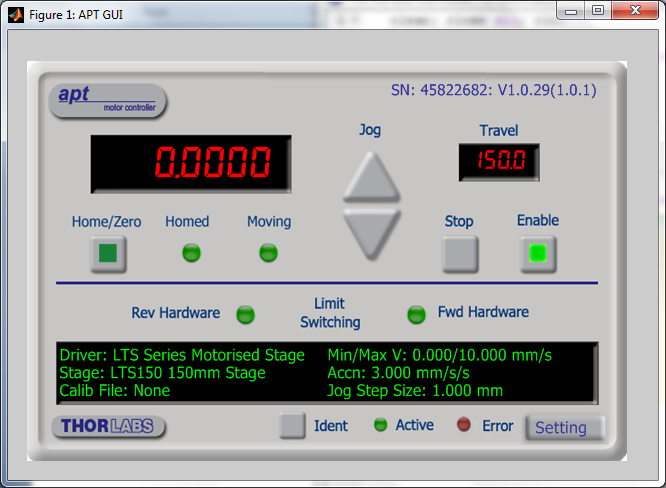
|  |
| --- |
| Thorlabs |
| Using Thorlabs APT ActiveX Control in MATLAB |
|  |
|  |
| **Wei Wang** |
| **3/8/2011** |

|  |
| --- |
| A brief demonstration on how to integrate Thorlabs APT ActiveX control into MATLAB figure container and control the hardware through MATLAB. This is a supplemental document to the MATLAB COM control document. Sample codes are included |

# Introduction

The Microsoft Component Object Model (COM) is a binary-interface standard for implementing objects that can be used in an environment different from the one in which they are created. It allows reuse of objects without knowledge of their internal implementation. A Microsoft ActiveX control is a type of in-process COM server that requires a control container. ActiveX controls typically have a user interface. A control container is an application capable of hosting ActiveX controls. A MATLAB figure window or a Simulink model are examples of control containers.

In a typical scenario, MATLAB creates ActiveX controls in figure windows, which are manipulated by MATLAB through the controls’ properties, methods and events. In this case, Thorlabs APT ActiveX control is used as the server and MATLAB is used as the client. Graphic user interfaces (GUI) components have been implemented as APT ActiveX controls. By creating an ActiveX object, MATLAB can control the hardware using the APT GUI components. Figure 1 shows an example of integrating the APT stepper controller into the MATLAB figure window to control a LTS150 integrated long travel stage.



Figure

Detailed documents on MATLAB ActiveX control can be access on Mathworks website

<http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_external/bqdwu3j.html>

# Getting Started with ActiveX

To start using ActiveX objects, you need to create the object and obtain the information (properties, methods and events) about it.

### Creating an Instance of a ActiveX object

h = actxcontrol('progid', position, fig\_handle)

### Getting Information about Methods

To get a list of the methods of the object handle (h), type

h.methods

### Getting Information about Properties

To get a list of the properties, type

get(h)

To see the value of the property *PropertyName*, type

get(h, ‘*PropertyName*’)

### Getting information about Events

To get a list of the events supported, type

h.events

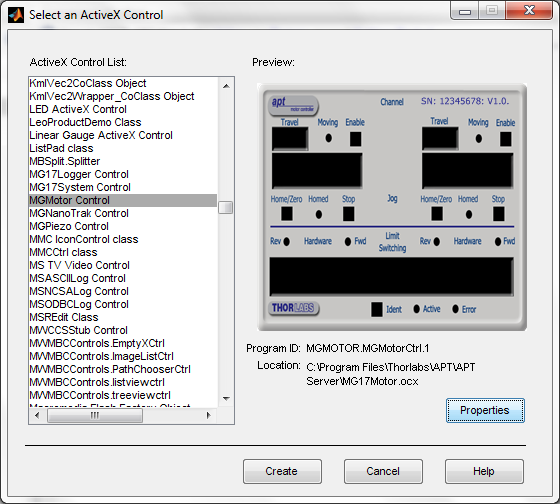
### Getting an Object’s ProgID

To use the actxcontrol command, you need to get the programmatic identifier (ProgID) of the ActiveX control that is already registered on your computer first. Use the command to list all the registered ActiveX ProgID. You can also use the ActiveX Control Selector, actxcontrolselect. Figure 2 shows the screen shot of choosing the APT step motor ActiveX control ProgID.

The ActiveX controls should be registered after you install the APT software package. If you cannot find the ProgID in the list, you must register it with the Microsoft Windows operation system. You can do this by typing in the MATLAB command:

!regsvr32 /s filename.ocx

where the filename is the name of the file containing the ActiveX control.



Figure

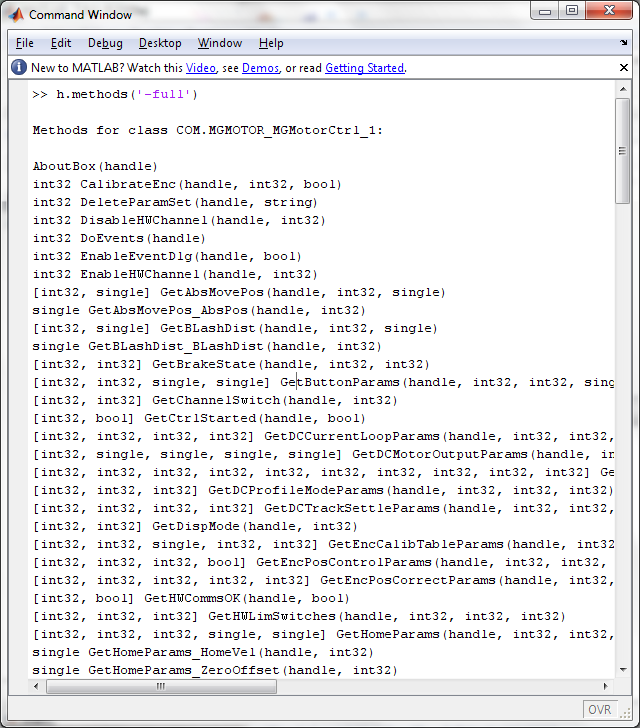
# Using Methods

You can execute, or invoke, ActiveX methods. To see what methods are supported by the ActiveX object, you can use methodsview, methods or invoke commands.

For example, type

h.methods(‘-full’)

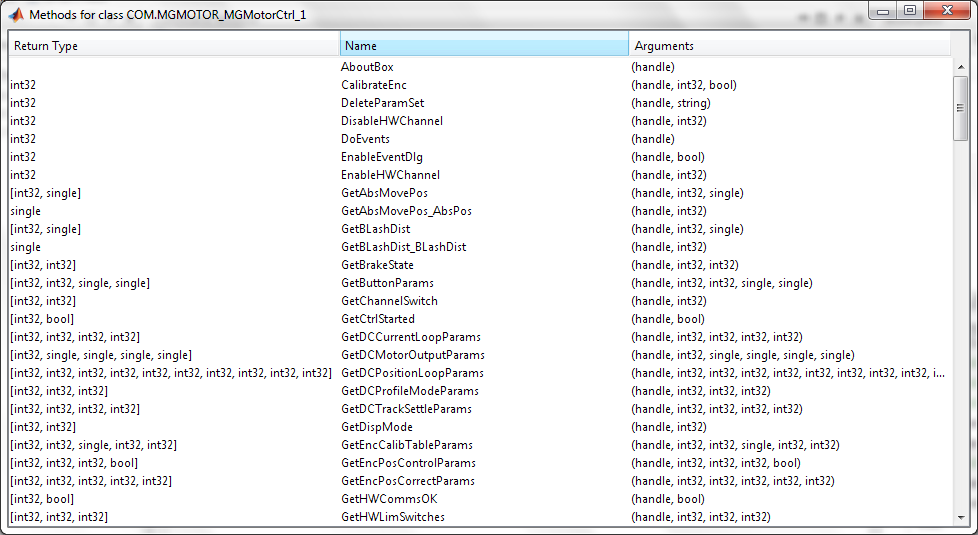
MATLAB displays as Fig. 3



Figure

Alternatively, you can use the methodsview command to open a new window with a table of all methods, as shown in Fig. 4.

Thorlabs also offers a detailed documentation on all the available methods. The document is a windows help file located in your hard drive after installation. The default position is C:\Program Files\Thorlabs\APT\APT Server\



Figure

### Calling Syntax

To invoke a method of an ActiveX object, use dot syntax:

outputvalue = object.methodname(‘arg1’, ’arg2’, …);

where the arguments, arg1, arg2, …., are defined in the document in the help file.

# Using Events

An event is typically a user-initiated action that takes place when an action is complete. For example, with a mouse, clicking a button triggers a “mouse click” event. An event handler program will be executed after the event is fired.

For the LTS150 stage in our example code, a text message “Moving Completed” is displayed after the move is completed. In a practical situation, the user may want to do more complicated applications, such as trigger camera and process the image data.

Instead of listening to the MoveComplete event, one can check the status of the motor by calling the method GetStatusBits\_Bits. It will return a number. The 5th bit shows if the motor shaft is moving clockwise (1- moving, 0 -stationary ), and the 6th bit shows if the motor shaft is moving counterclockwise (1- moving, 0 -stationary ). The function IsMoving.m reads the status number and check the two bits and return the status (1 – moving, 0 – moving completed).

# Appendix:

## Sample Code:

### APT\_GUI.m

clear; close all; clc;

global h; % make h a global variable so it can be used outside the main

% function. Useful when you do event handling and sequential move

%% Create Matlab Figure Container

fpos = get(0,'DefaultFigurePosition'); % figure default position

fpos(3) = 650; % figure window size;Width

fpos(4) = 450; % Height

f = figure('Position', fpos,...

'Menu','None',...

'Name','APT GUI');

%% Create ActiveX Controller

h = actxcontrol('MGMOTOR.MGMotorCtrl.1',[20 20 600 400 ], f);

%% Initialize

% Start Control

h.StartCtrl;

% Set the Serial Number

SN = 45822682; % put in the serial number of the hardware

set(h,'HWSerialNum', SN);

% Indentify the device

h.Identify;

pause(5); % waiting for the GUI to load up;

%% Controlling the Hardware

%h.MoveHome(0,0); % Home the stage. First 0 is the channel ID (channel 1)

% second 0 is to move immediately

%% Event Handling

h.registerevent({'MoveComplete' 'MoveCompleteHandler'});

%% Sending Moving Commands

timeout = 10; % timeout for waiting the move to be completed

%h.MoveJog(0,1); % Jog

% Move a absolute distance

h.SetAbsMovePos(0,7);

h.MoveAbsolute(0,1==0);

t1 = clock; % current time

while(etime(clock,t1)<timeout)

% wait while the motor is active; timeout to avoid dead loop

s = h.GetStatusBits\_Bits(0);

if (IsMoving(s) == 0)

pause(2); % pause 2 seconds;

h.MoveHome(0,0);

disp('Home Started!');

break;

end

end

#### MoveCompleteHandler.m

function MoveCompleteHandler(varargin)

pause(0.5); %dummy program

disp('Move Completed!');

#### IsMoving.m

function r = IsMoving(StatusBits)

% Read StatusBits returned by GetStatusBits\_Bits method and determine if

% the motor shaft is moving; Return 1 if moving, return 0 if stationary

r = bitget(abs(StatusBits),5)||bitget(abs(StatusBits),6);