Project Documentation DemoApplication

May 3, 2023

Contents

Part I
X2C Model

1 Version Information

1.1 X2C

- X2C Development: Version 6.4.2826

1.2 Operating System

- OS: Windows 10 10.0

1.3 Scilab

- Scilab: Version 6.1.1.1626343451
- Java: Version 1.8.0_292

2 Model Structure

2.1 Xcos Model

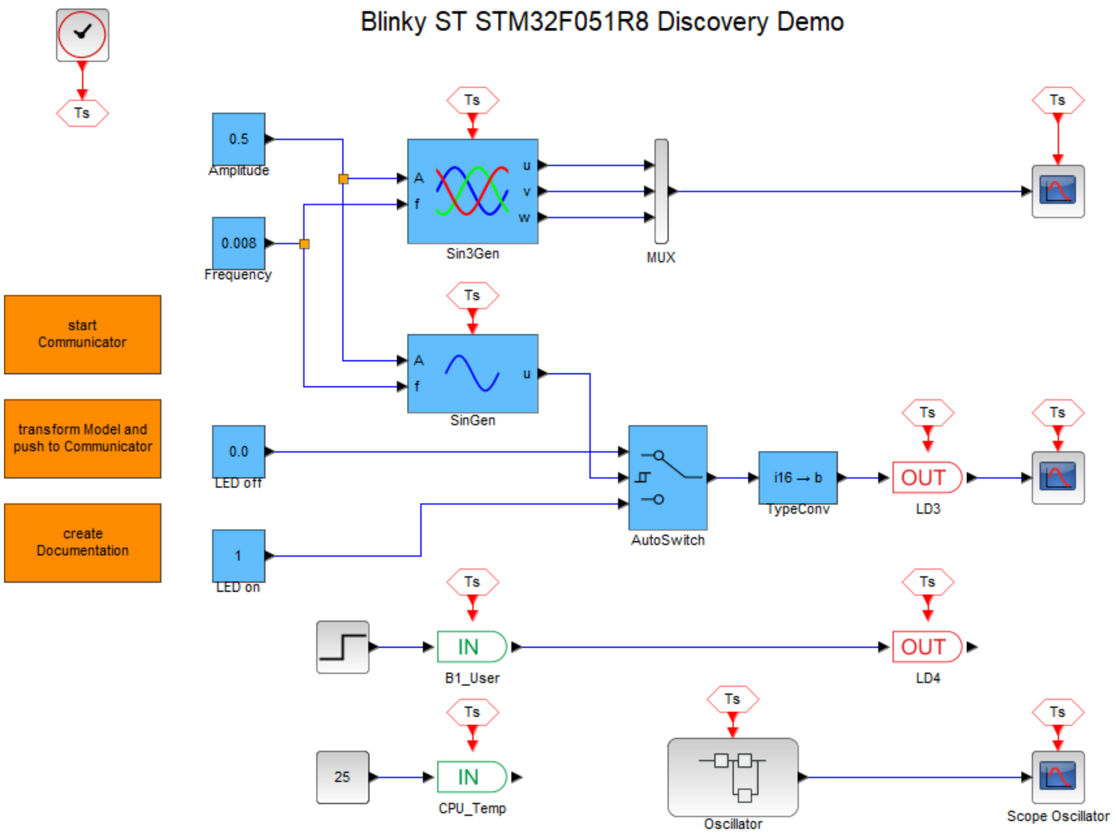


Figure 1: DemoApplication

2.2 Subsystems

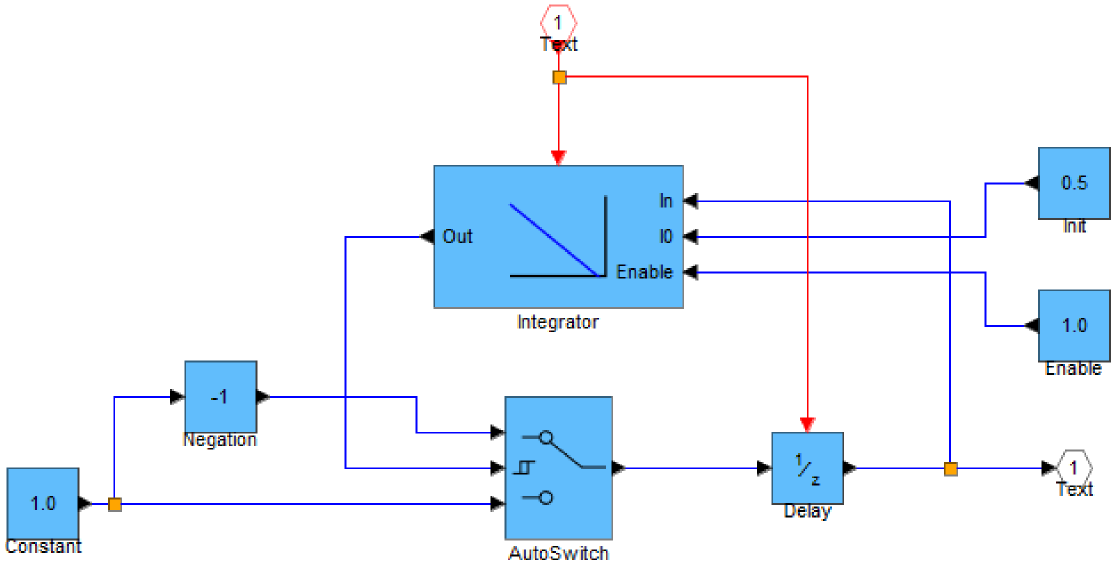


Figure 2: Oscillator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

4 Mask Parameter

4.1 Inports

4.1.1 Inports with auto generated ID

B1_User	
Block Type	Inport - bool
ts_fact	1.0
Simulation Gain	1.0
Simulation Offset	0.0

CPU_Temp	
Block Type	Inport - int16
ts_fact	1.0
Simulation Gain	1.0
Simulation Offset	0.0

4.2 Outports

4.2.1 Outports with auto generated ID

LD3	
Block Type	Outport - bool
ts_fact	1.0
Simulation Gain	1.0
Simulation Offset	0.0

LD4	
Block Type	Outport - bool
ts_fact	1.0
Simulation Gain	1.0
Simulation Offset	0.0

4.3 Blocks

4.3.1 Blocks with auto generated ID

Amplitude	
Block Type	Constant - FiP16
Value	0.5

AutoSwitch	
Block Type	AutoSwitch - FiP16
Thresh_up	0.0
Thresh_down	0.0

Frequency	
Block Type	Constant - FiP16
Value	0.008

LED off	
Block Type	Constant - FiP16
Value	0.0

LED on	
Block Type	Constant - FiP16
Value	1.0

Oscillator/AutoSwitch	
Block Type	AutoSwitch - FiP16
Thresh_up	0.5
Thresh_down	-0.5

Oscillator/Constant	
Block Type	Constant - FiP16
Value	1.0

Oscillator/Delay	
Block Type	Delay - FiP16
ts_fact	1.0

Oscillator/Enable	
Block Type	Constant - Bool
Value	1.0

Oscillator/Init	
Block Type	Constant - FiP16
Value	0.5

Oscillator/Integrator	
Block Type	I - FiP16
Ki	50.0
ts_fact	1.0

Oscillator/Negation	
Block Type	Negation - FiP16

Sin3Gen	
Block Type	Sin3Gen - FiP16
fmax	1000.0
Offset	0.0
ts_fact	1.0

SinGen	
Block Type	SinGen - FiP16
fmax	1000.0
Offset	0.0
Phase	0.0
ts_fact	1.0

TypeConv	
Block Type	TypeConv - FiP16_Bool

Part II
Frame Program Documentation

5 File Index

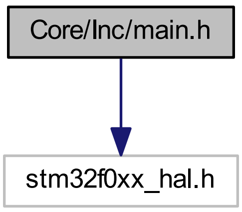
5.1 File List

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

6 File Documentation

6.1 Core/Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.
`#include "stm32f0xx_hal.h"`
Include dependency graph for main.h:



Macros

- #define PWM_FREQUENCY (float)10e3
- #define CONTROL_TASK_FREQUENCY_DIVIDER 1

6.1.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

6.1.2 Macro Definition Documentation

6.1.2.1 #define CONTROL_TASK_FREQUENCY_DIVIDER 1

Ratio between sample frequency and PWM frequency

6.1.2.2 #define PWM_FREQUENCY (float)10e3

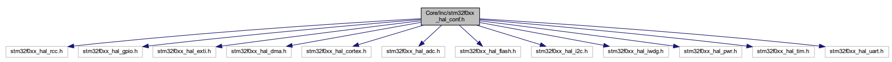
Though PWM frequency and and ADC trigger rate can be adjusted in the STM32CubeIDE, the settings will be overwritten with manually written code in main.c.
In STM32CubeIDE the PWM frequency can be found in TIM1 -> Parameter Settings -> Counter Period. Because of center aligned mode, this value can be calculated this way (example for 10 kHz): fTimer / desired PWM frequency / 2 = 48e6 / 10e3 / 2 = 2400.
The ADC trigger rate can be controlled by the TIM8 repetition counter (RCR). This timer causes an Update Event (UEV) on every over- and underflow. In center aligned mode, the timer counts in up- and down count mode and therefore causes an UEV twice per period. To get 1 ADC trigger per period, the RCR has to be set to 1. To get 1 ADC trigger on every 2nd period, the RCR has to be set to 3.PWM frequency of power stage [Hz]

6.2 Core/Inc/stm32f0xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f0xx_hal_rcc.h"
#include "stm32f0xx_hal_gpio.h"
#include "stm32f0xx_hal_exti.h"
#include "stm32f0xx_hal_dma.h"
#include "stm32f0xx_hal_cortex.h"
#include "stm32f0xx_hal_adc.h"
#include "stm32f0xx_hal_flash.h"
#include "stm32f0xx_hal_i2c.h"
#include "stm32f0xx_hal_iwdg.h"
#include "stm32f0xx_hal_pwr.h"
#include "stm32f0xx_hal_tim.h"
#include "stm32f0xx_hal_uart.h"
```

Include dependency graph for stm32f0xx_hal_conf.h:



Macros

- #define HAL_MODULE_ENABLED

  This is the list of modules to be used in the HAL driver.

- #define HSE_VALUE ((uint32_t)8000000)

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

- #define HSE_STARTUP_TIMEOUT ((uint32_t)100)

  In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.
- #define HSI_VALUE ((uint32_t)8000000)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI_STARTUP_TIMEOUT ((uint32_t)5000)

  In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.
- #define HSI14_VALUE ((uint32_t)14000000)

  Internal High Speed oscillator for ADC (HSI14) value.
- #define HSI48_VALUE ((uint32_t)48000000)

  Internal High Speed oscillator for USB (HSI48) value.
- #define LSI_VALUE ((uint32_t)40000)

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE ((uint32_t)32768)

  External Low Speed oscillator (LSI) value.
- #define LSE_STARTUP_TIMEOUT ((uint32_t)5000)

  Time out for LSE start up value in ms.
- #define VDD_VALUE ((uint32_t)3300)

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)0)
- #define assert_param(expr) ((void)0U)

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

6.2.1 Detailed Description

HAL configuration file.

Attention

6.2.2 Macro Definition Documentation

6.2.2.1 #define assert_param(*expr*) ((void)0U)

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
Include module's header file

6.2.2.2 #define HSE_STARTUP_TIMEOUT ((uint32_t)100)

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.
Time out for HSE start up, in ms

6.2.2.3 #define HSE_VALUE ((uint32_t)8000000)

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
Value of the External oscillator in Hz

6.2.2.4 #define HSI14_VALUE ((uint32_t)14000000)

Internal High Speed oscillator for ADC (HSI14) value.
Value of the Internal High Speed oscillator for ADC in Hz. The real value may vary depending on the variations in voltage and temperature.

6.2.2.5 #define HSI48_VALUE ((uint32_t)48000000)

Internal High Speed oscillator for USB (HSI48) value.
Value of the Internal High Speed oscillator for USB in Hz. The real value may vary depending on the variations in voltage and temperature.

6.2.2.6 #define HSI_STARTUP_TIMEOUT ((uint32_t)5000)

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.
Time out for HSI start up

6.2.2.7 #define HSI_VALUE ((uint32_t)8000000)

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
Value of the Internal oscillator in Hz

6.2.2.8 #define LSE_STARTUP_TIMEOUT ((uint32_t)5000)

Time out for LSE start up value in ms.
Time out for LSE start up, in ms

6.2.2.9 #define LSE_VALUE ((uint32_t)32768)

External Low Speed oscillator (LSI) value.
< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External Low Speed oscillator in Hz

6.2.2.10 #define TICK_INT_PRIORITY ((uint32_t)0)

tick interrupt priority (lowest by default)

6.2.2.11 #define VDD_VALUE ((uint32_t)3300)

This is the HAL system configuration section.
Value of VDD in mv

6.3 Core/Inc/stm32f0xx_it.h File Reference

This file contains the headers of the interrupt handlers.

6.3.1 Detailed Description

This file contains the headers of the interrupt handlers.

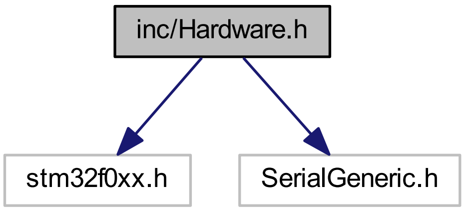
Attention

6.4 inc/Hardware.h File Reference

Hardware configuration.

`#include <stm32f0xx.h>`
`#include "SerialGeneric.h"`

Include dependency graph for Hardware.h:



Functions

- void initSerial (tSerial *serialSTM32F0)

    Initialization of serial interface.

6.4.1 Detailed Description

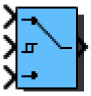
Hardware configuration.

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		
Name	**ID**	**Description**
Thresh_up	1	Threshold level for rising switch signal
Thresh_down	2	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

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		
Name	**ID**	**Description**
Value	1	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

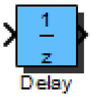
Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Outports Data Type	
Out	float64

Block: Delay



Inports	
In	Input In(k)

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

Mask Parameters		
Name	**ID**	**Description**
ts_fact	1	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:

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

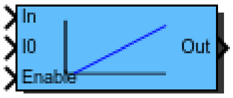
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: I



Inports	
In	Control error input
I0	Integral 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		
Name	**ID**	**Description**
Ki	1	Integral Factor
ts_fact	2	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):

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

8.0.0.1 Developer note:

The source code of block *ILimit* is used.

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	
In	int16
I0	int16
Enable	bool

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32
I0	int32
Enable	bool

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32
I0	float32
Enable	bool

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
I0	float32
Enable	bool

Outports Data Type	
Out	float64

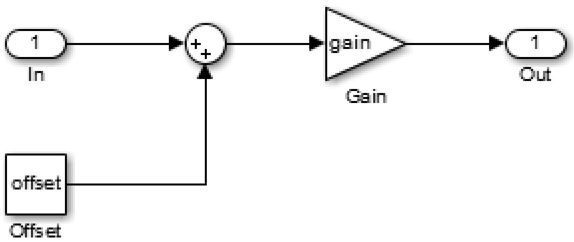
Block: Inport

Inports	
IN	Signal from frame program

Mask Parameters	
ts_fact	Multiplication factor of base sampling time (in integer format)
Gain	Gain value used in simulation
Offset	Offset value used in simulation

Description:

Serves as interface to the frame program. The input of this block is intended for simulation purposes and can be left unconnected if not used. Also the parameters *Gain* and *Offset* are only used during simulation. The schematic for simulation can be seen in the figure below.
Note: Currently, *Gain* and *Offset* parameters are only available in Matlab/Simulink.



Data Types:

int8	8 Bit Fixed Point
int16	16 Bit Fixed Point
int32	32 Bit Fixed Point
float32	32 Bit Floating Point
float64	64 Bit Floating Point

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

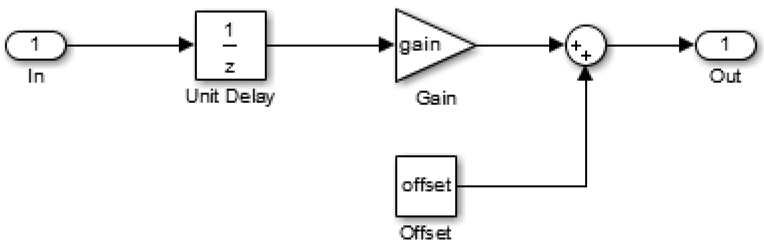
Block: Outport

Outports	
OUT	Signal to frame program

Mask Parameters	
ts_fact	Multiplication factor of base sampling time (in integer format)
Gain	Gain value used in simulation
Offset	Offset value used in simulation

Description:

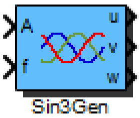
Serves as interface to the frame program. The output of this block is intended for simulation purposes and can be left unconnected if not used. Also the parameters *Gain*, and *Offset* are only used during simulation. The schematic for simulation can be seen in the figure below. The Unit Delay block is only used during simulation and should reflect the time delay caused by a discrete controller.
Note: Currently, *Gain* and *Offset* parameters are only available in Matlab/Simulink.



Data Types:

int8	8 Bit Fixed Point
int16	16 Bit Fixed Point
int32	32 Bit Fixed Point
float32	32 Bit Floating Point
float64	64 Bit Floating Point

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		
Name	**ID**	**Description**
fmax	1	Maximum Frequency in Hz
Offset	2	Offset
ts_fact	3	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

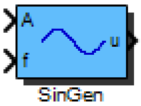
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64
v	float64
w	float64

Block: SinGen



Inports	
A	Amplitude
f	Frequency

Outports	
u	Sine wave output

Mask Parameters		
Name	**ID**	**Description**
fmax	1	Maximum Frequency in Hz
Offset	2	Offset
Phase	3	Phase [-Pi..Pi]
ts_fact	4	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:

$$u_k = A_k \sin\left(2 f_k f_{\mathrm{max}} k T_{\mathrm{s}} + \phi_{\mathrm{phase}}\right) + A_{\mathrm{offset}}$$

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):

$$u_k = A_k \sin\left(2\pi f_k k T_{\mathrm{s}} + \phi_{\mathrm{phase}}\right) + A_{\mathrm{offset}}$$

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

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

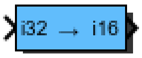
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64

Block: TypeConv



Inports	
In	

Outports	
Out	

Description:

Data Type Conversion

Implementations:

FiP8_16	8 to 16 Bit Fixed Point Implementation
FiP8_32	8 to 32 Bit Fixed Point Implementation
FiP16_8	16 to 8 Bit Fixed Point Implementation
FiP16_32	16 to 32 Bit Fixed Point Implementation
FiP32_8	32 to 8 Bit Fixed Point Implementation
FiP32_16	32 to 16 Bit Fixed Point Implementation
Bool_FiP16	Boolean to 16 Bit Fixed Point Implementation
Bool_FiP32	Boolean to 32 Bit Fixed Point Implementation
FiP16_Bool	16 Bit Fixed Point to Boolean Implementation
FiP32_Bool	32 Bit Fixed Point to Boolean Implementation

Implementation: FiP8_16

8 to 16 Bit Fixed Point Implementation

Inports Data Type	
In	int8

Outports Data Type	
Out	int16

Implementation: FiP8_32

8 to 32 Bit Fixed Point Implementation

Inports Data Type	
In	int8

Outports Data Type	
Out	int32

Implementation: FiP16_8

16 to 8 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int8

Implementation: FiP16_32

16 to 32 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int32

Implementation: FiP32_8

32 to 8 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int8

Implementation: FiP32_16

32 to 16 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int16

Implementation: Bool_FiP16

Boolean to 16 Bit Fixed Point Implementation

Inports Data Type	
In	bool

Outports Data Type	
Out	int16

Implementation: Bool_FiP32

Boolean to 32 Bit Fixed Point Implementation

Inports Data Type	
In	bool

Outports Data Type	
Out	int32

Implementation: FiP16_Bool

16 Bit Fixed Point to Boolean Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	bool

Implementation: FiP32_Bool

32 Bit Fixed Point to Boolean Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	bool