Using TDTRP:

Use Matlab to directly talk to TDT hardware

Requirements

- TDT Drivers
- TDT ActiveX package

About TDTRP

TDTRP is a Matlab class that serves as a wrapper for the RPcoX ActiveX control that installs with the TDT ActiveX package. It provides direct access to TDT hardware (connect, load/run circuit, read/write tags on the hardware, send software triggers) and is generally easier to use than calling the RPcoX object directly.

TDTRP Functionality

Methods, Examples, and RPcoX Methods Comparisons

Creating a TDTRP object

```
obj = TDTRP(CIRCUITPATH, DEVICETYPE, 'parameter', value, ...)

% obj is what you want the structure to be called for your code. In
% this document obj will be 'TDT' after we make our first call to
% TDTRP below. However, obj can be any unique variable name.
```

'parameter', value pairs:

'INTERFACE'	String, interface type. Can be 'USB', 'USB3', or 'GB' (default for fiber optic interface)
'NUMBER'	Integer, device number as enumerated by zBusMon (default 1)
'FS'	Float, sampling rate to run the circuit at (otherwise uses circuit default)

Example:

```
>> TDT = TDTRP...

('C:\TDT\ActiveX\ActXExamples\RP_Files\Continuous_Acquire.rcx', 'RZ6')
```

```
% TDT is the obj

% the target device is an RZ6 processor. I only have one, so no need
% to set the 'NUMBER' parameter, and it has a fiber optic interface
% built into it, so it uses the default 'GB' interface.

% Note: the '...' is an example of a continued line in MATLAB. It is
% used here for the sake of margins

Loading C:\TDT\ActiveX\ActXExamples\RP_Files\Continuous_Acquire.rcx

Circuit loaded and running
```

RPcoX Method Comparison:

```
RP=actxcontrol('RPco.x',[5 5 26 26]);

RP.ConnectRZ6('GB',1)

RP.LoadCOF...

('C:\TDT\ActiveX\ActXExamples\RP_Files\Continuous_Acquire.rcx');

e = RP.Run

if e == 0

     disp 'error running circuit'

else

     disp 'Circuit ready to run'

end
```

Setting the mode with TDTRP

```
TDT.halt          % stop any existing processing chain

TDT.load          % load the processing chain

TDT.run           % run the processing chain
```

Example:

```
>> TDT.halt;

>> circuit =
'C:\TDT\ActiveX\ActXExamples\RP_Files\Continuous_Acquire.rcx';

>> TDT.load(circuit);

>> TDT.run;
```

```
% Calling TDTRP() does the equivalent of calling halt, load, and run
% load and run the circuit onto the specified device
```

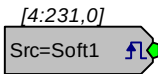
RPcoX Method Comparison:

```
RP.Halt

circuit =
'C:\TDT\ActiveX\ActXExamples\RP_Files\Continuous_Acquire.rcx';

e = RP.LoadCOF(circuit); % Loads circuit

if e==0

      disp 'Error loading circuit'

else

      disp 'Circuit ready to run'

end

e = RP.Run

if e==0

      disp 'error running circuit'

else

      disp 'Circuit ready to run'

end
```

Software Triggers

```
TDT.trg(TRIGNUM) % TRIGNUM is the software trigger index (1-10)
```

Example



Component to trigger. TRIGNUM determined by number after 'Soft'

```
TDT.trg(1);
```

RPcoX Method:

```
RP.SoftTrg(1);
```

Writing to Parameter Tags

```
TDT.write(TAGNAME, VALUE, 'parameter', value, ...) % TAGNAME is a
% string. VALUE is the single value or array that you want to push
% onto the tag. Whether the tag is pointed to a scalar component or
% buffer will determine what is a valid write. Examples of these
% paradigms are below in 'Example.'
```

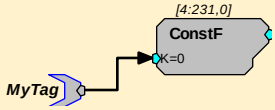
'parameter', value pairs

'FORMAT'	String, destination format (array only) options are 'F32' (default) 'I32' 'I16' 'I8'
'OFFSET'	Scalar, offset into buffer (array only). Default is 0

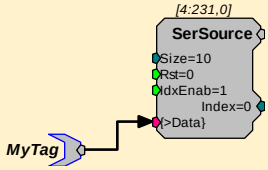
Example

Tag to write.

```
TDT.write('MyTag',3.14); % writing a single scalar value to a ConstF
%component
```




```
% or

TDT.write('MyTag',1:10); % writing an array of values to a SerSource,
% which is a serial buffer
```



RPcoX Method:

```
RP.SetTagVal('MyTag', 3.14); % for single value

RP.WriteTagVEX('MyTag', 0, 'F32', 1:10); % for array of values
```

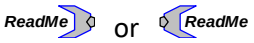
Reading from Parameter Tags

```
TDT.read(TAGNAME,'parameter',value,...) % Similar to the TDT.write
% method, what the tag is pointed to will determine what a valid read
% is. A scalar tag must be on a scalar component. An array must be on
% a buffer.
```

TDT.read 'parameter', value pairs

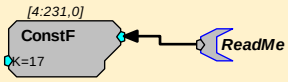
'SOURCE'	String, source format (array only) options are 'F32' (default) 'I32' 'I16' 'I8'
'DEST'	String, destination format (array only) options are 'F64' 'F32' (default) 'I32' 'I16' 'I8'
'SIZE'	Scalar, number of words to read (array only). Default is the entire buffer
'OFFSET'	Scalar, offset into buffer (array only). Default is 0
'NCHAN'	Scalar, number of channels in buffer (array only). Used for de-interlacing data. Default is 1

Example

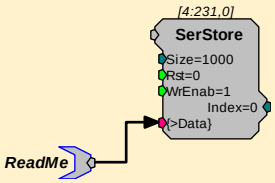


Tags to read.

```
ScalarVal = TDT.read('ReadMe'); % Will read single values or arrays
```




```
 % Tags always point TOWARDS a component

% or

ArrVals = TDT.read('ReadMe');
```




```
 % This will read all values stored in the buffer
```

RPcoX Method:

```
RP.GetTagVal('ReadMe'); % for single value

RP.ReadTagVEX('ReadMe',0,1000,'F32','F32',1); % for 1 chan array of
% 1000 values
```

Example ActiveX Code Using TDTRP

You can find examples of ActiveX code using TDTRP installed with ActiveX in C:\TDT\ActiveX\ActXExamples\Matlab or in the ActiveX Controls example zip on our download's page https://www.tdt.com/support/downloads/