A library for Hardware-In-Loop for OpenModelica

FOSSEE, IIT Bombay

March 28, 2018

1 Introduction

This document has been written with the intent of familiarising a user with the Modelica H-I-L toolbox developed at FOSSEE. The InterProcessCommunication package, which is central to this toolbox, was developed at ModeliCon with the intent of communicating between two PCs. The team at FOSSEE modified it to function with an Arduino Mega and a PC.

The reason the IPC package was chosen over conventional packages is its ability to transfer data independent of data type. This comes in handy when dealing with the vast range of values that can be assumed while working with a controller such as a PID.

2 Directory Structure

The directory structure for the library is described below. Also as package.mo and package.order are present in multiple directories within InterProcessCommunication, and are largely irrelevant to direct usage, only those at the top level have been listed.

```
└─ ArduinoCode
   ├─ ArduIPCWrite
   │  └─ ArduIPCWrite.ino
   ├─ basic_ write
   │  └─ basic_ write.ino
   └─ IPC_ PID
      └─ IPC_ PID.ino
```

```
|
└─ InterProcessCommunication
   ├─ Examples
   │  ├─ CombinedExamples
   │  │  ├─ PIDandMotor.mo
   │  │  └─ PulsePIDandMotor.mo
   │  └─ InterProcessExamples
   │     ├─ ArduinoIPC.mo
   │     └─ DC_ Motor_ Arduino.mo
   ├─ Info
   │  ├─ Tutorial
   │  │  ├─ Advanced.mo
   │  │  └─ GettingStarted.mo
   │  ├─ Contact.mo
   │  └─ Overview.mo
   ├─ Resources
   │  ├─ Include
   │  │  ├─ Arduino_ port.sh
   │  │  ├─ SerialMI.h
   │  │  ├─ Serial_ SHM.c
   │  │  └─ ShmMI.c
   │  └─ Library
   │     └─ linux64
   │        ├─ librt.a
   │        └─ librt.so
   ├─ SharedMemory
   │  ├─ SharedMemoryRead.mo
   │  └─ SharedMemoryWrite.mo
   ├─ package.mo
   └─ package.order
```

3 Usage

The following steps detail how to use the IPC package to run real time HIL routines. This has been illustrated using a couple of examples. However, before any examples, the user is advised to replace the files librt.so and librt.a in Resources/Libraries/linux64 with files with the same name in the usr/lib/x86_ 64-linux-gnu folder of their systems. The ArduinoIPC example will demonstrate basic capabilities of the Shared Memory paradigm being used for HIL. It is assumed that the user is running OpenModelica as root.

- First, change directory to /InterProcessCommunication/Resources/Include
- Next, run gcc -o Serial_SHM Serial_SHM.c -lrt
- Next, flash ArduIPCBasic.ino on the Arduino.
- Run sudo bash Arduino_ port.sh | ./Serial_ SHM [baudrate of choice] on the terminal
- Set up the experiment on OpenModelica. A minimum of 10 seconds is advisable in this case.
- Go into the Simulation Flags menu in the Simulation setup, and adding a '-rt' flag in the Additional Simulation Flags textbox.
- Execute.

The DC_ Motor_ arduino example will be used to illustrate more detailed usage. It is assumed that the user is running OpenModelica as root and has the Modelica Device Drivers package loaded, if they wish to obtain the reference wave as well. The reference wave in question is being generated on a function generator. In the absence of one, a virtual reference wave can be generated by commenting out line 57 on IPC_ PID.ino and uncommenting line 56.

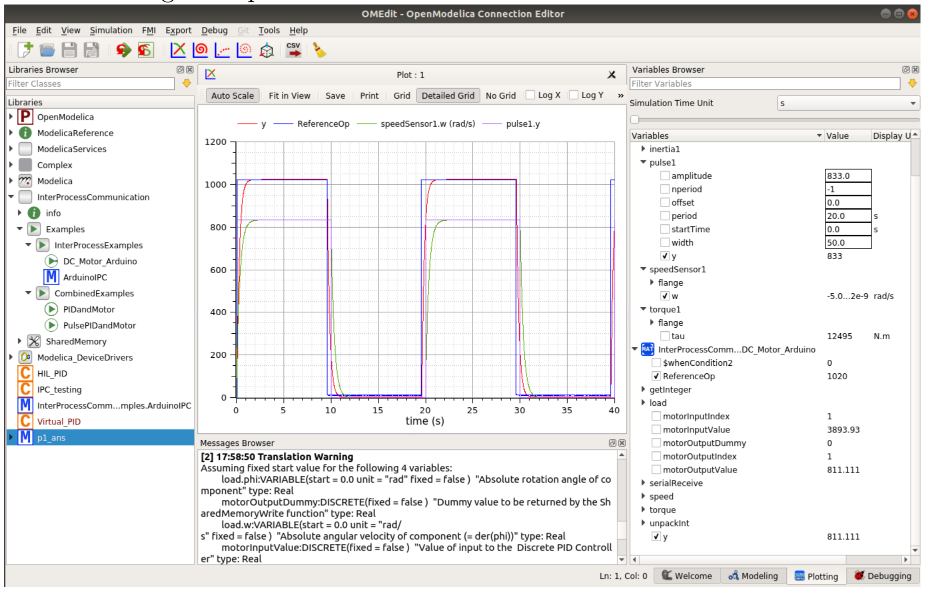
One of the bits of terminology that is being used here is that of the primary and secondary arduinos. The primary arduino is the one with the PID code, and the secondary arduino is the one that simply transmits a copy of the reference wave to OpenModelica.

- First, change directory to /InterProcessCommunication/Resources/Include
- Next, run gcc -o Serial_SHM Serial_SHM.c -lrt
- Next, flash IPC_ PID.ino on the primary Arduino.
- Flash basic_ write.ino on the secondary Arduino.
- If the user is making use of a function generator, connect it to pin A5 for both the arduinos, set a 4V sine wave with a period of 30 seconds or a square pulse with a similar period, on the function generator.

- If the user is not making use of a function generator,comment out line 6 on basic_ write.ino and uncomment line 7.
- Run sudo bash Arduino_ port.sh | ./Serial_ SHM [baudrate of choice] on the terminal
- Set up the experiment on OpenModelica. A minimum of 60 seconds is advisable in this case.
- Go into the Simulation Flags menu in the Simulation setup, and adding a '-rt' flag in the Additional Simulation Flags textbox.
- Execute.

4 Results

Here we will see the results of a successful run of the DC_ Motor_ arduino example, and compare the results with those obtained with a purely virtual run with the same PID tuning parameters. The amplitudes on the virtual run have been kept lower deliberately for ease of viewing. As is seen in the image, the results are identical, and indicate a fair degree of performance.


OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI Export Debug Git Tools Help
Libraries Browser
Plot : 1
Auto Scale Fit in View Save Print Grid Detailed Grid No Grid Log X Log Y
y ReferenceOp speedSensor1.w (rad/s) pulse1.y
time (s)
Variables Browser
Welcome Modeling Plotting Debugging

```
[2] 17:58:50 Translation Warning
Assuming fixed start value for the following 4 variables:
	load.phi:VARIABLE(start = 0.0 unit = "rad" fixed = false )  "Absolute rotation angle of co
mponent" type: Real
	motorOutputDummy:DISCRETE(fixed = false )  "Dummy value to be returned by the Sh
aredMemoryWrite function" type: Real
	load.w:VARIABLE(start = 0.0 unit = "rad/
s" fixed = false )  "Absolute angular velocity of component (= der(phi))" type: Real
	motorInputValue:DISCRETE(fixed = false )  "Value of input to the  Discrete PID Controll
er" type: Real
```

Simulation Time Unit: s

Variables	Value	Display Unit
inertia1		
pulse1		
amplitude	833.0	
nperiod	-1	
offset	0.0	
period	20.0	s
startTime	0.0	s
width	50.0	
✓ y	833	
speedSensor1		
flange		
✓ w	-5.0...2e-9	rad/s
torque1		
flange		
tau	12495	N.m
InterProcessComm...DC_Motor_Arduino		
$whenCondition2	0	
✓ ReferenceOp	1020	
getInteger		
load		
motorInputIndex	1	
motorInputValue	3893.93	
motorOutputDummy	0	
motorOutputIndex	1	
motorOutputValue	811.111	
serialReceive		
speed		
torque		
unpackInt		
✓ y	811.111	