



Project Documentation DemoApplication

September 1, 2016

© Linz Center of Mechatronics GmbH

Contents

I	X2C Model	2
1	Version Information	2
1.1	X2C	2
1.2	Operating System	2
1.3	Scilab	2
2	Model Structure	3
2.1	Xcos Model	3
2.2	Subsystems	4
3	Model Parameter	5
3.1	Sample Time	5
3.2	Scilab Parameter	5
4	Mask Parameter	6
II	Frame Program Documentation	8
5	File Index	8
5.1	File List	8
6	File Documentation	8
6.1	Hardware.h File Reference	8
6.1.1	Detailed Description	8
6.1.2	Function Documentation	9
6.2	Main.h File Reference	9
6.2.1	Detailed Description	9
6.2.2	Function Documentation	10
III	Used X2C-Blocks	11
7	Project Specific Blocks	11
8	Internal Library Blocks	11
	AutoSwitch	11
	Constant	15
	Delay	18
	I	21
	Negation	25
	Sin3Gen	28
	SinGen	33

Part I

X2C Model

1 Version Information

1.1 X2C

- X2Cfull: Version 1012

1.2 Operating System

- OS: Windows 7 6.1

1.3 Scilab

- Scilab: Version 5.5.1.1412169962
- Java: Version 1.6.0_41

2 Model Structure

2.1 Xcos Model

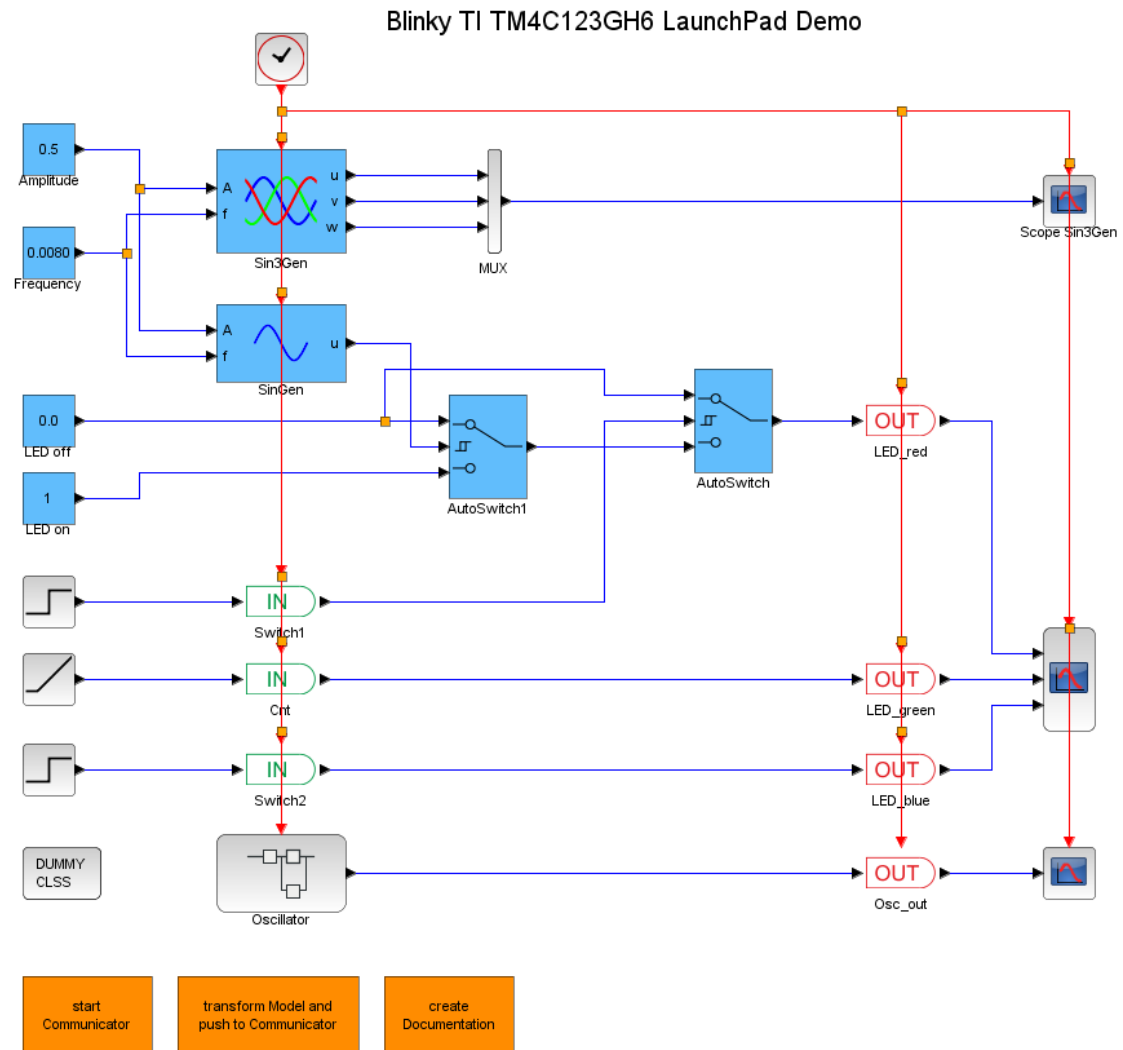


Figure 1: DemoApplication

2.2 Subsystems

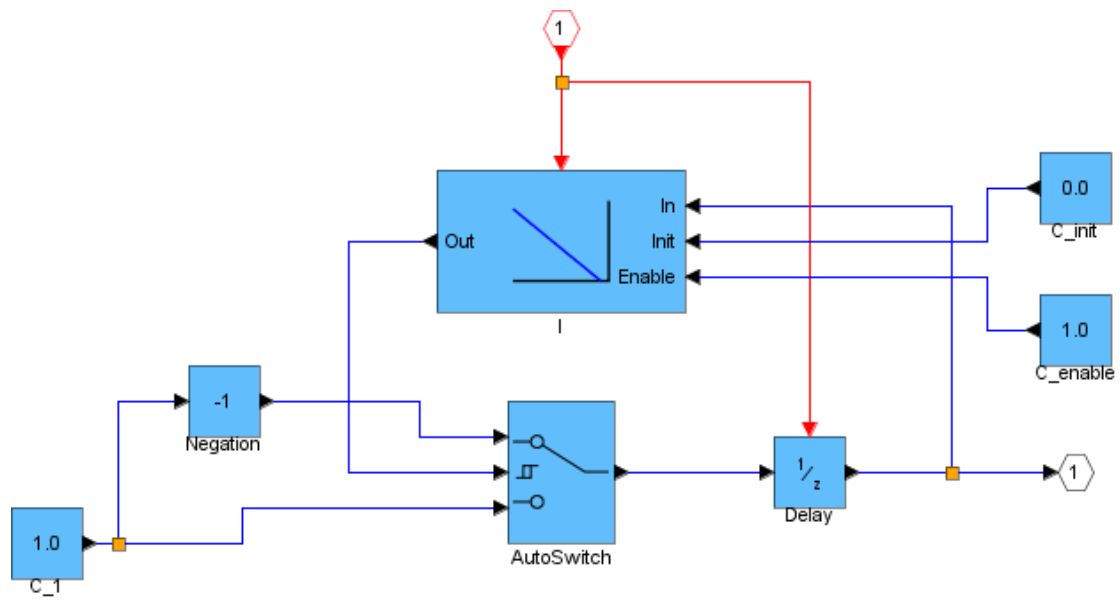


Figure 2: DemoApplication_Oscillator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$200\mu s$

3.2 Scilab Parameter

```
1 // File with model parameters such as sample time, scaling factors, etc...
2 //
3 // Copyright (c) 2016, Linz Center of Mechatronics GmbH (LCM) http://www.lcm.at/
4 // All rights reserved.
5 //
6 // $LastChangedRevision: 894 $
7 // $LastChangedDate:: 2016-05-25 15:59:23 +0200#$
8 //
9 // This file is part of X2C. http://www.mechatronic-simulation.org/
10
11 // Sampling time
12 X2C_sampleTime = 1/5000; // 5kHz sampling frequency
13
14 // Scaling factors
15
16 // Controller parameters
```

Listing 1: ModelParameter.sce

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: Oscillator__AutoSwitch	
Thresh_up	0.5
Thresh_down	-0.5
Used Implementation	FiP16

Constant: Oscillator__C_1	
Value	1.0
Used Implementation	FiP16

Constant: Oscillator__C_enable	
Value	1.0
Used Implementation	FiP8

Constant: Oscillator__C_init	
Value	0.0
Used Implementation	FiP16

Delay: Oscillator__Delay	
ts_fact	1.0
Used Implementation	FiP16

I: Oscillator__I	
Ki	50.0
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:

Hardware.h	
Hardware configuration	8
Main.h	
Main function	9

6 File Documentation

6.1 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 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

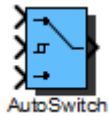
Part III

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

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 \geq Threshold up \rightarrow Out = In1

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

Name	FiP8
ID	128
Revision	0.1
C filename	AutoSwitch_FiP8.c
H filename	AutoSwitch_FiP8.h

8 Bit Fixed Point Implementation

Controller Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal
Status	Current hysteresis state

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int8        *In1;  
    int8        *Switch;  
    int8        *In3;  
    int8        Out;  
    int8        Thresh_up;  
    int8        Thresh_down;  
    int8        Status;  
} AUTOSWITCH_FIP8;
```

Implementation: FiP16

Name	FiP16
ID	129
Revision	0.1
C filename	AutoSwitch_FiP16.c
H filename	AutoSwitch_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal
Status	Current hysteresis state

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int16        *In1;  
    int16        *Switch;  
    int16        *In3;
```

```

    int16      Out;
    int16      Thresh_up;
    int16      Thresh_down;
    int8       Status;
} AUTOSWITCH_FIP16;

```

Implementation: FiP32

Name FiP32
ID 130
Revision 0.1
C filename AutoSwitch_FiP32.c
H filename AutoSwitch_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal
Status	Current hysteresis state

Data Structure:

```

typedef struct {
    uint16      ID;
    int32       *In1;
    int32       *Switch;
    int32       *In3;
    int32       Out;
    int32       Thresh_up;
    int32       Thresh_down;
    int8       Status;
} AUTOSWITCH_FIP32;

```

Implementation: Float32

Name Float32
ID 131
Revision 0.1
C filename AutoSwitch_Float32.c
H filename AutoSwitch_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal
Status	Current hysteresis state

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float32     *In1;  
    float32     *Switch;  
    float32     *In3;  
    float32     Out;  
    float32     Thresh_up;  
    float32     Thresh_down;  
    int8        Status;  
} AUTOSWITCH_FLOAT32;
```

Implementation: Float64

Name	Float64
ID	132
Revision	0.1
C filename	AutoSwitch_Float64.c
H filename	AutoSwitch_Float64.h

64 Bit Floating Point Implementation

Controller Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal
Status	Current hysteresis state

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float64     *In1;  
    float64     *Switch;  
    float64     *In3;  
    float64     Out;  
    float64     Thresh_up;  
    float64     Thresh_down;  
    int8        Status;  
} AUTOSWITCH_FLOAT64;
```

Block: Constant



Outputs	
Out	Constant output

Mask Parameters	
Value	Constant factor

Description:

Constant value.

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

Name	FiP8
ID	48
Revision	0.3
C filename	Constant_FiP8.c
H filename	Constant_FiP8.h

8 Bit Fixed Point Implementation

Controller Parameters	
K	Constant factor

Data Structure:

```
typedef struct {  
    uint16 ID;  
    int8 Out;  
    int8 K;  
} CONSTANT_FIP8;
```


Implementation: FiP16

Name FiP16
ID 49
Revision 0.3
C filename Constant_FiP16.c
H filename Constant_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
K	Constant factor

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int16       Out;  
    int16       K;  
} CONSTANT_FIP16;
```

Implementation: FiP32

Name FiP32
ID 50
Revision 0.3
C filename Constant_FiP32.c
H filename Constant_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
K	Constant factor

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int32       Out;  
    int32       K;  
} CONSTANT_FIP32;
```

Implementation: Float32

Name Float32
ID 51
Revision 0.1
C filename Constant_Float32.c
H filename Constant_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
K	Constant factor

Data Structure:

```
typedef struct {
    uint16      ID;
    float32     Out;
    float32     K;
} CONSTANT_FLOAT32;
```

Implementation: Float64

Name Float64
ID 52
Revision 0.1
C filename Constant_Float64.c
H filename Constant_Float64.h

64 Bit Floating Point Implementation

Controller Parameters	
K	Constant factor

Data Structure:

```
typedef struct {
    uint16      ID;
    float64     Out;
    float64     K;
} CONSTANT_FLOAT64;
```

Block: Delay



Inports	
In	Input In(k)
Outputs	
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

Implementation: FiP16

Name	FiP16
ID	3425
Revision	0.1
C filename	Delay_FiP16.c
H filename	Delay_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
In_old	Input value from previous cycle

Data Structure:

```
typedef struct {  
    uint16    ID;
```

```

    int16      *In ;
    int16      Out;
    int16      In_old;
} DELAY_FIP16;

```

Implementation: FiP32

Name FiP32
ID 3426
Revision 0.1
C filename Delay_FiP32.c
H filename Delay_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
In_old	Input value from previous cycle

Data Structure:

```

typedef struct {
    uint16      ID;
    int32       *In ;
    int32       Out;
    int32       In_old;
} DELAY_FIP32;

```

Implementation: Float32

Name Float32
ID 3427
Revision 0.1
C filename Delay_Float32.c
H filename Delay_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
In_old	Input value from previous cycle

Data Structure:

```

typedef struct {
    uint16      ID;
    float32     *In ;
    float32     Out;
    float32     In_old;
} DELAY_FLOAT32;

```

Implementation: Float64

Name	Float64
ID	3428
Revision	0.1
C filename	Delay_Float64.c
H filename	Delay_Float64.h

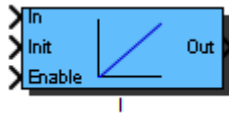
64 Bit Floating Point Implementation

Controller Parameters	
In_old	Input value from previous cycle

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float64     *In;  
    float64     Out;  
    float64     In_old;  
} DELAY_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:

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

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:

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

Name	FiP8
ID	3200
Revision	1.0
C filename	I_FiP8.c
H filename	I_FiP8.h

8 Bit Fixed Point Implementation

Controller Parameters	
b0	Integral coefficient
sfr	Shift factor for I coefficient b0
i_old	Integrator value from previous cycle
enable_old	Enable value of previous cycle

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int8        *In;  
    int8        *InIt;  
    int8        *Enable;  
    int8        Out;  
    int8        b0;  
    int8        sfr;  
    int16       i_old;  
    int8        enable_old;  
} I_FIP8;
```

Implementation: FiP16

Name	FiP16
ID	3201
Revision	1.0
C filename	I_FiP16.c
H filename	I_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
b0	Integral coefficient
sfr	Shift factor for I coefficient b0
i_old	Integrator value from previous cycle
enable_old	Enable value of previous cycle

Data Structure:

```
typedef struct {
```

```

uint16      ID;
int16       *In;
int16       *Init;
int8        *Enable;
int16       Out;
int16       b0;
int8        sfr;
int32       i_old;
int8        enable_old;
} I_FIP16;

```

Implementation: FiP32

Name FiP32
ID 3202
Revision 1.0
C filename I_FiP32.c
H filename I_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
b0	Integral coefficient
sfr	Shift factor for I coefficient b0
i_old	Integrator value from previous cycle
enable_old	Enable value of previous cycle

Data Structure:

```

typedef struct {
    uint16      ID;
    int32       *In;
    int32       *Init;
    int8        *Enable;
    int32       Out;
    int32       b0;
    int8        sfr;
    int64       i_old;
    int8        enable_old;
} I_FIP32;

```

Implementation: Float32

Name Float32
ID 3203
Revision 0.1
C filename I_Float32.c
H filename I_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
b0	Integral coefficient
i_old	Integrator value from previous cycle
enable_old	Enable value of previous cycle

Data Structure:

```
typedef struct {
    uint16      ID;
    float32     *In;
    float32     *Init;
    int8        *Enable;
    float32     Out;
    float32     b0;
    float32     i_old;
    int8        enable_old;
} I_FLOAT32;
```

Implementation: Float64

Name Float64
ID 3204
Revision 0.1
C filename I_Float64.c
H filename I_Float64.h

64 Bit Floating Point Implementation

Controller Parameters	
b0	Integral coefficient
i_old	Integrator value from previous cycle
enable_old	Enable value of previous cycle

Data Structure:

```
typedef struct {
    uint16      ID;
    float64     *In;
    float64     *Init;
    int8        *Enable;
    float64     Out;
    float64     b0;
    float64     i_old;
    int8        enable_old;
} I_FLOAT64;
```

Block: Negation



Inports	
In	Input

Outputs	
Out	Negated input value

Description:

Negation of input signal.

Calculation:

$$Out = -In$$

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

Name	FiP8
ID	5040
Revision	0.1
C filename	Negation_FiP8.c
H filename	Negation_FiP8.h

8 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {  
    uint16 ID;  
    int8 *In;  
    int8 Out;  
} NEGATION_FIP8;
```

Implementation: FiP16

Name	FiP16
ID	5041
Revision	0.1
C filename	Negation_FiP16.c
H filename	Negation_FiP16.h

16 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int16       *In;  
    int16       Out;  
} NEGATION_FIP16;
```

Implementation: FiP32

Name	FiP32
ID	5042
Revision	0.1
C filename	Negation_FiP32.c
H filename	Negation_FiP32.h

32 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {  
    uint16      ID;  
    int32       *In;  
    int32       Out;  
} NEGATION_FIP32;
```

Implementation: Float32

Name	Float32
ID	5043
Revision	0.1
C filename	Negation_Float32.c
H filename	Negation_Float32.h

32 Bit Floating Point Implementation

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float32     *In;  
    float32     Out;
```

```
} NEGATION_FLOAT32;
```

Implementation: Float64

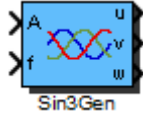
Name	Float64
ID	5044
Revision	0.1
C filename	Negation_Float64.c
H filename	Negation_Float64.h

64 Bit Floating Point Implementation

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float64     *In;  
    float64     Out;  
} NEGATION_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:

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

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):

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

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

Implementation: FiP8

Name	FiP8
ID	432
Revision	1.0
C filename	Sin3Gen_FiP8.c
H filename	Sin3Gen_FiP8.h

8 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {  
    uint16    ID;  
    int8      *A;  
    int8      *f;  
    int8      u;  
    int8      v;  
    int8      w;  
    int8      delta_phi;  
    int8      offset;  
    int8      phi;  
} SIN3GEN_FIP8;
```

Implementation: FiP16

Name	FiP16
ID	433
Revision	1.0
C filename	Sin3Gen_FiP16.c
H filename	Sin3Gen_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {
    uint16      ID;
    int16       *A;
    int16       *f;
    int16       u;
    int16       v;
    int16       w;
    int16       delta_phi;
    int16       offset;
    int16       phi;
} SIN3GEN_FIP16;
```

Implementation: FiP32

Name FiP32
ID 434
Revision 1.0
C filename Sin3Gen_FiP32.c
H filename Sin3Gen_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {
    uint16      ID;
    int32       *A;
    int32       *f;
    int32       u;
    int32       v;
    int32       w;
    int32       delta_phi;
    int32       offset;
    int32       phi;
} SIN3GEN_FIP32;
```

Implementation: Float32

Name	Float32
ID	435
Revision	0.1
C filename	Sin3Gen_Float32.c
H filename	Sin3Gen_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
delta_phi	Angle increment
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {  
    uint16      ID;  
    float32     *A;  
    float32     *f;  
    float32     u;  
    float32     v;  
    float32     w;  
    float32     delta_phi;  
    float32     offset;  
    float32     phi;  
} SIN3GEN_FLOAT32;
```

Implementation: Float64

Name	Float64
ID	436
Revision	0.1
C filename	Sin3Gen_Float64.c
H filename	Sin3Gen_Float64.h

64 Bit Floating Point Implementation

Controller Parameters	
delta_phi	Angle increment
offset	Amplitude offset
phi	Current angle

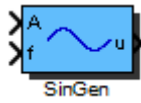
Data Structure:

```
typedef struct {  
    uint16      ID;  
    float64     *A;  
    float64     *f;
```



```
    float64    u;  
    float64    v;  
    float64    w;  
    float64    delta_phi;  
    float64    offset;  
    float64    phi;  
} SIN3GEN_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:

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

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):

$$u_k = A_k \cdot \sin(2\pi f_k \cdot kT_S + \phi_{Phase}) + A_{Offset}$$

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

Implementation: FiP8

Name FiP8
ID 416
Revision 1.0
C filename SinGen_FiP8.c
H filename SinGen_FiP8.h

8 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
phase	Angle offset
offset	Amplitude offset
phi	Current angle

Data Structure:

```

typedef struct {
    uint16    ID;
    int8      *A;
    int8      *f;
    int8      u;
    int8      delta_phi;
    int8      phase;
    int8      offset;
    int8      phi;
} SINGEN_FIP8;
  
```

Implementation: FiP16

Name FiP16
ID 417
Revision 1.0
C filename SinGen_FiP16.c
H filename SinGen_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
phase	Angle offset
offset	Amplitude offset
phi	Current angle

Data Structure:

```

typedef struct {
    uint16    ID;
    int16      *A;
    int16      *f;
  
```

```

    int16    u;
    int16    delta_phi;
    int16    phase;
    int16    offset;
    int16    phi;
} SINGEN_FIP16;

```

Implementation: FiP32

Name FiP32
ID 418
Revision 1.0
C filename SinGen_FiP32.c
H filename SinGen_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
delta_phi	Angle increment
phase	Angle offset
offset	Amplitude offset
phi	Current angle

Data Structure:

```

typedef struct {
    uint16    ID;
    int32     *A;
    int32     *f;
    int32     u;
    int32     delta_phi;
    int32     phase;
    int32     offset;
    int32     phi;
} SINGEN_FIP32;

```

Implementation: Float32

Name Float32
ID 419
Revision 0.1
C filename SinGen_Float32.c
H filename SinGen_Float32.h

32 Bit Floating Point Implementation

Controller Parameters	
delta_phi	Angle increment
phase	Angle offset
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {
    uint16      ID;
    float32     *A;
    float32     *f;
    float32     u;
    float32     delta_phi;
    float32     phase;
    float32     offset;
    float32     phi;
} SINGEN_FLOAT32;
```

Implementation: Float64

Name Float64
ID 420
Revision 0.1
C filename SinGen_Float64.c
H filename SinGen_Float64.h

64 Bit Floating Point Implementation

Controller Parameters	
delta_phi	Angle increment
phase	Angle offset
offset	Amplitude offset
phi	Current angle

Data Structure:

```
typedef struct {
    uint16      ID;
    float64     *A;
    float64     *f;
    float64     u;
    float64     delta_phi;
    float64     phase;
    float64     offset;
    float64     phi;
} SINGEN_FLOAT64;
```