Documentation on CALIBCTRL.WSH (ALFOSC calibration lamp/mirror control.

Jacob Wang Clasen
Nordic Optical Telescope

November 15, 1996

Contents

Chapter 1

Description

This program is used for controlling the FASU/ALFOSC calibration lamps and the calibration lamp mirror. It is a graphical user interface written in Tcl/Tk.

Figure 1.1: The calibration unit interface. (Sorry about the placement ,-)

Chapter 2

Compilation

This program is interpreted. It does not need compilation.

Chapter 3

Installation

The program should be copied to `$UIF_DIR/io/calibctrl.wsh` where it can be found by the executing script `uif.csh`. Remember also to create the FIFO pipes

- `$UIF_DIR/pipes/calib_pipe_i`
- `$UIF_DIR/pipes/calib_pipe_o`

These pipes are created with the command

- `mknod calib_pipe_i p`
- `mknod calib_pipe_o p`

Chapter 4

Usage

4.1 Normal usage

The program is started automatically by the script `uif.csh`, in turn started by the commandscript `runuif`. Therefore, the usage is confined to pushing the buttons according to the desired action.
To swich a calibration lamp on or off, push the corresponding button. To move the mirror into or out of the lightpath, push the mirrorbutton. Two constraints are made on the funtion of the interface.

1. If the mirror is out and a lamp is turned on, the mirror will automatically be moved in.

2. If any lamps are on and the mirror is moved out, all lamps will be turned off.

To change the buttonlabels (if lamps are changed) edit the following lines in the file `$UIF_DIR/io/calibctrl.wsh`

```
set Lamp1Text "   HgCd"
set Lamp2Text "   Ne "
set Lamp3Text "   Halogen"
set Lamp4Text "   H. Cath. (Copper)"
```

4.2 Debug mode

There is no debug mode for this program.

Chapter 5

Bugs

As of nov. 15. 1996, no bugs have been found or reported.

Chapter 6

Program listing.

```
#!/usr/bin/wish -f
set outfile [open /home/staff/alfosc/src/uif/pipes/calib_pipe_o w 0666]
set infile [open /home/staff/alfosc/src/uif/pipes/calib_pipe_i r 0666]
# Before eventloop optain the status of the lamps and the mirror by
#  sending the command "LP" og "MP" on the serial port (i.e. the pipe)

set Lamp1Text "   HgCd"
set Lamp2Text "   Ne "
set Lamp3Text "   Halogen"
set Lamp4Text "   H. Cath. (Copper)"

#**** Procedure forwards *************
proc FasuIO { mode value } {
    global infile
    global outfile
    for {set i 0} {$i < 2000} {incr i 1} {
    }
    if {$mode == "w"} {
      puts $outfile $value
      flush $outfile
      puts stdout "FasuIO w: $value"
      return 1
    }
    if {$mode == "r"} {
      gets $infile value
      puts stdout "FasuIO r: $value"
      return $value
    }
}

proc FasuRqst { rqst } {
    set status [FasuIO w $rqst]
    set reply ""
    set status [FasuIO r $reply]
    set reply $status
    switch $reply {
      A {set reply 10}
      a {set reply 10}
      B {set reply 11}
      b {set reply 11}
      C {set reply 12}
      c {set reply 12}
      D {set reply 13}
      d {set reply 13}
      E {set reply 14}
      e {set reply 14}
      F {set reply 15}
```

```
      f {set reply 15}
    }
    switch $reply {
      "mn\r" {puts stdout "FasuRqst: $reply"
         return $reply
         }
      "my\r" {puts stdout "FasuRqst: $reply"
         return $reply
         }
    }
    puts stdout "FasuRqst: $reply"
    return [format "%x" $reply]
}

proc FasuGetPos { device } {
    append device P
    set position [FasuRqst $device]
    return $position
}

proc investigate { } {
    global lamp_status
    global mirror_status
    # Sorry, but this sucks!
    set tmp $lamp_status
    switch $lamp_status {
      A {set lamp_status 10}
      a {set lamp_status 10}
      B {set lamp_status 11}
      b {set lamp_status 11}
      C {set lamp_status 12}
      c {set lamp_status 12}
      D {set lamp_status 13}
      d {set lamp_status 13}
      E {set lamp_status 14}
      e {set lamp_status 14}
      F {set lamp_status 15}
      f {set lamp_status 15}
    }
    if {   $lamp_status & 0x1} then {.lamp1 select}
    if {   $lamp_status & 0x2} then {.lamp2 select}
    if {   $lamp_status & 0x4} then {.lamp3 select}
    if {   $lamp_status & 0x8} then {.lamp4 select}
    if {$mirror_status == 0} {
      .mirror deselect
      .mirror configure -text "Mirror is out"
    }
    if {$mirror_status == 1} {
      .mirror select
      .mirror configure -text "Mirror is in"
    }
    set lamp_status $tmp
}
proc DeselectLamps { } {
    global lamp_status
    global mirror_status
    # Sorry, but this sucks!
    set tmp $lamp_status
    switch $lamp_status {
      A {set lamp_status 10}
      a {set lamp_status 10}
      B {set lamp_status 11}
      b {set lamp_status 11}
      C {set lamp_status 12}
      c {set lamp_status 12}
      D {set lamp_status 13}
      d {set lamp_status 13}
      E {set lamp_status 14}
```

```
      e {set lamp_status 14}
      F {set lamp_status 15}
      f {set lamp_status 15}
    }
    if {   $lamp_status & 0x1} then {.lamp1 deselect}
    if {   $lamp_status & 0x2} then {.lamp2 deselect}
    if {   $lamp_status & 0x4} then {.lamp3 deselect}
    if {   $lamp_status & 0x8} then {.lamp4 deselect}
    set lamp_status $tmp
}

proc DoIt { a but_state } {
    global lamp_status
    global mirror_status
    if {$but_state == 1} {
      set temp_status [expr $a | $lamp_status]
    } elseif {$but_state == 0} {
      set temp_status [expr $a ^ $lamp_status]
    }
    set lamp_status $temp_status
    #DEBUG puts stdout [format "New status: %#x" $lamp_status]

    if {$mirror_status == 0} {
      MiCon I
      .mirror select
    }

    set outcmd [string toupper [format "L%x" $lamp_status]]
    FasuIO  w $outcmd
}

proc MiCon { but_action } {
    global lamp_status
    global mirror_status
    ### Action if the mirror is being taken in ###
    if { $but_action == "I" } {
      #DEBUG puts stdout "Mirror in"
      set outcmd [string toupper [format "M%s" $but_action]]
      set mirror_status 1
      ### Send the 'MI' command ###
      FasuIO w $outcmd
      ### And wait until mirror answers 'y' ###

      set a "my"
      while {[set bah [string compare $a [set ready [FasuRqst m]]]] >= 0} {
        for {set i 0} {$i < 500} {incr i 1} {
        }
      .mirror configure -text "Mirror is moving"
      .mirror flash
      #DEBUG puts stdout $ready
      #DEBUG puts stdout "bah in: $bah"
      }

      ### Get the new position of mirror and make it the new mirror_status ###
      set mirror_status [FasuGetPos M]
      ### Set the buttons according to the new status ###
      investigate
      ### Action if the mirror is being taken out ###
    } elseif { $but_action == "O" } {
      ### If any lamps are on, turn them off ###
      if {$lamp_status != 0} {
        DeselectLamps
        set lamp_status 0
        set outcmd [string toupper [format "L%x" $lamp_status]]
        FasuIO w $outcmd
      }
      #DEBUG puts stdout "Mirror out"
      set outcmd [string toupper [format "M%s" $but_action]]
```

```
      set mirror_status 0
      ### Send the 'MO' command ###
      FasuIO w $outcmd
      ### And wait until mirror answers 'y' ###

      set a "my"
      while {[set bah [string compare $a [set ready [FasuRqst m]]]] >= 0} {
        for {set i 0} {$i < 500} {incr i 1} {
        }
      .mirror configure -text "Mirror is moving"
      .mirror flash
      #DEBUG puts stdout $ready
      #DEBUG puts stdout "bah out: $bah"
      }

      ### Get the new position of mirror and make it the new mirror_status ###
      set mirror_status [FasuGetPos M]
      ### Set the buttons according to the new status ###
      investigate
    }
}
#*********** main ********************
set dummy [exec sleep 15]

set lamp_status [FasuGetPos L]
set mirror_status [FasuGetPos M]

#DEBUG puts stdout "lamp pos. $lamp_status"
#DEBUG puts stdout "mirror pos. $mirror_status"

checkbutton .lamp1 -text $Lamp1Text \
  -variable but_1_state \
  -onvalue 1 \
  -offvalue 0 \
  -relief raised \
  -anchor w \
  -width 20 \
  -command {
        DoIt 1 $but_1_state
       }

checkbutton .lamp2 -text $Lamp2Text \
  -variable but_2_state \
  -onvalue 1 \
  -offvalue 0 \
  -relief raised \
  -anchor w \
  -width 20 \
  -command {
        DoIt 2 $but_2_state
       }
checkbutton .lamp3 -text $Lamp3Text \
  -variable but_3_state \
  -onvalue 1 \
  -offvalue 0 \
  -relief raised \
  -anchor w \
  -width 20 \
  -command {
        DoIt 4 $but_3_state
       }
checkbutton .lamp4 -text $Lamp4Text \
  -variable but_4_state \
  -onvalue 1 \
  -offvalue 0 \
  -relief raised \
  -anchor w \
  -width 20 \
```

```
  -command {
        DoIt 8 $but_4_state
       }
checkbutton .mirror -text "Mirror" \
  -relief raised \
  -variable value \
  -onvalue I \
  -offvalue 0 \
  -width 20 \
  -command {
        MiCon $value
       }
button .bye -text "Exit" \
  -width 20 \
  -command {
        exit
       }
.lamp1 configure -selectcolor green
.lamp2 configure -selectcolor green
.lamp3 configure -selectcolor green
.lamp4 configure -selectcolor green
.mirror configure -selectcolor green
pack .lamp1
pack .lamp2
pack .lamp3
pack .lamp4
pack .mirror

#pack .bye
wm geometry . -20-30
wm title . "Calibration unit"


#*********** After widget creation ***************
investigate
```