X2C

LINZ CENTER OF MECHATRONICS GMBH

Project Documentation DemoApplication

September 6, 2019

Contents

Part I

X2C Model

1 Version Information

1.1 X2C

- X2C: Version 6.1.1740

1.2 Operating System

- OS: Windows 7 6.1

1.3 Scilab

- Scilab: Version 5.5.2.1427793548
- Java: Version 1.6.0_41

2 Model Structure

2.1 Xcos Model

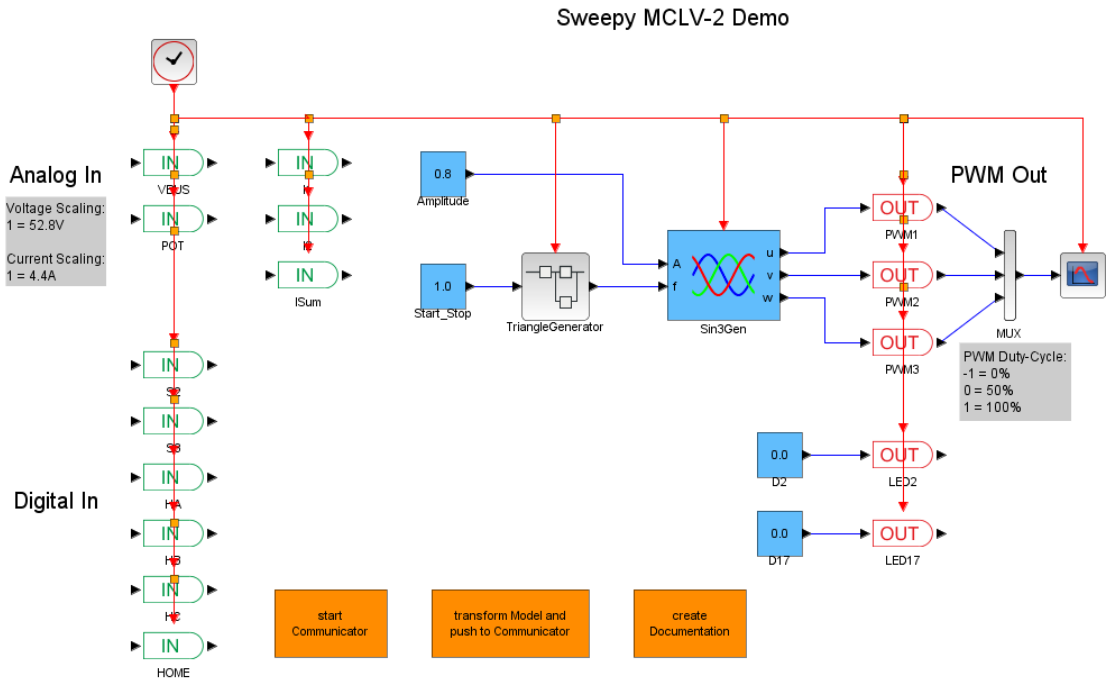


Figure 1: DemoApplication

2.2 Subsystems

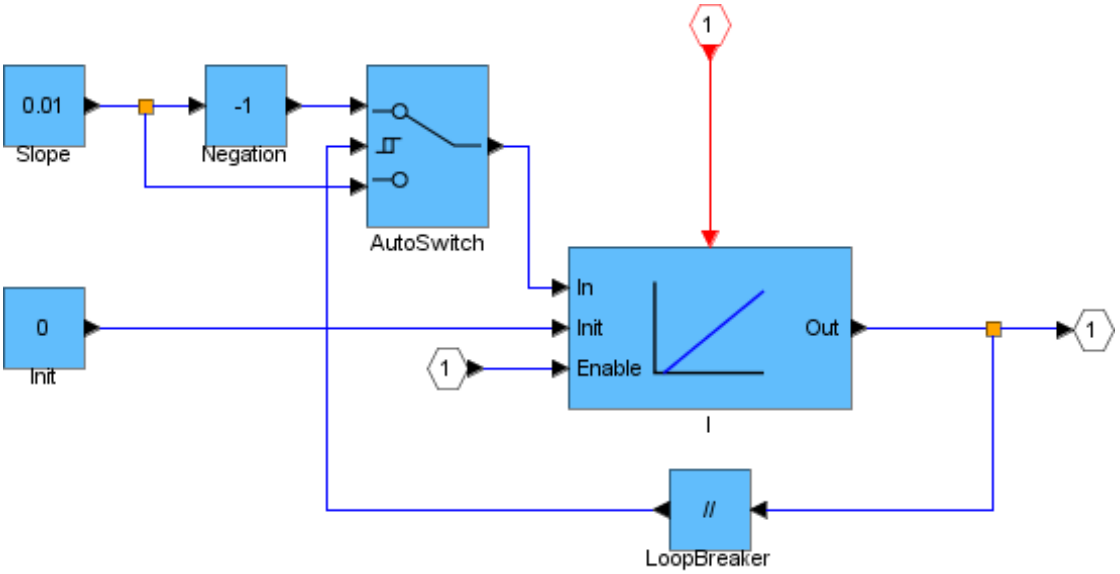


Figure 2: DemoApplication_TriangleGenerator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

3.2 Scilab Parameter

```
// File with model parameters such as sample time, scaling factors, etc...
//
// Copyright (c) 2019, Linz Center of Mechatronics GmbH (LCM) http://www.lcm.at/
// All rights reserved.
//
// This file is licensed according to the BSD 3-clause license as follows:
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//     * Redistributions of source code must retain the above copyright
//       notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above copyright
//       notice, this list of conditions and the following disclaimer in the
//       documentation and/or other materials provided with the distribution.
//     * Neither the name of the "Linz Center of Mechatronics GmbH" and "LCM" nor
//       the names of its contributors may be used to endorse or promote products
//       derived from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
// ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
// WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
// IN NO EVENT SHALL "Linz Center of Mechatronics GmbH" BE LIABLE FOR ANY
// DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
// LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
// ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
// SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// $LastChangedRevision: 1603 $
// $LastChangedDate:: 2019-01-21 19:02:13 +0100#$
//
// This file is part of X2C. https://x2c.lcm.at/

// Sampling time
X2C_sampleTime = 100e-6;    // 10kHz sampling frequency

// Scaling factors

// Controller parameters
```

Listing 1: ModelParameter.sce

4 Mask Parameter

Constant: Amplitude	
Value	0.8
Used Implementation	FiP16

Constant: D17	
Value	0.0
Used Implementation	Bool

Constant: D2	
Value	0.0
Used Implementation	Bool

Sin3Gen: Sin3Gen	
fmax	1000.0
Offset	0.0
ts_fact	1.0
Used Implementation	FiP16

Constant: Start_Stop	
Value	1.0
Used Implementation	Bool

AutoSwitch: AutoSwitch	
Thresh_up	0.03
Thresh_down	0.0
Used Implementation	FiP16

I: I	
Ki	1.0
ts_fact	1.0
Used Implementation	FiP16

Constant: Init	
Value	0.0
Used Implementation	FiP16

LoopBreaker: LoopBreaker	
Used Implementation	FiP16

Negation: Negation	
Used Implementation	FiP16

Constant: Slope	
Value	0.01
Used Implementation	FiP16

Part II
Frame Program Documentation

5 File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

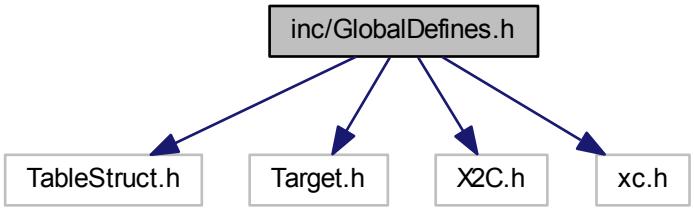
6 File Documentation

6.1 inc/GlobalDefines.h File Reference

Collection of globally needed defines.

```
#include "TableStruct.h"
#include "Target.h"
#include "X2C.h"
#include <xc.h>
```

Include dependency graph for GlobalDefines.h:



Macros

- #define X2C_SAMPLETIME SAMPLETIME_100US
- #define PWM_FREQUENCY PWM_20KHZ /∗ fPWM = 20kHz ∗/

6.1.1 Detailed Description

Collection of globally needed defines.
Available Preprocessor Definitions:

- none

6.1.2 Macro Definition Documentation

6.1.2.1 #define PWM_FREQUENCY PWM_20KHZ /∗ fPWM = 20kHz ∗/

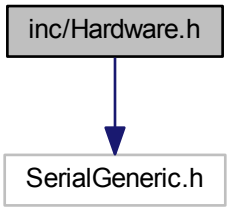
PWM frequency

6.1.2.2 #define X2C_SAMPLETIME SAMPLETIME_100US

Sample time

6.2 inc/Hardware.h File Reference

Hardware initialization.
`#include "SerialGeneric.h"`
Include dependency graph for Hardware.h:



Functions

- void initHardware (void)
  Hardware initialization.
- void initSerial (tSerial ∗serial)
  Initialization of serial interface.

6.2.1 Detailed Description

Hardware initialization.

6.2.2 Function Documentation

6.2.2.1 void initHardware (void)

Hardware initialization.

- Configuration of oscillator
  - Internal oscillator (fast RC oscillator with PLL)

  - fCY = 50MHz

- Configuration of serial port
  - Baudrate: 115.2kB/s
  - Data bits: 8
  - Parity: none
  - Stop bits: 1
- Configuration of IO ports
- Configuration of ADC
  - 12 bit resolution
  - internal RC clock source
  - AVdd as reference
  - unsigned, right justified output data format
  - Channels: AN0, ANA1, AN15, AN18, AN19
- Configuration of Timer 1 unit for sampling time (100us)
- Configuration of Timer 3 unit for CPU load measurement
- Configuration of PWM

6.2.2.2 void initSerial (tSerial * *serial*)

Initialization of serial interface.

Parameters

serial	Serial interface object.

6.3 inc/InputControl.h File Reference

Handling of inputs.

Functions

- void readAnalogIn (void)

  Routine to read values from ADC.
- void readDigitalIn (void)

  Routine to read digital input pins.

6.3.1 Detailed Description

Handling of inputs.

- Reading of digital inputs
- Reading of analog inputs

6.3.2 Function Documentation

6.3.2.1 void readAnalogIn (void)

Routine to read values from ADC.

- Bridge currents
- Supply voltage
- Potentiometer

6.3.2.2 void readDigitalIn (void)

Routine to read digital input pins.

- read push buttons
- read hall sensor signals
- read home signal

6.4 inc/Main.h File Reference

Main function.

Functions

- void mainTask (void)
  Main control task.

6.4.1 Detailed Description

Main function.

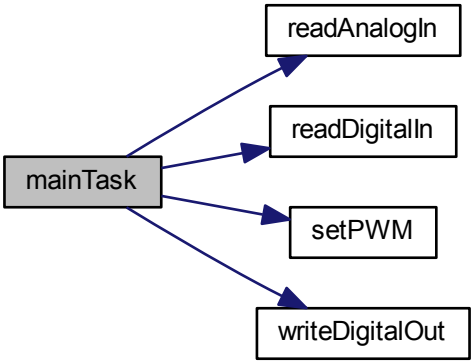
6.4.2 Function Documentation

6.4.2.1 void mainTask (void)

Main control task.
This task is/has to be called periodically. Calling rate = Sample time defined in GlobalDefines.h

- assign inports
- update X2C
- update outports

Here is the call graph for this function:



6.5 inc/OutputControl.h File Reference

Handling of outputs.

Functions

- void setPWM (void)
  Routine to set PWM duty cycle.
- void writeDigitalOut (void)
  Routine to write to digital output pins.

6.5.1 Detailed Description

Handling of outputs.

- Setting duty cycle of PWM signals
- Setting of digital outputs

6.5.2 Function Documentation

6.5.2.1 void setPWM (void)

Routine to set PWM duty cycle.

- Update duty cycle of PWM

6.5.2.2 void writeDigitalOut (void)

Routine to write to digital output pins.

- LEDs

Part III

Used X2C-Blocks

7 Project Specific Blocks

8 Internal Library Blocks

Block: AutoSwitch

Inports	
In1	Input #1
Switch	Input #2: Threshold signal
In3	Input #3

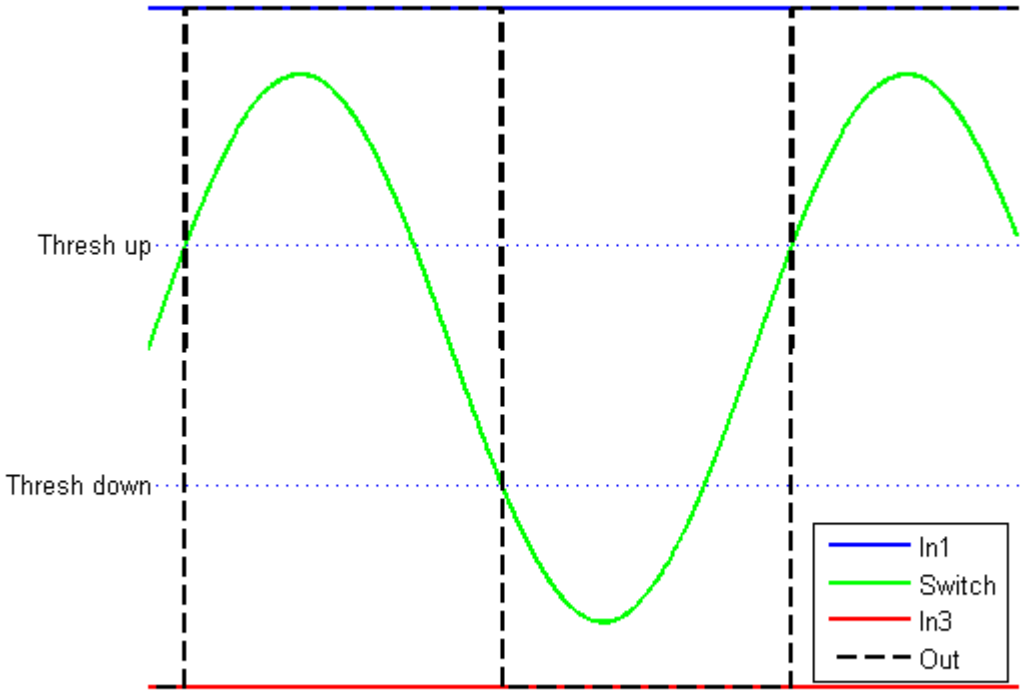
Outports	
Out	Either value of input #1 or input #3 dependent on value of input #2

Mask Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal

Description:

Switch between In1 and In3 dependent on Switch signal:
Switch signal rising: Switch >= Threshold up –> Out = In1
Switch signal falling: Switch < Threshold down –> Out = In3

The hysteresis behaviour of the block is illustrated in the figure below.



Implementations:

FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In1	int16
Switch	int16
In3	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In1	int32
Switch	int32
In3	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In1	float32
Switch	float32
In3	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
Switch	float64
In3	float64

Outports Data Type	
Out	float64

Block: Constant

Outports	
Out	Constant output

Mask Parameters	
Value	Constant factor

Description:

Constant value.

Implementations:

Bool	Boolean Implementation
Int8	8 Bit Integer Implementation
Int16	16 Bit Integer Implementation
Int32	32 Bit Integer Implementation
FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: Bool

Boolean Implementation

Outports Data Type	
Out	bool

Implementation: Int8

8 Bit Integer Implementation

Outports Data Type	
Out	int8

Implementation: Int16

16 Bit Integer Implementation

Outports Data Type	
Out	int16

Implementation: Int32

32 Bit Integer Implementation

Outports Data Type	
Out	int32

Implementation: FiP8

8 Bit Fixed Point Implementation

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

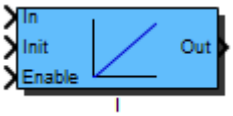
Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Outports Data Type	
Out	float64

Block: I



Inports	
In	Control error input
Init	Value which is loaded at initialization function call
Enable	Enable == 0: Deactivation of block; Out set to 0 Enable 0->1: Preload of integral part Enable == 1: Activation of block

Outports	
Out	Control value

Mask Parameters	
Ki	Integral Factor
ts_fact	Multiplication factor of base sampling time (in integer format)

Description:

I controller:
G(s) = Ki/s = 1/(Ti*s)

Each fixed point implementation uses the next higher integer datatype for the integrational value storage variable.
A rising flank at the *Enable* inport will preload the integrational part with the value present on the *Init* inport.

Transfer function (zero-order hold discretization method):

$$G(z) = K_{\mathrm{i}} T_{\mathrm{s}} \frac{1}{z-1}$$

Implementations:

FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In	int8
Init	int8
Enable	bool

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16
Init	int16
Enable	bool

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32
Init	int32
Enable	bool

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32
Init	float32
Enable	bool

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
Init	float64
Enable	bool

Outports Data Type	
Out	float64

Block: LoopBreaker

Inports	
In	Input In(k)

Outports	
Out	Output Out(k)=In(k-1)

Description:

Block to break algebraic loops.

Implementations:

Bool	Boolean Integration
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: Bool

Boolean Integration

Inports Data Type	
In	bool

Outports Data Type	
Out	bool

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Negation

Inports	
In	Input

Outports	
Out	Negated input value

Description:

Negation of input signal.

Calculation:

$$\mathrm{Out} = -\mathrm{In}$$

Implementations:

FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32

Outports Data Type	
Out	float32

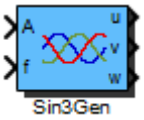
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Sin3Gen



Inports	
A	Amplitude
f	Frequency

Outports	
u	Sine wave output phase u
v	Sine wave output phase v
w	Sine wave output phase w

Mask Parameters	
fmax	Maximum Frequency in Hz
Offset	Offset
ts_fact	Multiplication factor of base sampling time (in integer format)

Description:

Generation of a 3 sine waves with amplitude (A) and frequency (f).

Calculation fixed point implementation:

$$
\begin{aligned}
u_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s}\right) + A_\mathrm{offset} \\
v_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s} - \frac{2\pi}{3}\right) + A_\mathrm{offset} \\
w_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s} + \frac{2\pi}{3}\right) + A_\mathrm{offset}
\end{aligned}
$$

For sine calculation a lookup table with 256 entries is used. This results in a short computation time but with the downside of reduced accuracy for the FiP32 implementation.

Calculation floating point implementation (parameter *f_max* is ignored):

$$
\begin{aligned}
u_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s}\right) + A_\mathrm{offset} \\
v_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s} - \frac{2\pi}{3}\right) + A_\mathrm{offset} \\
w_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s} + \frac{2\pi}{3}\right) + A_\mathrm{offset}
\end{aligned}
$$

Implementations:

FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
A	int16
f	int16

Outports Data Type	
u	int16
v	int16
w	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
A	int32
f	int32

Outports Data Type	
u	int32
v	int32
w	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
A	float32
f	float32

Outports Data Type	
u	float32
v	float32
w	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64
v	float64
w	float64