Example TCPComm Experimental File

written by John S. McCaskill and Carla Verhaelen

Ruhr University Bochum

June 2008

This is an example file for taking complex programmed control of the Omega Machine

via the interpreted language environment provided by *Mathematica*.

This file uses the mathematica package TCPComm (ver 1.2.6) written by J.S.McCaskill.

It communicates via Mathlink with an executable written in C for TCP Communication.

This in turn communicates across the network via TCP with the external command interface

for the Ng-biopro software (leading author Uwe Tangen, RUB-BioMIP) running on the experimental machine

that is connected via the Biofox stand alone interface computer with the Omega Machine.

Users can write their own command sequences (macros) and execute these in a timed interpretative environment using this software. A simple example collects data from various subimages, plotting results interactively and controlling the laser illumination synchronized with microscope images.

Set the directory to your current one containing a copy of the TCPComm executable.

The TCPComm.m mathematica package written by J.S.McCaskill also needs to be installed in your *Mathematica*/Applications folder.

```
In[1]:= SetDirectory["/Users/john/Library/Mathematica/Applications/TCPComm/Build/Release"]
Out[1]= /Users/john/Library/Mathematica/Applications/TCPComm/build/Release
```

This loads the definitions from the TCPComm package (note the backquote, necessary for context name)

```
In[2]:= Needs["TCPComm`"]
```

The following list of active links gives the usage information for the functions exported from the TCPComm package.

In[3]:= **? TCPComm`***

▼ TCPComm`

imbin	imextr˙. act	imrow˙. diag	imwrite	NG	NGCo˙. mm˙. une	NGPa˙. lette	verbo˙. se
imco˙. mbi˙. ne	imread	imset˙. ROI	MA	NGCl˙. ose	NGExit	status	□
imcon˙. vert	imROI	imshow	MAC˙. mds	NGC˙. md	NGIns˙. tall	Timed˙. Loo˙. p	
imcsu˙. ROI	imrow˙. col	imsize	MAPa˙. lette	NGC˙. mds	NGlink	untool	

Setup palettes once on your machine

NGPalette[]

Sess open | Sess close | Quit | Status | BackOff | BackOn
Synch | Window | Cam par | Snap | AOTF | Filter
Pump par | Pump | List pins | Electrode | Cycle | List sensors
Sensor data | Intensity | List names | Temp | XY move | XY def
ZStage | Verbose

MAPalette[]

Timed Loop | Set ROI | I Snap | I Convert | I Display | I Dispsub
I Color | I Write | I Read | I Size | I RowCol | I Line
I Bin | I ROI | I Comb | NG Start | NG Exit

1. Install connection to ng-biopro

In[9]:= **MA[NGInstall["dyck"]]**

```
Command completed successfully in session Mathematica_TCP

Command completed successfully in session session1

verbose mode turned off
```

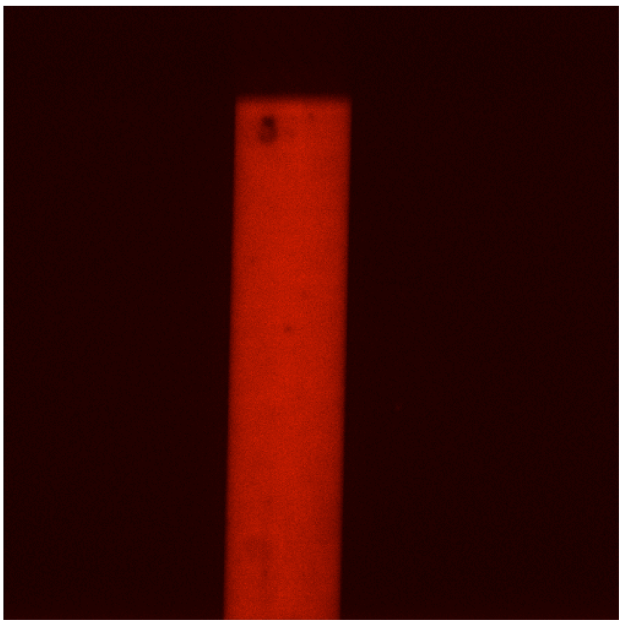
Out[9]= {LinkObject[./TCPComm, 7, 5], {Mathematica_TCP}, {session1},
False, {Pump use desired, Graphics are up and operational, 0}}

2. Experimental macro session

In[10]:= **im1 = imconvert[NG["i"]];**

```
Image[1004,1002] Thu 21 Aug 2008 13:19:36
```

In[11]:=
```
Print[imshow[im1, Red, 0, {{0, 0}, {0, 0}}]];
```



e.g. Low bleaching acquisition with measurement field

Definitions for region extraction

In[13]:=
```
size[roi_] := (roi[[2, 1]] - roi[[1, 1]] + 1) (roi[[2, 2]] - roi[[1, 2]] + 1);
```

In[14]:=
```
flipROI[{{x1_, y1_}, {x2_, y2_}}, {ydim_, xdim_}] :=
  Reverse[{{x2, ydim - 1 - y1}, {x1, ydim - 1 - y2}}];
```

Macro for low bleaching

In[27]:=
```
laser[j_Integer] := Module[{mim},
  NG["a", "biofox_blue", 63];
  Pause[0.2]; im1 = imconvert[NG["i"]];
  NG["a", "biofox_blue", 0];
  mim = Flatten[imextract[im1[[1]], ROImess[1]]];
  mv[1][j] = Fold[Plus, 0, mim] / size[ROImess[1]];
  mim = Flatten[imextract[im1[[1]], ROImess[2]]];
  mv[2][j] = Fold[Plus, 0, mim] / size[ROImess[2]];
  PrintTemporary["Printing image"];
  Print[GraphicsRow[{Show[imshow[im1, White, 0, imcsuROI], PlotLabel → j],
     ListPlot[Array[mv[1], j]], ListPlot[Array[mv[2], j]]}]];
 ]
```

Custom version of TimedLoop macro

In[16]:=
```
TimedLoop1[name_String, body_, intvlsec_, niter_Integer] := Module[{i, t, t1, nwait, d},
  For[t = SessionTime[]; i = 1, i ≤ niter, i++,
   t1 = SessionTime[]; Print["\n", name , "[", i, "] Time = ", t1 - t, "s"];
   body[i];
   If[i == niter, Break[]];
   t1 = SessionTime[];
   nwait = t + intvlsec * i - t1;
   If[nwait ≤ 0, Print[nwait, " is too little time for macro timing"],
    Print["waiting..."]; Pause[nwait]];
  ]]
```

Take sample image

In[21]:=
```
im1 = imconvert[NG["i"]][[1]];
```

```
Image[1004,1002] Thu 21 Aug 2008 13:21:23
```

First guess, ROI for a sample measurement field

In[22]:=
```
roi = ROImess[1] = ROImess[2] = imsetROI[{{300, 300}, {350, 350}}]
```

Out[22]=
```
{{300, 300}, {350, 350}}
```

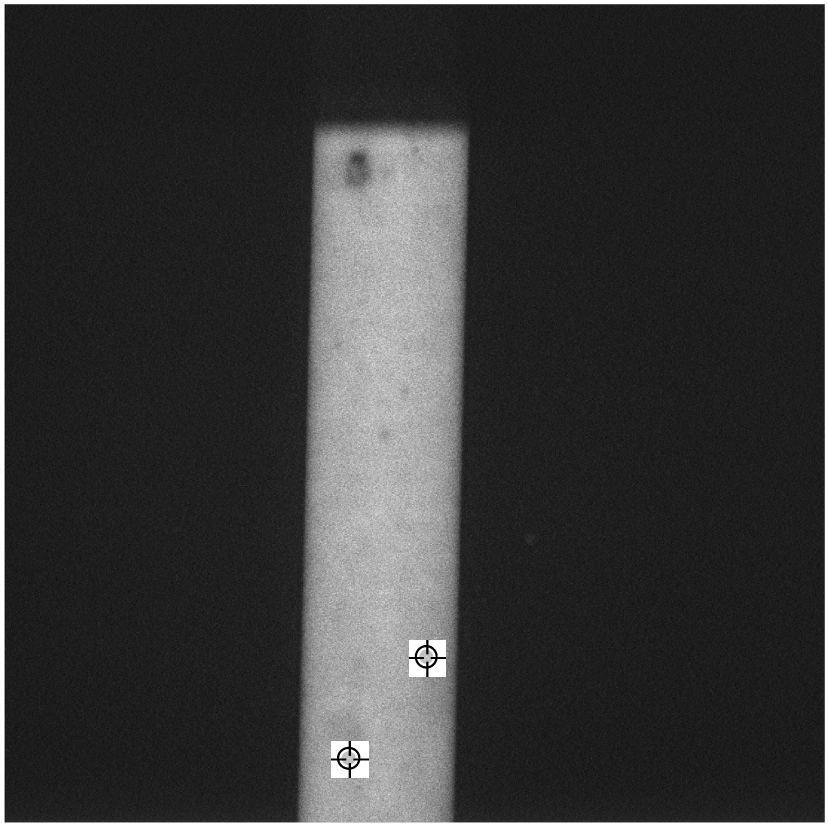
Optimize interactively

Optimize measurement field

In[23]:=
```
Dynamic[
 {LocatorPane[Dynamic[roi], Show[imshow[im1, White, 0, {{0, 0}, {0, 0}}, False],
    ImageSize → {400, 400}], Automatic, {1, 1}],
  Dynamic[
   roi]}]
```

Optimize measurement field

Out[23]=
```
{
```



```
 , {{423., 78.}, {517., 200.}}}
```

Store vertically flipped ROI as measurement field 1

In[24]:=
```
ROImess[1] = flipROI[Round[roi], Dimensions[im1]]
```

Out[24]=
```
{{421, 506}, {538, 587}}
```

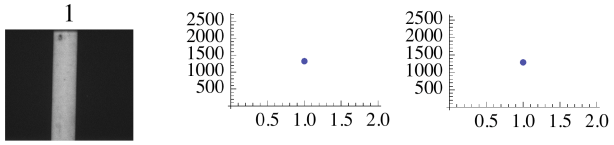
Repeat optimization for second measurement field. The Store vertically flipped ROI as measurement field 2

In[25]:=
```
ROImess[2] = flipROI[Round[roi], Dimensions[im1]]
```

Out[25]=
```
{{423, 801}, {517, 923}}
```

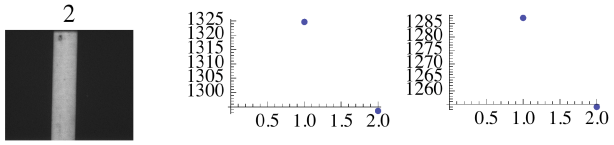
In[28]:=
```
NG["s 0x8000 0x8000"]; TimedLoop1["laser", laser, 15, 3]; NG["s 0x8000 0"];
```

```
laser[1] Time = 0.000049s
Image[1004,1002] Thu 21 Aug 2008 13:25:14
```



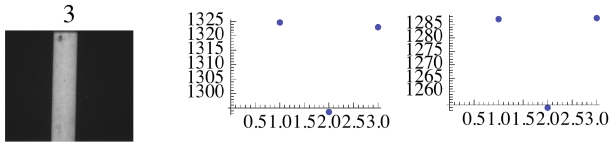

```
waiting...

laser[2] Time = 15.006485s
Image[1004,1002] Thu 21 Aug 2008 13:25:14
```




```
waiting...

laser[3] Time = 30.009190s
Image[1004,1002] Thu 21 Aug 2008 13:25:14
```



3. To finish session

In[29]:= **MA[NGExit["dyck"]]**

```
verbose mode turned on
No return parameter, returned with stype è
```

Out[29]= {True, $Failed, ./TCPComm}