

Getting started with Phidgets in Simulink

Environment and Libraries

Phidgets support development in Simulink for many types of Phidgets using MATLAB calls to the C library. First, we need to set up the proper environment and get the necessary files off the Phidgets website. Visit the drivers and programming section at <http://www.phidgets.com> and get the latest:

- Phidget Framework
- Simulink examples

You will need the phidget21Matlab.h from the MATLAB examples to program with Phidgets, and the Phidget Framework to use them. We also recommend that you download the following reference materials from the programming section:

- C API Manual
- Programming Manual
- The Product Manual for your device

The C API manual contains calls for every type of Phidget and can be used as a reference. You can find a high level discussion about programming with Phidgets in general in the Programming Manual. The Product manual for your device also contains an API section that describes limitations, defaults, and implementation details specific to your Phidget. You may want to have these manuals open while working through these instructions.

Due to the nature of Simulink, some function calls and Phidget classes may not be supported. These will be discussed later in the tutorial.

Simulink offers several different types of user defined blocks that can interact with MATLAB or C code. This tutorial will use MATLAB code to demonstrate how user defined functions can be used to interact with Phidgets through the use of MATLAB Fcn, M Level 1 S Functions, and M Level 2 S Function blocks. As writing C code is beyond the scope of this guide, C S Function blocks will not be discussed. However, C S Function blocks work similar to M S Function blocks.

MATLAB Fcn blocks are the easiest blocks to use, but have limited features(i.e., only 1 input port). Level 1 S Functions blocks have more features, but are more complicated to implement. Level 2 S Functions blocks are the most complex, but benefit in supporting the most features.

This tutorial will assume the reader is familiar with Simulink, Simulink user defined functions and interacting with Phidgets using MATLAB. For more information about interacting with Phidgets using MATLAB, please see the Getting Started Guide for MATLAB.

Setting up a Phidgets Project

The Phidget examples were written using Simulink 6.1/MATLAB 7 in Windows XP. Other recent versions should work as well and would be set up in a similar manner. To begin,

- The first step is to set the path for phidget21Matlab.h in MATLAB(File->Set Path). The

phidget21Matlab.h can be found in the MATLAB examples.

- Launch MATLAB and enter "simulink" in the MATLAB command window to open up the Simulink Library Browser.
- Create a new model

The project now has access to Phidgets and we are ready to begin coding.

Connecting to the Phidget

The Phidget object will need to be declared and initialized. For example, we can declare and initialize an instance of a PhidgetInterfaceKit inside a M file of a Level 1 S Function block with:

```
global handle;
loadlibrary('phidget21', 'phidget21Matlab.h');
ptr = libpointer('int32Ptr',0);

calllib('phidget21', 'CPhidgetInterfaceKit_create', ptr);
handle = get(ptr, 'Value');

calllib('phidget21', 'CPhidget_open', handle, -1)
calllib('phidget21', 'CPhidget_waitForAttachment', handle, 2500)
```

The handle variable is declared as a global, and can be easily accessed by other blocks. Since blocks repeatedly run until the simulation is completed, the above code will need to be restricted so that it runs only once during a simulation. It is placed inside the mdlInitializeSizes function.

At the end of the simulation, the Phidget object will need to be closed and deleted. The following lines are added to mdlTerminate.

```
calllib('phidget21', 'CPhidget_close', handle);
calllib('phidget21', 'CPhidget_delete', handle);
```

Working directly with the Phidget

In this section, a MATLAB Fcn block will be used to set the output states of the digital output ports on a PhidgetInterfaceKit 8/8/8. The following code is placed inside a M file. The block's input is the desired output state.

```
function setOutputStates(outputState)

    global handle;
    for i = 0:7
        if (calllib('phidget21', 'CPhidgetInterfaceKit_setOutputState', handle, i,
outputState) == 1)
            display('Set output state failed')
        end
    end
end
```

Note that this handle is the same handle that was created earlier.

Next, a M Level 2 S Function will be used to retrieve sensor values of analog sensor ports and output it to a scope. The block will contain no inputs and 1 output, with 8 dimensions(1 dimension for each analog sensor of a PhidgetInterfaceKit 8/8/8). The following code is placed in the Output function in the M file.

```
function Output(block)

    dataptr = libpointer('int32Ptr',0);
    global handle;
    for i = 0:7
        if (calllib('phidget21', 'CPhidgetInterfaceKit_getSensorValue', handle, i,
dataptr) == 0)
            block.OutputPort(1).Data(i+1) = get(dataptr, 'Value');
        end
    end
end
```

Event Driven Programming

MATLAB does not support event handling.

Working with multiple Phidgets

Multiple Phidgets of the same type can easily be run inside the same model. Each instance of a Phidget corresponding to a particular handle. Create more handles to work with more Phidgets. If the application needs to distinguish between the devices, open can be called with the serial number of a specific Phidget.

Other Phidgets

The design given in this document can also be followed for almost all Phidgets. For example, if you were using a PhidgetRFID instead of an PhidgetInterfacekit, you would declare an RFID object instead of an InterfaceKit. The methods and events available would change but they can be accessed in a similar manner.

Working with Strings

It is possible for a user to specify a string of characters in the block parameters dialog box of a user defined function such that it is passed into a user defined function. Please see the PhidgetInterfaceKit_RemoteIP block library in the Simulink examples for details.

Simulink cannot display strings in the GUI, however it is possible to write a C S function that displays strings in the MATLAB console.

Working in Real Time

Even though a model may finish simulating before a user has a chance to interact with Phidgets, there are workarounds that can be used to slow down simulation time to allow for user interaction. One method is to modify the Solver Options(under Simulation->Configuration Parameters->Solver). Changing the step size and stop time may slow down the simulation. Another method is to implement a user defined function that reduces the simulation speed until the system clock matches the

Simulink clock. The Phidget Simulink example uses the "Real Time Block" to simulate the model in approximate real time. For more information about the "Real Time Block", please see

<http://www.mathworks.com/support/solutions/en/data/1-15JAW/>

To use the "Real Time Block", place the following files in your MATLAB working directory: slblocks.m, rtc.mdl, waitforreal.m. These files can be found in the Simulink examples. Open up the Simulink Library Browser and select "Real Time Clock". Place the Real Time Clock block onto your model.

For applications where accurate real time simulation is needed, xPC Target can be used.

Please note that the simulation speed of these methods may differ depending on the CPU speed, complexity of the model, the amount of text being displayed in the MATLAB console, etc....