LINZ CENTER OF MECHATRONICS GMBH

Project Documentation DemoApplication

August 1, 2017

Contents

Part I

X2C Model

1 Version Information

1.1 X2C

- X2Cfull: Version 1194

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

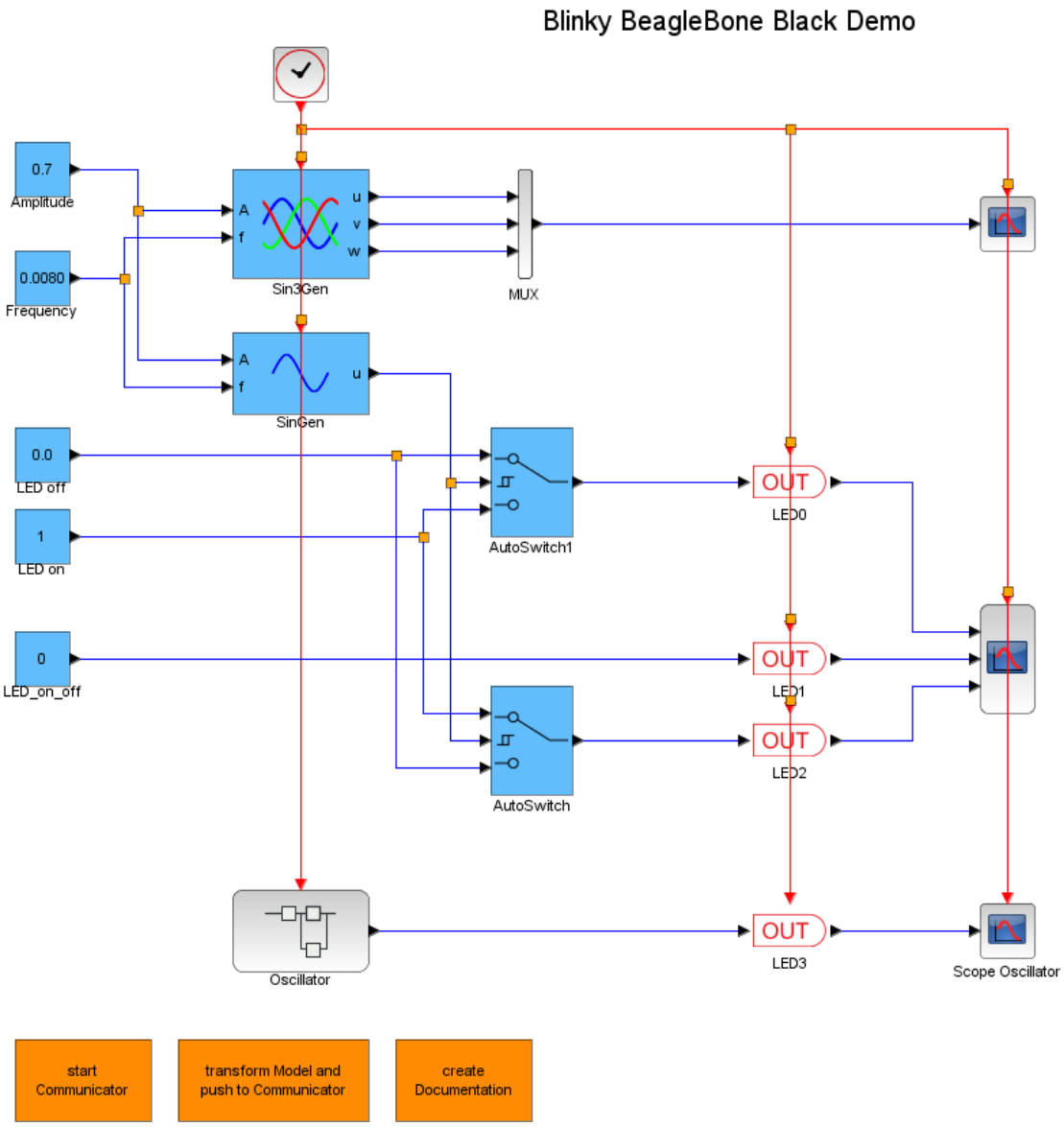


Figure 1: DemoApplication

2.2 Subsystems

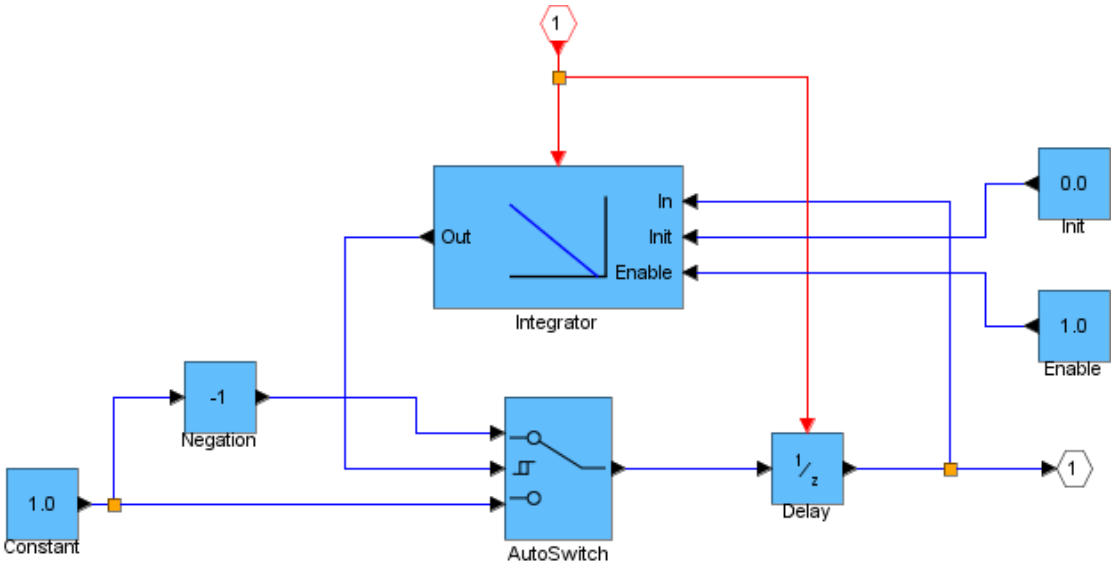


Figure 2: DemoApplication_Oscillator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

4 Mask Parameter

Constant: Amplitude	
Value	0.7
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

Constant: LED_on_off	
Value	0.0
Used Implementation	FiP16

AutoSwitch: Oscillator__AutoSwitch	
Thresh_up	0.5
Thresh_down	-0.5
Used Implementation	FiP16

Constant: Oscillator__Constant	
Value	1.0
Used Implementation	FiP16

Delay: Oscillator__Delay	
ts_fact	1.0
Used Implementation	FiP16

Constant: Oscillator__Enable	
Value	1.0
Used Implementation	Bool

Constant: Oscillator__Init	
Value	0.0
Used Implementation	FiP16

I: Oscillator__Integrator	
Ki	25.0
ts_fact	1.0
Used Implementation	FiP16

Negation: Oscillator__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 File Index

5.1 File List

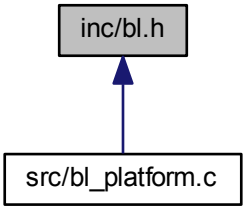
Here is a list of all documented files with brief descriptions:

6 File Documentation

6.1 inc/bl.h File Reference

This file defines boot macros and objects.

This graph shows which files directly or indirectly include this file:

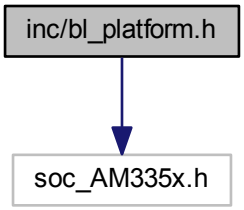


6.1.1 Detailed Description

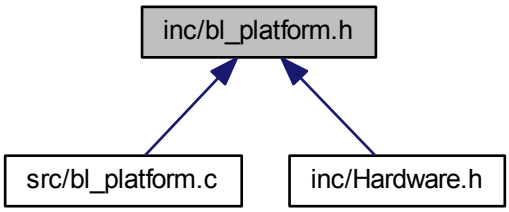
This file defines boot macros and objects.

6.2 inc/bl_platform.h File Reference

This file exports the APIs used for configuring devices required during boot.
`#include "soc_AM335x.h"`
Include dependency graph for bl_platform.h:



This graph shows which files directly or indirectly include this file:



6.2.1 Detailed Description

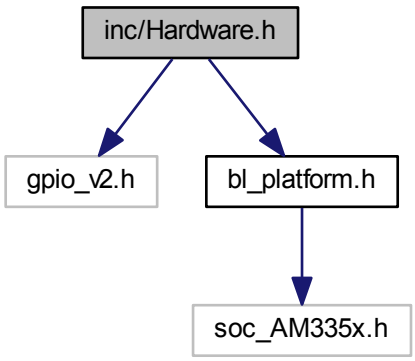
This file exports the APIs used for configuring devices required during boot.

6.3 inc/Hardware.h File Reference

Hardware initialization.

```
#include "gpio_v2.h"
#include "bl_platform.h"
```

Include dependency graph for Hardware.h:



Functions

- void initHardware (void)

  Initialization of hardware.

6.3.1 Detailed Description

Hardware initialization.

6.3.2 Function Documentation

6.3.2.1 void initHardware (void)

Initialization of hardware.

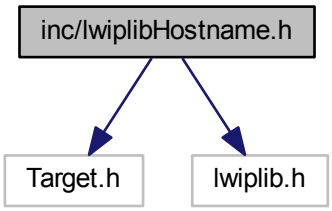
- Configuration of IO ports
- Configuration of timer 2
  - 24MHz timer clock
  - Generation of cyclic interrupt with selected sample time
  - Interrupt calls X2C main task

6.4 inc/lwiplibHostname.h File Reference

Extended lwIP configuration.

```
#include "Target.h"
#include "lwiplib.h"
```

Include dependency graph for lwiplibHostname.h:



Functions

- uint32 lwIPInitHostname (LWIP_IF ∗lwipIf, char ∗hostname)

  Initializes lwip TCP/IP stack with hostname.

6.4.1 Detailed Description

Extended lwIP configuration.
Modified lwiplib from TI StarterWare to support hostname configuration. Requires lwIP 1.4.0.

6.4.2 Function Documentation

6.4.2.1 uint32 lwIPInitHostname (LWIP_IF ∗ *lwipIf,* char ∗ *hostname*)

Initializes lwip TCP/IP stack with hostname.

Parameters

lwipIf	lwIP interface structure
hostname	Hostname

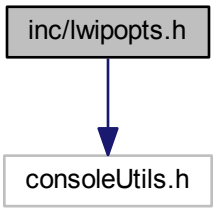
Returns

IP address

6.5 inc/lwipopts.h File Reference

```
#include "consoleUtils.h"
```

Include dependency graph for lwipopts.h:



Macros

- #define LWIP_NETIF_HOSTNAME 1

6.5.1 Detailed Description

- Configuration options for lwIP

Copyright (c) 2010 Texas Instruments Incorporated

6.5.2 Macro Definition Documentation

6.5.2.1 #define LWIP_NETIF_HOSTNAME 1

User specific macros.

6.6 inc/Main.h File Reference

Main function.

Functions

- void mainTask (void)

  Main control task.

6.6.1 Detailed Description

Main function.
X2C maintenance table, protocol & hardware initialization.
Uses Atmel Software Framework (ASF).

6.6.2 Function Documentation

6.6.2.1 void mainTask (void)

Main control task.
TODO: This task has to be called periodically. Calling rate = 100us

- assign inports (not available in this demo)
- update X2C
- update outports

6.7 inc/MMUConfig.h File Reference

MMU configuration.

6.7.1 Detailed Description

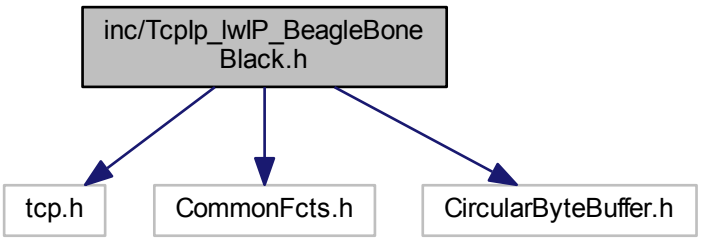
MMU configuration.

6.8 inc/TcpIp_lwIP_BeagleBoneBlack.h File Reference

TCP/IP driver for BeagleBone Black.

```
#include "tcp.h"
#include "CommonFcts.h"
#include "CircularByteBuffer.h"
```

Include dependency graph for TcpIp_lwIP_BeagleBoneBlack.h:



Functions

- uint32 initTcpIp (tTcpIp *tcpip, uint32 ipAddress, uint16 port, char *hostname, uint8 *rxBuffer, uint16 rxSize, uint8 *txBuffer, uint16 txSize)
- void linkTcpIp (tProtocol *protocol, tTcpIp *tcpip)
- void closeConnection (tTcpIp *tcpip)
- uint16 getAvailableTcpIpBytes (tTcpIp *tcpip)
- uint16 getTcpIpData (tTcpIp *tcpip, uint8 *data, uint16 len)
- uint16 getFreeTcpIpBytes (tTcpIp *tcpip)
- uint16 putTcpIpData (tTcpIp *tcpip, uint8 *data, uint16 len)

6.8.1 Detailed Description

TCP/IP driver for BeagleBone Black.
TCP/IP driver implementation for BeagleBone Black board. Uses lwIP 1.4.0 stack for communication.

6.8.2 Function Documentation

6.8.2.1 void closeConnection (tTcpIp * *tcpip*)

Closes connection.
Parameters

tcpip	TCP/IP interface

6.8.2.2 uint16 getAvailableTcpIpBytes (tTcpIp * *tcpip*)

Returns available bytes in receive buffer.
Parameters

tcpip	TCIP/IP interface

Returns

The number of bytes being available to receive

6.8.2.3 uint16 getFreeTcpIpBytes (tTcpIp * *tcpip*)

Returns free bytes in transmit buffer.
Parameters

tcpip	TCP/IP interface

Returns

The number of free bytes in the transmit buffer

6.8.2.4 uint16 getTcpIpData (tTcpIp * *tcpip,* uint8 * *data,* uint16 *len*)

Reads data from communication buffers.
Parameters

tcpip	TCIP/IP interface
data	User data buffer
len	User data buffer len

Returns

The number of bytes being transferred

Note

The caller must guarantee correct buffer size.

6.8.2.5 uint32 initTcplp (tTcplp * *tcpip,* uint32 *ipAddress,* uint16 *port,* char * *hostname,* uint8 * *rxBuffer,* uint16 *rxSize,* uint8 * *txBuffer,* uint16 *txSize*)

Initializes TCP/IP interface.

Parameters

tcpip	TCP/IP interface
IP	address
port	TCP port
hostname	Hostname being used in DHCP (option #12)
rxBuffer	Receive buffer
rxSize	Receive buffer length
txBuffer	Transmit buffer
txSize	Transmit buffer length

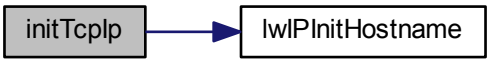
Returns

Assigned IP address

Note

Static IP address example: 192.168.0.1 => 0xC0A80001
If DHCP IP address is desired, the IP address has to be set to zero

Here is the call graph for this function:



6.8.2.6 void linkTcpIp (tProtocol * *protocol,* tTcpIp * *tcpip*)

Links protocol with TCP/IP interface.

Parameters

protocol	Protocol
tcpip	TCP/IP interface

6.8.2.7 uint16 putTcpIpData (tTcpIp * *tcpip,* uint8 * *data,* uint16 *len*)

Writes data to communication buffers.

Parameters

tcpip	TCIP/IP connection
data	User data buffer
len	User data buffer len

Returns

The number of bytes being transferred

Note

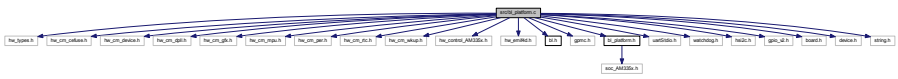
The caller must guarantee correct buffer size.

6.9 src/bl_platform.c File Reference

Initializes AM335x Device Peripherals.

```
#include "hw_types.h"
#include "hw_cm_cefuse.h"
#include "hw_cm_device.h"
#include "hw_cm_dpll.h"
#include "hw_cm_gfx.h"
#include "hw_cm_mpu.h"
#include "hw_cm_per.h"
#include "hw_cm_rtc.h"
#include "hw_cm_wkup.h"
#include "hw_control_AM335x.h"
#include "hw_emif4d.h"
#include "bl.h"
#include "gpmc.h"
#include "bl_platform.h"
#include "uartStdio.h"
#include "watchdog.h"
#include "hsi2c.h"
#include "gpio_v2.h"
#include "board.h"
#include "device.h"
#include "string.h"
```

Include dependency graph for bl_platform.c:



Functions

- void ConfigureVdd2 (unsigned int opVolMultiplier, unsigned maxLoadCurrent, unsigned int timeStep, unsigned int supplyState)
  - *Configure vdd2 for various parameters such as Multiplier, Maximum Load Current etc .*
- void SelectVdd2Source (unsigned int vddSource)

  Select the VDD2 value. VDD2_OP_REG or VDD2_SR_REG.
- void SetVdd2OpVoltage (unsigned int opVolSelector)

set VDD2_OP voltage value.

- void SetVdd2SrVoltage (unsigned int opVolSelector)

  set VDD2_SR voltage value
- void SelectI2CInstance (unsigned int i2cInstance)

  Select I2C interface whether SR I2C or Control I2C.
- void ConfigureVdd1 (unsigned int opVolMultiplier, unsigned maxLoadCurrent, unsigned int timeStep, unsigned int supplyState)
  - *Configure vdd1 for various parameters such as Multiplier, Maximum Load Current etc .*
- void SelectVdd1Source (unsigned int vddSource)

  Select the VDD1 value. VDD1_OP_REG or VDD1_SR_REG.
- void SetVdd1OpVoltage (unsigned int opVolSelector)

  set VDD1_OP voltage value.

6.9.1 Detailed Description

Initializes AM335x Device Peripherals.

6.9.2 Function Documentation

6.9.2.1 void ConfigureVdd1 (unsigned int *opVolMultiplier,* unsigned *maxLoadCurrent,* unsigned int *timeStep,* unsigned int *supplyState*)

- Configure vdd1 for various parameters such as Multiplier, Maximum Load Current etc .

Parameters

opVolMultiplier	- Multiplier.
maxLoadCurrent	- Maximum Load Current.
timeStep	- Time step - voltage change per us(micro sec).
supplyState	- Supply state (on (high/low power mode), off)

Returns

: None.

6.9.2.2 void ConfigureVdd2 (unsigned int *opVolMultiplier,* unsigned *maxLoadCurrent,* unsigned int *timeStep,* unsigned int *supplyState*)

- Configure vdd2 for various parameters such as Multiplier, Maximum Load Current etc .

Parameters

opVolMultiplier	- Multiplier.
maxLoadCurrent	- Maximum Load Current.
timeStep	- Time step - voltage change per us(micro sec).
supplyState	- Supply state (on (high/low power mode), off)

Returns

: None.

6.9.2.3 void SelectI2CInstance (unsigned int *i2cInstance*)

Select I2C interface whether SR I2C or Control I2C.

Parameters

i2cInstance	- I2c instance to select.

Returns

None.

6.9.2.4 void SelectVdd1Source (unsigned int *vddSource*)

Select the VDD1 value. VDD1_OP_REG or VDD1_SR_REG.

Parameters

vddSource	- VDD2 value.

Returns

None.

6.9.2.5 void SelectVdd2Source (unsigned int *vddSource*)

Select the VDD2 value. VDD2_OP_REG or VDD2_SR_REG.

Parameters

vddSource	- VDD2 value.

Returns

None.

6.9.2.6 void SetVdd1OpVoltage (unsigned int *opVolSelector*)

set VDD1_OP voltage value.

Parameters

opVolSelector	- VDD2_OP voltage value.

Returns

None.

6.9.2.7 void SetVdd2OpVoltage (unsigned int *opVolSelector*)

set VDD2_OP voltage value.

Parameters

opVolSelector	- VDD2_OP voltage value.

Returns

None.

6.9.2.8 void SetVdd2SrVoltage (unsigned int *opVolSelector*)

set VDD2_SR voltage value

Parameters

opVolSelector	- VDD2_SR voltage value.

Returns

None.

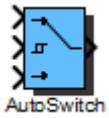
Part III

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

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

Block: Constant

Outports	
Out	Constant output

Mask Parameters	
Value	Constant factor

Description:

Constant value.

Implementations:

Bool	Boolean Integration
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

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:

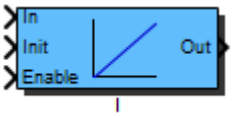
Output delay by one sample time interval.

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

Implementations:

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

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):

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

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

Block: Negation

Inports	
In	Input

Outports	
Out	Negated input value

Description:

Negation of input signal.

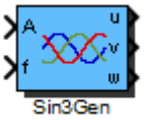
Calculation:

$$Out \quad = \quad -In$$

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

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:

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

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.

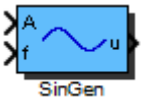
Calculation floating point implementation (parameter *f_max* is ignored):

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

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

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:

$$u_k = A_k \cdot \sin\left(2 f_k \cdot f_{max} \cdot k T_S + \phi_{Phase}\right) + A_{Offset}$$

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):

$$u_k = A_k \cdot \sin\left(2\pi f_k \cdot k T_S + \phi_{Phase}\right) + A_{Offset}$$

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