Project Documentation mc_foc_sl_fip_dspic33ck_mclv2

July 28, 2020

Contents

Part I

X2C Model

1 Version Information

1.1 X2C

- X2C: Version 6.2.1993

1.2 Operating System

- OS: Windows 8 6.2

1.3 Scilab

- Scilab: Version 5.5.2.1427793548

- Java: Version 1.6.0_41

2 Model Structure

2.1 Xcos Model

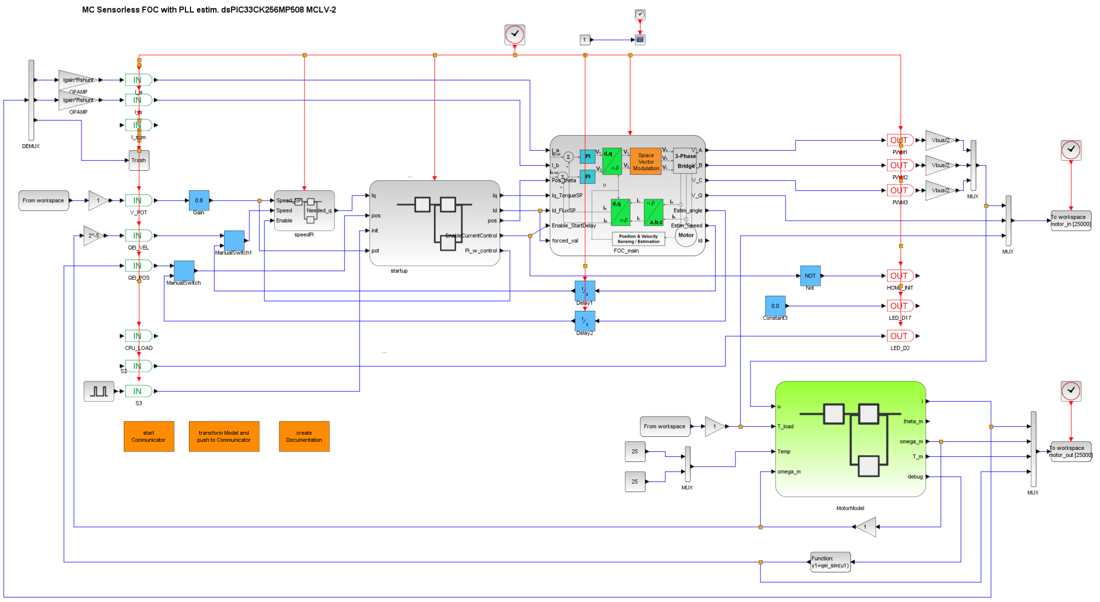


Figure 1: mc_foc_sl_fip_dspic33ck_mclv2

2.2 Subsystems

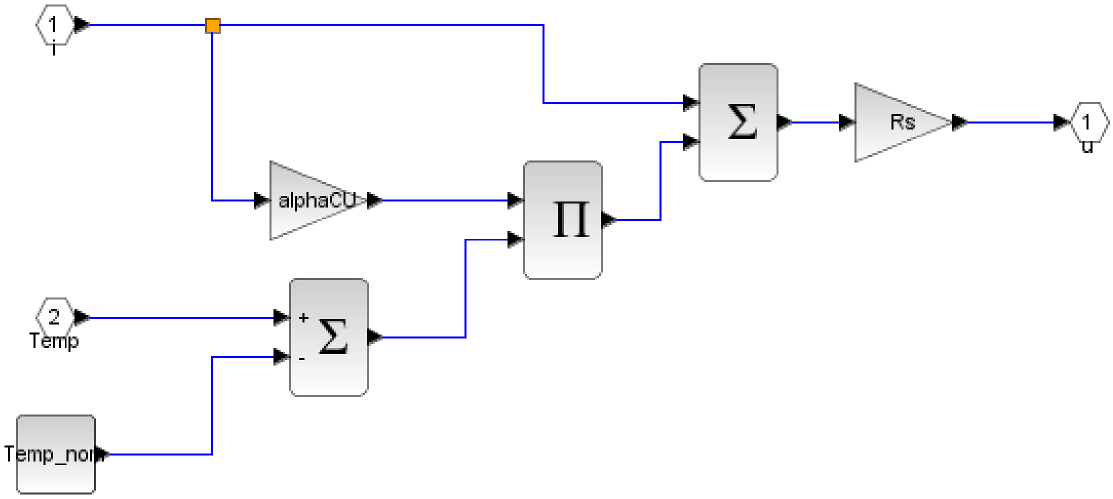


Figure 2: mc_foc_sl_fip_dspic33ck_mclv2_startup

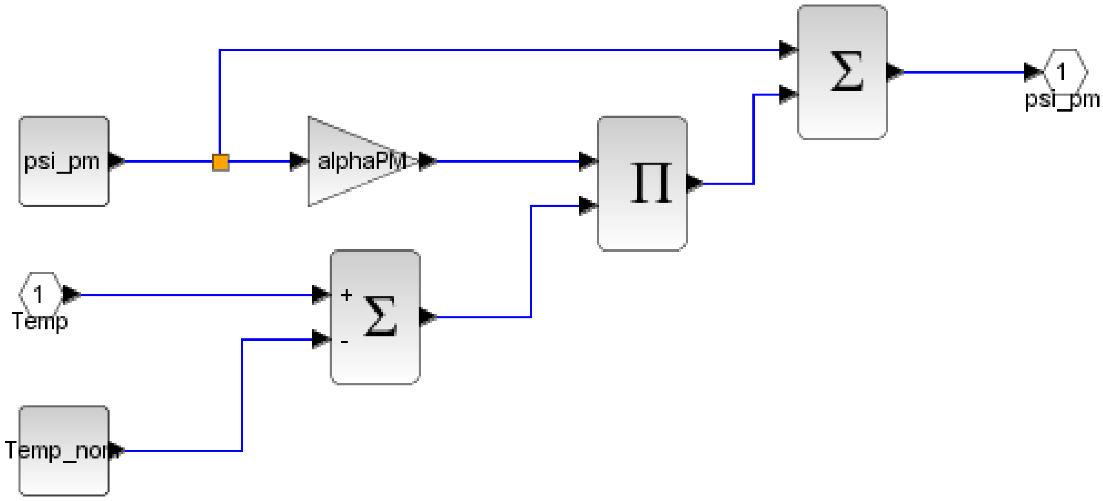


Figure 3: mc_foc_sl_fip_dspic33ck_mclv2_speedPI

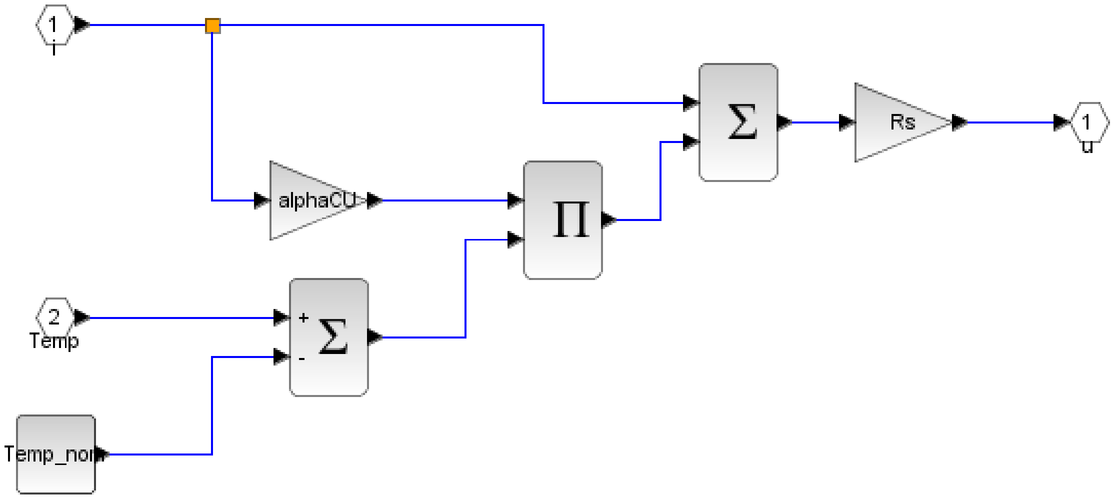


Figure 4: mc_foc_sl_fip_dspic33ck_mclv2_MotorModel

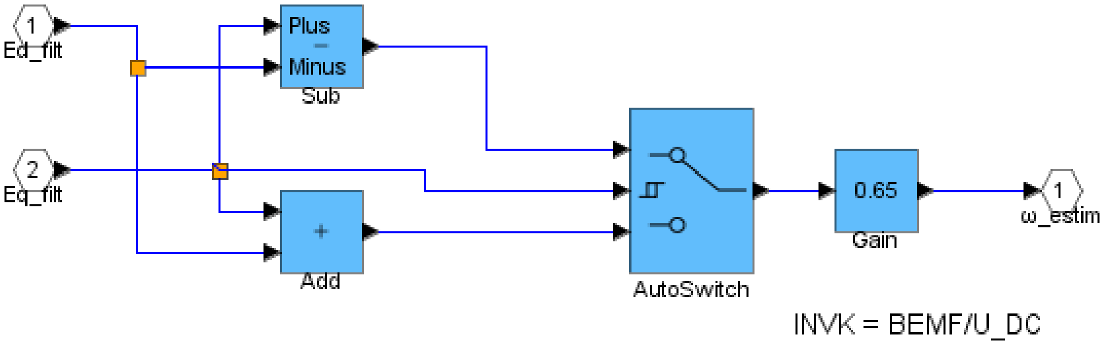


Figure 5: mc_foc_sl_fip_dspic33ck_mclv2_FOC_main

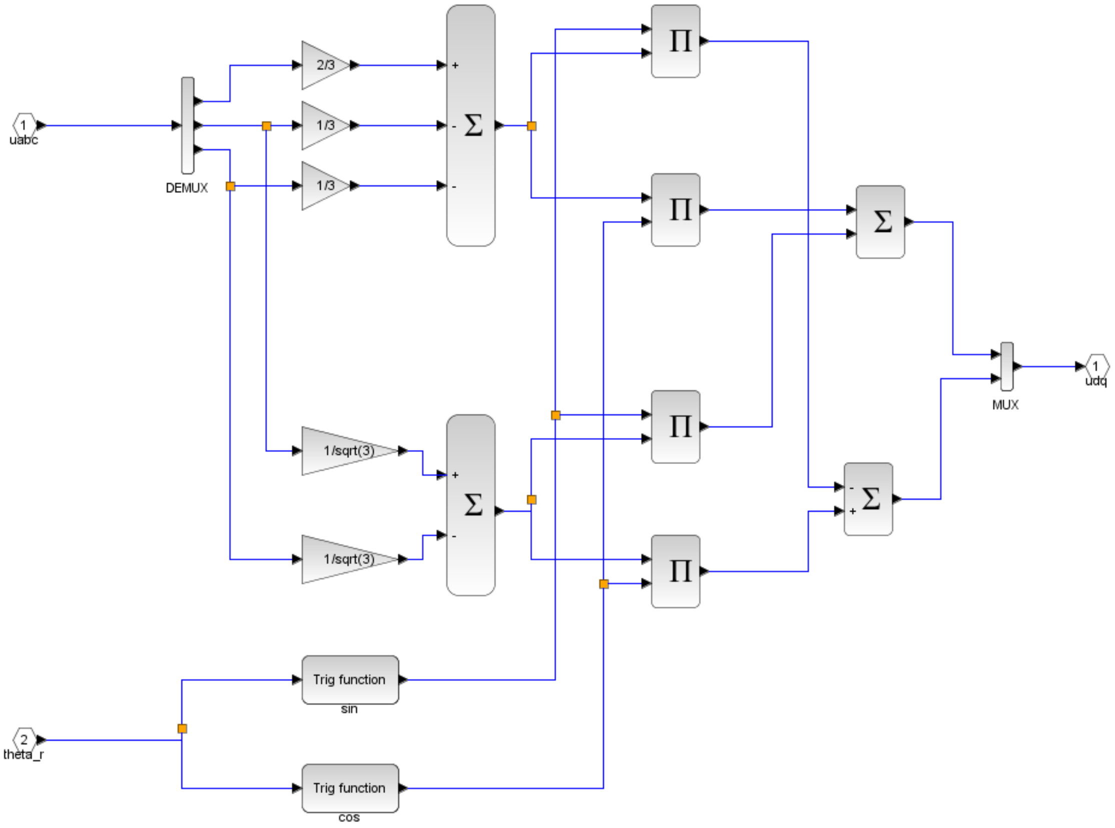


Figure 6: mc_foc_sl_fip_dspic33ck_mclv2_4

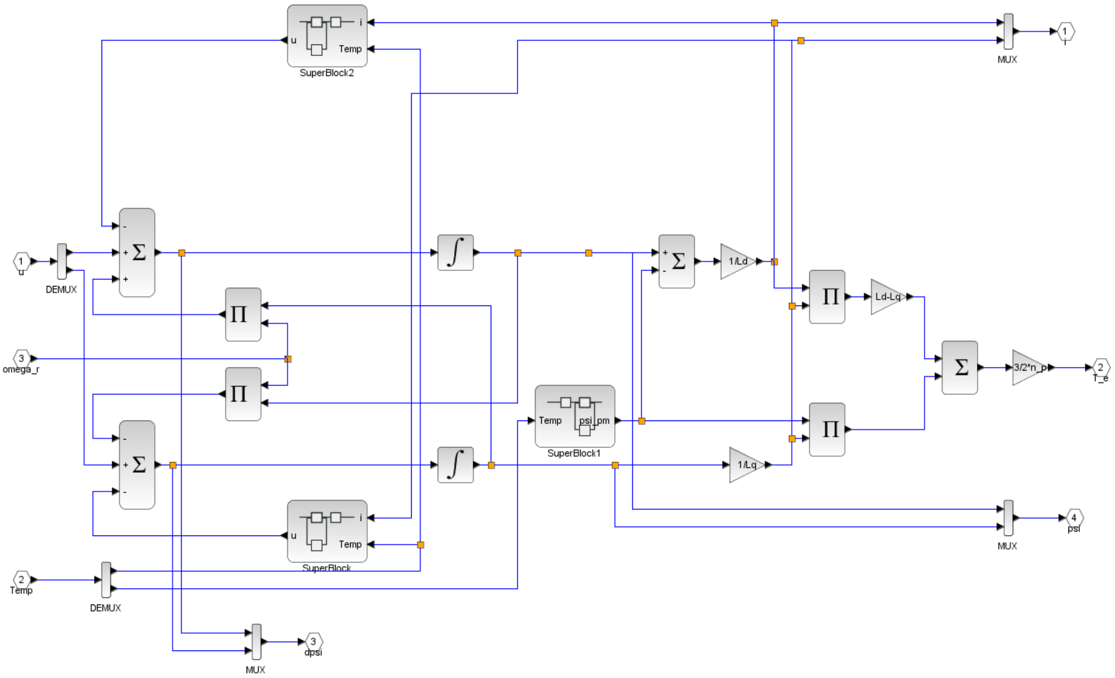


Figure 7: mc_foc_sl_fip_dspic33ck_mclv2_3

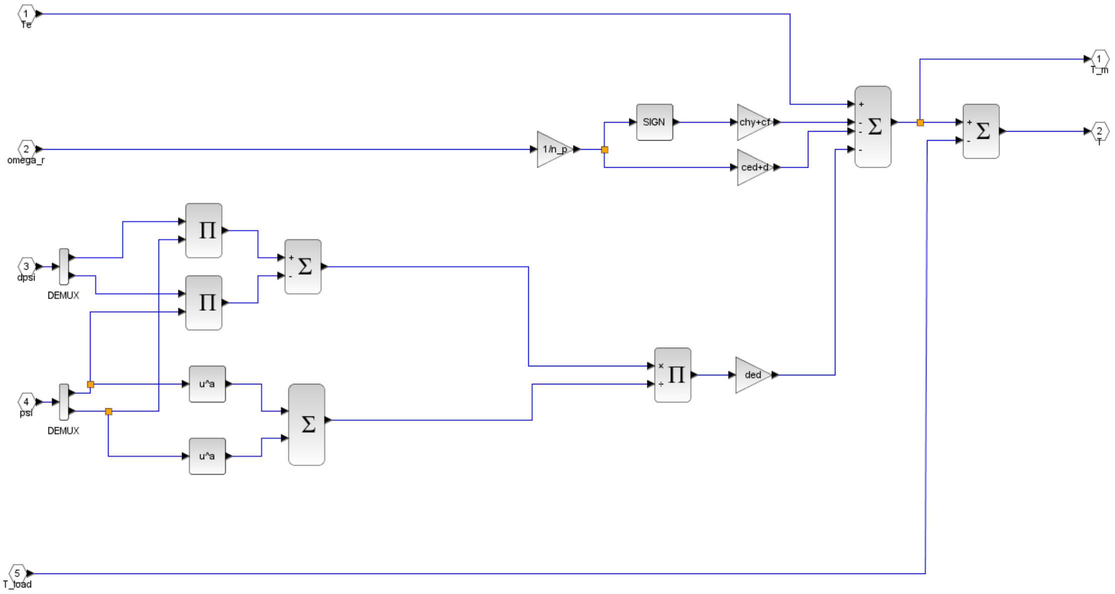


Figure 8: mc_foc_sl_fip_dspic33ck_mclv2_2

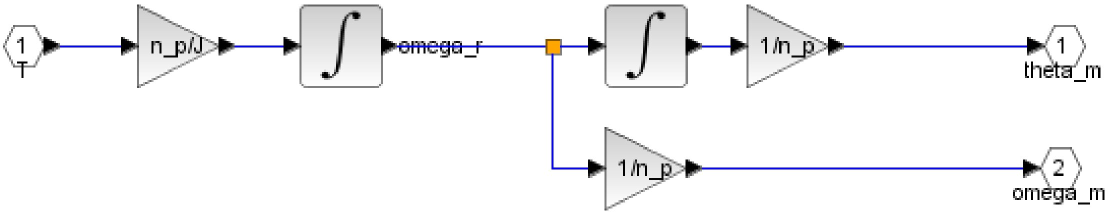


Figure 9: mc_foc_sl_fip_dspic33ck_mclv2_1

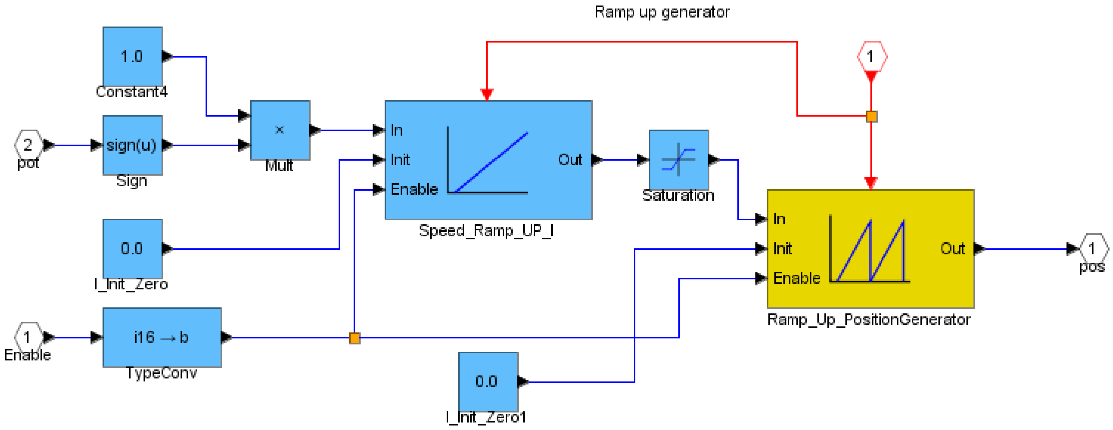


Figure 10: mc_foc_sl_fip_dspic33ck_mclv2_0

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

3.2 Scilab Parameter

```
// Scilab script file to store Model parameters
// This file is automatically executed by initProject.sce
// initScript.sce and this script is executed automatically, if model is opened from MPLAB X MCC

// Simulation settings
endTime      = 5;
stepSize     = 1.0E-2;
X2C_sampleTime =50E-6;

// CODE GENERATION PARAMETERS

// Speed PI
SpeedKi = 1;
SpeedKi = 0.5;

// Current PI
PIFluxKp = 0.8;
PIFluxKi = 0.5;

PITorqueKp = 0.8;
PITorqueKi = 0.5;

// PLL parameters
MotorLs = 0.41;
MotorRs = 0.35;
U_DCLINK = 24;
I_MAX = 3.95;
BEMF_D_UDC = 0.65;
PLL_INT = 800;









// POWERSTAGE DATA
Vbus = 24;
Rshunt = 0.025;
Igain = -15;

// MOTORDATEN
Nm_ozin = 7.061552e-3;
KRPM_rads = 0.060/2/%pi;

Rs = 4.03;
Ld = 4.60e-3;
Lq = 4.60e-3;
Kell = 7.24;
n_p = 5;        // number of polepairs


J = 0.000628 * Nm_ozin;   // inertia
cf = 0;
chy = 0;
d = 0;
ced = 0;
```

```
ded = 0;

alphaCU = 0;
alphaPM = 0;
Temp_nom = 25;
omega_m0 = 0;
theta_m0 = -0.5;
psi_pm = Kell/sqrt(3)*60/(2*%pi*1000)/n_p;
theta_r0 = theta_m0*2*%pi*n_p;
omega_r0 = omega_m0/60*2*%pi*n_p;










// Umrechnen auf Rechnenwerte fuer Modell
Ld = Ld/2;
Lq = Lq/2;
Rs = Rs/2;




// initalize input for simulation
exec("./gen_inputs.sci");
exec("./qei_sim.sce");
exec("./qei_sim2.sce");
```

Listing 1: ModelParameter.sce

4 Mask Parameter

Constant: Constant3	
Value	0.0
Used Implementation	Bool

Delay: Delay1	
ts_fact	1.0
Used Implementation	FiP16

Delay: Delay2	
ts_fact	1.0
Used Implementation	FiP16

Clarke_Park_MCHP: Clarke_Park_MCHP	
Used Implementation	FiP16

Constant: Constant	
Value	0.2
Used Implementation	FiP16

Constant: Constant1	
Value	1.0
Used Implementation	FiP16

Gain: Gain	
Gain	0.5
Used Implementation	FiP16

Gain: Gain1	
Gain	0.98
Used Implementation	FiP16

Gain: Gain2	
Gain	0.98
Used Implementation	FiP16

ManualSwitch: ManualSwitch	
Toggle	1.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch1	
Toggle	1.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch2	
Toggle	1.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch3	
Toggle	1.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch4	
Toggle	1.0
Used Implementation	FiP16

Not: Not	
Used Implementation	Bool

Constant: OpenLoop_Vd	
Value	0.0
Used Implementation	FiP16

Constant: OpenLoop_Vq	
Value	0.3
Used Implementation	FiP16

PI: PI_flux	
Kp	0.8
Ki	0.5
ts_fact	1.0
Used Implementation	FiP16

PI: PI_torque	
Kp	0.8
Ki	5.0
ts_fact	1.0
Used Implementation	FiP16

BEMF_MCHP: BEMF_MCHP	
Ls	0.1
Rs	0.1
U0	24.0
I0	1.0
ts_fact	1.0
CurrentSampleFactor	1.0
Used Implementation	FiP16

Constant: Constant1	
Value	0.0
Used Implementation	FiP16

Delay: Delay1	
ts_fact	1.0
Used Implementation	FiP16

PT1: Edfilter	
V	1.0
fc	400.0
ts_fact	1.0
method	zoh
Used Implementation	FiP16

PT1: Eqfilter	
V	1.0
fc	400.0
ts_fact	1.0
method	zoh
Used Implementation	FiP16

Park_MCHP: Park_MCHP	
Used Implementation	FiP16

Add: Add	
Used Implementation	FiP16

AutoSwitch: AutoSwitch	
Thresh_up	0.0
Thresh_down	0.0
Used Implementation	FiP16

Gain: Gain	
Gain	0.65
Used Implementation	FiP16

Sub: Sub	
Used Implementation	FiP16

uI: uI	
Ki	800.0
ts_fact	1.0
Used Implementation	FiP16

Park_Clarke_inv_SVM_MCHP: Park_Clarke_inv_SVM_MCHP	
Used Implementation	FiP16

Saturation: Saturation	
max	0.98
min	-0.98
Used Implementation	FiP16

Saturation: Saturation1	
max	0.98
min	-0.98
Used Implementation	FiP16

Sin3Gen: Sin3Gen	
fmax	100.0
Offset	0.0
ts_fact	1.0
Used Implementation	FiP16

Sub: Sub_flux	
Used Implementation	FiP16

Sub: Sub_torque	
Used Implementation	FiP16

Gain: Gain	
Gain	0.8
Used Implementation	FiP16

ManualSwitch: ManualSwitch	
Toggle	1.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch1	
Toggle	1.0
Used Implementation	FiP16

Not: Not	
Used Implementation	Bool

Constant: Constant	
Value	0.05
Used Implementation	FiP16

Constant: Constant1	
Value	0.0
Used Implementation	FiP16

ManualSwitch: ManualSwitch	
Toggle	0.0
Used Implementation	FiP16

PI: PI_speed	
Kp	1.0
Ki	0.5
ts_fact	10.0
Used Implementation	FiP16

RateLimiter: RateLimiter	
Tr	0.0010
Tf	0.0010
ts_fact	10.0
Used Implementation	FiP16

Constant: Speed_Init	
Value	1.0
Used Implementation	Bool

Constant: Speed_Init1	
Value	0.0
Used Implementation	FiP16

Sub: Speed_error	
Used Implementation	FiP16

Add: Add	
Used Implementation	FiP16

AutoSwitch: AutoSwitch	
Thresh_up	0.0
Thresh_down	0.0
Used Implementation	FiP16

Constant: Constant	
Value	1.0
Used Implementation	FiP16

Constant: Constant1	
Value	0.01
Used Implementation	FiP16

Constant: Constant2	
Value	1.0
Used Implementation	Bool

Constant: Constant3	
Value	0.0
Used Implementation	FiP16

Gain: Gain	
Gain	-1.0
Used Implementation	FiP16

RateLimiter: RateLimiter	
Tr	0.0
Tf	0.0
ts_fact	10.0
Used Implementation	FiP16

SinGen: SinGen	
fmax	100.0
Offset	0.0
Phase	0.0
ts_fact	10.0
Used Implementation	FiP16

Constant: Constant1	
Value	0.0
Used Implementation	FiP16

Constant: Constant2	
Value	0.0
Used Implementation	FiP16

Constant: Constant5	
Value	1.0
Used Implementation	Bool

AutoSwitch: Flux_select	
Thresh_up	0.5
Thresh_down	0.5
Used Implementation	FiP16

AutoSwitch: Flux_select1	
Thresh_up	0.5
Thresh_down	0.5
Used Implementation	FiP16

RateLimiter: IdRateLimiter	
Tr	0.5
Tf	3.0
ts_fact	1.0
Used Implementation	FiP16

AutoSwitch: Iq_select	
Thresh_up	0.5
Thresh_down	0.5
Used Implementation	FiP16

PI: PI	
Kp	0.05
Ki	0.0050
ts_fact	1.0
Used Implementation	FiP16

uSub: PosError	
Used Implementation	FiP16

AutoSwitch: PosSwitch	
Thresh_up	0.5
Thresh_down	0.5
Used Implementation	FiP16

Constant: Ramp_Up_Current	
Value	0.3
Used Implementation	FiP16

Constant: Constant4	
Value	1.0
Used Implementation	FiP16

Constant: I_Init_Zero	
Value	0.0
Used Implementation	FiP16

Constant: I_Init_Zero1	
Value	0.0
Used Implementation	FiP16

Mult: Mult	
Used Implementation	FiP16

uI: Ramp_Up_PositionGenerator	
Ki	100.0
ts_fact	1.0
Used Implementation	FiP16

Saturation: Saturation	
max	2.0
min	-2.0
Used Implementation	FiP16

Sign: Sign	
Used Implementation	FiP16

I: Speed_Ramp_UP_I	
Ki	1.0
ts_fact	1.0
Used Implementation	FiP16

TypeConv: TypeConv	
Used Implementation	FiP16_Bool

Sequencer: Sequencer	
Delay1	0.2
Delay2	1.0
Delay3	2.2
Delay4	3.0
ts_fact	1.0
Used Implementation	FiP16

TypeConv: TypeConv	
Used Implementation	FiP16_Bool

TypeConv: TypeConv1	
Used Implementation	FiP16_Bool

TypeConv: TypeConv2	
Used Implementation	FiP16_Bool

Part II

Frame Program Documentation

5 Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

6 Data Structure Documentation

6.1 _TMR_OBJ_STRUCT Struct Reference

6.1.1 Detailed Description

TMR1 Generated Driver API Source File
@Company Microchip Technology Inc.
@File Name tmr1.c
@Summary This is the generated source file for the TMR1 driver using PIC24 / dsPIC33 / PIC32MM MCUs
@Description This source file provides APIs for driver for TMR1. Generation Information : Product Revision : PIC24 / dsPIC33 / PIC32MM MCUs - 1.167.0 Device : dsPIC33CK256MP508 The generated drivers are tested against the following: Compiler : XC16 v1.50 MPLAB : MPLAB X v5.35Section: Included FilesSection: File specific functionsSection: Data Type DefinitionsTMR Driver Hardware Instance Object
@Summary Defines the object required for the maintenance of the hardware instance.
@Description This defines the object required for the maintenance of the hardware instance. This object exists once per hardware instance of the peripheral.
Remarks: None.
The documentation for this struct was generated from the following file:

- tmr1.c

7 Example Documentation

7.1 C:/LCM/X2C/_WorkApplications/mc_foc_sl_fip_dspic33ck_mclv2/mcc_generated_files/re

It handles the reset cause by clearing the cause register values. Its a weak function user can override this function.

Returns

None

```
RESET_CauseHandler();
/*
    (c) 2020 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/
#ifndef RESET_H
#define RESET_H
#include <stdint.h>
#include "reset_types.h"
uint16_t RESET_GetCause(void);
void __attribute__ ((weak)) RESET_CauseHandler(void);
void RESET_CauseClearAll();
#endif  /* RESET_H */
```

7.2 C:/LCM/X2C/_WorkApplications/mc_foc_sl_fip_dspic33ck_mclv2/mcc_generated_files/sy

Initializes the CPU core control register.

`SYSTEM_CORCONInitialize();`

```
/*
    (c) 2020 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/
#ifndef _XTAL_FREQ
#define _XTAL_FREQ  200000000UL
#endif
#define WDT_CLR_KEY 0x5743
#include "xc.h"
#include "stdint.h"
#include "system_types.h"
#ifndef SYSTEM_H
#define SYSTEM_H
inline static void SYSTEM_CORCONInitialize()
{
    CORCON = (CORCON & 0x00F2) | CORCON_MODE_PORVALUES;    // POR value
}
inline static void SYSTEM_CORCONModeOperatingSet(SYSTEM_CORCON_MODES modeValue)
{
    CORCON = (CORCON & 0x00F2) | modeValue;
}
inline static void SYSTEM_CORCONRegisterValueSet(uint16_t value)
{
    CORCON = value;
}
inline static uint16_t SYSTEM_CORCONRegisterValueGet(void)
{
    return CORCON;
}
inline static uint32_t SYSTEM_DeviceIdRegisterAddressGet(void)
{
    return __DEVID_BASE;
}
void SYSTEM_Initialize(void);
#endif  /* SYSTEM_H */
```

7.3 C:/LCM/X2C/_WorkApplications/mc_foc_sl_fip_dspic33ck_mclv2/mcc_generated_files/wa

Enables Watch Dog Timer (WDT) using the software bit.

```
WATCHDOG_TimerSoftwareEnable();
```

```
/*
    (c) 2020 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/
#ifndef WATCHDOG_H
#define WATCHDOG_H
#define WATCHDOG_CLR_KEY 0x5743
inline static void WATCHDOG_TimerSoftwareEnable(void)
{
    WDTCONLbits.ON = 1;
}
inline static void WATCHDOG_TimerSoftwareDisable(void)
{
    WDTCONLbits.ON = 0;
}
inline static void WATCHDOG_TimerClear(void)
{
    WDTCONH = WATCHDOG_CLR_KEY;
}
#endif  /* WATCHDOG_H */
```

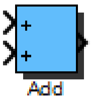
Part III

Used X2C-Blocks

8 Project Specific Blocks

9 Internal Library Blocks

Block: Add



Inports	
In1	Addend 1
In2	Addend 2

Outports	
Out	Sum

Description:

Addition of input 1 and input 2.

Calculation:

$$\mathrm{Out} = \mathrm{In}_1 + \mathrm{In}_2$$

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
In2	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In1	int16
In2	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In1	int32
In2	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In1	float32
In2	float32

Outports Data Type	
Out	float32

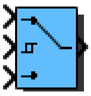
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
In2	float64

Outports Data Type	
Out	float64

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

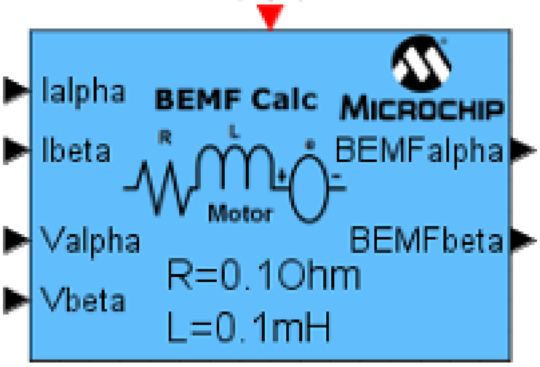
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
Switch	float64
In3	float64

Outports Data Type	
Out	float64

Block: BEMF_calc



Inports	
Ialpha	Clarke transformed A phase current
Ibeta	Clarke transformed B phase current
Valpha	Clarke transformed Output A phase voltage
Vbeta	Clarke transformed Output B phase voltage

Outports	
BEMFalpha	back electromotive force alpha voltage
BEMFbeta	back electromotive force beta voltage

Mask Parameters	
Ls	Motor phase inductance for Y connection (MilliHenry)
Rs	Motor phase resistance for Y connection (Ohm)
U0	U0: DC link voltage (Only in Fip implementation)
I0	The maximum peak current per phase (Only in Fip implementation)
ts_fact	Multiplication factor of base sampling time (in integer format)
CurrentSampleFactor	Division factor for current sample. (in integer format) Current sample = ModelSampleTime/CurrentSampleFactor

Description:

BEMFalpha=Va-Rs0*Ia-Ls0*deltaIa
BEMFbeta=Vb-Rs0*Ib-Ls0*deltaIb
FIP:
Rs0 = Rs*(I0/U0)
Ls0 = Ls*(I0/U0)
Float:

Rs0 = Rs
Ls0 = Ls

Implementations:

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

Implementation: FiP16

Name	FiP16
ID	20992
Revision	0.2
C filename	BEMF_calc_FiP16.c
H filename	BEMF_calc_FiP16.h

16 Bit Fixed Point Implementation

Controller Parameters	
Ls	Motor phase inductance. Scales by I0/U0
Rs	Motor phase resistance. Scales by I0/U0
sfrLs	Shift factor for Ls
sfrRs	Shift factor for Rs
Ib_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
Ia_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
CurrentSampleFactor	Division factor for current sample. Current sample: ModelSampleTime/CurrentSampleFactor
V_Ls_alpha	Voltage of inductance. (Save data for next calculation)
V_Ls_beta	Voltage of inductance (Save data for next calculation)
FactCounter	Current Sample Factor loop counter

Data Structure:

```
typedef struct {
          uint16          ID;
          int16           *Ialpha;
          int16           *Ibeta;
          int16           *Valpha;
          int16           *Vbeta;
          int16           BEMFalpha;
          int16           BEMFbeta;
          int16           Ls;
```

```
    int16          Rs;
    int8           sfrLs;
    int8           sfrRs;
    int16          Ib_old;
    int16          Ia_old;
    uint8          CurrentSampleFactor;
    int16          V_Ls_alpha;
    int16          V_Ls_beta;
    uint8          FactCounter;
} BEMF_CALC_FIP16;
```

Implementation: FiP32

Name	FiP32
ID	20993
Revision	0.2
C filename	BEMF_calc_FiP32.c
H filename	BEMF_calc_FiP32.h

32 Bit Fixed Point Implementation

Controller Parameters	
Ls	Motor phase inductance. Scales by I0/U0
Rs	Motor phase resistance Scales by I0/U0
sfrLs	Shift factor for Ls
sfrRs	Shift factor for Rs
Ia_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
Ib_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
CurrentSampleFactor	Division factor for current sample. Current sample: ModelSampleTime/CurrentSampleFactor
V_Ls_alpha	Voltage of inductance. (Save data for next calculation)
V_Ls_beta	Voltage of inductance. (Save data for next calculation)
FactCounter	Current Sample Factor loop counter

Data Structure:

```
typedef struct {
    uint16         ID;
    int32          *Ialpha;
    int32          *Ibeta;
    int32          *Valpha;
    int32          *Vbeta;
    int32          BEMFalpha;
```

```
        int32           BEMFbeta;
        int32           Ls;
        int32           Rs;
        int8            sfrLs;
        int8            sfrRs;
        int32           Ia_old;
        int32           Ib_old;
        uint8           CurrentSampleFactor;
        int32           V_Ls_alpha;
        int32           V_Ls_beta;
        uint8           FactCounter;
} BEMF_CALC_FIP32;
```

Implementation: Float32

Name	Float32
ID	20994
Revision	0.2
C filename	BEMF_calc_Float32.c
H filename	BEMF_calc_Float32.h

32 Bit Floating Point Implementation

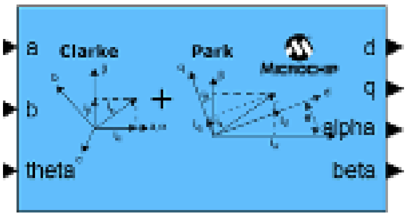
Controller Parameters	
Ls	Motor phase inductance
Rs	Motor phase resistance
Ia_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
Ib_old	Input value from previous cycle. Sample time divided by CurrentSampleFactor
CurrentSampleFactor	Division factor for current sample. Current sample: ModelSampleTime/CurrentSampleFactor
V_Ls_alpha	Voltage of inductance. (Save data for next calculation)
V_Ls_beta	Voltage of inductance. (Save data for next calculation)
FactCounter	Current Sample Factor loop counter

Data Structure:

```
typedef struct {
        uint16          ID;
        float32         *Ialpha;
        float32         *Ibeta;
        float32         *Valpha;
        float32         *Vbeta;
        float32         BEMFalpha;
        float32         BEMFbeta;
        float32         Ls;
        float32         Rs;
        float32         Ia_old;
```

```
    float32         Ib_old;
    uint8           CurrentSampleFactor;
    float32         V_Ls_alpha;
    float32         V_Ls_beta;
    uint8           FactCounter;
} BEMF_CALC_FLOAT32;
```

Block: Clark_Park



Inports	
a	A phase current
b	B phase current
theta	Rotating position

Outports	
d	d steady axis (typically flux)
q	q steady axis (typically torque)
alpha	Shifted A phase current
beta	Shifted B phase current

Description:

Merged Clark and park transform.
d=a*cos(theta)+((a*2b)*1.732)*sin(theta)
q=-a*sin(theta)+((a*2b)*1.732)*cos(theta)
Moves a three-axis, two-dimensional
coordinate system, referenced to the stator, onto a
two-axis system, keeping the same reference.
The two-axis orthogonal system with the axis called
alpha-beta transform into another two-axis
system that is rotating with the rotor flux.
Two-axis rotating coordinate system is
called the d-q axis.
Theta represents the rotor angle.

Implementations:

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

Implementation: FiP16

Name	FiP16
ID	20960
Revision	0.1
C filename	Clark_Park_FiP16.c
H filename	Clark_Park_FiP16.h

16 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {
     uint16         ID;
     int16          *a;
     int16          *b;
     int16          *theta;
     int16          d;
     int16          q;
     int16          alpha;
     int16          beta;
} CLARK_PARK_FIP16;
```

Implementation: FiP32

Name	FiP32
ID	20961
Revision	0.1
C filename	Clark_Park_FiP32.c
H filename	Clark_Park_FiP32.h

32 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {
     uint16         ID;
     int32          *a;
     int32          *b;
     int32          *theta;
     int32          d;
     int32          q;
     int32          alpha;
     int32          beta;
} CLARK_PARK_FIP32;
```

Implementation: Float32

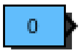
Name	Float32
ID	20962
Revision	0.1
C filename	Clark_Park_Float32.c
H filename	Clark_Park_Float32.h

32 Bit Floating Point Implementation

Data Structure:

```
typedef struct {
     uint16         ID;
     float32        *a;
     float32        *b;
     float32        *theta;
     float32        d;
     float32        q;
     float32        alpha;
     float32        beta;
} CLARK_PARK_FLOAT32;
```

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

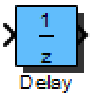
Outports Data Type	
Out	float32

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: Gain

Inports	
In	Input

Outports	
Out	Amplified input

Mask Parameters		
Name	**ID**	**Description**
Gain	1	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

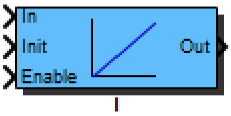
Implementation: Float64

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		
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):

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

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

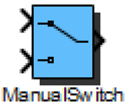
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
Init	float64
Enable	bool

Outports Data Type	
Out	float64

Block: ManualSwitch



Inports	
In1	Input #1
In2	Input #2

Outports	
Out	

Mask Parameters		
Name	**ID**	**Description**
Toggle	1	Toggle

Description:

Toggling between inputs by double-clicking on block.

Doubleclicking of the *ManualSwitch* block changes the routing of the input signals and doesn't open the *Function Block Parameters* dialog. So if changing the implementation is required, one has to open the dialog via *Mask Parameters* command of the context menu.

9.0.0.1 Developer note:

To get the double-click feature the callback function of *OpenFnc* in *Block Properties* is manually altered to

```
if get_param(gcb,'Toggle') == '0'
    set_param(gcb,'Toggle', '1');
else
    set_param(gcb,'Toggle', '0');
end
setBlockData(gcs, gcb);
initSFunction(gcb);
```

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

Inports Data Type	
In1	bool
In2	bool

Outports Data Type	
Out	bool

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In1	int8
In2	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In1	int16
In2	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In1	int32
In2	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In1	float32
In2	float32

Outports Data Type	
Out	float32

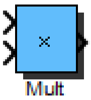
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
In2	float64

Outports Data Type	
Out	float64

Block: Mult



Inports	
In1	Multiplicand 1
In2	Multiplicand 2

Outports	
Out	Product

Description:

Multiplication of input 1 with input 2.

Calculation:

$$\mathrm{Out} = \mathrm{In}_1 \cdot \mathrm{In}_2$$

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
In2	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In1	int16
In2	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In1	int32
In2	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In1	float32
In2	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
In2	float64

Outports Data Type	
Out	float64

Block: Not

Inports	
In	

Outports	
Out	

Description:

Logical inverter block.

Implementations:

Bool	Boolean Implementation

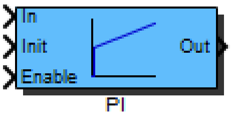
Implementation: Bool

Boolean Implementation

Inports Data Type	
In	bool

Outports Data Type	
Out	bool

Block: PI



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	

Mask Parameters		
Name	**ID**	**Description**
Kp	1	Proportional Factor
Ki	2	Integral Factor
ts_fact	3	Multiplication factor of base sampling time (in integer format)

Description:

PI controller:
G(s) = Kp + Ki/s

Each fixed point implementation uses the next higher integer data type for the integral value storage variable.
A rising flank at the *Enable* inport will preload the integral part with the value present on the *Init* inport.

Transfer function (zero-order hold discretization method):

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

9.0.0.2 Developer note:

For the fixed point implementations the source code of block **??** is used.

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	int8

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

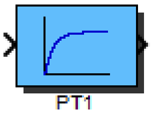
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
Init	float64
Enable	bool

Outports Data Type	
Out	float64

Block: PT1



Inports	
In	Input In(k)

Outports	
Out	Output Out(k)

Mask Parameters		
Name	**ID**	**Description**
V	1	Gain
fc	2	Cut off frequency of low pass filter
ts_fact	3	Multiplication factor of base sampling time (in integer format)
method	4	Discretization method

Description:

First order low pass:
G(s) = V/(s/w + 1)

Due to limited value range in the 8 bit fixed point implementation rather high deviations from expected output values may occur.

9.0.0.3 Developer note:

The source code of block *TF1* is used.

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

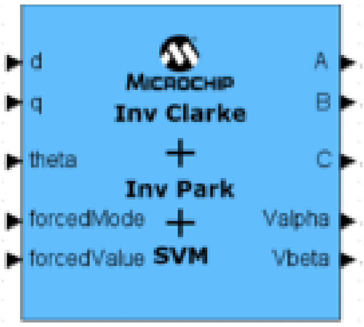
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Park_Clark_inv



Inports	
d	d voltage axis in rotating system
q	q voltage axis in rotating system
theta	Rotating position
forcedMode	Force Motor to base position. 0 -> No force. Normal mode. 1 -> Force the motor to base position
forcedValue	The forced voltage value

Outports	
A	Phase A out
B	Phase B out
C	Phase C out
Valpha	Inverse Park Voltage output
Vbeta	Inverse Park Voltage output

Description:

Merged Inverse Park, Iverse Clark and Space Vector Modulation
Park_inv:
alpha = d*cos(theta) - q*sin(theta)
beta = d*sin(theta) + q*cos(theta)
Clark_inv:
a = beta
b = (-beta + 1.732*alpha)/2
c = (-beta - 1.732*alpha)/2
SVM:
Optimize output for PWM
Description:
Transform from the two-axis rotating d-q frame
to the two-axis stationary frame ?-?.
Transform from the stationary two-axis ?-? frame
to the stationary three-axis, 3-phase reference frame.
Optimize output for PWM

Implementations:

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

Implementation: FiP16

Name	FiP16
ID	20976
Revision	0.1
C filename	Park_Clark_inv_FiP16.c
H filename	Park_Clark_inv_FiP16.h

16 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {
     uint16            ID;
     int16             *d;
     int16             *q;
     int16             *theta;
     int8              *forcedMode;
     int16             *forcedValue;
     int16             A;
     int16             B;
     int16             C;
     int16             Valpha;
     int16             Vbeta;
} PARK_CLARK_INV_FIP16;
```

Implementation: FiP32

Name	FiP32
ID	20977
Revision	0.1
C filename	Park_Clark_inv_FiP32.c
H filename	Park_Clark_inv_FiP32.h

32 Bit Fixed Point Implementation

Data Structure:

```
typedef struct {
     uint16            ID;
     int32             *d;
     int32             *q;
     int32             *theta;
     int8              *forcedMode;
     int32             *forcedValue;
     int32             A;
```

```
    int32          B;
    int32          C;
    int32          Valpha;
    int32          Vbeta;
} PARK_CLARK_INV_FIP32;
```

Implementation: Float32

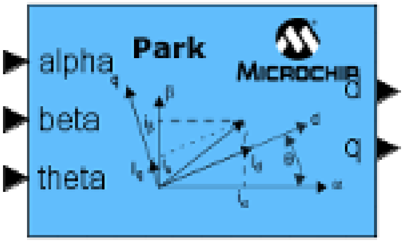
Name	Float32
ID	20978
Revision	0.1
C filename	Park_Clark_inv_Float32.c
H filename	Park_Clark_inv_Float32.h

32 Bit Floating Point Implementation

Data Structure:

```
typedef struct {
    uint16         ID;
    float32        *d;
    float32        *q;
    float32        *theta;
    int8           *forcedMode;
    float32        *forcedValue;
    float32        A;
    float32        B;
    float32        C;
    float32        Valpha;
    float32        Vbeta;
} PARK_CLARK_INV_FLOAT32;
```

Block: Park



Inports	
alpha	
beta	
theta	

Outports	
d	
q	

Description:

The two-axis orthogonal system with the axis called
alpha-beta transform into another two-axis
system that is rotating with the rotor flux.
Two-axis rotating coordinate system is
called the d-q axis.
Theta represents the rotor angle.

Implementations:

FiP16	16 Bit Fixed Point Implementation (uses MCHP dsp if possible)
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation

Implementation: FiP16

Name	FiP16
ID	20880
Revision	0.2
C filename	Park_FiP16.c
H filename	Park_FiP16.h

16 Bit Fixed Point Implementation (uses MCHP dsp if possible)

Data Structure:

```c
typedef struct {
    uint16      ID;
    int16       *alpha;
    int16       *beta;
    int16       *theta;
    int16       d;
    int16       q;
} PARK_FIP16;
```

Implementation: FiP32

Name	FiP32
ID	20881
Revision	0.1
C filename	Park_FiP32.c
H filename	Park_FiP32.h

32 Bit Fixed Point Implementation

Data Structure:

```c
typedef struct {
    uint16      ID;
    int32       *alpha;
    int32       *beta;
    int32       *theta;
    int32       d;
    int32       q;
} PARK_FIP32;
```

Implementation: Float32

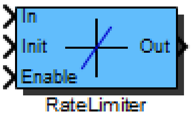
Name	Float32
ID	20882
Revision	0.1
C filename	Park_Float32.c
H filename	Park_Float32.h

32 Bit Floating Point Implementation

Data Structure:

```c
typedef struct {
    uint16      ID;
    float32     *alpha;
    float32     *beta;
    float32     *theta;
    float32     d;
    float32     q;
} PARK_FLOAT32;
```

Block: RateLimiter



Inports	
In	
Init	Value which is loaded at rising flanke of enable signal
Enable	Enable == 0: Deactivation of block; Out is set to In. Enable != 0: Activation of block; Out is rate limited. Enable 0->1: Preloading of output; Out is set to value of Init input

Outports	
Out	

Mask Parameters		
Name	**ID**	**Description**
Tr	1	Rising time in seconds. Slew rate will be 1/Tr
Tf	2	Falling time in seconds. Slew rate will be 1/Tf
ts_fact	3	Multiplication factor of base sampling time (in integer format)

Description:

Limitation of rising and falling rate.
Function of Enable:
0: rate limiting disabled, signal is passed through
1: rate limiting enabled, signal is rate limited
0->1: preload of output with value from init input

Rising and falling time refer to a step from 0 to 1. Entries for *Tr: Rising time* and *Tf: Falling time* smaller than the actual sample time will be limited to the sample time internally.
The 16- and 32-Bit fixed point implementations are based on an internal 32-Bit wide slew-rate variable while the 8-Bit fixed point implementation uses a 16-Bit wide slew-rate variable.

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
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: Saturation

Inports	
In	Input

Outports	
Out	Limited output

Mask Parameters		
Name	**ID**	**Description**
max	1	Upper Limit
min	2	Lower Limit

Description:

Saturation of output to adjustable upper and lower limit.

If the entry for *Upper Limit* is lower than the entry for *Lower Limit* then the limits will be swapped internally.

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

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

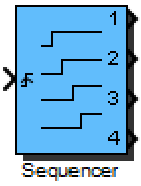
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Sequencer



Inports	
Start	Start signal. Rising flank triggers sequence

Outports	
Out1	Output #1
Out2	Output #2
Out3	Output #3
Out4	Output #4

Mask Parameters		
Name	**ID**	**Description**
Delay1	1	Time delay for output 1
Delay2	2	Time delay for output 2
Delay3	3	Time delay for output 3
Delay4	4	Time delay for output 4
ts_fact	5	Multiplication factor of base sampling time (in integer format)

Description:

Generation of time delayed (enable) sequence.

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	
Start	int8

Outports Data Type	
Out1	int8
Out2	int8
Out3	int8
Out4	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
Start	int16

Outports Data Type	
Out1	int16
Out2	int16
Out3	int16
Out4	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
Start	int32

Outports Data Type	
Out1	int32
Out2	int32
Out3	int32
Out4	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
Start	float32

Outports Data Type	
Out1	float32
Out2	float32
Out3	float32
Out4	float32

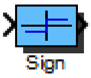
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
Start	float64

Outports Data Type	
Out1	float64
Out2	float64
Out3	float64
Out4	float64

Block: Sign



Inports	
In	Input u

Outports	
Out	Value corresponding to sign of u

Description:

Signum function.

Calculation:

$$\mathrm{Out} = \operatorname{sgn}(\mathrm{In}) = \begin{cases} 1 & \mathrm{In} \geq 0 \\ -1 & \mathrm{In} < 0 \end{cases}$$

Implementations:

FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed 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

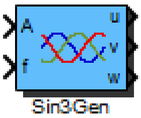
Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int32

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

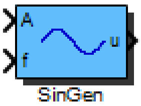
Implementation: Float64

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:

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

For sine calculation a lookup table with 256 entries is used. This results in a short computation time but with the downside of reduced accuracy for the FiP32 implementation.

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

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

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

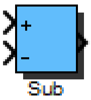
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64

Block: Sub



Inports	
Plus	Minuend
Minus	Subtrahend

Outports	
Out	Difference

Description:

Subtraction of input Minus from input Plus.

Calculation:

$$\mathrm{Out} = \mathrm{Plus} - \mathrm{Minus}$$

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	
Plus	int8
Minus	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
Plus	int16
Minus	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
Plus	int32
Minus	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
Plus	float32
Minus	float32

Outports Data Type	
Out	float32

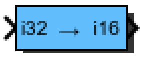
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
Plus	float64
Minus	float64

Outports Data Type	
Out	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

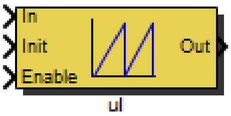
Implementation: FiP32_Bool

32 Bit Fixed Point to Boolean Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	bool

Block: uI



Inports	
In	Control error input
Init	Value which is loaded at initialization function call
Enable	Enable == 0: Deactivation of block; Out is set to 0. Enable 0->1: Preload of integral part. Enable == 1: Activation of block

Outports	
Out	Integrator output

Mask Parameters		
Name	**ID**	**Description**
Ki	1	Integral Factor
ts_fact	2	Multiplication factor of base sampling time (in integer format)

Description:

Integrator for angle signals:
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

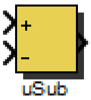
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
Init	float64
Enable	bool

Outports Data Type	
Out	float64

Block: uSub



Inports	
Plus	Minuend
Minus	Subtrahend

Outports	
Out	Difference

Description:

Subtraction of input Minus from input Plus with output wrapping.

Calculation:

$$\mathrm{Out} = \mathrm{Plus} - \mathrm{Minus}$$

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	
Plus	int8
Minus	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
Plus	int16
Minus	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
Plus	int32
Minus	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
Plus	float32
Minus	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
Plus	float64
Minus	float64

Outports Data Type	
Out	float64