple RS232 connectors.

In MKIR designs the concept of software objects is adapted to these entities. One object represents the communication pathway of the TIP. It can pass messages and be initialized and reset. Other objects represent the actual smartmotors that are attached to real mechanisms. During normal user mode operation the operation of the TIP will be invisible to the user.

7.2. Kinds of mechanisms and Low Level Command Interface

The kinds of objects (classes in software lingo) are as varied as the kinds of mechanisms. These are few in number and rather simple.

7.2.1. Rotary continuous

The mechanism is rotary and can rotate without stopping in either direction. There is a single preset, detectable position called the index position for each complete revolution of the mechanism. Positioning a rotary continuous mechanism is done on a degree basis (0-360), where multiples and negative rotations are possible. Essentially, the command is given to go forward or backward a number of degrees. Wrap-around is possible and expected. 360 degrees is exactly the same position of the mechanism as 0, 720 or -360 degrees. Fractional degrees are permitted. Each time the mechanism is asked to pass through the index position, the calculated position is internally recalibrated to the index position.

The Animatics motors attached to this kind of mechanism will each have exactly the same program loaded: 'rotcon.sm'. In addition to the manufacturer command set, the rotcon.sm program will enable the rotary continuous smartmotor to respond to the following commands:

7.2.1.1. Rotate to home: report how much movement is needed.

This will be done twice during reset to calculate the entire 360 rotation of the mechanism)

7.2.1.2. Rotate a positive (or negative) amount.

7.2.1.3. Report on version of SW that is present.

This is done to detect communication failures and to insure the proper program has been loaded to the smartmotor.

7.2.2. Rotary discrete

The mechanism is rotary and can rotate without stopping in either direction. There is a single detectable position called the index position for each complete revolution of the mechanism. In addition there are other positions that are detectable during the rotation of the mechanism. The index position is unique in that the distances between it and the positions on either side are distinguishable from the other distances. The key or index position does not correspond to a legal position of the wheel, but is used to detect the adjacent legal "home" position. The home and the other positions are numbered linearly from one to N where N is the total legal positions of the wheel.

The Animatics motors attached to this kind of mechanism will each have exactly the same program loaded: 'rotdis.sm'. In addition to the manufacturer command set, the rotdis.sm program will enable the rotary continuous smartmotor to respond to the following commands:

Rev 1.5

7.2.2.1. Rotate to next (or previous) detent: report how much movement is needed.

This will be done during reset to calculate the number of detents around the entire 360 rotation of the mechanism as well as find the home position. It will also be used during normal movements. To eliminate backlash, the smartmotor will rock the wheel to insure centering in the detent.

7.2.2.2. Report on version of SW that is present.

This is done to detect communication failures and to insure the proper program has been loaded to the smartmotor.

7.2.3. Linear continuous

Linear motion presents a special problem in MKIR instruments. With certain mechanisms, pushing beyond a certain position can cause damage. This damage may be expensive to repair and may result in lost observation time of several days or weeks. For this reason, the ends of the travel are important and must be absolutely reliably detected by the smartmotor. Each endpoint must be distinguishable. That is, the left end and the right end must activate different switches. This is because it is legal to go to the right from the left end stopper, but not the other way around. Deciding the legal motion must be simple and reliable. For this reason, there must be no chance that the mechanism be in a position beyond the endpoint switches (because of manual motion when the instrument is turned off, for example)

The Animatics motors attached to this kind of mechanism will each have exactly the same program loaded: 'lincon.sm'. In addition to the manufacturer command set, the lincon.sm program will enable the rotary continuous smartmotor to respond to the following commands:

7.2.3.1. Report on version of SW that is present.

This is done to detect communication failures and to insure the proper program has been loaded to the smartmotor.

7.2.3.2. Report at home.

Is the mechanism at either left or right stopper?

7.2.3.3. Go forward or reverse to home: report how much movement is needed.

The extent of the travel between the two limits is known.

7.2.3.4. Go forward or reverse an amount.

If the motion would reach either left or right limit, motion is to stop immediately and report the completion of movement. The total amount traveled is reported.

7.2.4. Linear discrete

Linear discrete has exactly the same situation as regards the left and right limits of travel, so all of the above considerations apply. In addition, there are discrete positions that are signaled to the smartmotor, which indicate resting positions. There are a number of these positions and intermediate positions are to be ignored. The Animatics motors attached to this kind of mechanism will each have exactly the same program loaded: 'lindis.sm'. In addition to the manufacturer command set, the lindis.sm program will enable the rotary continuous smartmotor to respond to the following commands:

7.2.4.1. Report on version of SW that is present.

This is done to detect communication failures and to insure the proper program has been loaded to the smartmotor.

7.2.4.2. Report at home.

Is the mechanism at either left or right stopper?

7.2.4.3. Go forward or reverse to home

report how much movement is needed.

The extent of the travel between the two limits is known.

7.2.4.4. Go forward or reverse to next position.

If the motion would reach either left or right limit, motion is to stop immediately and report the completion of movement. The total amount traveled is reported.

7.3. User level Command Object interface

During normal user operation, the smartmotors are pre-loaded with the proper firmware, and the initialization sequences have been done. Hence the software knows the position, number of possible positions, and limits for each mechanism. These operations are engineering level commands and not covered here.

The normal command objects for each of these mechanisms correspond to the kind of mechanism:

7.3.1. Rotary Continuous:

The command

```
<CO> move <degree>
```

The mechanism will move to the desired position and report when that position is reached.

The command

```
<CO> park
```

The mechanism will move to the home or index position and report when that position is reached.

The command:

```
<CO> cget -position
```

Will report the current position.

7.3.2. Linear discrete

Linear discrete is handled the same as Rotary Discrete, below.

7.3.3. Rotary Discrete

Mechanisms with this characteristic will have each of the desired positions identified with a label (corresponding to well-know filter labels). There will be N of these labels, and each will be selected by a text string identifying the position.

The command is

```
<CO> move "string"
```

If the string corresponds to a position, the command is accepted and motion will start. As the wheel moves through intermediary positions, intermediate reports of these positions will be passed to the user. The final position will be identified by the keyword "stable" at the end of the status message. Intermediate positions will be tagged as "transient" at the end of these intermediate status messages.

```
<CO> park
```

The mechanism will move to the home or index position and report when that position is reached.

The command:

```
<CO> cget -position
```

Will report the current position.

If the user needs to have a list of these filter names and their number, the command:

```
<CO> cget
```

Will report that information (as well as the current position).

7.3.4. Linear continuous:

The command is

```
<CO> move <percent>
```

Where -100 represents the lower limit and 100 represents the upper limit of travel. Any position in between will be recognized. Positions below -100 or beyond 100 are rejected.

```
<CO> park
```

The mechanism will move to the home or index position and report when that position is reached.

The command

```
<CO> cget -position
```

Will report the current position.

8. Diagrams

8.1. The following diagrams break down the software interconnections among the classes and objects driving the various kinds of mechanisms.

Figure 1 - Rotary Discrete

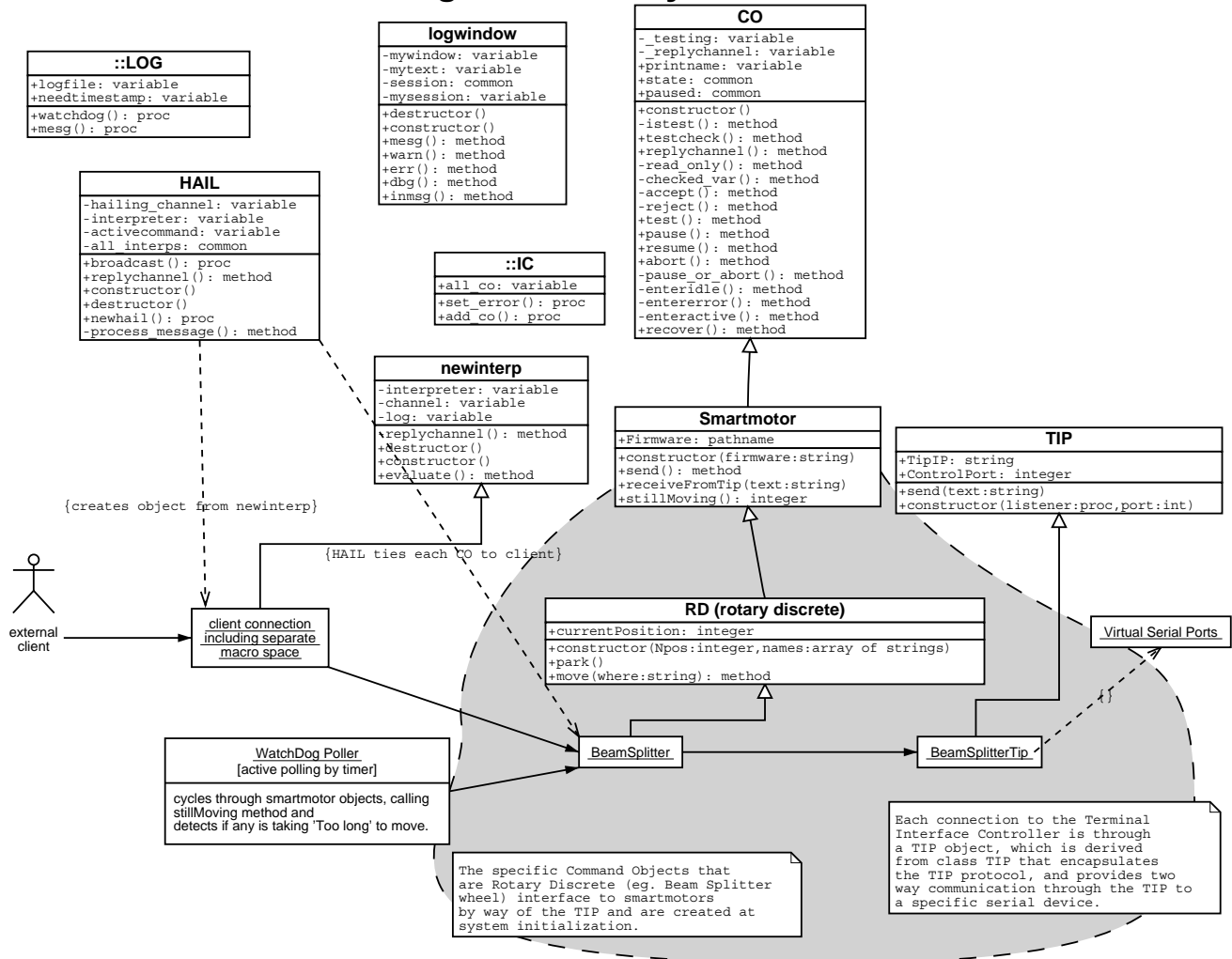


Figure 3 - Rotary Continuous

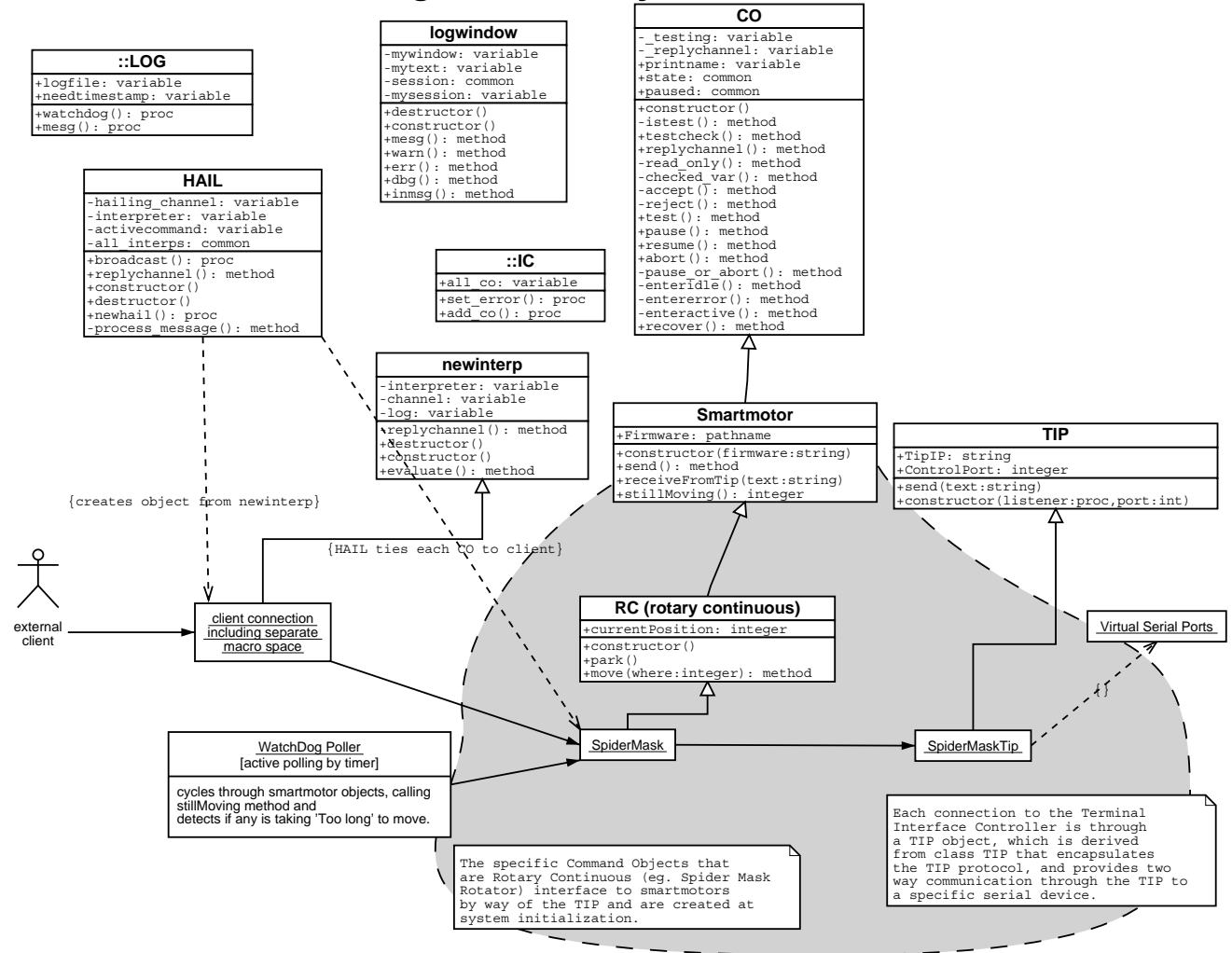


Figure 4 - Linear Continuous

This diagram is presented for completeness only. Currently, there are no linear continuous mechanisms in the NICI instrument.

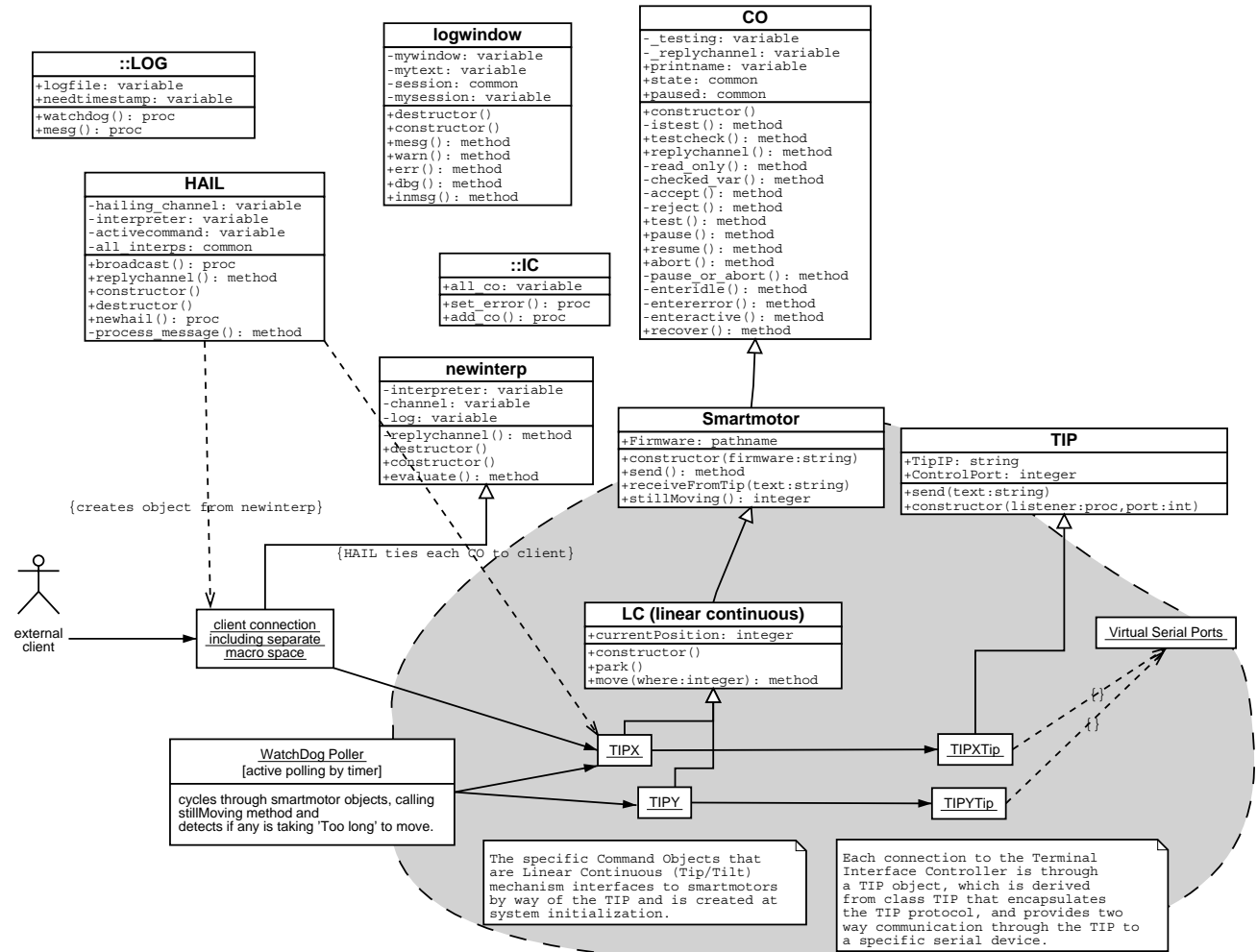


Figure 5 – NDFW: Neutral Density Filter Wheel

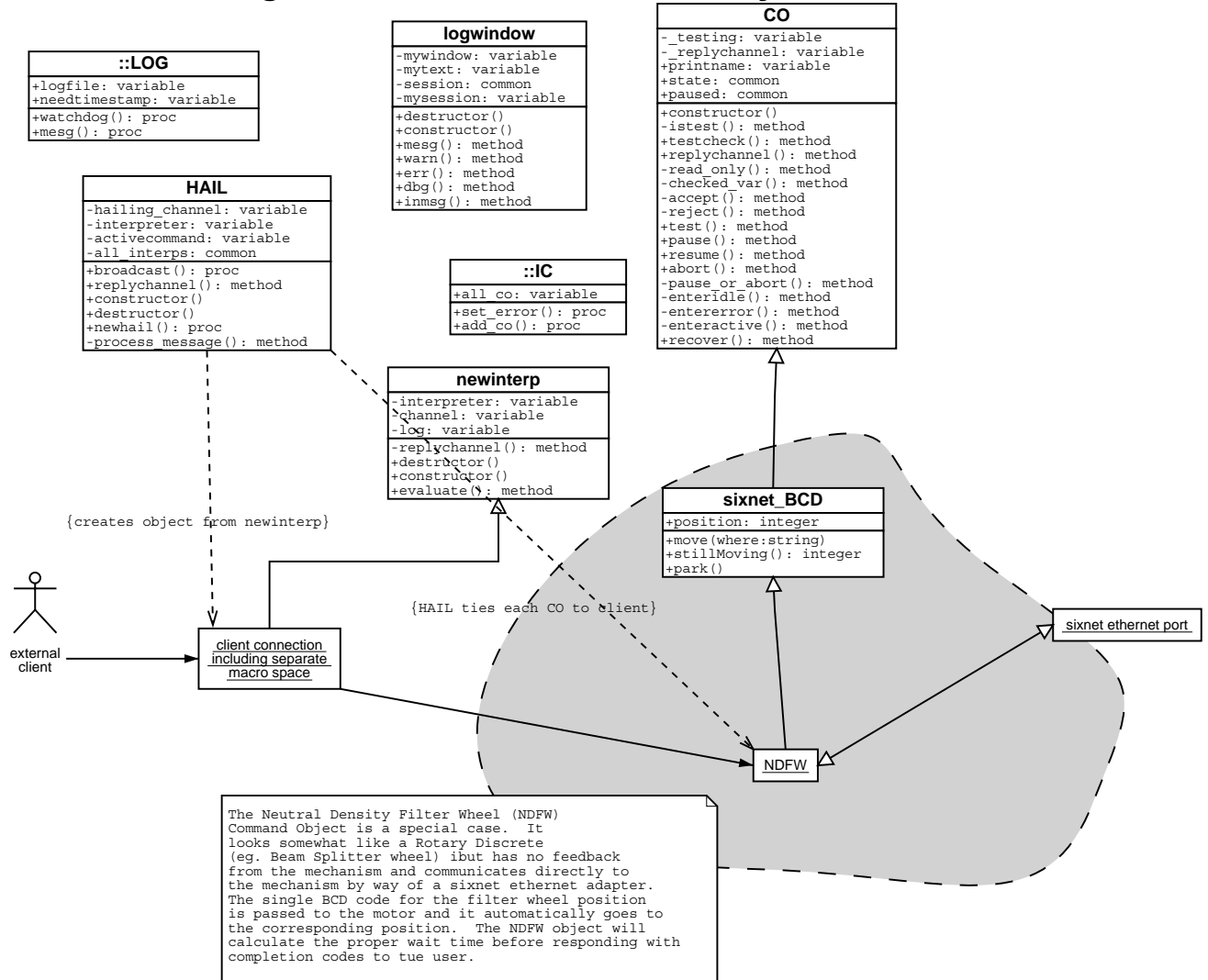


Figure 6 -- Tip/Tilt Steering Mirror

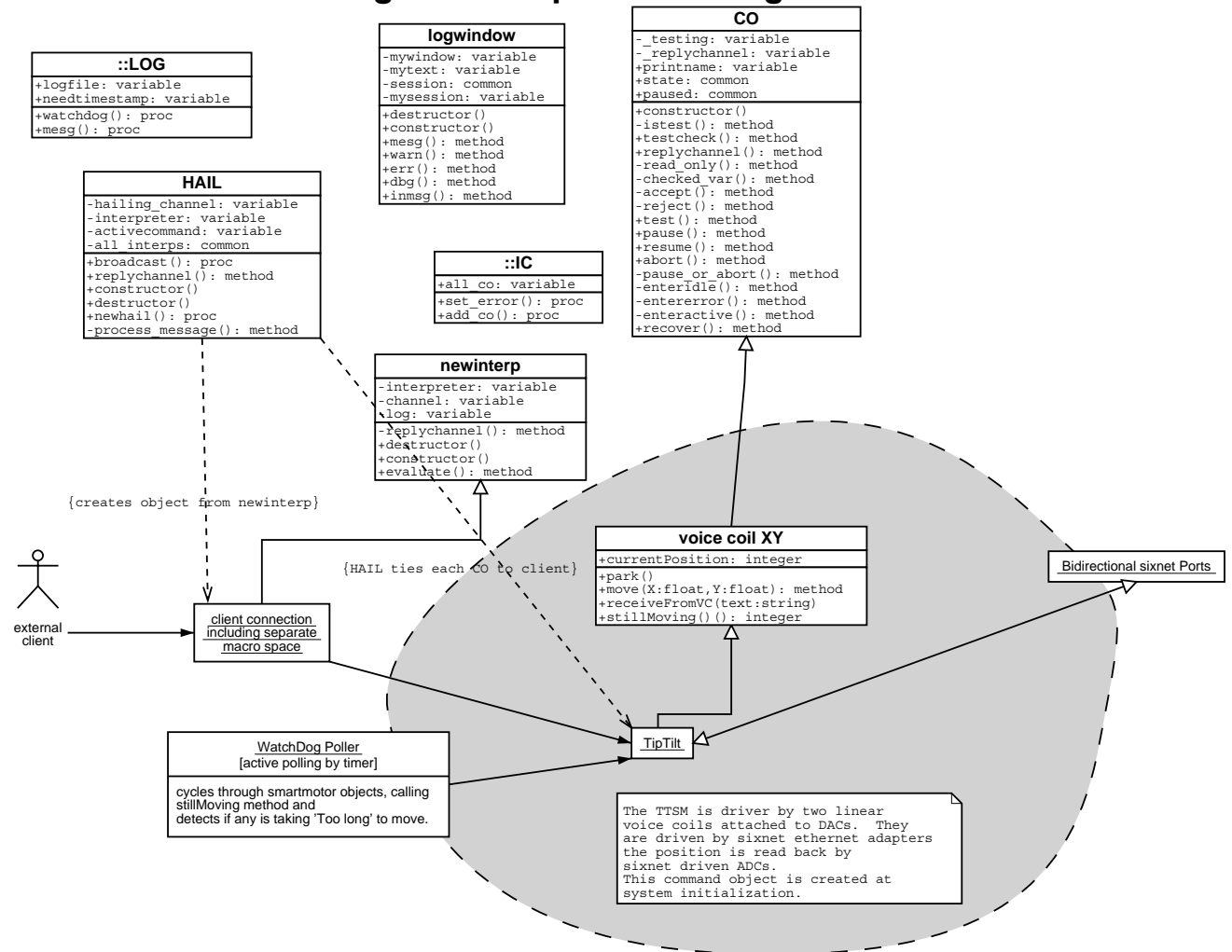
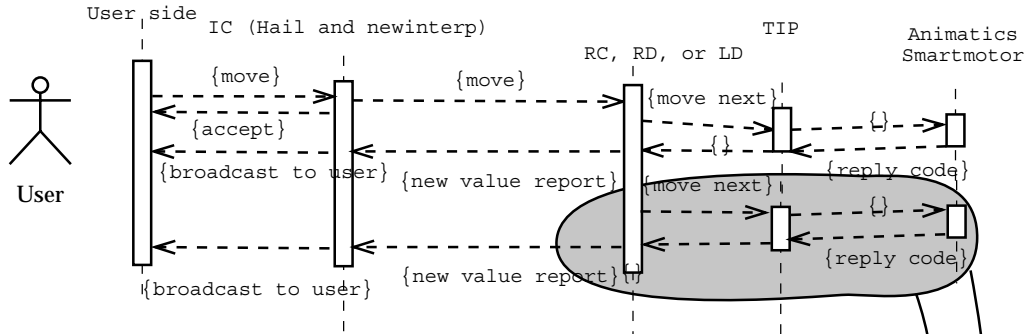
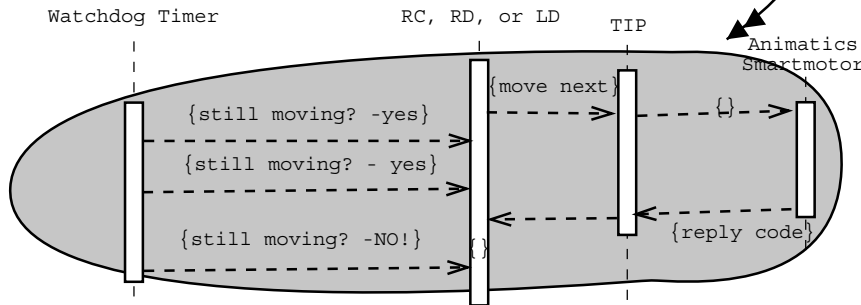


Figure 7-- Information Flow for mechanism movement

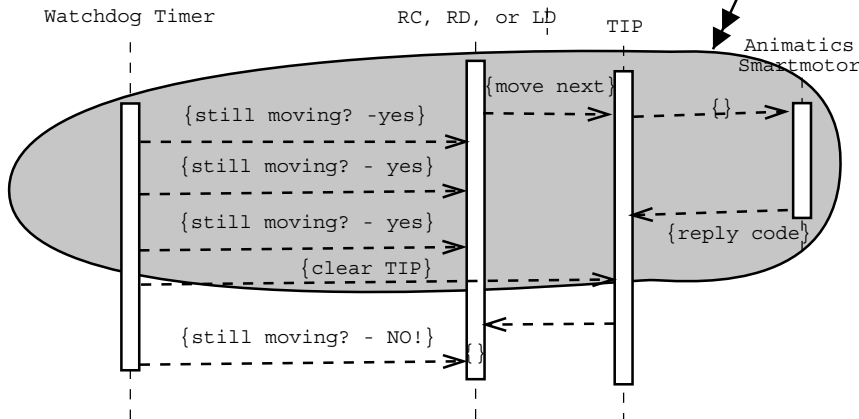


User receives updates as movement proceeds (here only two updates, there will be one per position moved through). The new positions is BROADCAST back to active user interfaces.

IC instrument controller main interface
 (composed of HAIL and newinterp objects)
 RC, RD or LD: Mechanism Command Objects
 Watchdog: timer driven object for polling
 TIP: Terminal interface controller
 Smartmotor: Animatics Smartmotor via TIP

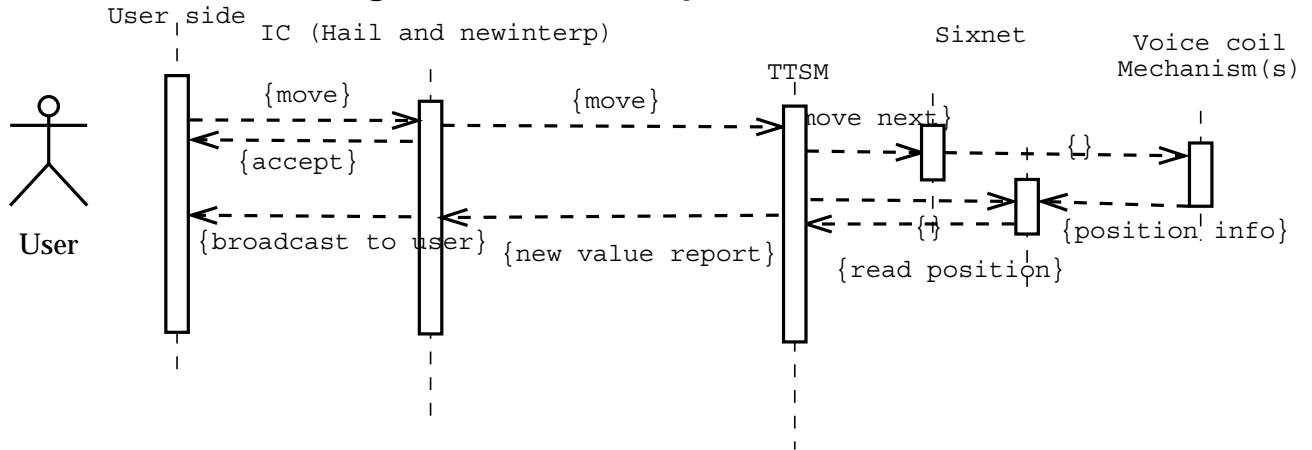


Detail of watchdog timer interaction to detect when movement completes normally. Here we use a count of three "still moving" calls to detect time-out. This count requires tuning on live system.



In this diagram, the watchdog time-out detects that the TIP needs recovery, and it responds properly releasing its stuck buffer. Operations now proceed normally. However, If the TIP could not clear it's error condition, or the poll fails twice in a row the error flag is set on the TIP and an error message is broadcast to the user interfaces. A failing TIP is an instrument level error (except in engineering mode) and no functions can be done until cleared and the recover command is sent. This also implies that mechanisms need re-homing.

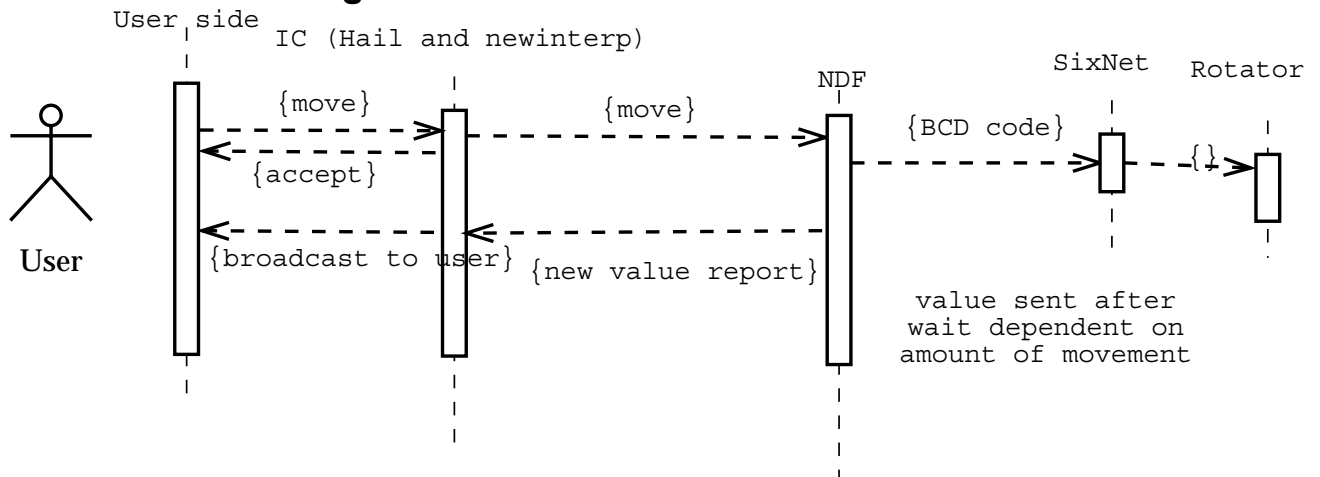
Figure 8 -- Use of Tip/Tilt mechanism



User receives updates after time for movement. The new positions is BROADCAST back to active user interfaces.

IC instrument controller main interface
 (composed of HAIL and newinterp objects)
 TISM: XY command object for Tip/Tilt Mirror
 Watchdog: timer driven object for polling
 SixNet: Ethernet to DAC and ADC to Ethernet
 Voice Coil: linear positioning mechanism

Figure 9-- Use of NDF with BCD Rotator



User receives updates after time for movement. The new positions is BROADCAST back to active user interfaces.

IC instrument controller main interface
 (composed of HAIL and newinterp objects)
 NDF: Adaptive Optics Neutral density filter Command Object
 SixNet: Ethernet to BCD
 Rotator: OTS filter assembly