

# Commanding picomotors with a python script

Jean-Philippe Berger

August 9, 2005

## 1 Setup

This documents describes how to control the picomotors used for alignment purposes on the MIRC combiner.

The equipment in use is the Model 8750 “Intelligent Picomotor Network Controller” (NC) and the Model 8753 picomotor driver which can open-loop drive up to 3 motors. The newtwork controller is accessible via telnet provided it has a valid IP address. It has its own machine language called called MCL.

Drivers are ethernet and power daisy chained to the Network controller  
To setup things do the following:

1. connect the different elements together and the NC to the network;
2. get a valid IP/host address. For that you need to communicate the system admin a MAC address. This is easely done bye issuing the following command with the hand terminal (connected to the NC) who understands MCL commands.

```
> MACADDR
```

3. once connected you can check the IP adress by issuing the following command

```
> IPADDR
```

## 2 Sending commands

Sending commands require to use the hand terminal or a telnet connection. In telnet mode you need to telnet the ethernet controller.

```
telnet monnierpico1.astro.lsa.umich.edu
```

The prompt is a “>” and you can enter MCL commands directly

There are examples in the manual: MCL commands and examples (p61-100)  
A summary of the commands can be found p57 to 59

## 2.1 Low level commands

I have written a PicoControl python class that defines low level python commands which are almost direct images of MCL commands. The main issue I had to deal with (thanks John) was to introduce a time-out pause between each telnet command inside the python commands.

The following table summarizes them.

Description	Python function	Comments
Move to a relative position by	MoveRelative(driver,motor,steps)	Flaky
Change speed of one motor	SetVelocity(driver,motor,vel)	
Change acceleration of one motor	SetAcceleration(driver,motor,acc)	
Check one motor speed	ReadVelocity(driver,motor)	
Check one motor acceleration	ReadAcceleration(driver,motor)	
Checked plugged-in drivers	CheckDrivers()	
Stop all motors	StopMotors()	

## 3 A typical interactive sequence.

Drivers are daisy chained to the NC. The first one (the one that is directly connected to the NC) carries the name “a1” in the MCL language terminology the other ones “a2” ... and so on. The three motors on each driver are designated with a number (1,2,3).

Let's set velocity of motor 2 connected to driver a2 to 1000 Hz (and check it has been correctly set) and moving it by 200000 steps then stopping it while its moving.

```
prompt % python
>>> import picoLowLevel as pic
>>> pc = pic.PicoControl()
>>> pc.SetVelocity("a2",2,1000)
>>> pc.ReadVelocity("a2",2)
>>> pc.MoveRelative("a2",2,200000)
>>> pc.StopMotors()
```

## 4 The python code

For the record your can directly access the MCL terminal from a python interactive section by opening a telnet session using telnetlib class. `telnetlib.Telnet(HOST).interact()`

```

import getpass
import sys
import telnetlib
import time

class PicoControl:

    def __init__(self):
        self.HOST = "monnierpico1.astro.lsa.umich.edu"
        self.timing=0.2 # sleeping time (in s) required for telnet communication
        self.tn = telnetlib.Telnet(self.HOST)
        self.sl = time.sleep

    def MoveRelative(self,driver,motor,steps):

        print "Moving motor "+str(motor)+" on driver "+driver+" by "\
            +str(steps)+" steps"
        self.tn.write("chl "+driver+"="+str(motor)+"\n")
        self.sl(self.timing)
        self.tn.write("rel "+driver+"="+str(steps)+"\n")
        self.sl(self.timing)
        self.tn.write("go\n")
        self.sl(self.timing)

    def SetVelocity(self,driver,motor,vel):
# Set velocity in Hz
# Value: 1 to 2000
        print "Set velocity of motor "+str(motor)+" on driver "+driver+" to "\
            +str(vel)+" Hz"
        self.tn.write("vel "+driver+" "+str(motor)+"="+str(vel)+"\n")

    def SetAcceleration(self,driver,motor,acc):
# Set acceleration in steps/s^2
# Value: 16-20,000
        print "Set acceleration of motor "+str(motor)+" on driver "+driver+" to "\
            +str(acc)+" steps/s^2"
        self.tn.write("acc "+driver+" "+str(motor)+"="+str(acc)+"\n")

    def StopMotors(self):

        print "Stopping all active motors"
        self.tn.write("hal\n")

    def ReadVelocity(self,driver,motor):

        flush=self.tn.read_very_eager()

```

```

        self.sl(self.timing)
        self.tn.write("vel "+driver+"="+str(motor)+"\n")
        self.sl(self.timing)
        vel=self.tn.read_very_eager()
        print "Velocity of motor "+str(motor)+" on driver "+driver+" is "+vel.split()[1][3:]
        return vel.split()[1][3:]

def ReadAcceleration(self,driver,motor):

    flush=self.tn.read_very_eager()
    self.sl(self.timing)
    self.tn.write("acc "+driver+"="+str(motor)+"\n")
    self.sl(self.timing)
    acc=self.tn.read_very_eager()
    print "Acceleration of motor "+str(motor)+" on driver "+driver+" is "+acc.split()[1][3:]
    return acc.split()[1][3:]

def CheckDrivers(self):

    flush=self.tn.read_very_eager()
    self.sl(self.timing)
    self.tn.write("drt\n")
    self.sl(self.timing)
    dri=self.tn.read_eager()
    print str(len(dri.split())-1)+ " drivers are connected."

def __del__(self):

    self.tn.close()

```