Overview

This DAPL-code is used for measurements on intact trabeculae.

The DAPL-code is an adaption on the existing isotwitch code. The following adaptions have been made in comparison with the existing isotwitch:

* 1 Digital output channel signal. This digital output opens the light inlet shutter at the beginning of the measurement and closes it when the measurement has finished.
* 1 Analog output signal. This signal will control the monochrometer that sends out the light at different wavelengths, thus causing the contractions of the cell. The signal itself is a squarewave with predefined amplitude and a frequency of 500Hz.
* The manual command for isotwitch has been adapted (see the C++ code for isotwitch\_ramp), now including a ramp before returning to the resting position. As of this, the isotwitch command in DAPL will include 2 more inputs.
* In addition to the existing proportional control, integral control has been added.
* 4 Output channels have been added.

Additional resources

Need more help?

Check the resources, and then see Ken

Main content

**Part 1: When run…**

When run, the following occurs:

* The processor is **reset**, removing any lingering data in the system.
* **Options** are set as defined by Ken.

**reset**

**option underflowq=off**

**option errorq=off**

**option scheduling=fixed**

**option buffering=off**

**option quantum=100**

* Some standard values for isotwitch are defined. **Do not change the values for these variables in the main DAPL document**, they can be adapted in the matlab code used to run this DAPL-code. This is important as the matlab code will assume the standard values to be the same as defined below.

**// Header for triggered output**

**var pretriggersamples = 10000 LONG**

**var isoton = 0**

**var isoinc = 0**

**var nsamples = 30000 LONG ;\*\* Don't make bigger than 60000**

**var isopoints = 2500 LONG**

**var ramppoints = 2500 LONG**

**var interramppoints = 0 LONG**

**var interfldiff = 0 LONG**

**var propgain = 200**

**var Ki = 20     ; max is 32767**

**var flpolarity = -1**

**var flswitch = 1**

**var flthreshold = 1**

**var smoothing\_points = 10**

**var integral\_points = 10**

**var gain\_asymmetry = 0     ; positive value here helps the downward drift when the motor reverses**

**var wave\_center = 14811**

**var wave\_amplitude = 3146**

* The pipes that will be used are defined. Mind that the digital out channel that will be used depends on the value that the pipe pOpen\_Shutter is filled with. This is a binary number, representing the digital out channel. So for instance filling the pipe with value 4 will open digital out channel 3.  
    
  **pOpen\_Shutter value**               **Digital out channel**  
  1                                               1  
  2                                               2  
  3                                               1 & 2  
  4                                               3  
  5                                               1 & 3  
  etc.

**pipes pSquarewave**

**pipes pSquarewave2**

**pipes pReset**

**pipes pOpen\_Shutter**

**pipes pClose\_Shutter**

**fill  pReset 0**

**fill  pOpen\_Shutter 4**

**fill  pClose\_Shutter 0**

* **pdef SqWave**: this process creates the squarewave that will be used to control the monochrometer. First a squarewave with amplitude "wave\_amplitude" and a frequency of 500Hz are created. After this the wave center is still zero, so has to be adjusted to get the proper high and low values of the squarewave.  
  The frequency of the squarewave is determined by a number of values. First the sample time length is determined in the output process that sets the output pipes to the analog outputs. The sample time is the number defined after the TIME command times the number of analog outputs. So in this case two analog outputs and a defined TIME of 100 sets the time samples to be 200 microseconds long. Another number of importance is the oscillation length as defined in the squarewave process. In this case each oscillation is 50 time samples long.  
  Since 1 second is 1,000,000 microseconds, the frequency can thus be calculated:  
  1,000,000 / (analog outputs \* TIME ### \* number of samples)  
    
  In this case: 1,000,000 / (2 \* 100 \* 50) = 100Hz

**pdef SqWave**

**squarewave (Wave\_Amplitude, 50, pSquarewave)**

**pSquarewave2 = pSquarewave+Wave\_Center**

**end**

* **pdef control**: This control process will run the manual command "isotwitchpicont". Isotwitchpicont is a C++ written manual DAPL command and was based on the original isotwitch. In comparison with the original code, the proportional control has been improved to both Proportional en Integral control (PI-control). Furthermore a ramp has been added after the sample has been held at isotonic level, but before it will be held on isometric level. Because of the addition of integral control and the extra ramp, isotwitchpicont will require four more inputs, adding to a total of 19 in- and outputs. All inputs and the output pipe (op1) are defined earlier in the code and can be adjusted in the Matlab file.  
  For technical reasons, the copying of the aforementioned squarewave to analog output pipe 1 is also included in this process. DAPL has the "limitation" that it can only write out data to output pipes in one process at a time. Since all updating will happen real-time, it cannot process writing to output pipes in two different processes at the same moment. For this reason, both the data from isotwitch to output pipe 0 and the data from the squarewave to output pipe 1 are written in the control process.  
  Finally, this process will merge the input pipes 0 through 7 (so 8 inputs in total).

**pdef control**

**isotwitchpicont(ip0,ip2,pretriggersamples,isoton,isoinc,nsamples,isopoints,ramppoints,interramppoints,interfldiff,propgain,Ki,flpolarity,flswitch,flthreshold,smoothing\_points,integral\_points,gain\_asymmetry,op0)**

**copy (pSquarewave2, op1)**

**merge(ip(0..7),$binout)**

**end**

* **pdef Open\_Shutter**: Opens the shutter for the light inlet on the sample. The advantage of opening the shutter with a process rather than through a digital output pipe is that you will not have 3 outputs over 8 inputs. Three over eight would have been impossible to time correctly, since you want to update both outputs and inputs over the same interval in order to prevent aliasing. Fortunately, this method gives us two outputs over eight inputs, which is easy to time.

**pdef Open\_Shutter**

**digitalout (pOpen\_Shutter, 0)**

**end**

* **pdef Close\_Shutter**: Closes the shutter after the measurements are taken.

**pdef Close\_Shutter**

**digitalout (pClose\_Shutter, 0)**

**end**

* **pdef EndWave**: Resets the analog output of both isotwitch and the squarewave back to zero after the system has finished taking measurements.

**pdef EndWave**

**copy (pReset, op0)**

**copy (pReset, op1)**

**end**

* **odef output\_def 2**: Since two analog outputs are defined here, odef contains a 2 after the output definition's name.  
  Outputwait orders the system to wait until there are at least 2 samples are contained in the output pipe before the output is updated.  
  Update burst orders the system to stop updating the outputs when the output pipes are exhausted and to continue only after the number is larger again than the number specified in outputwait.  
  Output pipes 0 and 1 are set to analog ouputs 0 and 1 respectively. (As a matter of fact, DAPL won't even allow a user to set op0 to analog 1 or vise versa.)  
  TIME sets the length of each time sample in microseconds. However, the real number for which each output is updated is sample time \* number of analog outputs, this is because the system switches between both outputs. So since in this case TIME is 100 microseconds, the system updates op0 for 100μs, op1 for 100μs and op0 for 100μs and so on. So effectively each output is updated every 200 microseconds.

**odef output\_def 2**

**outputwait 2**

**update burst**

**set op0 a0**

**set op1 a1**

**time 100**

**end**

* **odef EndWave\_output 2**: sets the aforementioned reset of the isotwitchpicont command and the squarewave to the analog outputs.

**odef EndWave\_output 2**

**set op0 A0**

**set op1 A1**

**cycle 1**

**end**

* **idef input\_def 8**: Sets the eight input signals to input pipes 0 through 7. The number 8 is contained after the input definition name since there are eight input signals.  
  TIME is again the sample time length for input updating. In order to prevent aliasing, it is best if both output updating cycle and input updating cycle are equal. For this, TIME has been set to 25μs, since 2\*100 = 8\*25 = 200μs.  
  The count command sets a fixed number of input numbers. If this number is surpassed, the input configuration suspends operation.

**idef input\_def 8**

**set ip0 s0**

**set ip1 s1**

**set ip2 s2**

**set ip3 s3**

**set ip4 s4**

**set ip5 s5**

**set ip6 s6**

**set ip7 s7**

**time 25**

**count 260000**

**end**

* Finally the **start** command starts the defined processes, inputs and outputs.  
  These are continued for the duration of pause, which is determined in the MATLAB m.file that writes the DAP code. The pause duration will be 10 miliseconds longer than the actual measurement, this is done to make sure that the system has finished taking the measurement before it is reset.  
  Then the **stop** command order the defined processes, inputs and outputs to stop.  
  Ultimately, some processes and outputs are run to reset the analog outputs and the configuration stops completely.

**start SqWave, control, Open\_Shutter, output\_def, input\_def**

**pause 6510**

**stop SqWave, control, Open\_Shutter, output\_def, input\_def**

**//For resetting to zero**

**Start Close\_Shutter, EndWave, EndWave\_output**

**pause 100**

**stop**

[to edit digital out channels, edit fill **pOpen\_shutter N**, not **digital out (pOpen\_Shutter, N);** the current method prevents digital pacing via DAP during this out.]

**Part 2: The Complete DAPL Code**

**reset**

**option underflowq=off**

**option errorq=off**

**option scheduling=fixed**

**option buffering=off**

**option quantum=100**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Header for triggered output**

**var pretriggersamples = 10000 LONG**

**var isoton = 0**

**var isoinc = 0**

**var nsamples = 30000 LONG ;\*\* Don't make bigger than 60000**

**var isopoints = 2500 LONG**

**var ramppoints = 2500 LONG**

**var interramppoints = 0 LONG**

**var interfldiff = 0 LONG**

**var propgain = 200**

**var Ki = 20     ; max is 32767**

**var flpolarity = -1**

**var flswitch = 1**

**var flthreshold = 1**

**var smoothing\_points = 10**

**var integral\_points = 10**

**var gain\_asymmetry = 0     ; positive value here helps the downward drift when the motor reverses**

**var wave\_center = 14811**

**var wave\_amplitude = 3146**

**pipes pSquarewave**

**pipes pSquarewave2**

**pipes pReset**

**pipes pOpen\_Shutter**

**pipes pClose\_Shutter**

**fill  pReset 0**

**fill  pOpen\_Shutter 4**

**fill  pClose\_Shutter 0**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**pdef SqWave**

**squarewave (Wave\_Amplitude, 50, pSquarewave)**

**pSquarewave2 = pSquarewave+Wave\_Center**

**end**

**pdef control**

**isotwitchpicont(ip0,ip2,pretriggersamples,isoton,isoinc,nsamples,isopoints,ramppoints,interramppoints,interfldiff,propgain,Ki,flpolarity,flswitch,flthreshold,smoothing\_points,integral\_points,gain\_asymmetry,op0)**

**copy (pSquarewave2, op1)**

**merge(ip(0..7),$binout)**

**end**

**pdef Open\_Shutter**

**digitalout (pOpen\_Shutter, 0)**

**end**

**pdef Close\_Shutter**

**digitalout (pClose\_Shutter, 0)**

**end**

**pdef EndWave**

**copy (pReset, op0)**

**copy (pReset, op1)**

**end**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**odef output\_def 2**

**outputwait 2**

**update burst**

**set op0 a0**

**set op1 a1**

**time 100**

**end**

**odef EndWave\_output 2**

**set op0 A0**

**set op1 A1**

**cycle 1**

**end**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**idef input\_def 8**

**set ip0 s0**

**set ip1 s1**

**set ip2 s2**

**set ip3 s3**

**set ip4 s4**

**set ip5 s5**

**set ip6 s6**

**set ip7 s7**

**time 25**

**count 260000**

**end**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**start SqWave, control, Open\_Shutter, output\_def, input\_def**

**pause 6510**

**stop SqWave, control, Open\_Shutter, output\_def, input\_def**

**//For resetting to zero**

**Start Close\_Shutter, EndWave, EndWave\_output**

**pause 100**

**stop**