X2C

LINZ CENTER OF MECHATRONICS GMBH

Project Documentation DemoApplication

July 23, 2018

Contents

Part I

X2C Model

1 Version Information

1.1 X2C

- X2Cfull: Version 6.0.1498

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

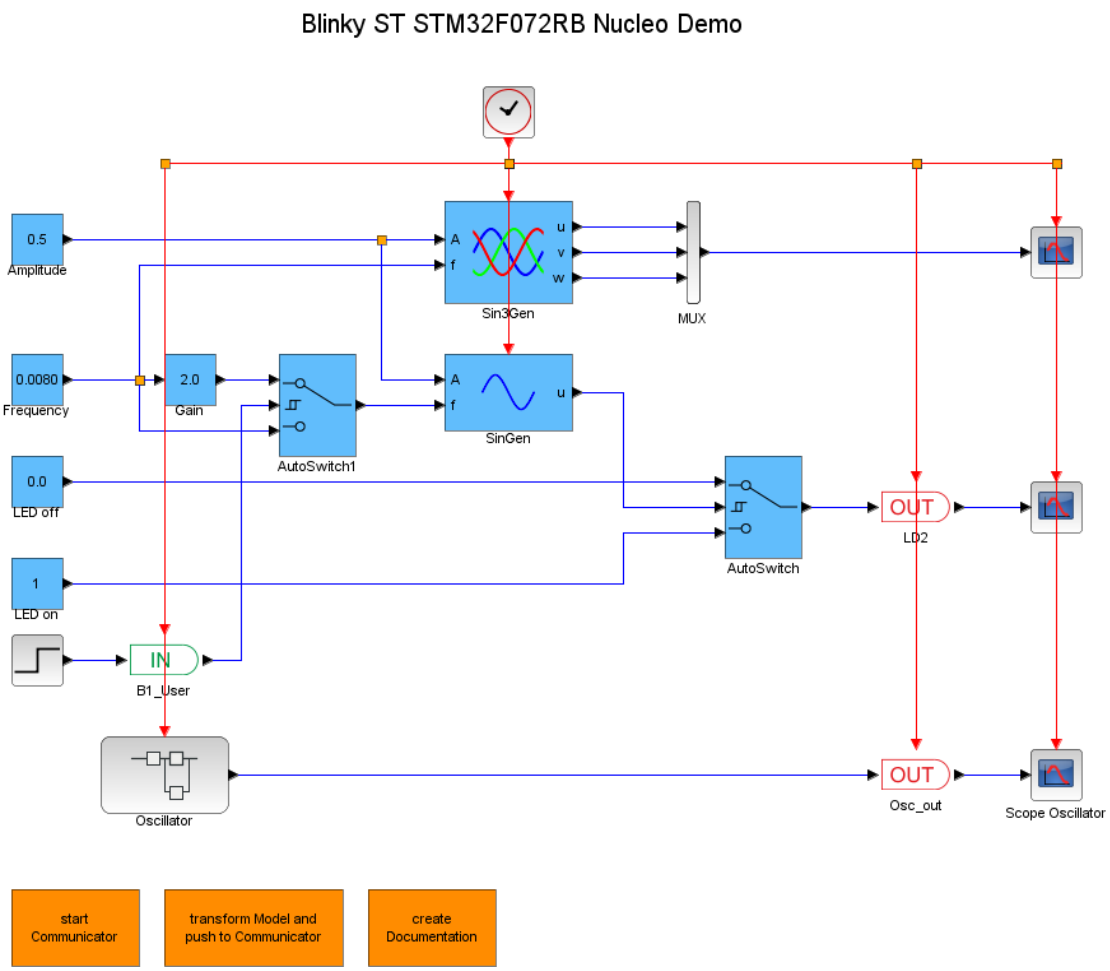


Figure 1: DemoApplication

2.2 Subsystems

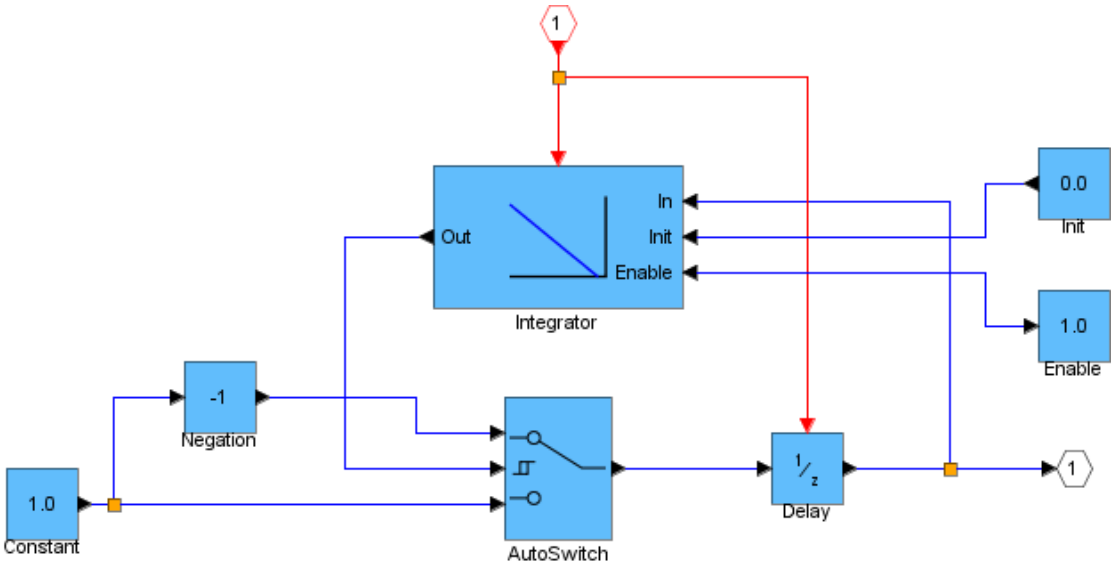


Figure 2: DemoApplication_Oscillator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

4 Mask Parameter

Constant: Amplitude	
Value	0.5
Used Implementation	FiP16

AutoSwitch: AutoSwitch	
Thresh_up	0.0
Thresh_down	0.0
Used Implementation	FiP16

AutoSwitch: AutoSwitch1	
Thresh_up	0.5
Thresh_down	0.5
Used Implementation	FiP16

Constant: Frequency	
Value	0.0080
Used Implementation	FiP16

Gain: Gain	
Gain	2.0
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: Constant	
Value	1.0
Used Implementation	FiP16

Delay: Delay	
ts_fact	1.0
Used Implementation	FiP16

Constant: Enable	
Value	1.0
Used Implementation	Bool

Constant: Init	
Value	0.0
Used Implementation	FiP16

I: Integrator	
Ki	50.0
ts_fact	1.0
Used Implementation	FiP16

Negation: Negation	
Used Implementation	FiP16

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

SinGen: SinGen	
fmax	1000.0
Offset	0.0
Phase	0.0
ts_fact	1.0
Used Implementation	FiP16

Part II
Frame Program Documentation

5 Module Index

5.1 Modules

Here is a list of all modules:

6 File Index

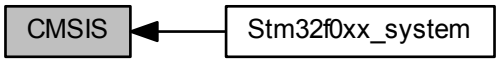
6.1 File List

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

7 Module Documentation

7.1 CMSIS

Collaboration diagram for CMSIS:



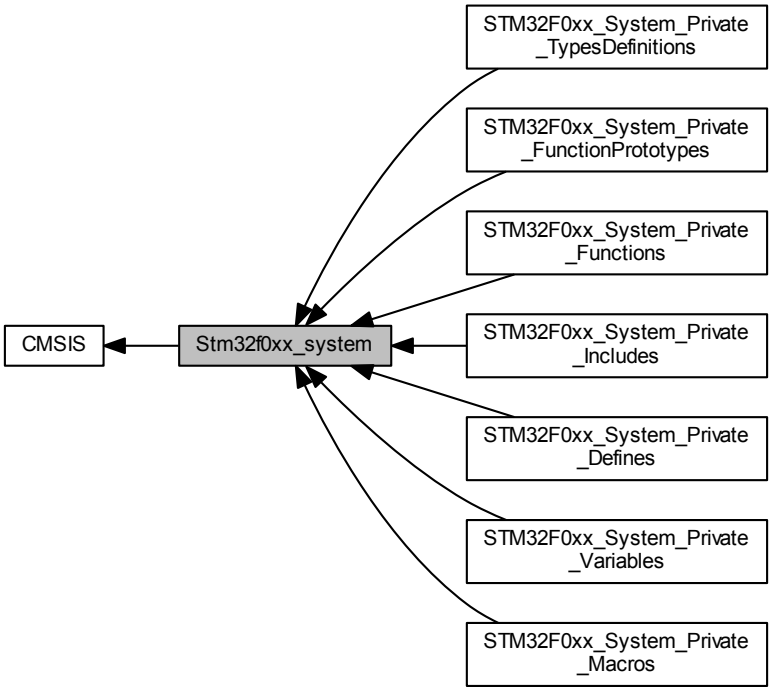
Modules

- Stm32f0xx_system

7.1.1 Detailed Description

7.2 Stm32f0xx_system

Collaboration diagram for Stm32f0xx_system:



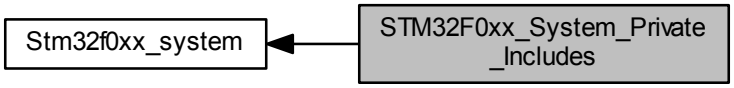
Modules

- STM32F0xx_System_Private_Includes
- STM32F0xx_System_Private_TypesDefinitions
- STM32F0xx_System_Private_Defines
- STM32F0xx_System_Private_Macros
- STM32F0xx_System_Private_Variables
- STM32F0xx_System_Private_FunctionPrototypes
- STM32F0xx_System_Private_Functions

7.2.1 Detailed Description

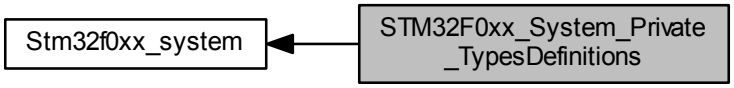
7.3 STM32F0xx_System_Private_Includes

Collaboration diagram for STM32F0xx_System_Private_Includes:



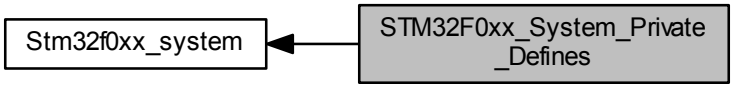
7.4 STM32F0xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F0xx_System_Private_TypesDefinitions:



7.5 STM32F0xx_System_Private_Defines

Collaboration diagram for STM32F0xx_System_Private_Defines:



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

7.5.1 Detailed Description

7.5.2 Macro Definition Documentation

7.5.2.1 #define HSE_VALUE ((uint32_t)8000000)

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

7.5.2.2 #define HSI48_VALUE ((uint32_t)48000000)

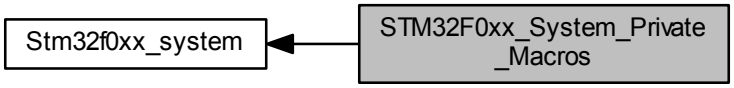
Default value of the HSI48 Internal oscillator in Hz. This value can be provided and adapted by the user application.

7.5.2.3 #define HSI_VALUE ((uint32_t)8000000)

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

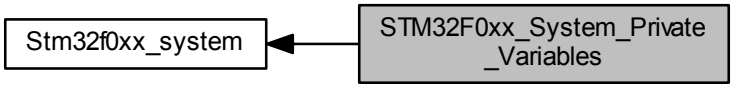
7.6 STM32F0xx_System_Private_Macros

Collaboration diagram for STM32F0xx_System_Private_Macros:



7.7 STM32F0xx_System_Private_Variables

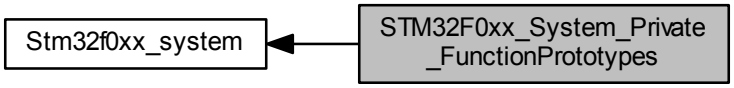
Collaboration diagram for STM32F0xx_System_Private_Variables:



7.7.1 Detailed Description

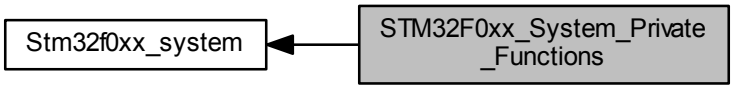
7.8 STM32F0xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F0xx_System_Private_FunctionPrototypes:



7.9 STM32F0xx_System_Private_Functions

Collaboration diagram for STM32F0xx_System_Private_Functions:



Functions

- void SystemInit (void)

  Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

7.9.1 Detailed Description

7.9.2 Function Documentation

7.9.2.1 void SystemCoreClockUpdate (void)

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f0xx_hal.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.
(∗∗) HSE_VALUE is a constant defined in stm32f0xx_hal.h file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

7.9.2.2 void SystemInit (void)

Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.

Parameters

None	

Return values

None	

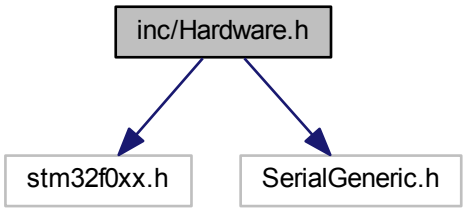
8 File Documentation

8.1 inc/Hardware.h File Reference

Hardware configuration.

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

Include dependency graph for Hardware.h:



Functions

- void initHardware (void)

  Initialization of hardware peripherials.
- void initClock (void)

  Initialization of system clock.
- void initSerial (tSerial ∗serialSTM32F0)

  Initialization of serial interface.
- void ADC1_COMP_IRQHandler (void)

  Interrupt service routine for ADC end of conversion interrupt.

8.1.1 Detailed Description

Hardware configuration.

8.1.2 Function Documentation

8.1.2.1 void initClock (void)

Initialization of system clock.
Configuration:

- Internal high speed clock with PLL
- 48 MHz system clock

8.1.2.2 void initSerial (tSerial ∗ *serialSTM32F0*)

Initialization of serial interface.

- enable port A clock

- setup pins A2 & A3 as USART2 pins
- setup USART Rx- & Tx pins
- enable USART2 clock
- setup baudrate, parity, data bits, stop bits, and flow control
- hook serial functions

8.2 inc/Main.h File Reference

Main function.

Functions

- void mainTask (void)

  Main control task.

8.2.1 Detailed Description

Main function.

8.2.2 Function Documentation

8.2.2.1 void mainTask (void)

Main control task.
Calling rate = 100us

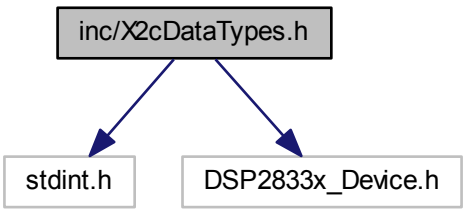
- assign inports
- update X2C
- update outports

8.3 inc/X2cDataTypes.h File Reference

X2C data type definitions.

```
#include <stdint.h>
#include <DSP2833x_Device.h>
```

Include dependency graph for X2cDataTypes.h:

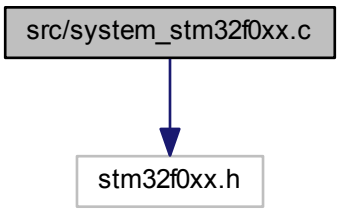


8.3.1 Detailed Description

X2C data type definitions.

8.4 src/system_stm32f0xx.c File Reference

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.
`#include "stm32f0xx.h"`
Include dependency graph for system_stm32f0xx.c:



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

8.4.1 Detailed Description

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

Version

V2.3.0

Date

27-May-2016

1. This file provides two functions and one global variable to be called from user application:
   - SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f0xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then SystemInit() function is called, in "startup_stm32f0xx.s" file, to configure the system clock before to branch to main program.

3. This file configures the system clock as follows:

Supported STM32F0xx device

System Clock source | **HSI**

SYSCLK(Hz) | **8000000**

HCLK(Hz) | **8000000**

AHB Prescaler | **1**

APB1 Prescaler | **1**

==========

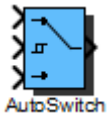
Attention

Part III
Used X2C-Blocks

9 Project Specific Blocks

10 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

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	
In1	int8
Switch	int8
In3	int8

Outports Data Type	
Out	int8

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
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: 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:

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: Gain

Inports	
In	Input

Outports	
Out	Amplified input

Mask Parameters	
Gain	Gain factor in floating point format

Description:

Amplification of input by gain factor.

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

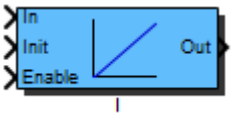
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: 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: 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

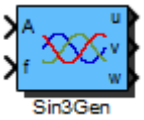
Implementation: Float64

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

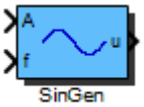
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64
v	float64
w	float64

Block: SinGen



Inports	
A	Amplitude
f	Frequency

Outports	
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:

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

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