



# ***Project Documentation DemoApplication***

October 14, 2019

© Linz Center of Mechatronics GmbH

# Contents

<b>I</b>	<b>X2C Model</b>	<b>2</b>
<b>1</b>	<b>Version Information</b>	<b>2</b>
1.1	X2C . . . . .	2
1.2	Operating System . . . . .	2
1.3	Scilab . . . . .	2
<b>2</b>	<b>Model Structure</b>	<b>3</b>
2.1	Xcos Model . . . . .	3
2.2	Subsystems . . . . .	4
<b>3</b>	<b>Model Parameter</b>	<b>5</b>
3.1	Sample Time . . . . .	5
<b>4</b>	<b>Mask Parameter</b>	<b>6</b>
<b>II</b>	<b>Frame Program Documentation</b>	<b>8</b>
<b>5</b>	<b>File Index</b>	<b>8</b>
5.1	File List . . . . .	8
<b>6</b>	<b>File Documentation</b>	<b>8</b>
6.1	inc/Hardware.h File Reference . . . . .	8
6.1.1	Detailed Description . . . . .	8
6.1.2	Function Documentation . . . . .	9
6.2	inc/Main.h File Reference . . . . .	9
6.2.1	Detailed Description . . . . .	9
6.2.2	Function Documentation . . . . .	10
<b>III</b>	<b>Used X2C-Blocks</b>	<b>11</b>
<b>7</b>	<b>Project Specific Blocks</b>	<b>11</b>
<b>8</b>	<b>Internal Library Blocks</b>	<b>11</b>
	AutoSwitch . . . . .	11
	Constant . . . . .	14
	Delay . . . . .	17
	I . . . . .	19
	Negation . . . . .	22
	Sin3Gen . . . . .	24
	SinGen . . . . .	27

## **Part I**

# **X2C Model**

## **1 Version Information**

### **1.1 X2C**

- X2C: Version 6.1.1749

### **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

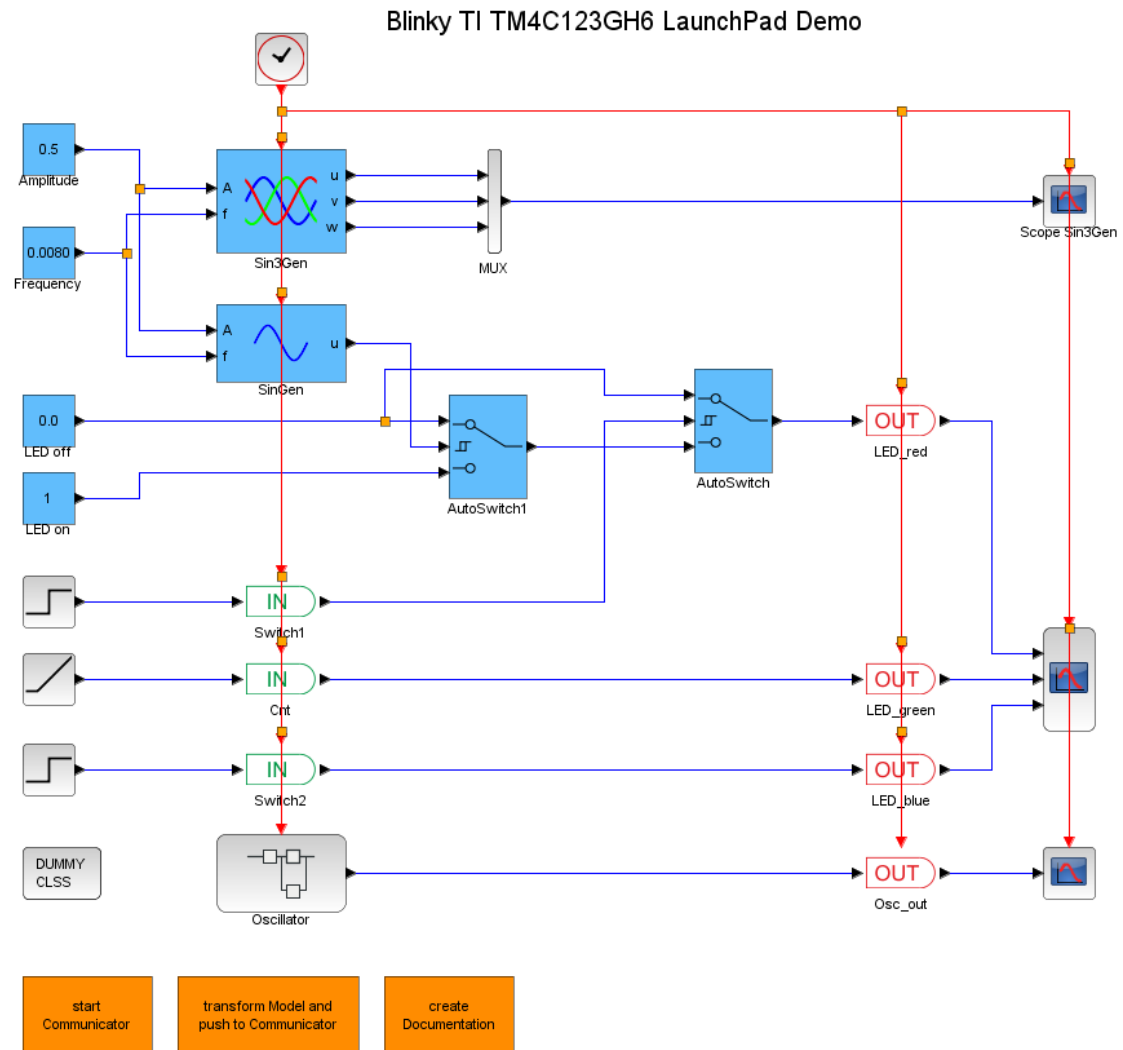


Figure 1: DemoApplication

## 2.2 Subsystems

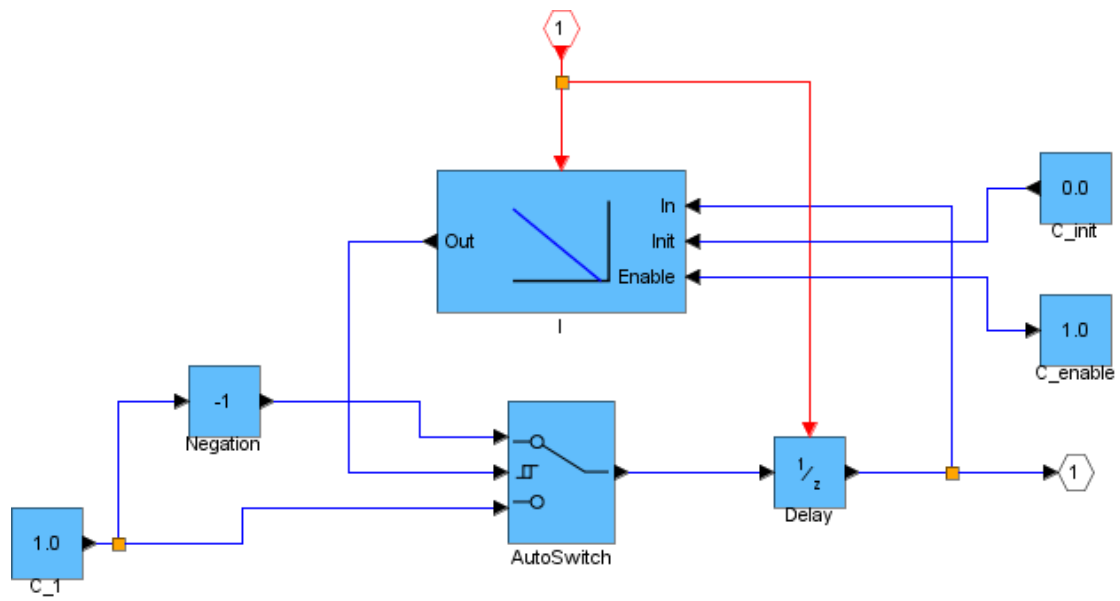


Figure 2: DemoApplication\_Oscillator

### 3 Model Parameter

#### 3.1 Sample Time

Sample Time	
$T_S$	$200\mu s$

## 4 Mask Parameter

Constant: Amplitude	
Value	0.5
Used Implementation	FiP16

AutoSwitch: AutoSwitch	
Thresh_up	0.6
Thresh_down	0.4
Used Implementation	FiP16

AutoSwitch: AutoSwitch1	
Thresh_up	0.0
Thresh_down	0.0
Used Implementation	FiP16

Constant: Frequency	
Value	0.0080
Used Implementation	FiP16

Constant: LED off	
Value	0.0
Used Implementation	FiP16

Constant: LED on	
Value	1.0
Used Implementation	FiP16

AutoSwitch: AutoSwitch	
Thresh_up	0.5
Thresh_down	-0.5
Used Implementation	FiP16

Constant: C_1	
Value	1.0
Used Implementation	FiP16

Constant: C_enable	
Value	1.0
Used Implementation	Bool

<b>Constant: C_init</b>	
Value	0.0
Used Implementation	FiP16

<b>Delay: Delay</b>	
ts_fact	1.0
Used Implementation	FiP16

<b>I: I</b>	
Ki	50.0
ts_fact	1.0
Used Implementation	FiP16

<b>Negation: Negation</b>	
Used Implementation	FiP16

<b>Sin3Gen: Sin3Gen</b>	
fmax	1000.0
Offset	0.0
ts_fact	1.0
Used Implementation	FiP16

<b>SinGen: SinGen</b>	
fmax	1000.0
Offset	0.0
Phase	0.0
ts_fact	1.0
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:

<a href="#">inc/Hardware.h</a>	Hardware configuration	8
<a href="#">inc/Main.h</a>	Main function	9

## 6 File Documentation

### 6.1 inc/Hardware.h File Reference

Hardware configuration.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_memmap.h"
#include "driverlib/rom.h"
#include "driverlib/uart.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/sysctl.h"
#include "driverlib/watchdog.h"
#include "SerialGeneric.h"
```

Include dependency graph for Hardware.h:



### Functions

- void [initHardware](#) (void)  
*Initialization of hardware.*
- void [initSerial](#) (tSerial \*serial)  
*Initialization of serial interface.*

#### 6.1.1 Detailed Description

Hardware configuration.

## 6.1.2 Function Documentation

### 6.1.2.1 void initHardware ( void )

Initialization of hardware.

- Configuration of system clock:
  - 16MHz external quartz
  - PLL
  - 2.5 frequency divider
  - -> 80 MHz system clock
- Configuration of serial interface:
  - UART0
  - 115200 baud
  - 8 data bits
  - 1 stop bit
  - no parity
- Configuration of I/O-ports:
  - PF1, PF2 & PF3 as outputs for LEDs
  - PF0 & PF4 as inputs for switch buttons
- Configuration of 32-bit timer 0 for interrupt generation:
  - periodic mode
  - 5 kHz
- Enable processor interrupts
- Enable timers

### 6.1.2.2 void initSerial ( tSerial \* serialP )

Initialization of serial interface.

Parameters

<i>serialP</i>	Serial object
----------------	---------------

## 6.2 inc/Main.h File Reference

Main function.

### Functions

- void [mainTask](#) (void)  
*Main control task.*

### 6.2.1 Detailed Description

Main function.

## **6.2.2 Function Documentation**

### **6.2.2.1 void mainTask ( void )**

Main control task.

The main control task is called by the timer 0 interrupt service routine with a frequency of 5kHz.

- Assign inports
- Update X2C
- Update outports

## 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

Outputs	
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

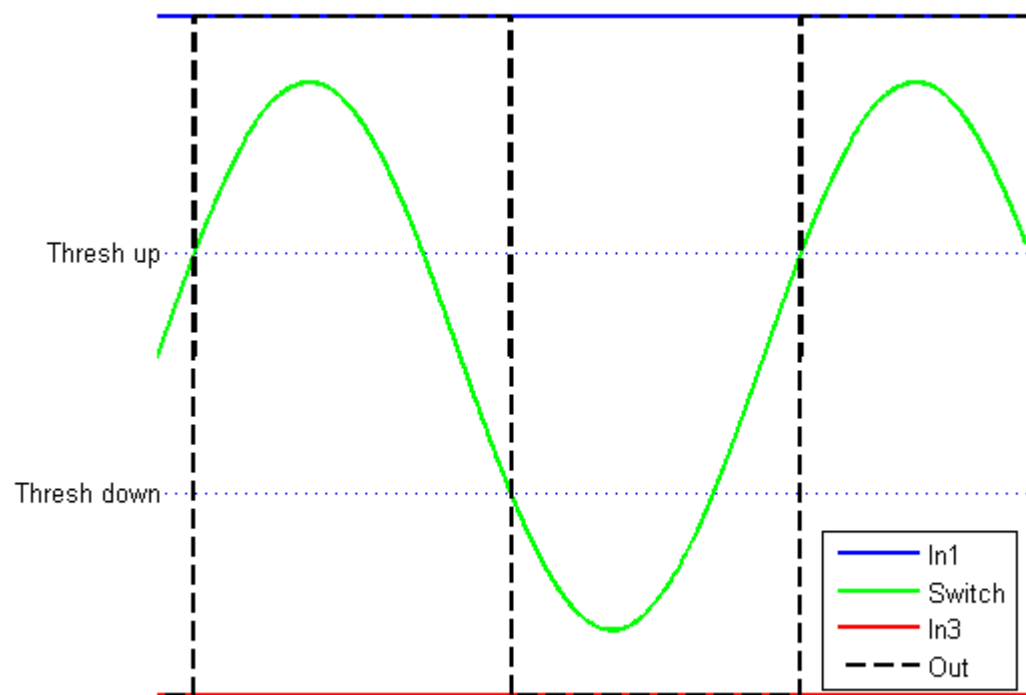
#### Description:

Switch between In1 and In3 dependent on Switch signal:

Switch signal rising: Switch  $\geq$  Threshold up  $\rightarrow$  Out = In1

Switch signal falling: Switch  $<$  Threshold down  $\rightarrow$  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

---



Outputs	
Out	Constant output

Mask Parameters	
Value	Constant factor

### Description:

Constant value.

### Implementations:

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

### Implementation: Bool

---

Boolean Implementation

Outputs Data Type	
Out	bool

### Implementation: Int8

---

8 Bit Integer Implementation

Outputs 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: Delay

---



Inports	
In	Input In(k)

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

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

### Description:

Output delay by one sample time interval.

This block can be used to enable feedback loops in the model.

### Implementations:

<b>Bool</b>	Boolean Integration
<b>FiP16</b>	16 Bit Fixed Point Implementation
<b>FiP32</b>	32 Bit Fixed Point Implementation
<b>Float32</b>	32 Bit Floating Point Implementation
<b>Float64</b>	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

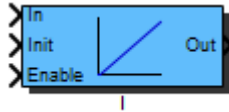
64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

## Block: 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

Outputs	
Out	Control value

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

### Description:

I controller:

$$G(s) = K_i/s = 1/(T_i \cdot s)$$

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):

$$G(z) = K_i T_s \frac{1}{z - 1}$$

### Implementations:

<b>FiP8</b>	8 Bit Fixed Point Implementation
<b>FiP16</b>	16 Bit Fixed Point Implementation
<b>FiP32</b>	32 Bit Fixed Point Implementation
<b>Float32</b>	32 Bit Floating Point Implementation
<b>Float64</b>	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: Negation

---



Inports	
In	Input

Outputs	
Out	Negated input value

### Description:

Negation of input signal.

Calculation:

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

### Implementations:

<b>FiP8</b>	8 Bit Fixed Point Implementation
<b>FiP16</b>	16 Bit Fixed Point Implementation
<b>FiP32</b>	32 Bit Fixed Point Implementation
<b>Float32</b>	32 Bit Floating Point Implementation
<b>Float64</b>	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

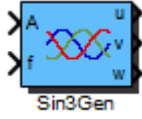
64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64



## Block: Sin3Gen



Inports	
A	Amplitude
f	Frequency

Outputs	
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:

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

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):

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

**Implementations:**

<b>FiP16</b>	16 Bit Fixed Point Implementation
<b>FiP32</b>	32 Bit Fixed Point Implementation
<b>Float32</b>	32 Bit Floating Point Implementation
<b>Float64</b>	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

---

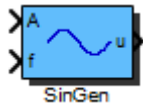
64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64
v	float64
w	float64

## Block: SinGen

---



Inports	
A	Amplitude
f	Frequency

Outputs	
u	Sine wave output

Mask Parameters	
fmax	Maximum Frequency in Hz
Offset	Offset
Phase	Phase [-Pi..Pi]
ts_fact	Multiplication factor of base sampling time (in integer format)

### Description:

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

Calculation fixed point implementation:

$$u_k = A_k \sin(2f_k f_{\max} kT_s + \phi_{\text{phase}}) + A_{\text{offset}}$$

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):

$$u_k = A_k \sin(2\pi f_k kT_s + \phi_{\text{phase}}) + A_{\text{offset}}$$

### Implementations:

<b>FiP16</b>	16 Bit Fixed Point Implementation
<b>FiP32</b>	32 Bit Fixed Point Implementation
<b>Float32</b>	32 Bit Floating Point Implementation
<b>Float64</b>	64 Bit Floating Point Implementation

### Implementation: FiP16

---

16 Bit Fixed Point Implementation

Inports Data Type	
A	int16
f	int16

Outports Data Type	
u	int16

### Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
A	int32
f	int32

Outports Data Type	
u	int32

### Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
A	float32
f	float32

Outports Data Type	
u	float32

### Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64