SimCoder^{TM}

User's Guide

Powersim Inc.

SimCoder User's Guide

Version 10.0

Release 1

January 2015

Copyright © 2008-2015 Powersim Inc.

All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of Powersim Inc.

Disclaimer

Powersim Inc. ("Powersim") makes no representation or warranty with respect to the adequacy or accuracy of this documentation or the software which it describes. In no event will Powersim or its direct or indirect suppliers be liable for any damages whatsoever including, but not limited to, direct, indirect, incidental, or consequential damages of any character including, without limitation, loss of business profits, data, business information, or any and all other commercial damages or losses, or for any damages in excess of the list price for the licence to the software and documentation.

Powersim Inc.

email: info@powersimtech.com
http://www.powersimtech.com

Contents

7 PE-Pro/F28335 Hardware Target

8 PE-Expert3 Hardware Target

9 TI DMC Library

1

SimCoder Overview

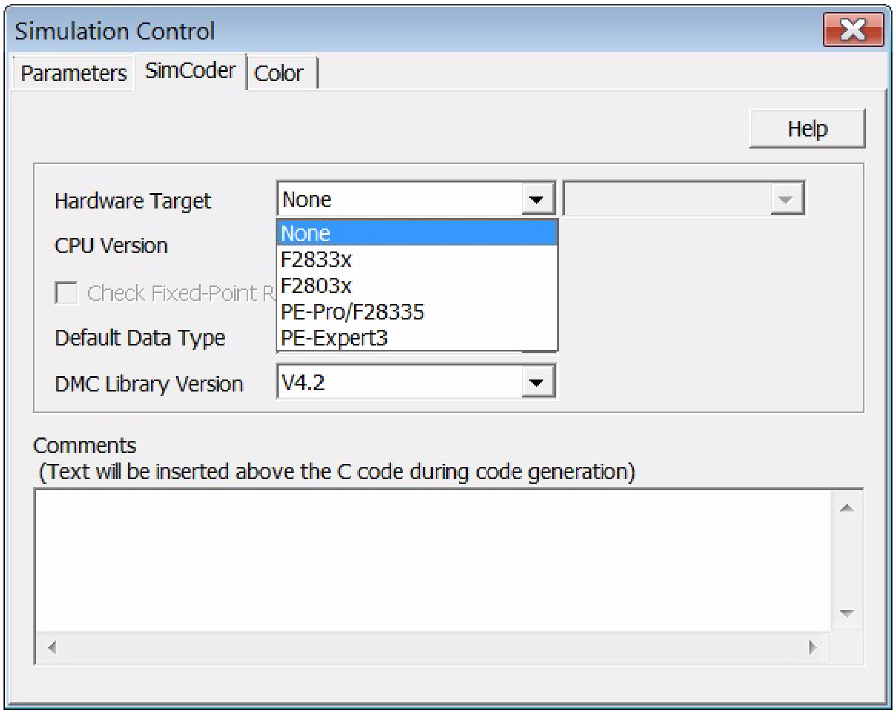
1.1 Introduction

SimCoder[1] is an add-on option of the PSIM software. It generates C code from PSIM schematics. With specific hardware target libraries, the C code generated by SimCoder can run directly on DSP hardware platforms.

This manual describes how to use SimCoder.

1.2 SimCoder Setup in Simulation Control

SimCoder setup is done in the SimCoder tab in the Simulation Control block, as shown below. One must select and set these parameters properly in order for SimCoder to generate code correctly.



Simulation Control

Hardware Target: None

DMC Library Version: V4.2

Comments
(Text will be inserted above the C code during code generation)

The setup process is explained below.

Supported Hardware Targets:

SimCoder supports the following hardware targets:

- *None:* Generating code for simulation only, and not for any specific hardware setting.
- *F2833x*: Generating code for hardware using the floating-point TMSF2833x family DSP from Texas Instruments (TI).
- *F2803x*: Generating code for hardware using the TI fixed-point TMSF2803x family DSP
- *PE-Pro/F28335*: Generating code for the PE-Pro/F28335 DSP board. PE-Pro/F28335 is a DSP board produced by Myway Co. (www.myway.co.jp). It uses TI's floating-point DSP TMSF28335 and Myway's PE-OS library.

[1] SimCoder^{TM} is a trademark of Powersim Inc., and is copyright by Powersim Inc., 2008-2015

- *PE-Expert3*: Generating code for the PE-Expert3 DSP hardware. PE-Expert3 is a DSP development platform produced by Myway Co. It uses TI's floating-point DSP TMS320C6713 and Myway's PE-OS library.

Project Configuration

For *F2833x* and *F2803x* Targets, the project configuration can be set as *RAM Debug*, *RAM Release*, *Flash Release*, or *Flash RAM Release*. For *PE-Expert3* Target, the project configuration can be set as *PE-View9* or *PE -View8*.

CPU Version:

For F2803x and F2833x Targets, there are subsequently the CPU version choices.

- For *F2833x* Target, the choices are F28335, F28334, and F28332
- For *F2803x* Target, the choices are from F28030 to F28035 with various pin configurations

Check Fixed-Point Range:

This is for the F2803x Target only. If the box is checked, the SimCoder will check the data in the simulation and provide a list of data range. In that list, the data which are near or over the range will be highlighted.

Default Data Type

When the hardware target is of floating-point type, this section is automatically chosen. If the hardware target is of fixed-point type or *None*, one must select one of the available default data types in the pull-down menu.

- For *F2803x* Target, It can be one of the following: Integer, IQ1, IQ2, .., IQ30.
- When there is no target, it can be one of the following: Float, Integer, IQ1, IQ2,..., IQ30

DMC Library Version

Texas Instruments's DMC (Digital Motor Control) library is composed of C functions (or macros) developed for C2000 DSP devices for motor control users.

To take advantage of this resource, SimCoder integrated the DMC library functions into the PSIM element library for code generation.

Some of these macros have different versions released by TI. SimCoder supports versions V4.0, V4.1, and V4.2. Please note that, once a specific version is selected, other versions' macros are disabled.

Comments

The Comments area at the bottom of the SimCoder tab allows users to add comments to the code generated by SimCoder. All text in this area will be added as comments to the beginning of the C code.

1.3 Elements for Code Generation

All the elements under the menus **Elements** >> **Event Control** and **Elements** >> **SimCoder** are for code generation.

Elements for each type of hardware targets can be found under the menu **Elements** >> **SimCoder**, under the sub-menus **F2833x Target**, **F2803x Target**, **PE-Pro/F28335 Target**, and **PE-Expert3 Target**.

Many elements in the standard PSIM library can also be used for code generation. In order to differentiate the elements in the standard library that can be used for code generation from the ones that can not, under **Options** >> **Settings** >> **Advanced**, if the option box "*Show image next to elements that can be used for code generation*" is checked, a small image CG will appear next to these elements that can be used for code generation.

Also, when this option box is checked, the image of TI will appear next to each of the elements for F2833x and F2803x Target, PR for PE-Pro/F28335 Target, and PE for PE-Expert3 Target.

For a list of elements in the standard library that can be used for code generation, please refer to Section 4.2.

2

Code Generation - A Step-by-Step Approach

2.1 Overview

In general, automatic code generation using SimCoder involves the following steps:

- Design and simulate a system in PSIM with the control in continuous domain.
- Convert the control section of the system into discrete domain and simulate the system.
- If there is no hardware target, place the control section in a subcircuit, and generate the code.
- If there is a hardware target, modify the system by including hardware elements, and run the simulation to validate the results. Then generate the code.

The first two steps, however, are not mandatory. One could, for example, create a schematic in PSIM and generate the code directly without simulating the system.

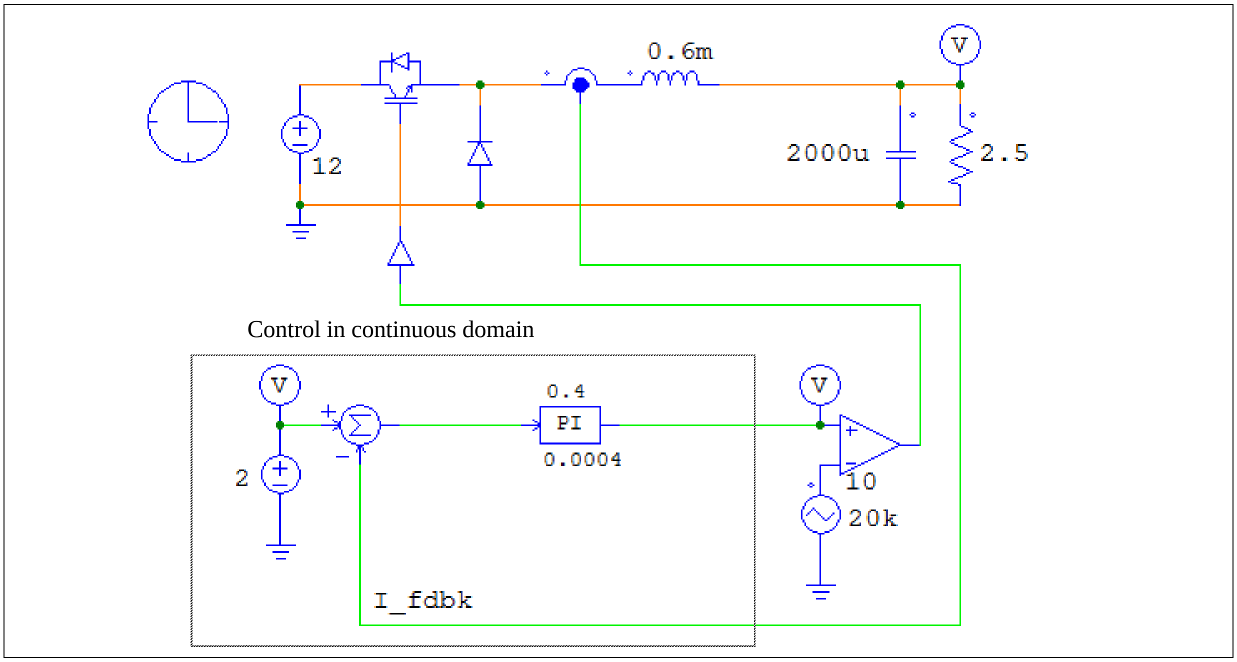
Please note that code can only be generated when control is in discrete domain, not in continuous domain. Therefore, Digital control Module is needed for SimCoder.

Simple examples are used in the sections below to illustrate the code generation process.

2.2 System in Continuous Domain

Often a system is designed and simulated in continuous domain first. Below is a simple dc converter circuit with current feedback. The PI (proportional-integral) controller in the control circuit is designed in the continuous s-domain. The PI gain *k* and time constant *T* are: *k* = 0.4, and *T* = 0.0004. The switching frequency is 20 kHz.

The objective of this exercise is to generate the C code for the control circuit in the dotted box. To perform the code generation, the first step is to convert the analog PI controller in s-domain to the digital PI controller in discrete z-domain.



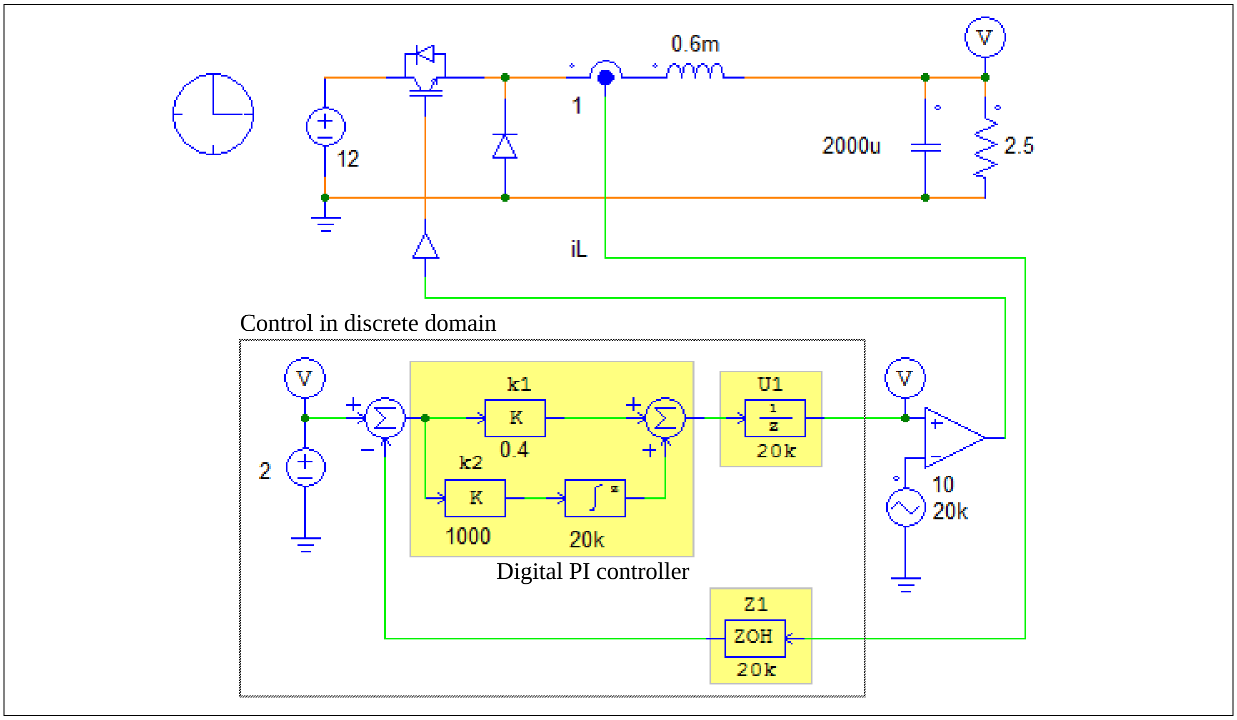
2.3 System in Discrete Domain

To convert an analog controller into a digital controller, one can use the *s2z Converter* program that comes with the Digital Control Module. To launch the program, in PSIM, choose **Utilities** >> **s2z Converter**.

Different conversion methods can be used to convert an analog controller to a digital controller. The most commonly used methods are Bilinear (also called Tustin or Trapezoidal) method and Backward Euler method.

In this example, we will use the Backward Euler method. With the sampling frequency the same as the switching frequency of 20 kHz, we will convert the analog PI controller to the digital PI controller. From the conversion program, we have the digital PI controller parameters as: k_1 = 0.4 for the proportional portion and k_2= 1000 for the integral portion.

The circuit with the digital controller is shown below:



As compared to the control circuit in continuous domain, there are three changes in this circuit, as highlighted by the yellow boxes. First, the analog PI controller is replaced by the digital PI controller. The "Algorithm Flag" of the digital integrator is set to 1 (for Backward Euler method), and the sampling frequency is set to 20 kHz. The gains k_1 and k_2 are obtained from the conversion program as described above.

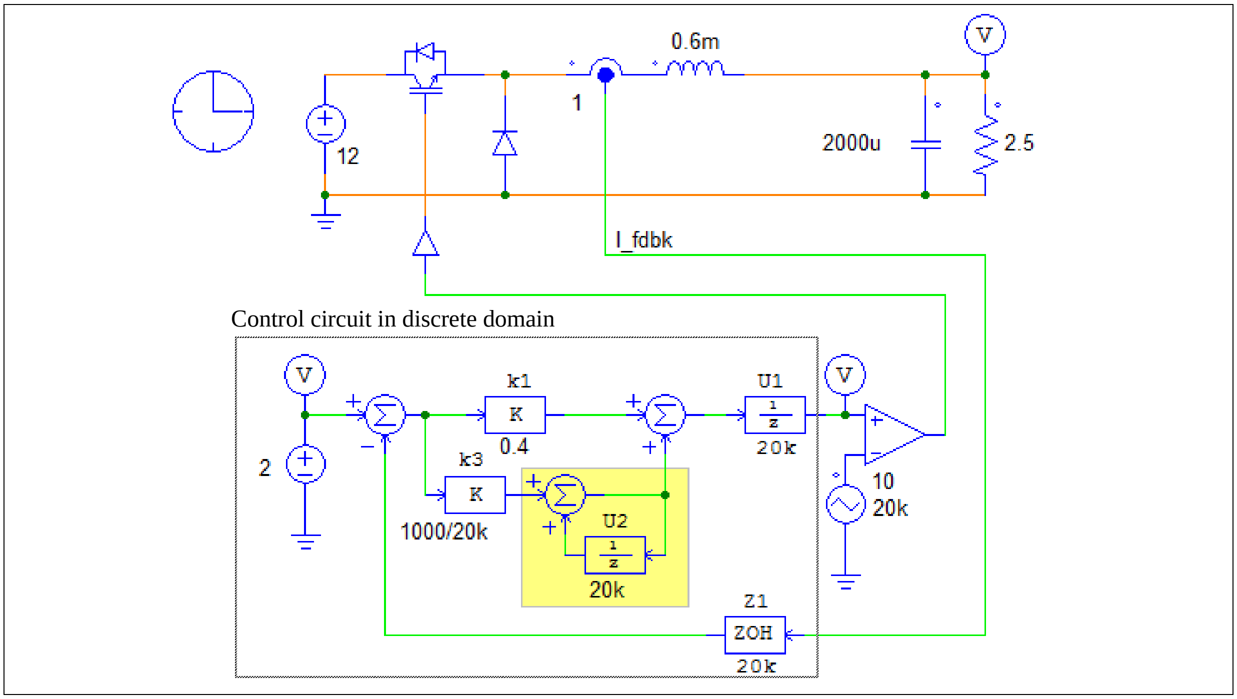
In addition, a zero-order-hold block Z_1 is used to simulate the A/D converter in digital hardware implementation for sampling the feedback current i_L. A unit-delay block U_1 is used to model the one-cycle delay inherent in digital control implementation. The delay is due to the fact that, usually quantities are sampled at the beginning of a cycle, and controller parameters are calculated within the cycle. But since it takes time to perform the calculation, the newly calculated quantities are normally not used until the beginning of the next cycle.

Note that the converted digital controller should result in a stable control loop and desired performance. If the simulation results with the digital control are not stable or not as desired, one needs to go back to the analog control system, re-design the analog controller, and repeat the process.

With the Backward Euler method, we can also represent the output-input relationship in the time domain as follows:

$$y(n) = y(n\text{-}1) + T_s * u(n)$$

where $y(n)$ and $u(n)$ are the output and input at the current time, $y(n\text{-}1)$ is the output at the previous sampling period, and T_s is the sampling period. Using the equation above, we can replace the discrete integrator in the above circuit with a summer and a unit-delay block, as shown below:



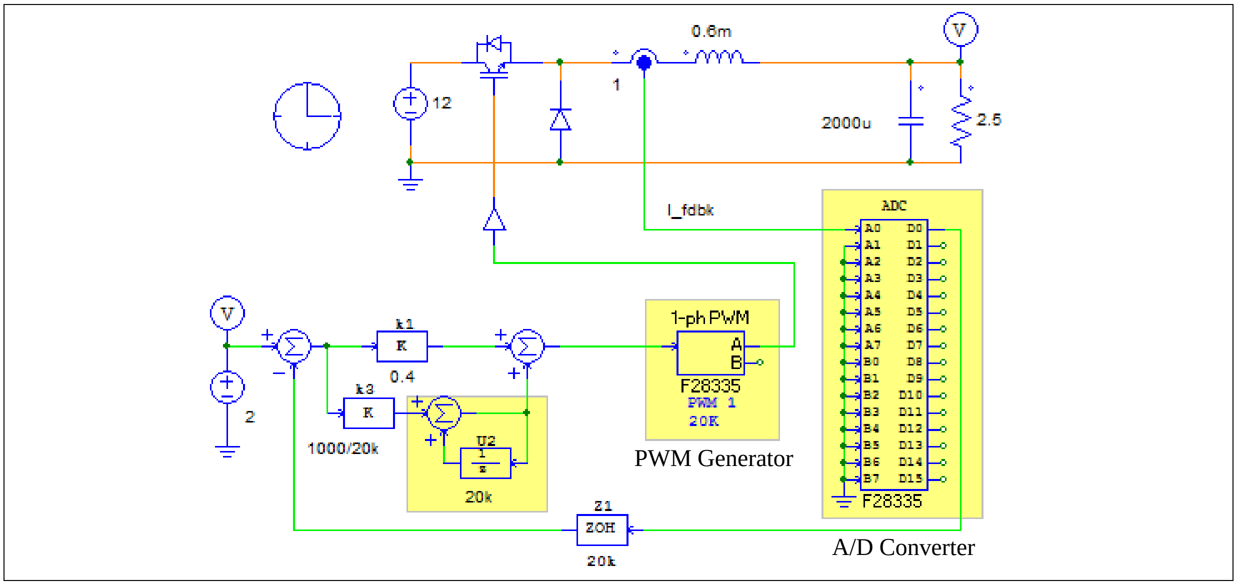
Note that, due to the factor T_s in the equation, the gain of the proportional block k_2 needs to be divided by the sampling frequency of 20kHz. The advantage of this circuit is that it is easier to start or stop the integration of the integrator.

With the control circuit in discrete domain, one is now ready to move on to the next step.

2.4 System Code Generation for Hardware Target

To generate system code for specific hardware target, the circuit schematic must be modified to include relevant hardware elements. Also, variables must be scaled properly to take into account the valid ranges of hardware elements.

Below is the same example circuit but with F2833x Target hardware elements added. For better illustration, the hardware elements are highlighted in yellow.



In the example circuit schematic, the following changes are made:

- An A/D converter is inserted between the current sensor and the control subcircuit. Special attention must be paid to the input range of the A/D converter. If the current sensor output is out of the range of the A/D converter, it must be scaled back accordingly. For this particular example, the A/D converter settings are as shown below.
    - **ADC Mode:** *Start-stop (8-channel)*. This means the A/D conversion will be triggered by PWM generator, and only half of the ADC converter is used.
    - **Ch A0 Mode:** *DC*. This set the input signal range from 0 to +3V
    - **Ch A0 Gain:** *1.0*.
- The comparator and the carrier wave are replaced with the hardware PWM generator. The PWM generator settings relevant to this example are:
    - **PWM Source:** *PWM1*. This defines the PWM module in F28335 processor.
    - **Output Mode:** *Use PWM A*. This defines the PWM output port
    - **Sampling Frequency:** *20k*. This defines the sampling frequency at 20 kHz.
    - **Carrier Wave Type:** *Sawtooth (start high)*. This setting will be explained further in Chapter 5.
    - **Trigger ADC:** *Trigger ADC Group A*. This setting will be explained further in Chapter 5.
    - **ADC Trigger Position:** *0*. This setting will be explained further in Chapter 5.
    - **Peak-to-Peak Value:** *10*. This defines the range of the input signal to this PWM generator.
- The unit delay block U1 is removed. This is because the PWM generator contains one sampling period delay inherently,
- In **Simulation Control**, go to **Parameters** tab, in **SimCoder** section**:**
    - **Hardware Type** is set to *F2833x*, with *RAM Debug*.
    - **CPU Version** is set to *F28335*
    - **Default Fixed-Point Position** is not applicable because it is set to Float
- User can add a comment section to the beginning of the generated code. To create and edit the comments, in **Simulation Control**, go to the **SimCoder** tab, and enter or edit the comments.

To check the validity of the changes after the hardware elements are added, run simulation for this system. The results of this system should be very close to the results of the system with the digital control in Section 2.2.

Once the simulation results are verified, C code can be generated by clicking **Simulate** >> **Generate Code**. The code generated for F2833x hardware is ready to run without any changes.

Below is the C code generated by the SimCoder for the system described above.

```
/*******************************************************************************
// This code is created by SimCoder Version 9.3.3 for TI F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2013
//
// Date: February 24, 2014 14:36:33
*******************************************************************************/
#include<math.h>
#include"PS_bios.h"
typedef float DefaultType;
#defineGetCurTime() PS_GetSysTimer()

interrupt void Task();

DefaultTypefGbliref = 0;
DefaultTypefGblU2 = 0;

interrupt void Task()
{
   DefaultTypefU2, fSUMP1, fSUMP3, fk3, fk1, fSUM1, fZ1, fTI_ADC1, fVDC2;

   PS_EnableIntr();
   fU2 = fGblU2;
```

```
    fTI_ADC1 = PS_GetDcAdc(0);
    fVDC2 = 2;
    fZ1 = fTI_ADC1;
    fSUM1 = fVDC2 - fZ1;
    fk1 = fSUM1 * 0.4;
    fk3 = fSUM1 * (1000.0/20000);
    fSUMP3 = fk3 + fU2;
    fSUMP1 = fk1 + fSUMP3;
    PS_SetPwm1RateSH(fSUMP1);
#ifdef_DEBUG
    fGbliref = fVDC2;
#endif
    fGblU2 = fSUMP3;
    PS_ExitPwm1General();
}

void Initialize(void)
{
    PS_SysInit(30, 10);
    PS_StartStopPwmClock(0);
    PS_InitTimer(0, 0xffffffff);
    PS_InitPwm(1, 0, 20000*1, (4e-6)*1e6, PWM_POSI_ONLY, 42822);// pwnNo, waveType, frequency, deadtime, 
outtype
    PS_SetPwmPeakOffset(1, 10, 0, 1.0/10);
    PS_SetPwmIntrType(1, ePwmIntrAdc0, 1, 0);
    PS_SetPwmVector(1, ePwmIntrAdc0, Task);
    PS_SetPwmTzAct(1, eTZHighImpedance);
    PS_SetPwm1RateSH(0);
    PS_StartPwm(1);

    PS_ResetAdcConvSeq();
    PS_SetAdcConvSeq(eAdc0Intr, 0, 1.0);
    PS_AdcInit(1, !1);

    PS_StartStopPwmClock(1);
}

void main()
{
    Initialize();
    PS_EnableIntr();   // Enable Global interrupt INTM
    PS_EnableDbgm();
    for (;;) {
    }
}
```

The generated code has the following structure:

Interrupt void Task (): The interrupt service routine for 20 kHz. It is called in every 20-kHz cycle.

void Initialize (): The initialization routine. It initializes hardware.

void main (): The main program. It calls the initialization routine, and runs an infinite loop.

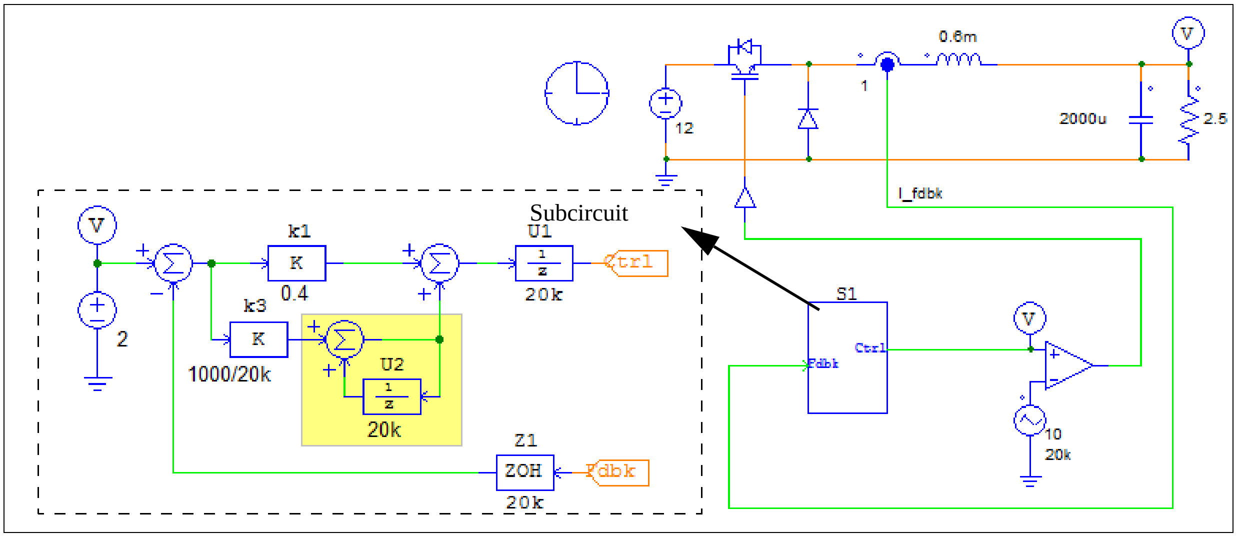
Note that in this example, all the control blocks run at the 20-kHz sampling rate. If there were blocks that run at a different sampling rate, another service routine would be created. One interrupt service routine will correspond to one sampling rate. For blocks that do not have sampling rates associated with them, the corresponding code will be placed in the main program.

This code and all necessary project files are stored in a sub-folder in the same directory as the main schematic file. User can load the project files into TI’s Code Composer Studio environment, compile the code, and upload it onto the DSP hardware for real-time operation.

2.5 Subcircuit Code Generation

In PSIM, a section of the circuit can form a subcircuit. SimCoder can generate code for each subcircuit, given the condition that all the elements in the subcircuit are supported by code generation. There are some other restrictions which are listed in Section 2.5.3

To illustrate the code generation for a subcircuit, we continue to use the example system in Section 2.3. The control circuit within the dotted line is placed inside a subcircuit, as shown below.



To create the subcircuit, select the circuit in the dotted box. Right click the mouse to display the pull-down menu. From the menu, select **Create Subcircuit**, and define the subcircuit file name.

The subcircuit for code generation excluded the comparator and the carrier wave source. One of the reasons for this arrangement is, in most of the hardware setup, these two functions are either implemented by external hardware or embedded in microcontroller’s peripheral interface. The other reason is, for simulation, the carrier wave and the comparator must be calculated at every time step, but the code is executed at the sampling rate of 20kHz. In SimCoder generated code, the sampling rate of every element must be defined. The comparator has two inputs: one is from the controller which has 20 kHz sampling rate, and the other is the carrier wave source which is undefined. In such cases, SimCoder will assume that both inputs of the comparator have the same sampling rate as the input which is defined.

SimCoder can generate code for a subcircuit either for simulation or for hardware target operation. These two types of code are not interchangeable. The subcircuit code generated for simulation can not be used in hardware target, and vice versa. These two situations are explained in the subsections below.

2.5.1 Subcircuit Code Generation for Simulation

To generate code for simulation, follow the steps below:

- While in the main circuit, right click the mouse at the subcircuit block and select **Attributes**.
- At the bottom of the Attributes dialog window, click the button **Generate Code**, and select the option **Generate Code for Simulation**.
- If wanted, user can add a comment section to the beginning of this code. To create and edit the comments, in **Simulation Control**, go to the **SimCoder** tab, and enter or edit the comments in the dialog window before clicking the **Generate Code** button.

The beginning of generated code is shown below.

```
/**********************************************************************
// This code is created by SimCoder Version 9.3.3
//
// SimCoder is copyright by Powersim Inc., 2009-2013
// Date: February 24, 2014 14:55:33
**********************************************************************/
```

```
#include    <stdio.h>
#include    <math.h>
#define    ANALOG_DIGIT_MID    0.5
#define INT_START_SAMPLING_RATE 1999999000L
#define NORM_START_SAMPLING_RATE 2000000000L

typedef void (*TimerIntFunc)(void);
typedef double DefaultType;
DefaultType    *inAry = NULL, *outAry;
DefaultType    *inTmErr = NULL, *outTmErr;

double fCurTime;
double GetCurTime() {return fCurTime;}

/* The input/output node definition for C/DLL block.
    The 2nd display node (outAry[1]): From element 'S1_iref'.
*/
/* The C block for the generated code has the following additional output port(s):
    2 - S1.iref
*/
void _SetVP6(int bRoutine, DefaultType fVal);
void InitInOutArray()
{... ...}

void FreeInOutArray()
{ ... ...}

void CopyInArray(double* in)
{ ... ...}

void CopyOutArray(double* out)
{ ... ...}

void Task();
void TaskS1(DefaultType fIn0, DefaultType *fOut0);

DefaultType    fGblS1_U1 = 0;
DefaultType    fGblS1_U2 = 0;

void TaskS1(DefaultType fIn0, DefaultType *fOut0)
{ ... ...}

void Task()
{
    TaskS1(inAry[0],&outAry[0]);
}

typedef struct {
    TimerIntFunc    func;
    long            samprate;
    double            tmLastIntr;
}    TimeChk;
#define NUM_TIMER_INTR    1
TimeChk    lGbl_TimeOverChk[NUM_TIMER_INTR] = {
    {Task, 20000, 0}};

void InitAllTaskPtr(void)
{
    lGbl_TimeOverChk[0].func = Task;
    lGbl_TimeOverChk[0].samprate = 20000;
}

void _SetVP6(int bRoutine, DefaultType fVal)
{
    static    DefaultType    val = 0.0;
    if (bRoutine) {
        val = fVal;
    }
    outAry[1] = val;
}
```

At the end of the generated code from the subcircuit, it contains *SimulationStep* function, *SimulationBegin* function, and *SimulationEnd* function at the end, as shown below. These functions can be used in C Block which can replace the subcircuit.

```
void SimulationStep(
            double t, double delt, double *in, double *out,
             int *pnError, char * szErrorMsg,
             void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{ ... ...}

void SimulationBegin(
            const char *szId, int nInputCount, int nOutputCount,
             int nParameterCount, const char ** pszParameters,
             int *pnError, char * szErrorMsg,
             void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
    InitInOutArray();
}

void SimulationEnd(const char *szId, void ** reserved_UserData, int reserved_ThreadIndex, void *
reserved_AppPtr)
{
    FreeInOutArray();
}
```

Replacing Subcircuit with C Block

In the subcircuit attribute dialog, there is a check box called **Replace subcircuit with generated code for simulation**. When this box is checked, the simulation will be done as if the system contains a C Block instead of a subcircuit. User do not need to go through the procedure to replace the subcircuit with a C block.

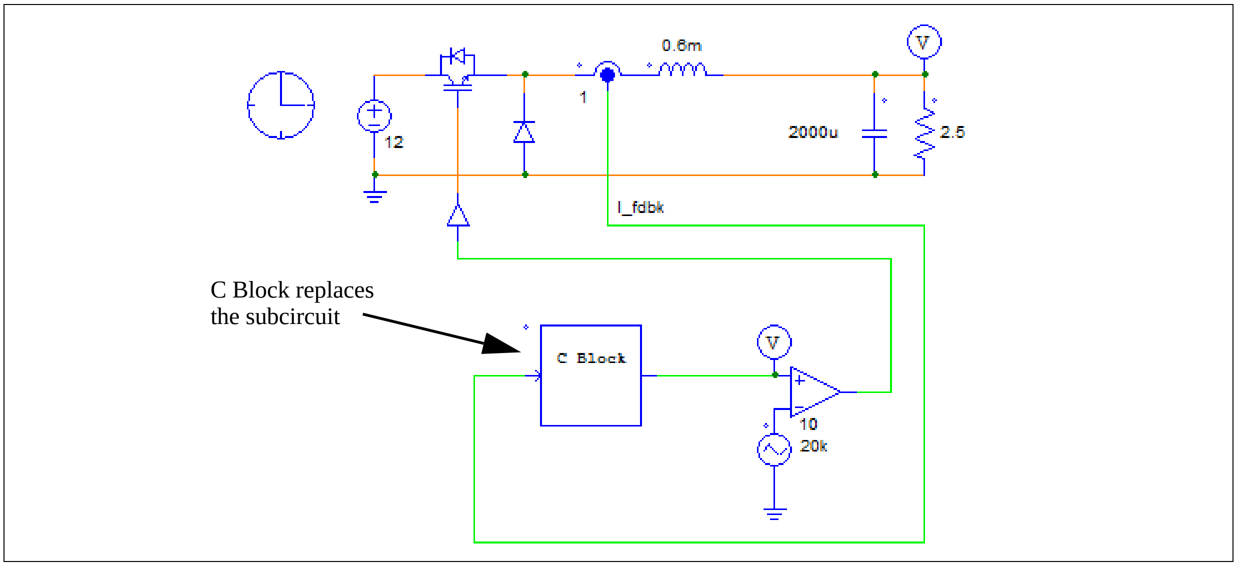
However, if the user prefer to modify the C-code at will, one can place the generated code in a C Block by following the steps below.

In the above example, the code generated from the subcircuit contains four sections: *SimulationStep*, *SimulationBegin*, *SimulationEnd* and the rest of the code. Similarly, the C Block is also composed of these four sections. A C Block contains the following sections:

```
#include <Stdlib.h>
#include <String.h>
#include <math.h>
#include <Psim.h>

// PLACE GLOBAL VARIABLES OR USER FUNCTIONS HERE...
... ...

/////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationStep
{
// ENTER YOUR CODE HERE...

}

/////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationBegin
{
// ENTER INITIALIZATION CODE HERE...

}

/////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationEnd
{

}
```

To create a C Block in the main circuit, use the pull-down menu **Elements** >> **Other** >> **Function Blocks** >> **C Block**. Then, copy the each section of the generated code to each section of the C Block: from the *SimulationStep* function to *SimulationStep* section in the C Block, from the *SimulationBegin* function to *SimulationBegin* section, from the *SimulationEnd* function to *SimulationEnd* section, and put the rest of the code to the *User Functions* section.

The example circuit with the subcircuit replaced with a C Block is shown below.



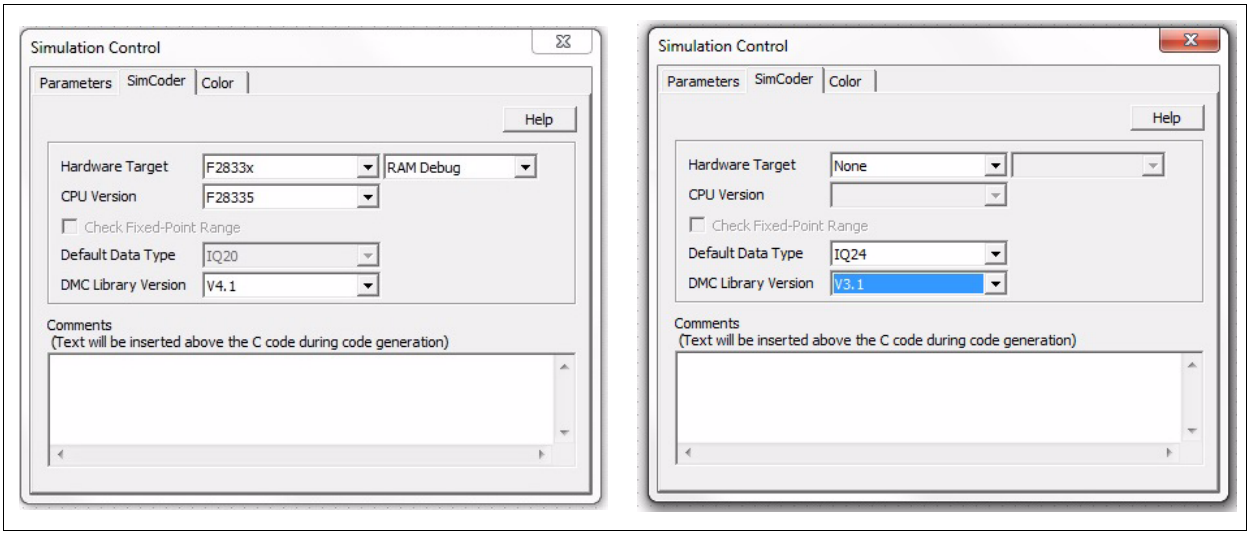
C Block replaces the subcircuit

2.5.2 Subcircuit Code Generation for Hardware Target

The same example circuit in Section 2.5.1 can also be used to generate code for targeted hardware.

One sets the desired hardware target type in the Simulation Control dialog’s SimCoder tab. As shown below on the left, the target is selected for *F2833x*, with CPU version of *F28335* for *RAM Debug* build.

One can select *None* as the hardware target but can still generate targeted code for a specific default data type. The example below on the right set the data type as Fixed Point with the position at IQ24.



Hardware Target: F2833x, RAM Debug
CPU Version: F28335
□ Check Fixed-Point Range
Default Data Type: IQ20
DMC Library Version: V4.1
Comments
(Text will be inserted above the C code during code generation)

Hardware Target: None
CPU Version:
□ Check Fixed-Point Range
Default Data Type: IQ24
DMC Library Version: V3.1
Comments
(Text will be inserted above the C code during code generation)

Go to the subcircuit’s attributes dialog. Click on the **Generate Code** button and select **Generate Code for Hardware Target**.

Below is the C code generated by the SimCoder for the subcircuit for F28335 CPU. Unlike the generated code for the whole system, the code for a subcircuit does not have the main program and the initialization routine. It can be inserted into one’s own code for F28335 target hardware implementation.

This subcircuit has only one sampling rate. As a result, the generated code has only one function *TaskS1*. The variables *fIn0* refers to the subcircuit input *I_fdbk*, and the variable *fOut0* corresponds to the subcircuit output *Ctrl*.

```
#defineNULL (0)
#ifndef DSP28_DATA_TYPES
#define DSP28_DATA_TYPES
typedef int                int16;
typedef long               int32;
typedef long long          int64;
typedef unsigned int       Uint16;
typedef unsigned long      Uint32;
typedef unsigned long long Uint64;
typedef float              float32;
typedef long double        float64;
#endif
DefaultTypefGblS1_U1 = 0;
DefaultTypefGblS1_iref = 0;

// Parameter list for S1
void TaskS1(DefaultType fIn0, DefaultType *fOut0)
{
   DefaultTypefS1_U1, fS1_SUMP1, fS1_B4, fS1_k2, fS1_k1, fS1_SUM1, fS1_Z1;
   DefaultTypefS1_VDC2;

   *fOut0 = fGblS1_U1;

   fS1_VDC2 = 2;
   fS1_Z1 = fIn0;
   fS1_SUM1 = fS1_VDC2 - fS1_Z1;
   fS1_k1 = fS1_SUM1 * 0.4;
   fS1_k2 = fS1_SUM1 * 1000;
   {
            static DefaultType out_A = 0;
            fS1_B4 = out_A + 1.0/20000 * (fS1_k2);
            out_A = fS1_B4;
   }
   fS1_SUMP1 = fS1_k1 + fS1_B4;
   fGblS1_U1 = fS1_SUMP1;
#ifdef_DEBUG
   fGblS1_iref = fS1_VDC2;
#endif
}
```

2.5.3 Restrictions for Subcircuit Code Generation

There are a few restrictions for building the subcircuit for code generation using SimCoder. These restrictions are listed below:

- All the elements in the sub-system must be supported for code generation. To find out if an element can be used for code generation, in PSIM, go to **Options** >> **Settings**, and check the box **Show image next to elements that can be used for code generation**. Any elements that have an image next to the elements in the PSIM Elements library can be used for code generation.
- Only uni-directional subcircuit ports can be used. That is, input signal ports must be used for subcircuit inputs, and output signal ports must be used for subcircuit outputs. Bi-directional ports are not allowed.
- No hardware input/output elements (such as A/D converter, digital input/output, encoder, counter, and PWM generator) as well as hardware interrupt elements are placed inside the subcircuit. They must be in the top-level main circuit only.
- If the input of a sub-system has a sampling rate, and the rate can not be derived from the circuit inside the sub-system, a zero-order-hold block must be connected at the input to explicitly define its sampling rate. If the zero-order-hold block is not used in this case, this input (and subsequent blocks that connect to this input) will be treated with no sampling rate.

- If the input of a subcircuit does not have a zero-order-hold block connected to it, SimCoder will derive its sampling rate from the blocks that connect to it in the sub-system. However, to avoid ambiguity, it is strongly suggested to place a zero-order-hold block at each input to explicitly define its sampling rate.

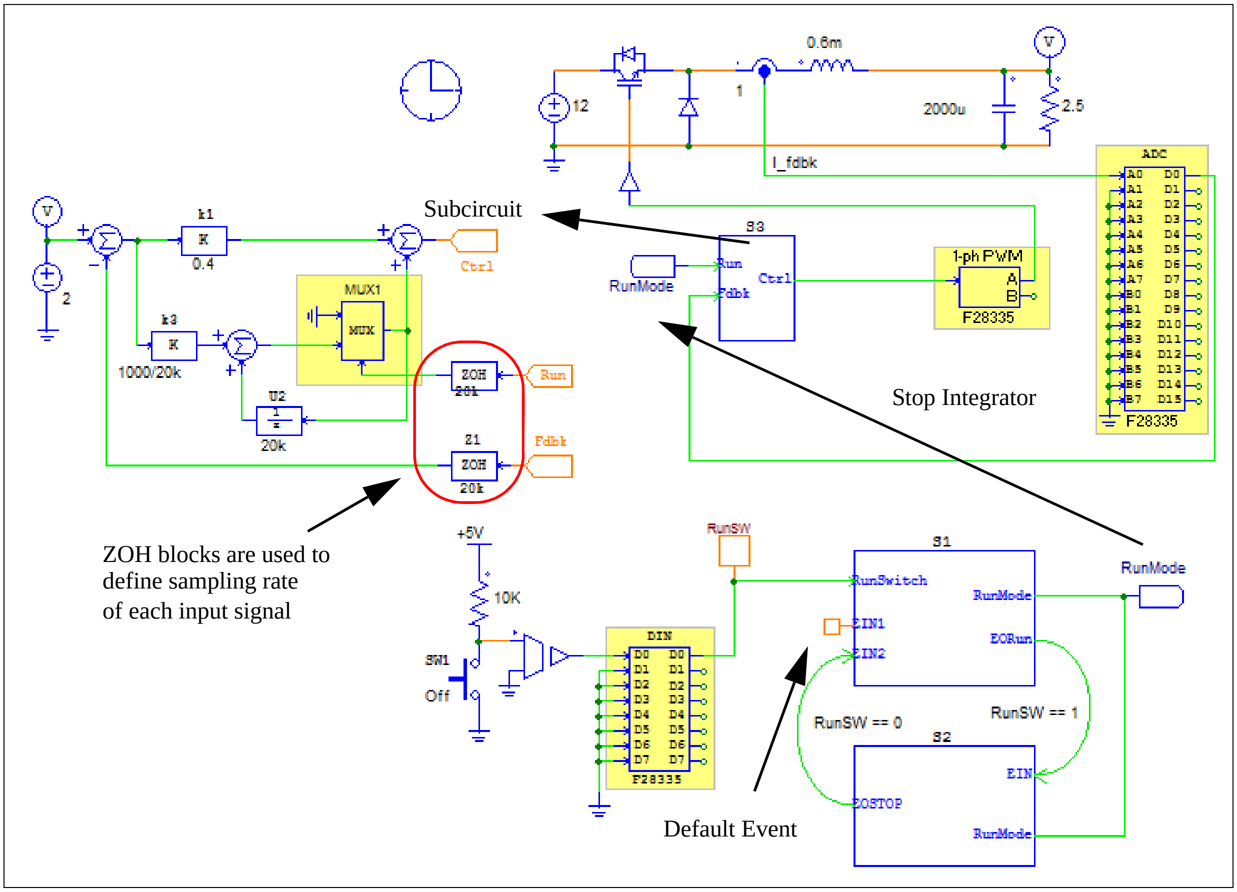
2.6 System with Event Control

Often a system may include event transitions. The system will transit from one state to another state when certain condition is met. SimCoder handles the event control through subcircuits. More detailed description of the event control can be found in Chapter 3.

To illustrate how event control works, the following considerations are added to the example in Section :

- A manual switch is employed to control the start/stop of the system. As a result, the system will have two operation modes: Stop Mode and Run Mode. The system will transit from one mode to another whenever the switch position changes.
- In Stop Mode, the integrator output is reset to 0 to prevent saturation.

The example system with event control is shown below.



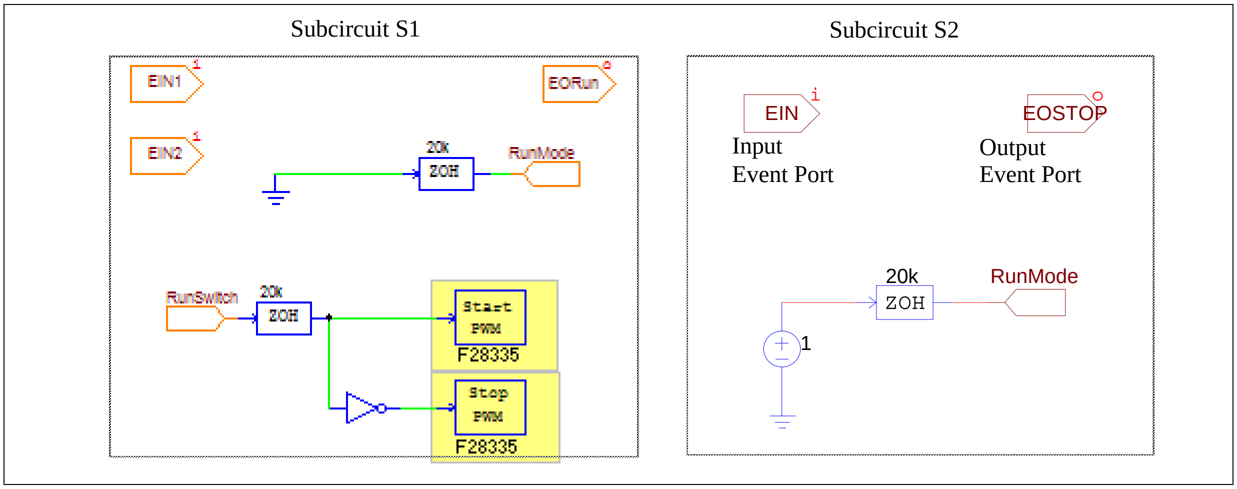
Subcircuit

Stop Integrator

ZOH blocks are used to define sampling rate of each input signal

Default Event

In the diagram, Blocks S1 and S2 are subcircuits, and the contents of the subcircuits are shown below.



Comparing with the circuit Section , the following changes are made:

- Two event control subcircuits are added to implement the two operation modes: Stop Mode (represented by Subcircuit S1) and Run Mode (represented by Subcircuit S2).
- The default mode of operation is the Stop Mode. This is defined by connecting the **Default Event** element to Port *EIN1* of the subcircuit S1.
- Subcircuit S1 has two input event ports *EIN1* and *EIN2*, one output event port *EORun*, one input signal port *RunSwitch*, and one output signal port *RunMode*. Subcircuit S2 has one input event port *EIN*, one output event port *EOSTOP*, and one signal port *RunMode*.
- Conditions are defined for the transition from the Stop Mode to the Run Mode, and vice versa. The variable *RunSW* used in the conditions is a global variable (refer to Section 5.2 for more details), and is defined by the global variable element connected to the output pin D8 of the digital input element.
- The hardware digital input element is used to measure the position of the push-button switch SW1. When the switch is off, the digital input voltage is high (1), so is the global variable *RunSW*, and the system is in the Run Mode. When RunSW is low (0), the system is in the Stop Mode.
- Multiplexer MUX1 is added to prevent the integrator from integrating in the Stop Mode. When the system is not running, the signal *RunMode* will be 0 and the integrator will not integrate. When the signal *RunMode* is 1, the integrator will start to work.

Below is how this system works:

- The position of the manual switch is read through the hardware digital input. This signal is sent to Subcircuit S1 (Stop Mode) through the input signal port *RunSwitch*. The same signal is also designated as the global variable *RunSw*.
- Initially the system is in the Stop Mode. When the condition "RunSW == 1" (or *RunSW* is equal to 1) is met, the system will transit from the Stop Mode to the Run Mode. This is defined by the connection of the output event port *EORun* of S1 to the input event port *EIN* of S2.
- While in the Run Mode, if the condition "RunSW == 0" (or if *RunSw* is equal to 0) is met, the system will transit from the Run Mode to the Stop Mode. This is defined by the connection of the output event port *E OSTOP* of S2 to the input event port *EIN2* of S1.
- In the Stop Mode subcircuit, the *RunMode* signal will be set to 0. As long as the *RunSwitch* signal is 0, the hardware PWM generator will be stopped. But when the *RunSwitch* is changed to 1, it will start PWM, and at the same time switch out of the Stop Mode into the Run Mode.
- In the Run Mode subcircuit, the *RunMode* signal will be set to 1 in order to enable the integrator operation.

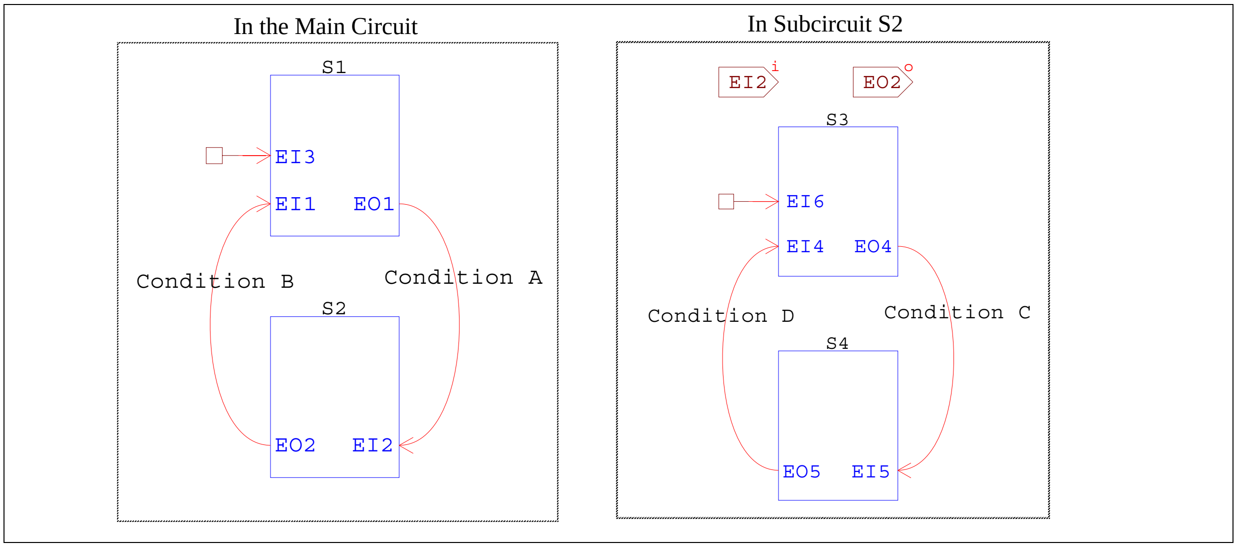
After the system is modified, user may run the simulation to verify that the changes are correct before generating system code for target hardware operation.

3

Event Handling

3.1 Basic Concept

Event is used to describe the transition of a system from one operation state to another. For example, the figure below shows several operation states and how the transition occurs.



In the main circuit, there are two states: S1 and S2, both in the form of subcircuits. The schematic of each state is included in a subcircuit. State S1 has two input event ports: *EI1* and *EI3*, and one output event port *EO1*. State S2 has one input event port *EI2* and one output event port *EO2*. By default, State S1 is the default state at the beginning. This is defined by the connection of the default event element to the input event port *EI3*.

The output event port *EO1* of S1 is connected to the input event port *EI2* of S2, with the transition Condition A. This means that when Condition A is met, the system will transit from State S1 to S2. Similarly, the output event port *EO2* of S2 is connected to the input event port *EI1*. When Condition B is met, the system will transit from State S2 to S1.

When two or more states can not co-exist and only one state can exist at any time, such as S1 and S2 in this case, we refer to these states as exclusive states.

The system on the right shows the content inside Subcircuit S2. It in turn has two states, S3 and S4. When the system transits to State S2, it will start with State S3 by default. If Condition C is met, it will transit from State S3 to S4. If Condition D is met, it will go back to State S3.

There is no limit on the number of states that a system can contain.

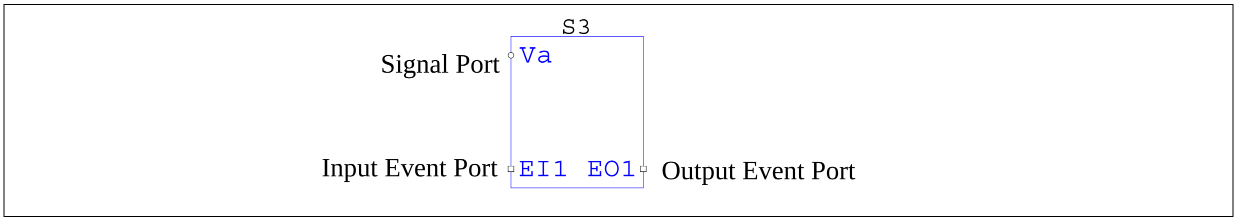
3.2 Elements for Event Handling

The following elements are used to define events and the state transition:

- Input event port
- Output event port
- Default event element
- Flag for event block first entry
- Hardware interrupt element (see Section 5.4)
- Global variable

Placing an input event port inside a subcircuit will create an event that allows the transition into the subcircuit. Similarly, placing an output event port inside a subcircuit will create an event that allows the transition out of the subcircuit.

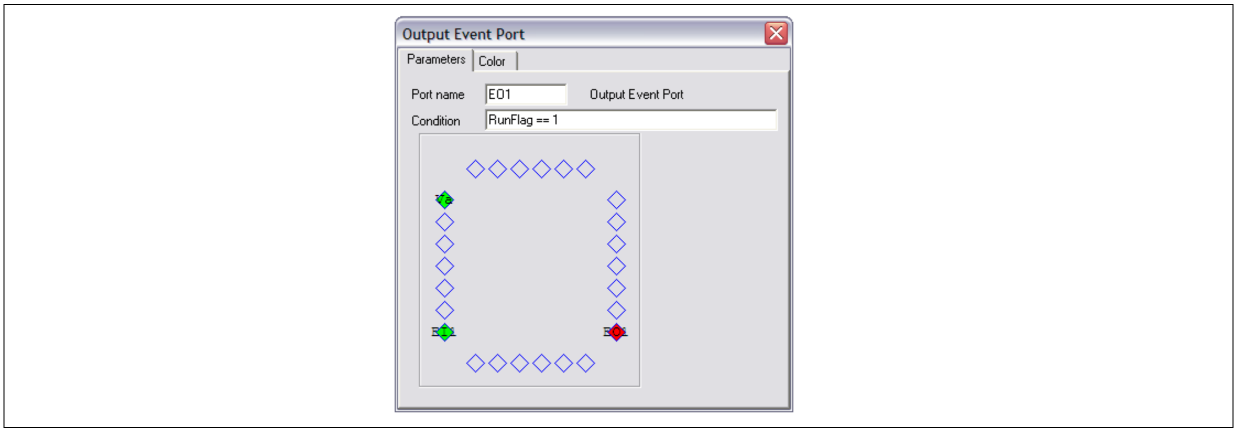
For example, the figure below shows the image of a subcircuit after an input event port and an output event port are placed inside the subcircuit.



The image of an event port is a square, which is different from the image of a signal port which is a circle.

The connection to an input event port can only come from an output event port or a hardware interrupt element, using the event connection wiring function. Input/output event ports and hardware interrupt elements can not be connected to other types of nodes.

For each output event port, a condition must be defined. The property window of the output event port *EO1* in Subcircuit S3 above, for example, is shown below:



Port name: EO1 — Output Event Port

Condition: RunFlag == 1

The condition "RunFlag == 1" is the condition that will trigger the output event to occur. The condition statement must be a valid C code expression. For example, the condition statement can be:

(RunFlag == 1) && (FlagA >= 250.) || (FlagB < Vconst)

Note that only global variables, numerical values, and parameter constants defined in parameter files or passed from the main circuit into subcircuits can be used in the condition expression. In the above expression, *RunFlag*, *FlagA*, and *FlagB* can be global variables, and *Vconst* can be a constant defined in a parameter file or passed into the subcircuit from the main circuit.

To create a global variable, connect the global variable element to a node.

3.3 Restrictions on Subcircuits with Events

A subcircuit that contains input or output event ports is considered to be a subcircuit with events. Also, everything inside a subcircuit with events will inherit the event property. That is, if Subcircuit A is within Subcircuit B, and Subcircuit B is a subcircuit with events, even if Subcircuit A does not have any input/output event ports, it is still considered as a subcircuit with events.

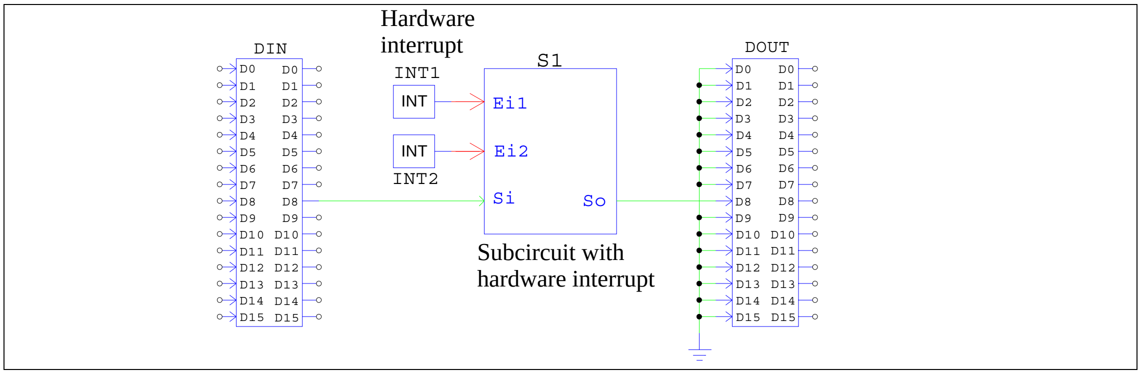
As subcircuits are used to handle events, there are now three types of subcircuits in PSIM:

- *Regular subcircuits*: This type of subcircuit does not contain any event ports and is the same as conventional subcircuits before.
- *Subcircuit with events*: This type of subcircuit contains input/output events ports, but there are no hardware interrupt elements connected to the input event ports.

- *Subcircuit with hardware interrupt*: This type of subcircuit contains input event ports only, and only hardware interrupt elements are connected to the input event ports. There is no output event port inside the subcircuit, and no output event ports are connected to the input event ports. This is a special case of the subcircuit with events as this type of subcircuit is dedicated to handle hardware interrupt only.

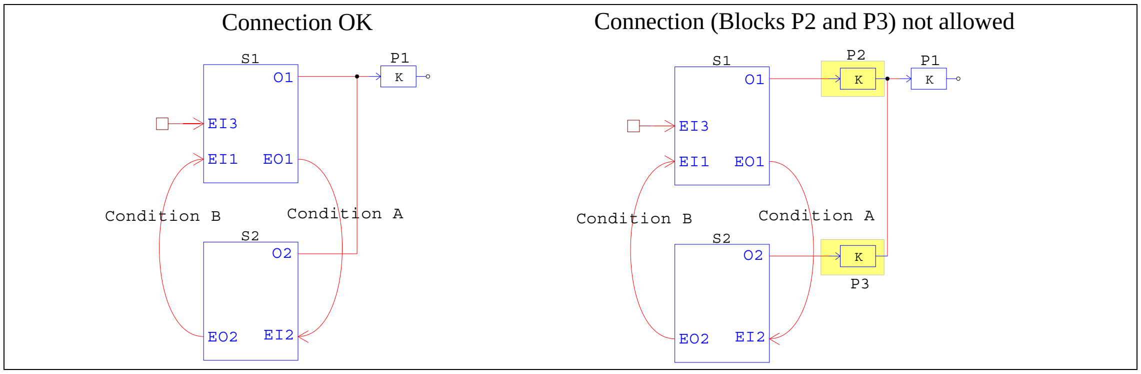
Since a subcircuit with events or with hardware interrupt is involved in the code generation only, the following restrictions are imposed on these two types of subcircuits:

- All the elements inside a subcircuit with events or with hardware interrupt must be supported by code generation. For example, such a subcircuit can not contain a resistor or a rms block, both not supported by the code generation.
- A subcircuit with hardware interrupt can have multiple input event ports, but can not have any output event ports. Also, only hardware interrupt elements can be connected to the input event ports. In addition, the signal input/output ports of the subcircuit can be connected to hardware elements only, not to other function blocks. The figure below shows how a subcircuit with hardware interrupt can be connected:


Hardware interrupt
DIN
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
INT1
INT
INT
INT2
S1
Ei1
Ei2
Si
So
Subcircuit with hardware interrupt
DOUT
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15

The subcircuit S1 is a subcircuit with hardware interrupt. It has two hardware interrupt elements connected to it, INT1 and INT2. It has one signal input port S_i connected to the hardware digital input, and one signal output port S_o connected to the hardware digital output.

- If a subcircuit with hardware interrupt contains z-domain blocks with sampling rates, these sampling rates will be ignored as the subcircuit will be called only when a hardware interrupt occurs. For example, if the subcircuit contains a discrete integrator, the sampling rate of the discrete integrator will be ignored. In the calculation for the integrator, the previous time will be the last time that a hardware device triggers an interrupt.
- If the signal outputs of two subcircuits are connected, they should be connected directly, not through other elements. To illustrate this, consider the following circuits:



In the circuit on the left, both subcircuits S1 and S2 have one output signal port, O1 and O2. They are connected externally together to the input of the proportional block P1. The way the circuit works is that the input of the block P1 will come from either Port O1 or O2, depending on which state is active. This connection is allowed.

However, in the circuit on the right, Port O1 is connected to Block P2, and Port O2 is connected to P3, and the outputs of P2 and P3 are then connected together. Such a connection is not allowed. In this case, Block P2 should be moved into the subcircuit S1, and Block P2 moved into the subcircuit S2.

4

SimCoder Libraries

4.1 Overview

SimCoder can be used with or without a hardware target. When it is used without a hardware target, it will convert a control schematic into C code. While the code can be simulated in PSIM, it is not for a specific hardware. On the other hand, with a hardware target, SimCoder can generate code that is ready to run on the specific hardware, or can be adopted for a specific hardware.

SimCoder element libraries include two types of elements: these that are not associated with any hardware targets or are shared by all hardware targets, and these that are specific to a particular hardware.

SimCoder elements that are independent of any hardware include the following:

- Some of the elements of the standard PSIM library.
- All the elements under **Elements** >> **Event Control**.
- The *Global Variable* element under **Elements** >> **SimCoder**.

Simcoder elements that are shared by all hardware targets include the following:

- The *Interrupt* element under **Elements** >> **SimCoder**.
- The *TI DMC* element under **Elements** >> **SimCoder >> TI DMC Library**.

SimCoder elements that are hardware-specific include the following:

- F2833x Target: All the elements under **Elements** >> **SimCoder** >> **F2833x Target**.
- F2803x Target: All the elements under **Elements** >> **SimCoder** >> **F2803x Target**.
- PE-Pro/F28335 Target: All the elements under **Elements** >> **SimCoder** >> **PE-Pro/F28335 Target**.
- PE-Expert3 Target: All the elements under **Elements** >> **SimCoder** >> **PE-Expert3 Target**.

The SimCoder elements that are independent or common to all hardware targets are described in this Chapter. The elements for each specific hardware target are described in Chapter 5 to 8. The elements in the TI DMC Library are described in Chapter 9.

4.2 Elements from Standard PSIM Library

Many elements in the standard PSIM library can be used for code generation. Under **Options** >> **Settings** >> **Advanced**, check the option box "*Show image next to elements that can be used for code generation*", a small image CG will appear next to those elements that can be used for code generation.

Please note that, for all the math function blocks and the Simplified C Block, variables t (for time) and delta (for time step) can not be used in SimCoder for code generation.

Also, the parameter file element and the sawtooth voltage source element have special usage in SimCoder, as described in the sections below.

4.2.1 Defining Global Parameters in Parameter File

The parameter file element can be used in the same way as before. In SimCoder, it can also be used to define global parameters.

In order to make generated code more readable and manageable, sometimes it is better to use parameter names instead of the actual numerical values in the code. For example, if the gain of a controller is 1.23, rather than using the number 1.23 in the code, we can define the parameter Kp = 1.23, and use the parameter name Kp in the code instead.

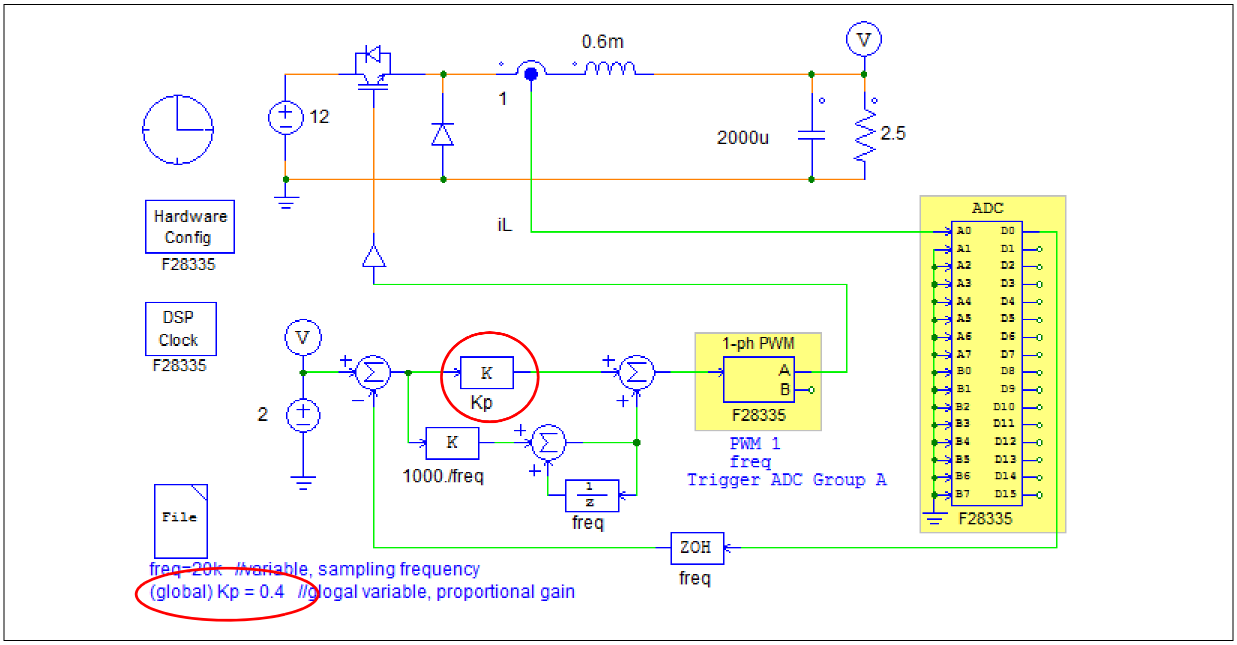
This type of parameters is global to the code, and can be used anywhere in the code. To define a parameter as a global parameter, in the content of a parameter file, use the "(global)" definition before the parameter name.

As shown below in the buck converter example circuit, the gain of the proportional controller is defined as Kp. In the parameter file, if the DSP target is floating-point F28335, the parameter Kp is defined as:

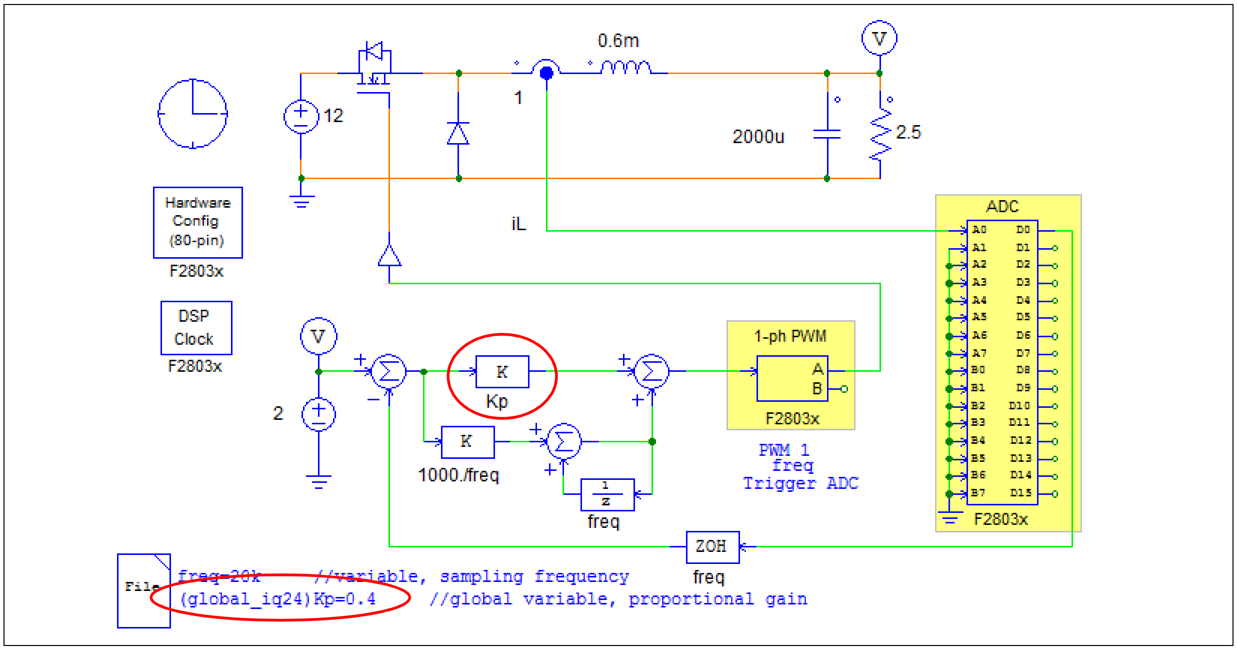
(global) Kp = 0.4

If the DSP target is F28035 with fixed-point at IQ-24, the parameter Kp is defined as:

(global_Iq24) = 0.4

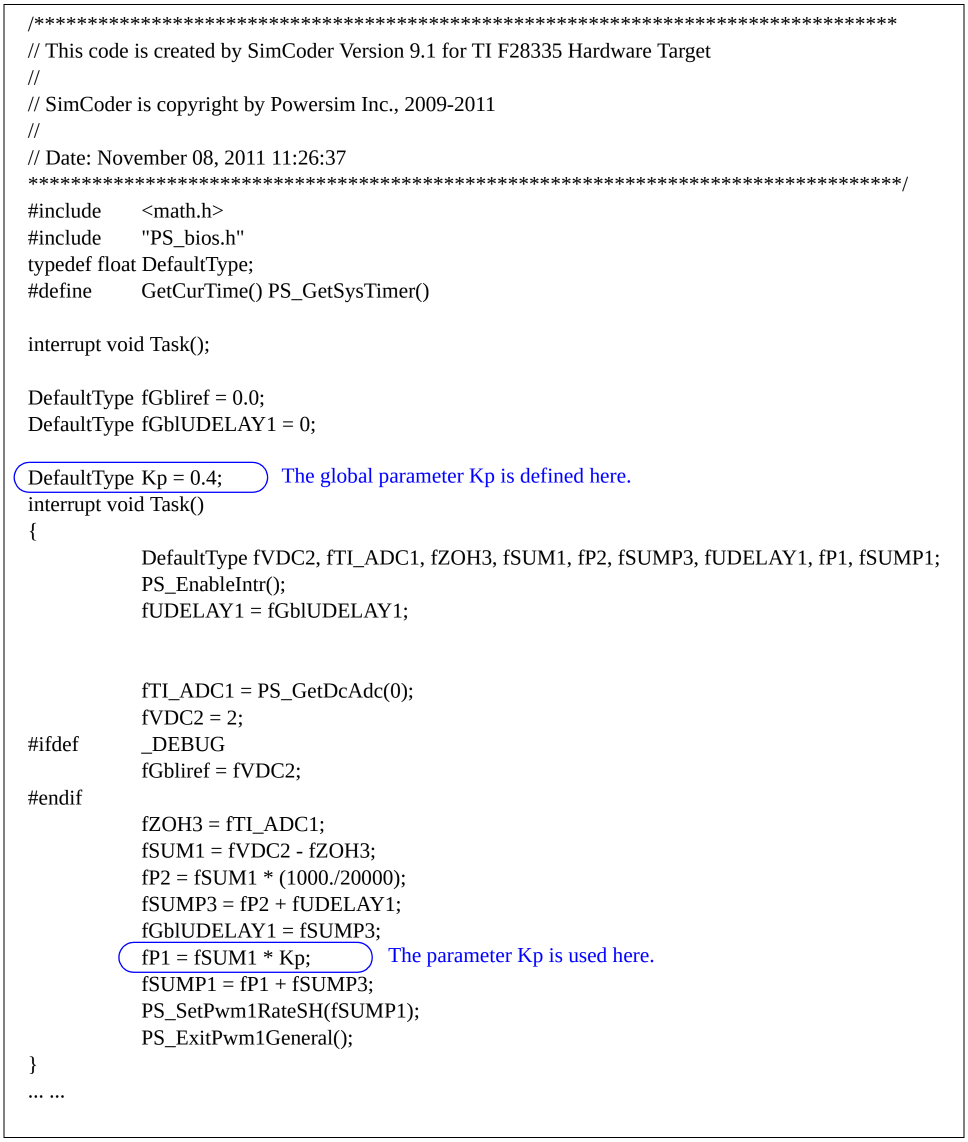


0.6m
V
12
1
2000u
2.5
Hardware Config
F28335
iL
DSP Clock
F28335
ADC
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
F28335
V
2
K
Kp
K
1000./freq
1/z
freq
1-ph PWM
A
B
F28335
PWM 1
freq
Trigger ADC Group A
ZOH
freq
File
freq=20k //variable, sampling frequency
(global) Kp = 0.4 //glogal variable, proportional gain



0.6m
V
12
1
2000u
2.5
Hardware Config (80-pin)
F2803x
iL
DSP Clock
F2803x
ADC
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
F2803x
V
2
K
Kp
K
1000./freq
1/z
freq
1-ph PWM
A
B
F2803x
PWM 1
freq
Trigger ADC
ZOH
freq
File
freq=20k //variable, sampling frequency
(global_iq24)Kp=0.4 //global variable, proportional gain

The generated code for F28335 is shown below. Note that in the code, the parameter Kp is defined as 0.4 at the beginning, and the parameter name Kp is used in the calculation.



```
/*****************************************************************************
// This code is created by SimCoder Version 9.1 for TI F28335 Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2011
//
// Date: November 08, 2011 11:26:37
******************************************************************************/
#include        <math.h>
#include        "PS_bios.h"
typedef float DefaultType;
#define         GetCurTime() PS_GetSysTimer()

interrupt void Task();

DefaultType fGbliref = 0.0;
DefaultType fGblUDELAY1 = 0;

DefaultType Kp = 0.4;
interrupt void Task()
{
                DefaultType fVDC2, fTI_ADC1, fZOH3, fSUM1, fP2, fSUMP3, fUDELAY1, fP1, fSUMP1;
                PS_EnableIntr();
                fUDELAY1 = fGblUDELAY1;


                fTI_ADC1 = PS_GetDcAdc(0);
                fVDC2 = 2;
#ifdef          _DEBUG
                fGbliref = fVDC2;
#endif
                fZOH3 = fTI_ADC1;
                fSUM1 = fVDC2 - fZOH3;
                fP2 = fSUM1 * (1000./20000);
                fSUMP3 = fP2 + fUDELAY1;
                fGblUDELAY1 = fSUMP3;
                fP1 = fSUM1 * Kp;
                fSUMP1 = fP1 + fSUMP3;
                PS_SetPwm1RateSH(fSUMP1);
                PS_ExitPwm1General();
}
... ...
```

The global parameter Kp is defined here.

The parameter Kp is used here.

4.2.2 Generating Sawtooth Waveform

A sawtooth waveform can be used in hardware as the system timer, or to generate other periodic waveforms (such as sinusoidal waveform). The sawtooth voltage source under **Elements** >> **Sources** >> **Voltage** is implemented using an actual counter in the hardware.

It assumes that there exists a 32-bit counter in the hardware, incrementing in every 20 ns, to generate the sawtooth waveform.

For the PE-Expert3 Hardware Target, it used the 32-bit free-run counter on the PEV Board, incrementing in every 20 ns, to generate the sawtooth waveform.

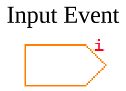
4.3 Event Control Elements

The following elements are used to implement event control.

4.3.1 Input Event

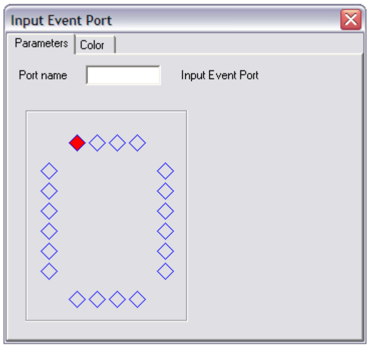
The image of an input event element is shown below.

Image:



The letter "i" in the image refers to "input".

The input event element is a type of subcircuit interface port. It should be used inside a subcircuit only. After double clicking on the element, one will define the port name and location, as shown below:



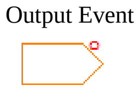
Port name Input Event Port

In the main circuit that calls this subcircuit, if there is an event connection wire connecting to this port, when the condition of the event connection is met, the system will transit to this subcircuit through this input event port.

4.3.2 Output Event

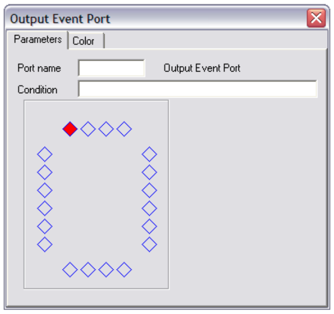
The image of an output event element is shown below.

Image:



The letter "o" in the image refers to "output".

The output event element is also a type of subcircuit interface port. It should be used inside a subcircuit only. After double clicking on the element, one will define the port name, location, as well as a condition, as shown below:



Port name Output Event Port

Condition

When the condition defined in the output event port is met, the system will transit out of this subcircuit into another subcircuit.

The condition statement must use format and operators supported by the C language. For example, the statements below are acceptable condition statements:

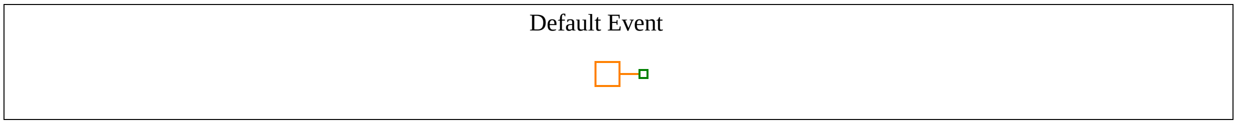
A == 1
A >= B
(A > B) && (C > D)

where A, B, C, and D are global variables or numerical constants.

4.3.3 Default Event

The image of a default event element is shown below.

Image:



When there are several exclusive states, the default event element is used to define which state is the default state. It is connected to the input event port of a subcircuit outside the subcircuit.

4.3.4 Event Connection

The event connection is a SimCoder element which connects an output event port or a hardware interrupt element to an input event port. Note that it should not be confused with the regular wiring tool to connect other PSIM elements. The event connection can be used for event connection only.

Event connection element can be found at **Elements >> Event Control >> Event Connection**.

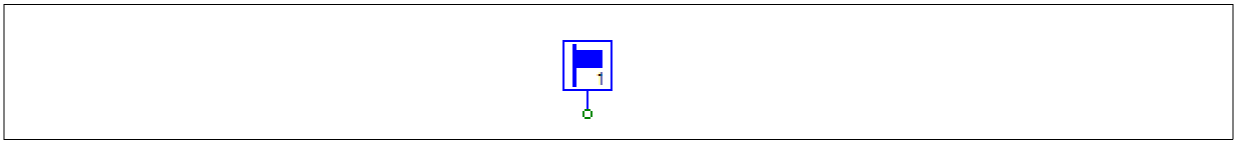
When double clicking on the event connection wire, one can edit the condition statement of the output event port that the event wire connects to.

Besides the starting point and the ending point, an event connection wire has two points in between. By modifying the locations of these two points, the shape of the connection wire can be changed. To modify these two points, highlight the event connection wire. Right click and choose "Modify handle 1" or "Modify handle 2".

4.3.5 Flag for Event Block First Entry

Sometimes certain actions need to be performed when the program execution enters an event subcircuit block for the first time. To identify this, a flag for event block first entry is provided.

Image:



Attribute:

Parameters	Description
Event Subcircuit Block Name	The name of the event subcircuit block that the flag is for.

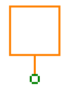
The flag node is an output node. When the event subcircuit block is entered for the first time, the node value will be 1. Otherwise, it will be 0. For example, to find out when the event subcircuit block S1 is entered the first time, set *Event Subcircuit Block Name* to S1.

4.4 Global Variable

A global variable is used in conditional statements and in special occasions.

Image:

Global Variable



Attributes:

Parameters	Description
Name	The name of the global variable name
Initial Value	The initial value of the global variable

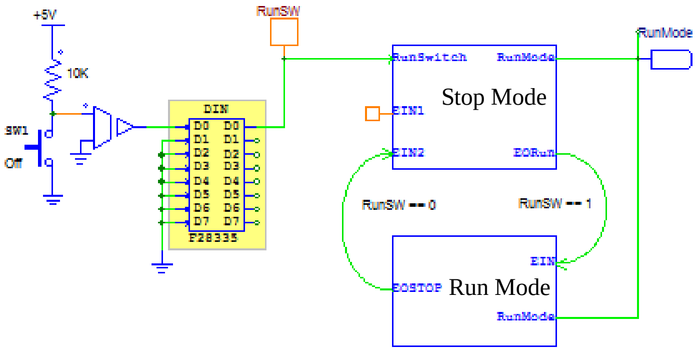
To define a signal as a global variable, connect the global variable element to the particular node. Note that only a signal in the control circuit for the code generation can be defined as a global variable.

As the name suggests, a global variable can be accessed globally. When the initial value of a global variable is changed, the initial values of all the global variables in that circuit, including subcircuits, are changed at the same time.

A global variable can be a signal sink or a signal source. When it is a signal sink, it reads the signal value from the node. When it is a signal source, it sets the value of the node.

One use of the global variables is in the event condition statements. All variables in the condition statements must be global variables. An example is shown below.

Global Variable



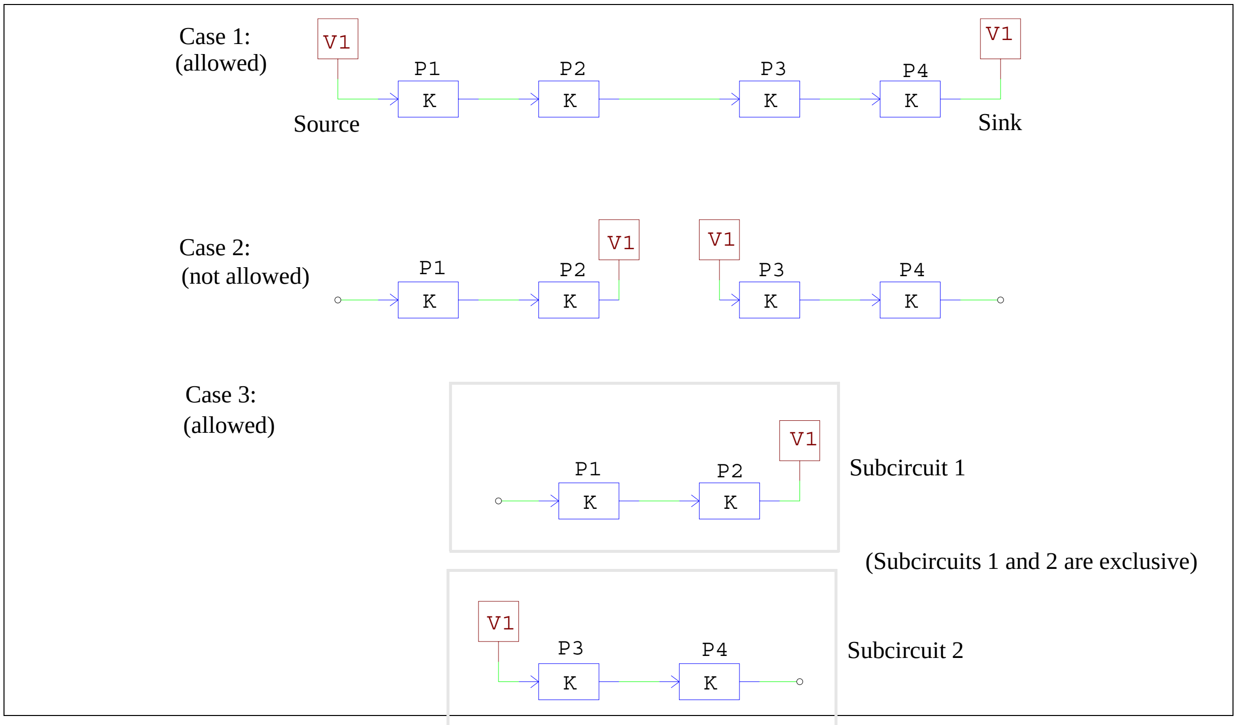
In this example, a global variable, *RunSW*, is connected to the output pin D0 of the digital input. This global variable is then used in the conditional statements between the transition of the two modes of operation.

Another use of the global variable is to use it as a signal source. For example, a global variable can be used as a signal source and passes the value to another block.

Note that global variable should not be used as a label to pass a value from one node to another, when two nodes can be physically connected by a wire. The use of the global variables has the following restrictions:

- Global variables of the same name can be used multiple times only if they are in the same signal flow path.
- If they are in different signal flow paths, global variables of the same name are not allowed, unless they are in different exclusive states (exclusive states are states that can not occur at the same time).

To illustrate this, the diagram below shows situations where global variables can and cannot be used.



In Case 1, a global variable V1 is first used as a source and it assigns the value to the input of the block P1. After a series of calculation, the output of the block P4 is assigned back to the same global variable V1. Since both global variables are in the same signal flow path, it is allowed.

In Case 2, however, the global variable V1 is used as a label to pass values from the output of the block P2 to the input of the block P3. This is not allowed. To pass the value from one node to another, labels should be used instead, or one should connect these two nodes with a wire.

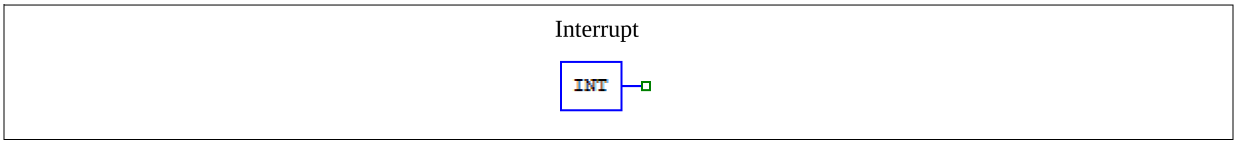
In Case 3, on the other hand, the global variable V1 is used in both Subcircuits 1 and 2. Subcircuit 1 and 2, however, are two exclusive states. That is, the system will run either Subcircuit 1, or Subcircuit 2, but not both. The use of the global variables is allowed in this case.

4.5 Interrupt

In a hardware target, elements such as digital input, encoder, capture, and PWM generators (for F2833x and F2803x DSPs) can generate hardware interrupt. The interrupt block allows users to associate the element that generates the interrupt with the corresponding subcircuit that represents the interrupt service routine.

Please note that the interrupt element cannot be placed inside a subcircuit. It must be in the top-level main circuit only.

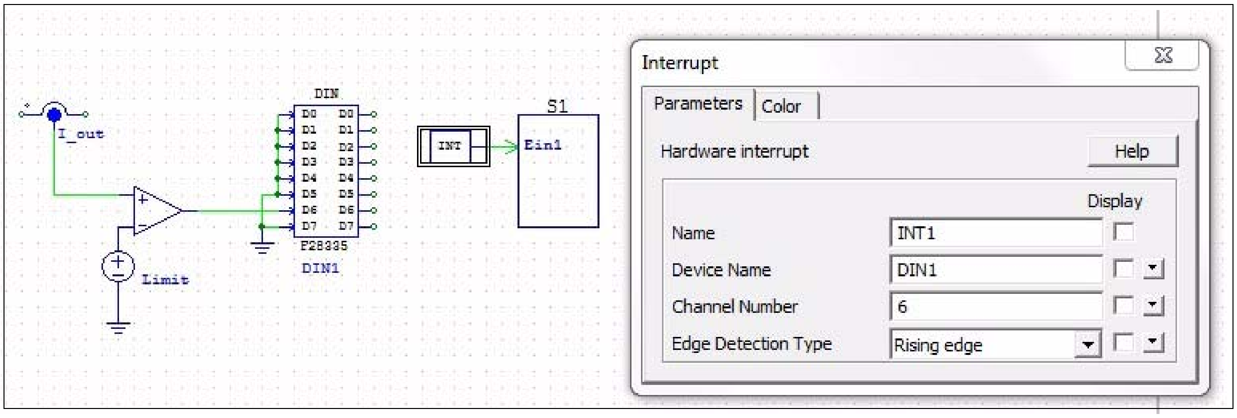
Image:



Attributes:

Parameters	Description
Device Name	The name of the hardware device that initiates the hardware interrupt
Channel Number	The input channel number of the device that initiates the interrupt. For example, if Channel D0 of a digital input generates the interrupt, the channel number should be set to 0. Note that this parameter is used only for: - Digital input - Capture (PE-Expert3 Target and General Hardware Target only) It does not apply to encoder and PWM generator.
Trigger Type	This applies to digital input and capture only. It can be one of the following: - *No edge detection*: No interrupt will be generated. - *Rising edge*: The rising edge of the input signal will generate interrupt. - *Falling edge*: The falling edge of the input signal will generate interrupt. - *Rising/falling edges*: Both the rising and falling edges of the input signal will generate interrupt.

The diagram below shows how the interrupt block is used.



Hardware interrupt

Name: INT1

Device Name: DIN1

Channel Number: 6

Edge Detection Type: Rising edge

In this circuit, the current *I_out* is measured and compared with the reference value *Limit*. If the current *I_out* exceeds the *Limit*, the output of the comparator will change from 0 to 1. This will generate a rising edge to channel *D6* of the digital input block *DIN1*. The interrupt block parameters are set as shown in the graph:

- Device Name: *DIN1* for the specified digital input block;
- Channel Number: *6* for the specified digital input channel *D6*;
- Edge Detection Type: Rising edge for the condition of *I_out* > *Limit.*

The rising edge at the output of the comparator will then generate a hardware interrupt and the operation will transit to Subcircuit *S1* through the input event port *EI1*.

Please note that the connection between the interrupt block *INT1* and the event subcircuit *S1* is an **Event Connection**, not a piece of wire.

5

F2833x Hardware Target

5.1 Overview

With the F2833x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2833x floating-point DSP.

The F2833x Hardware Target will work with all F2833x packages. The figures in the next two pages show the pin assignments of the F2833x DSP in the low-profile flatpack (LQFP) package. The main functions implemented in the F2833x Hardware Target are marked in color in the figures.

The F2833x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Start/Stop functions for PWM generators
- Trip-zone and trip-zone state
- A/D converter
- Digital input and output
- SCI configuration, input, and output
- SPI configuration, device, input, and output
- Capture and capture state
- Encoder and encoder state
- Up/Down counter
- DSP clock
- Hardware configuration

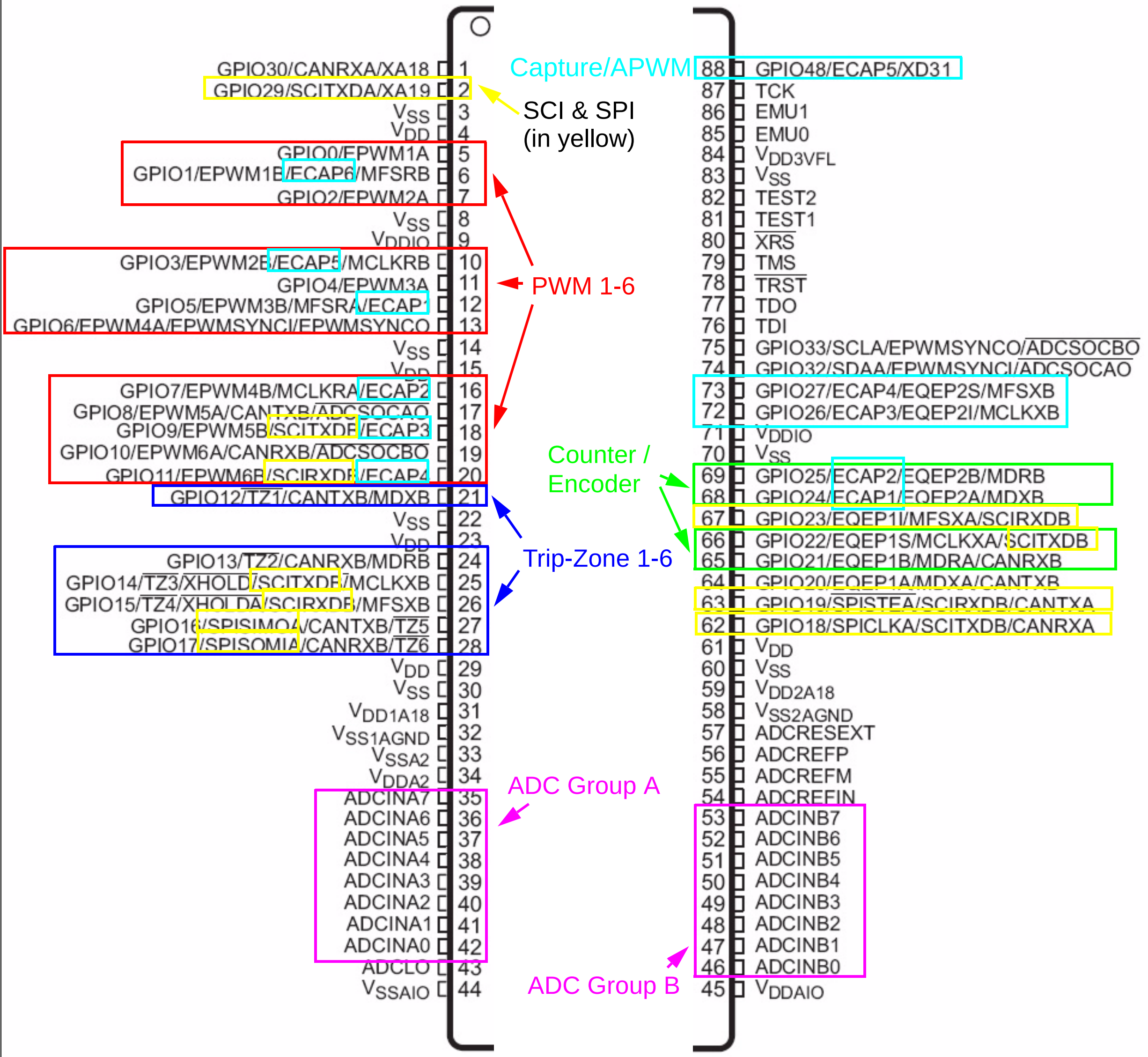
When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed, If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In TI F2833x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

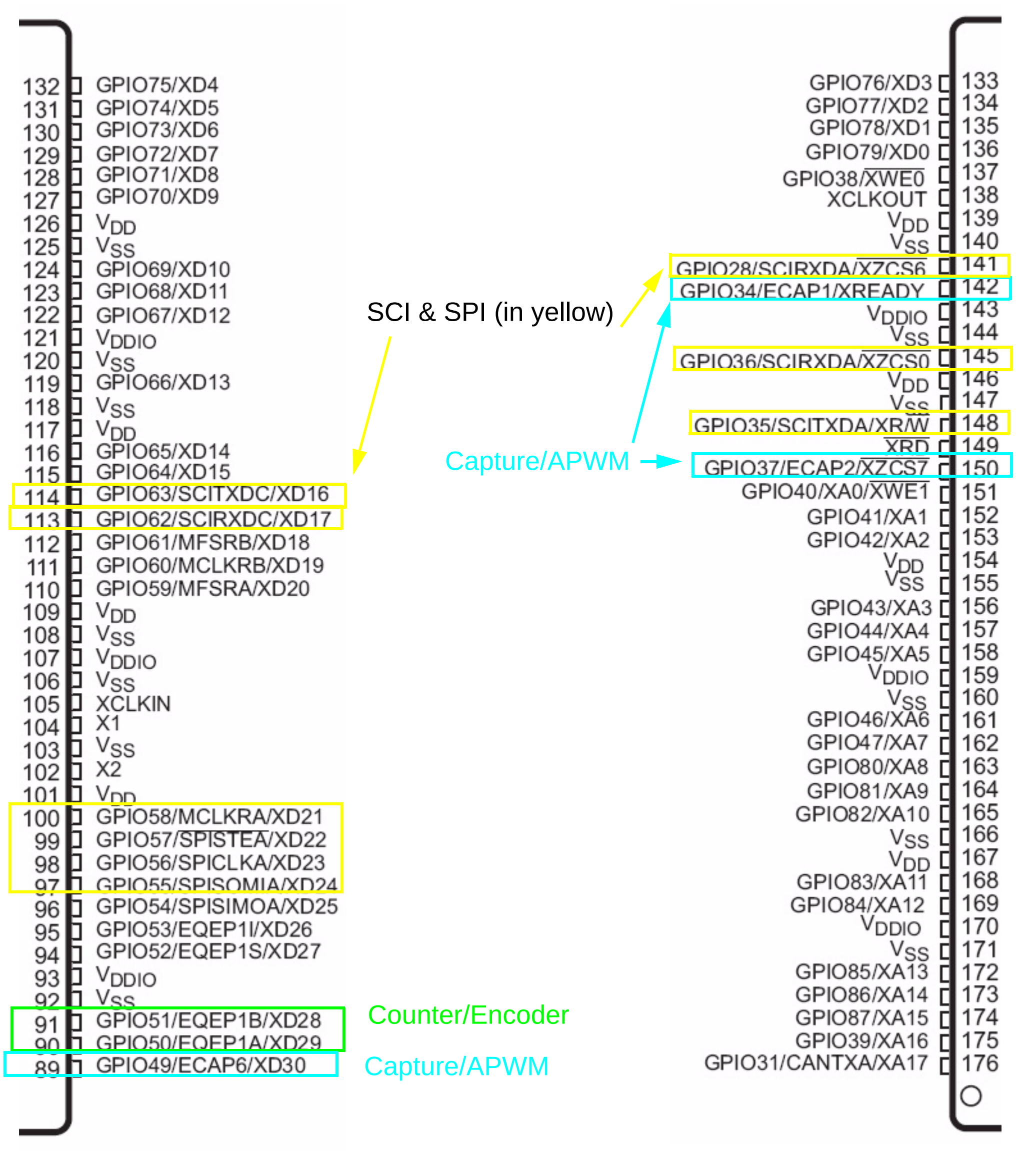
In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 5.4 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2833x Hardware Target library are described in this Chapter.

F28335 DSP Port Assignments (Pin 1 - 88)


GPIO30/CANRXA/XA18 1
GPIO29/SCITXDA/XA19 2
VSS 3
VDD 4
GPIO0/EPWM1A 5
GPIO1/EPWM1B/ECAP6/MFSRB 6
GPIO2/EPWM2A 7
VSS 8
VDDIO 9
GPIO3/EPWM2B/ECAP5/MCLKRB 10
GPIO4/EPWM3A 11
GPIO5/EPWM3B/MFSRA/ECAP1 12
GPIO6/EPWM4A/EPWMSYNCI/EPWMSYNCO 13
VSS 14
VDD 15
GPIO7/EPWM4B/MCLKRA/ECAP2 16
GPIO8/EPWM5A/CANTXB/ADCSOCAO 17
GPIO9/EPWM5B/SCITXDB/ECAP3 18
GPIO10/EPWM6A/CANRXB/ADCSOCBO 19
GPIO11/EPWM6B/SCIRXDB/ECAP4 20
GPIO12/TZ1/CANTXB/MDXB 21
VSS 22
VDD 23
GPIO13/TZ2/CANRXB/MDRB 24
GPIO14/TZ3/XHOLD/SCITXDB/MCLKXB 25
GPIO15/TZ4/XHOLDA/SCIRXDB/MFSXB 26
GPIO16/SPISIMOA/CANTXB/TZ5 27
GPIO17/SPISOMIA/CANRXB/TZ6 28
VDD 29
VSS 30
VDD1A18 31
VSS1AGND 32
VSSA2 33
VDDA2 34
ADCINA7 35
ADCINA6 36
ADCINA5 37
ADCINA4 38
ADCINA3 39
ADCINA2 40
ADCINA1 41
ADCINA0 42
ADCLO 43
VSSAIO 44
Capture/APWM
SCI & SPI
(in yellow)
PWM 1-6
Counter /
Encoder
Trip-Zone 1-6
ADC Group A
ADC Group B
88 GPIO48/ECAP5/XD31
87 TCK
86 EMU1
85 EMU0
84 VDD3VFL
83 VSS
82 TEST2
81 TEST1
80 XRS
79 TMS
78 TRST
77 TDO
76 TDI
75 GPIO33/SCLA/EPWMSYNCO/ADCSOCBO
74 GPIO32/SDAA/EPWMSYNCI/ADCSOCAO
73 GPIO27/ECAP4/EQEP2S/MFSXB
72 GPIO26/ECAP3/EQEP2I/MCLKXB
71 VDDIO
70 VSS
69 GPIO25/ECAP2/EQEP2B/MDRB
68 GPIO24/ECAP1/EQEP2A/MDXB
67 GPIO23/EQEP1I/MFSXA/SCIRXDB
66 GPIO22/EQEP1S/MCLKXA/SCITXDB
65 GPIO21/EQEP1B/MDRA/CANRXB
64 GPIO20/EQEP1A/MDXA/CANTXB
63 GPIO19/SPISTEA/SCIRXDB/CANTXA
62 GPIO18/SPICLKA/SCITXDB/CANRXA
61 VDD
60 VSS
59 VDD2A18
58 VSS2AGND
57 ADCRESEXT
56 ADCREFP
55 ADCREFM
54 ADCREFIN
53 ADCINB7
52 ADCINB6
51 ADCINB5
50 ADCINB4
49 ADCINB3
48 ADCINB2
47 ADCINB1
46 ADCINB0
45 VDDAIO

F28335 DSP Port Assignments (Pin 89 - 176)



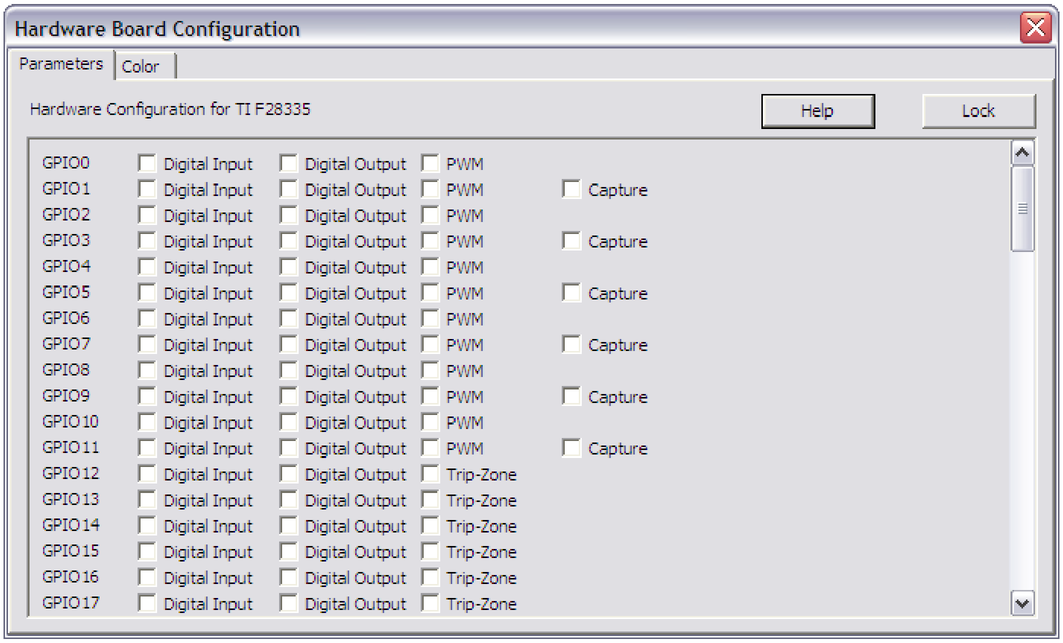
5.2 Hardware Configuration

The F2833x DSP provides 88 GPIO ports (GPIO0 to GPIO87), and each port may be configured for different functions. For a particular DSP board, however, not all the ports are accessible from outside, and often the functions of some ports are fixed. The Hardware Configuration block provides a way to configure SimCoder for a particular DSP board.

Image:

F28335
Board
Config

The dialog window of the block is shown below:



Hardware Configuration for TI F28335

GPIO0	□ Digital Input	□ Digital Output	□ PWM	
GPIO1	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO2	□ Digital Input	□ Digital Output	□ PWM	
GPIO3	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO4	□ Digital Input	□ Digital Output	□ PWM	
GPIO5	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO6	□ Digital Input	□ Digital Output	□ PWM	
GPIO7	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO8	□ Digital Input	□ Digital Output	□ PWM	
GPIO9	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO10	□ Digital Input	□ Digital Output	□ PWM	
GPIO11	□ Digital Input	□ Digital Output	□ PWM	□ Capture
GPIO12	□ Digital Input	□ Digital Output	□ Trip-Zone	
GPIO13	□ Digital Input	□ Digital Output	□ Trip-Zone	
GPIO14	□ Digital Input	□ Digital Output	□ Trip-Zone	
GPIO15	□ Digital Input	□ Digital Output	□ Trip-Zone	
GPIO16	□ Digital Input	□ Digital Output	□ Trip-Zone	
GPIO17	□ Digital Input	□ Digital Output	□ Trip-Zone	

For each GPIO port, a check box is provided for each of its available function. If this box is checked, only this function is used, and all other functions are not allowed in SimCoder. For example, Port GPIO1 can be used for "Digital Input", "Digital Output", "PWM" and "Capture". If a particular board uses Port GPIO1 as the "PWM" output, only the checkbox for "PWM" should be checked and all other check boxes should be left unchecked. If in the circuit Port GPIO1 is used as "Digital Input", SimCoder will report an error.

5.3 DSP Clock

The DSP Clock block defines the external clock frequency and the speed of the F2833x DSP, as well as the program space size.

Image:

DSP
Clock
F28335

Attributes:

Parameters	Description
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 30 MHz.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 150 MHz.

If the DSP Clock block is not used in a circuit, the default values of the DSP block are used.

5.4 PWM Generators

The F2833x DSP provides 6 sets of PWM outputs: PWM 1 (GPIO0 and GPIO1), PWM 2 (GPIO2 and GPIO3), PWM 3 (GPIO4 and GPIO5), PWM 4 (GPIO6 and GPIO7), PWM 5 (GPIO8 and GPIO9), and PWM 6 (GPIO10 and GPIO11). Each set has two outputs that are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM operates in a special operation mode.

In SimCoder, these 6 PWM's can be used in the following ways:

- Two 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);
- Six 2-phase PWM generators: PWM 1, 2, 3, 4, 5, and 6, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, 4, 5, and 6, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, 4, 5, and 6, with two outputs complementary to each other.

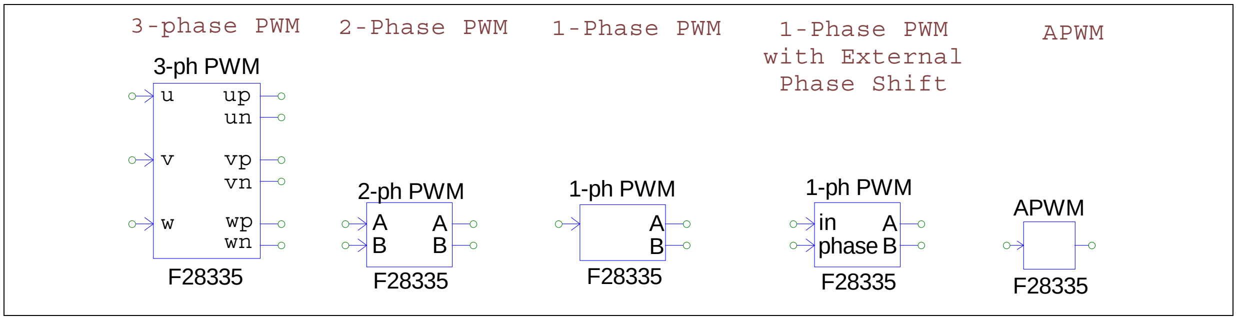
These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 APWM generators that use the same resources as the captures. These PWM generators have restricted functionality as compared to the 6 PWM generators (PWM 1 to 6) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

The images and parameters of the PWM generators are shown below.

Images:



5.4.1 3-Phase PWM Generator

In the 3-phase PWM generator image, "*u*", "*v*", and "*w*" refer to the three phases (alternatively they are called Phase "*a*", "*b*", and "*c*"). The letter "*p*" refers to the positive output, and "*n*" refers to the negative output. For example, for 3-phase PWM 123, "*up*" is PWM1A, and "*un*" is PWM1B.

Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be either "3-phase PWM 123" that uses PWM 1 to 3, or "3-phase PWM 456" that uses PWM 4 to 6.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.

PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Triangular* (*start low*): Triangular wave, and the initial PWM output state is low. - *Triangular* (*start high*): Triangular wave, and the initial output state is high. - *Sawtooth* (*start low*): Sawtooth wave, and the initial output state is low. - *Sawtooth* (*start high*): Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC Group A*: PWM will trigger Group A of the A/D converter. - *Trigger ADC Group B*: PWM will trigger Group B of the A/D converter. - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - *High impedance*: The PWM outputs are in high impedance. - *PWM A high & B low*: The PWM positive output is high, and the negative output is low. - *PWM A low & B high*: The PWM positive output is low, and the negative output is high. - *No action*: No action is taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u, v, w	Initial value of 3-phase inputs *u*, *v*, and *w*
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

5.4.2 1-Phase PWM Generators

The attributes are the same for 1-phase PWM generators with internal or external phase shift. the difference is the 1-phase PWM with external phase shift has one extra input for the phase shift (labeled as "phase"). The phase is in degree, the same as the 1-phase PWM without external phase shift.

Attributes:

Parameters	**Description**
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 6. With phase shift, it can be PWM 2 to PWM 6.
Output Mode	The output mode of the PWM generator. It can be one of the following: - *Use PWM A&B*: Both PWM outputs A and B are used, and they are complementary. - *Use PWM A*: Only PWM output A is used. - *Use PWM B*: Only PWM output B is used.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Triangular* (*start low*): Triangular wave, and the initial PWM output state is low. - *Triangular* (*start high*): Triangular wave, and the initial output state is high. - *Sawtooth* (*start low*): Sawtooth wave, and the initial output state is low. - *Sawtooth* (*start high*): Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC Group A*: PWM will trigger Group A of the A/D converter. - *Trigger ADC Group B*: PWM will trigger Group B of the A/D converter. - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - *High impedance*: The PWM outputs are in high impedance. - *PWM A high & B low*: The PWM positive output is high, and the negative output is low. - *PWM A low & B high*: The PWM positive output is low, and the negative output is high. - *No action*: No action is taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in degree. (1-phase PWM with phase shift has en input for this attribute)
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

Phase Shift

A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. There are two PWM series: PWM 1, 2, 3; and PWM 1, 4, 5, 6, as described below.

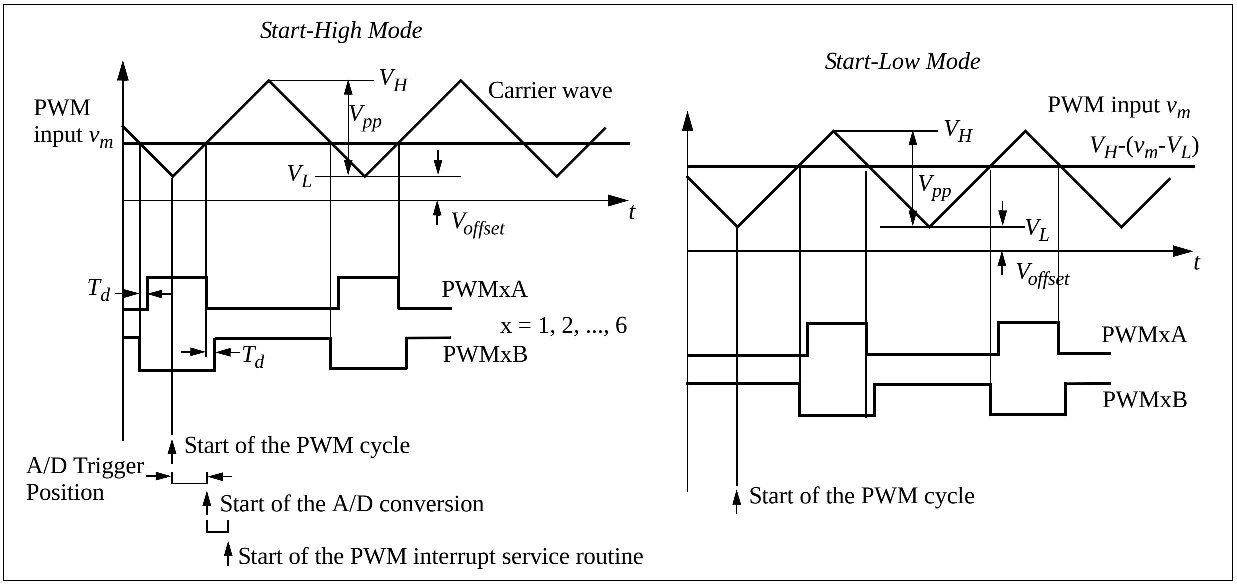
- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can be the reference, and PWM 2 and 3, or PWM 4, 5, and 6, can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3 can be phase shifted with respect to PWM 2. Similarly, PWM 4 (or 5) can be the reference, and PWM 5 (or 6) can be phase shifted with respect to PWM 4 (or 5). But using PWM 2 or 3 as the reference for PWM 4, 5, or 6 is not allowed.
- The reference PWM and the PWM being shifted must be consecutive in the series. That is, it is not permitted to use PWM 1 as the reference, and phase shift PWM 3 without PWM2, or PWM 5 or 6 wit.

The phase shift value is in degree. When the value is -30^o, the output will be shifted to the right (lagging) by 30^o of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30^o. When the phase value is 30^o, the output will be shifted to the left (leading) by 30^o.

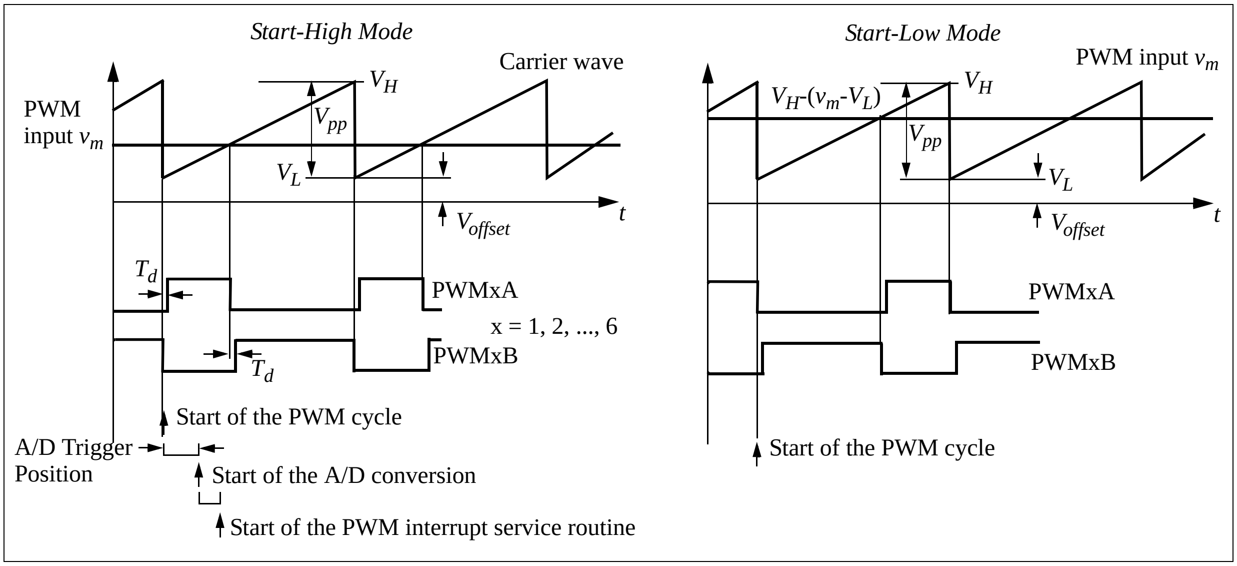
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below.

The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m, and the lowest value of the carrier wave is V_L and the highest value is V_H. In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, V_L=0, and V_H=1. If v_m=0.2, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value V_H-(v_m-V_L). For example, for a carrier wave from 0 to 1, V_L=0, and V_H=1. If v_m=0.2, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

Note: In the start-low mode, the PWM input v_m is converted to V_H-(v_m-V_L) internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with V_L=0 and V_H=1, or for a triangular wave with V_L= -V_H, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

5.4.3 2-Phase PWM Generator

Attributes:

Parameters	**Description**
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 6.
Mode Type	The operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC Group A*: PWM will trigger Group A of the A/D converter. - *Trigger ADC Group B*: PWM will trigger Group B of the A/D converter. - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - *High impedance*: The PWM outputs are in high impedance. - *PWM A high & B low*: The positive output of the PWM is high, and the negative output is low. - *PWM A low & B high*: The positive output of the PWM is low, and the negative output is high. - *No action*: No action is taken.

Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

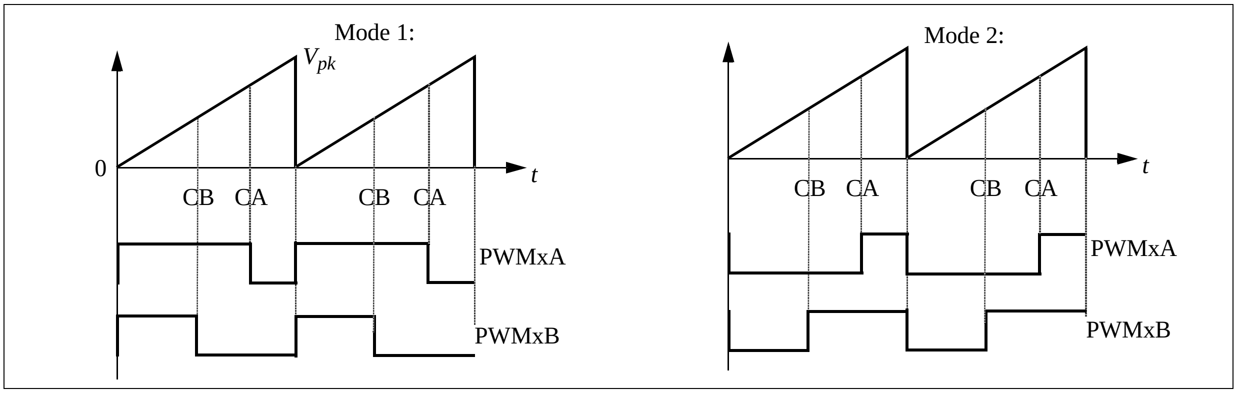
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk}, and there is no dc offset.

Operation Mode 1:

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

Operation Mode 2:

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

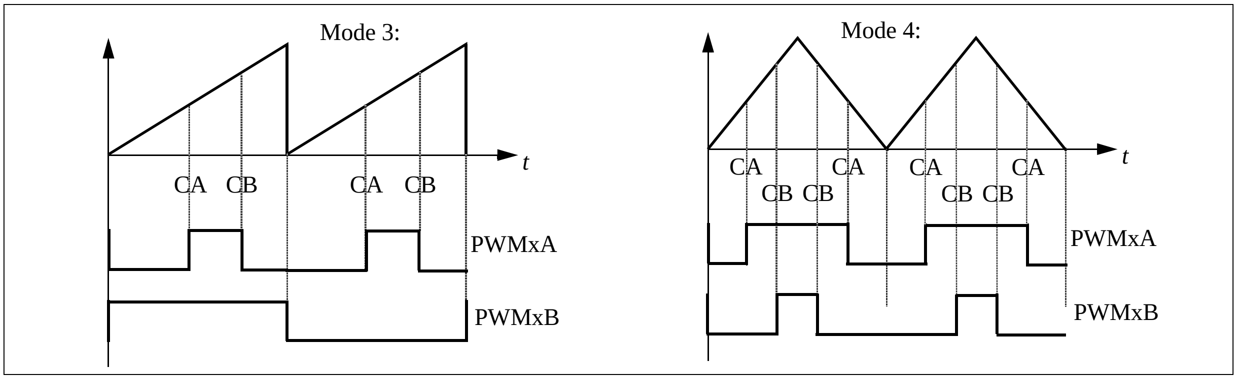


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

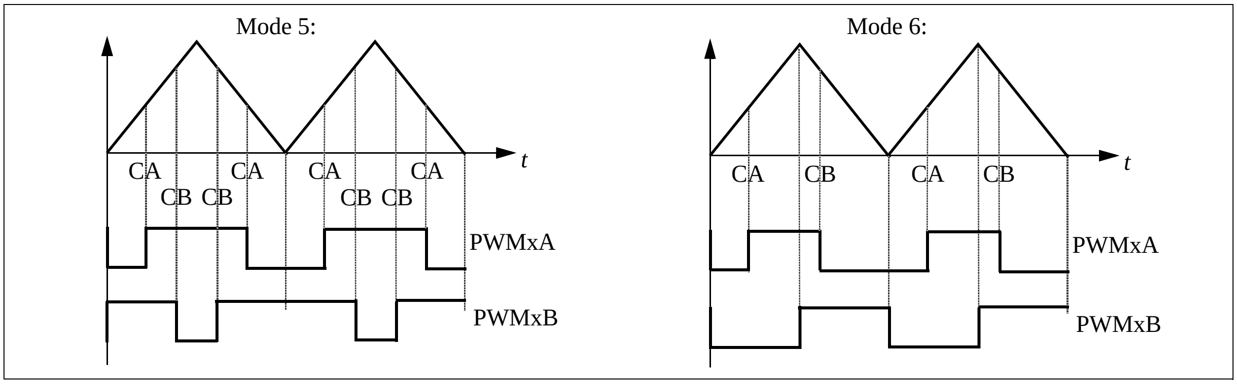


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

The figure below on the right shows the waveforms of Mode 6. Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



5.4.4 APWM Generator:

Attributes:

Parameters	**Description**
PWM Source	APWM generators share the same resource as captures. The PWM source can be one of the six APWM's in 14 designated GPIO ports, as listed below: - *APWM 1* (*GPIO5, GPIO24, GPIO34*) - *APWM 2* (*GPIO7, GPIO25, GPIO37*) - *APWM 3* (*GPIO9, GPIO26*) - *APWM 4* (*GPIO11, GPIO27*) - *APWM 5* (*GPIO3, GPIO48*) - *APWM 6* (*GPIO1, GPIO49*)
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Sawtooth* (*start low*): sawtooth wave, with the PWM output low initially. - *Sawtooth* (*start high*): Sawtooth wave, with the PWM output high initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: - Output low: The PWM output will be set to low. - Output high: The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

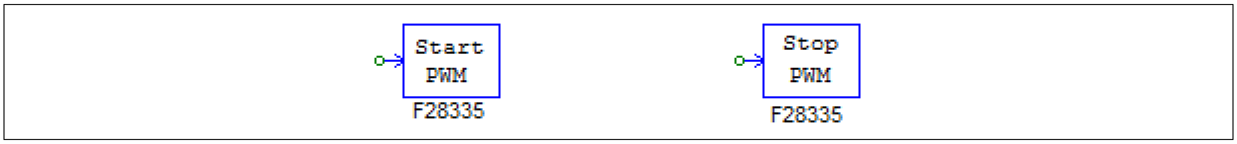
Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The phase shifting rules are also the same as the 1-phase PWM generators.

As noted before, the APWM generators has reduced number of functions than 1-phase PWM generators. It can not trigger the A/D converter and can not use the trip-zone signal.

5.5 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Images:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-6, 3-phase PWM 123 and PWM 456, and Capture 1-6.

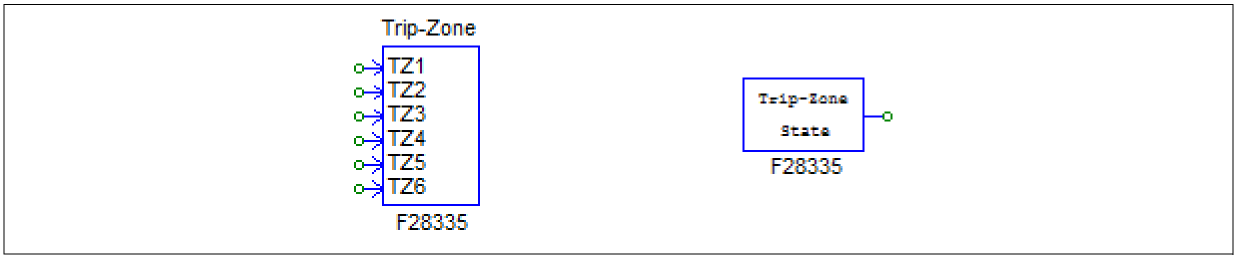
5.6 Trip-Zone and Trip-Zone State

The F2833x DSP provides 6 trip-zones, Trip-Zone 1 to 6 which use the ports GPIO12 to GPIO17. Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0).

Images:



Attributes for Trip-Zone:

Parameters	Description
Port GPIO12 as Trip-Zone 1	Define if Port GPIO12 is used as trip-zone 1.
Port GPIO13 as Trip-Zone 2	Define if Port GPIO13 is used as trip-zone 2.
Port GPIO14 as Trip-Zone 3	Define if Port GPIO14 is used as trip-zone 3.
Port GPIO15 as Trip-Zone 4	Define if Port GPIO15 is used as trip-zone 4.
Port GPIO16 as Trip-Zone 5	Define if Port GPIO16 is used as trip-zone 5.
Port GPIO17 as Trip-Zone 6	Define if Port GPIO17 is used as trip-zone 6.

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-6, and 3-phase PWM 123 and PWM 456.

The trip-zone interrupt can be generated in either one-shot mode of cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

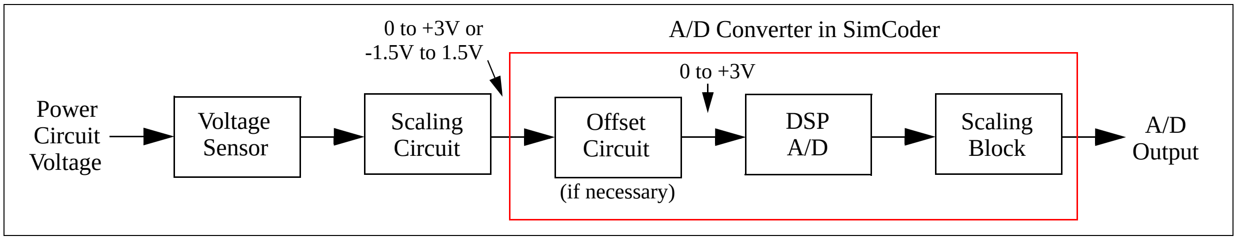
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

5.7 A/D Converter

The F2833x DSP provides a 12-bit 16-channel A/D converter. It is divided into two groups: Group A and Group B. The input range of the physical A/D converter on the DSP is from 0V to +3V.

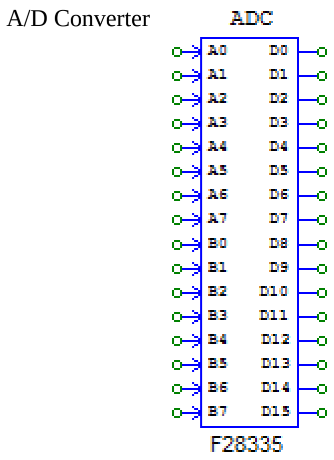
Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the 0V and +3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.



As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications.

The image and the parameters of the A/D converter in the SimCoder library are described below. In the following description, "A/D converter" refers to the A/D converter in the SimCoder library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	Description
ADC Mode	Define the A/D converter mode of operation. It can be one of the following: - *Continuous*: The A/D converter performs the conversion continuously. When the converter value is read, the result of the last conversion is read. - *Start/stop* (*8-channel*): The A/D converter only performs the conversion upon request, on only one of the 8-channel groups. - *Start/stop* (*16-channel*): The A/D converter only performs the conversion upon request, on all 16 channels.
Ch A_i *or* B_i Mode	Input mode of the A/D converter channel A_i or B_i, where *i* is from 0 to 7. The input mode can be one of the following: - *AC*: The input range is considered from -1.5V to +1.5V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3V range. - *DC*: The input is a dc value, and the range is from 0 to +3V.
Ch A_i *or* B_i Gain	Gain *k* of the A/D converter channel A_i (or B_i), where *i* is from 0 to 7.

Mode of Operation:

The A/D converter can perform conversion autonomously when it is set to the "Continuous" mode. The "Start/Stop" mode is for the conversion to be triggered by a PWM generator.

Note the following restrictions in using PWM generator triggered A/D converter:

- The A/D converter can be triggered by only one PWM generator. That is, if there are multiple PWM generators, only one can be set to trigger the A/D converter, and the rest should be set not to trigger the A/D converter.
- It is not permitted to have the A/D converter triggered by one PWM generator while some of the signals in this group are also used in a circuit that has a different sampling rate than the frequency of the PWM generator.

In these situations, it is recommended that the A/D converter be set to the "Continuous" mode.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input port of the A/D converter.

Input Offset and Scaling:

Note that the input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input channel mode is DC, the maximum input voltage be scaled to +3V; and when the input channel mode is AC, the maximum peak voltage be scaled to +1.5V.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.0V.

SimCoder's A/D converter for F2833x DSP provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

A/D Converter Channel DC Mode Example

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

V_{i_min} = 0 V, V_{i_max} = 150 V

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3V. Assume that the actual value of the voltage at a certain point is:

V_i = 100 V

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$V_{i\_max\_s}$ = 150 *0.01 = 1.5 V

V_{i_s} = 100 *0.01 = 1 V

To utilize the full range of the DSP, a conditioning circuit with a gain of 2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: 0.01*2 = 0.02. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$V_{i\_max\_s\_c}$ = 1.5 * 2 = 3 V

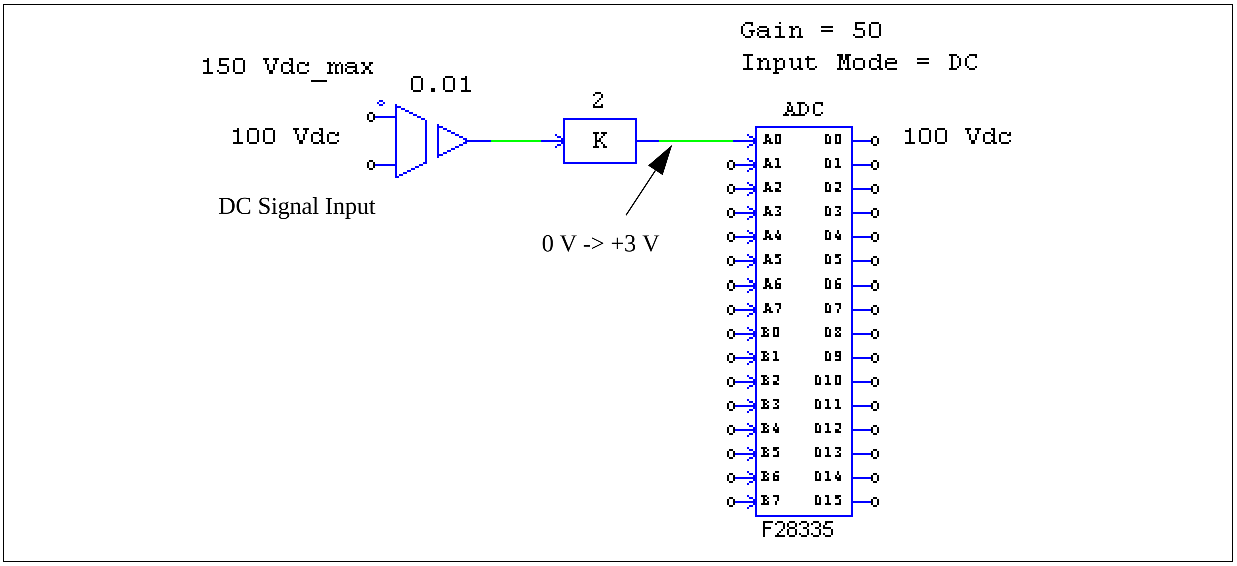
$V_{i\_s\_c}$ = 1 * 2 = 2 V

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

V_{o_max} = 50 * 3 = 150 V

V_o = 50 * 2 = 100 V

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.


Gain = 50
Input Mode = DC
150 Vdc_max
0.01
2
ADC
100 Vdc
K
100 Vdc
DC Signal Input
0 V -> +3 V
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
F28335

Please note that, in this example, if the gain of the proportional block is changed from 2 to 1, and the A/D gain is changed from 50 to 100, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3V.

A/D Converter Channel AC Mode Example

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

V_{i_max} = +/- 75 V

The input mode of the A/D converter will be set to ac, and the input range is from -1.5V to +1.5V. Assume that the actual value of the voltage has a peak value of:

V_i = +/- 50 V

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$V_{i\_max\_s}$ = +/- 0.75 V

V_{i_s} = +/- 0.5 V

Since the A/D converter input range is from -1.5V to +1.5V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2 is needed (i.e. 1.5/0.75 = 2). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$V_{i\_max\_s\_c}$ = +/- 1.5 V

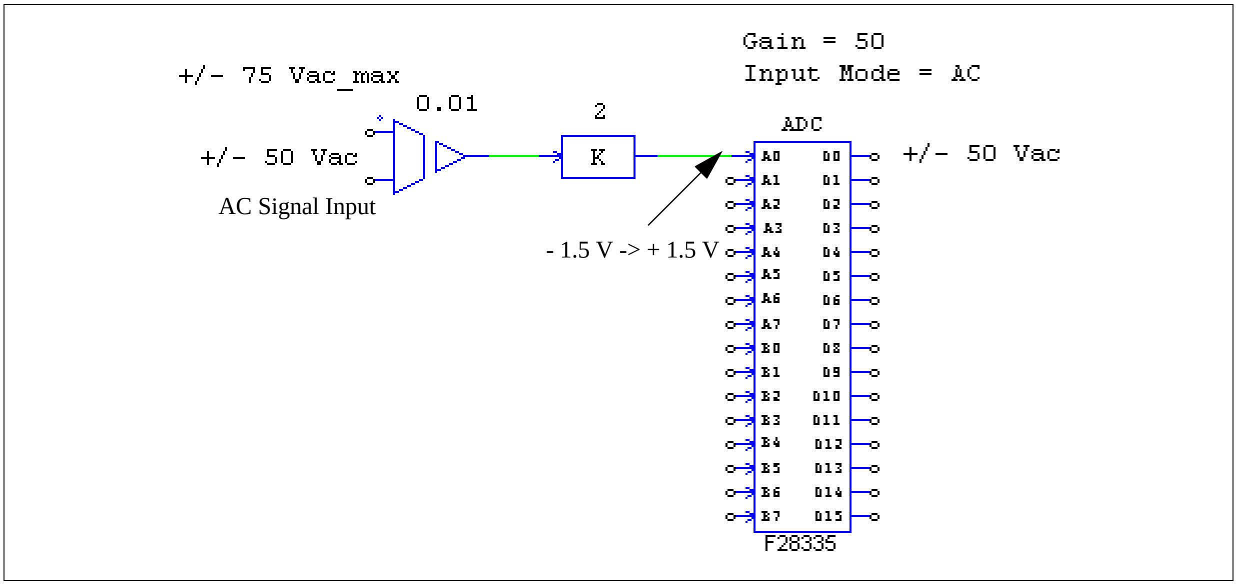
$V_{i\_s\_c}$ = +/- 1 V

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

V_{o_max} = +/- 75 V

V_o = +/- 50 V

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.



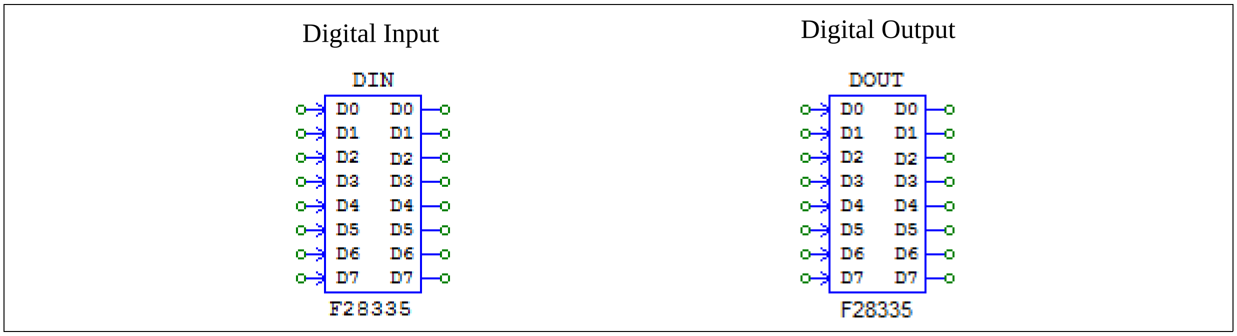
Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to ac, the input range is from -1.5V and +1.5V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal will need to be scaled and offset so that the range is within 0V to +3V required by the DSP A/D converter.

Also, to ensure the correctness of the generated code for the hardware, the maximum peak value of the input must be scaled to 1.5V at the input port of the A/D converter.

5.8 Digital Input and Digital Output

The F2833x DSP has 88 general-purpose-input-output (GPIO) ports that can be configured as either digital inputs or digital output. In SimCoder, an 8-channel block is provided for digital input or output. Multiple 8-channel digital input/output blocks can be used in a schematic.

Images:



Attributes for Digital Input:

Parameters	Description
Port Position for Input *i*	The port position of the Input *i*, where *i* is from 0 to 7. It can be one of the 88 GPIO ports, from GPIO0 to GPIO87.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	Description
Port Position for Output *i*	The port position of the Output *i*, where *i* is from 0 to 7. It can be one of the 88 GPIO ports, from GPIO0 to GPIO87.

Note that if a GPIO port is used as an input port, this same port cannot be used as another peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

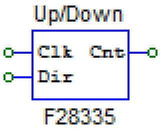
In the F2833x DSP, up to 7 external interrupt sources can be defined from ports GPIO0 to GPIO63 (specifically, up to 2 interrupt sources from Port GPIO0 to GPIO31, and up to 5 interrupt sources from Port GPIO32 to GPIO63). The priority of external interrupts in Port GPIO0 to GPIO31 is higher than the priority of the interrupt in Port GPIO32 to GPIO63.

5.9 Up/Down Counter

The F2833x DSP has 2 up/down counters. Counter 1 can be at either Port GPIO 20-21, or Port GPIO50-51. Counter 2 is at Port GPIO24-25.

Note that Counter 1 at Port GPIO20-21 and Port GPIO50-51 uses the same inner function blocks, and cannot be used at the same time.

Image:



Attributes:

Parameters	Description
Counter Source	Source of the counter. It can be one of the following: - *Counter 1* (*GPIO20, 21*): Counter 1 at Port GPIO20 and 21 is used. - *Counter 1* (*GPIO50, 51*): Counter 1 at Port GPIO50 and 51 is used. - *Counter 2* (*GPIO24, 25*): Counter 2 at Port GPIO24 and 25 is used.

In the image, "Clk" refers to the input clock signal, and "Dir" refers to the signal that defines the counting direction. When the "Dir" input is 1, the counter counts forward, and when the input is 0, the counter counts backward.

Note that the "Clk" input corresponds to the first port of the counter, and the "Dir" input corresponds to the second port. For example, for Counter 1 at Port GPIO20 and 21, GPIO20 is the "Clk" input and GPIO21 is the "Dir" input.

The output of the up/down counter gives the counter value.

Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

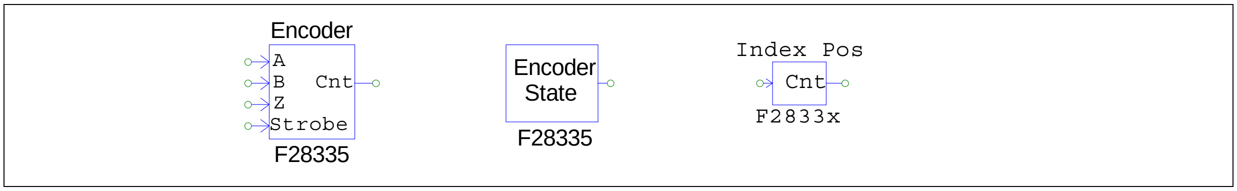
5.10 Encoder and Encoder State

The F2833x DSP has 2 **Encoders**. Encoder 1 can be at either Port GPIO 20-21, or Port GPIO50-51. Encoder 2 is at Port GPIO24-25. Note that Encoder 1 at Port GPIO20-21 and at Port GPIO50-51 uses the same inner function blocks, and cannot be used at the same time. The output of the encoder gives the counter value.

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's intial position. When the input of this block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Image:



Attributes for Encoder:

Parameters	**Description**
Encoder Source	Source of the encoder. It can be one of the followings: - *Encoder 1* (*GPIO20, 21*): Encoder 1 at Port GPIO20 and 21 is used, wth GPIO22 as Strobe and GPIO23 as Index (Z). - *Encoder 1* (*GPIO50, 51*): Encoder 1 at Port GPIO50 and 51 is used, wth GPIO52 as Strobe and GPIO53 as Index (Z). - *Encoder 2* (*GPIO24, 25*): Encoder 2 at Port GPIO24 and 25 is used, wth GPIO27 as Strobe and GPIO26 as Index (Z).
Use Z Signal	Define if the encoder uses the Z (or index) signal.
Use Strobe Signal	Define if the encoder uses the strobe signal.
Counting Direction	The counting direction can be either *Forward* or *Reverse*. When it is set to *Forward*, the encoder counts up. Otherwise, the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

Attributes for Encoder State:

Parameters	**Description**
Encoder Source	Define which encoder generates the interrupt. It can be one of the followings, must be the same Encoder used in the same schematic: - *Encoder 1* (*GPIO20, 21*): Encoder 1 at Port GPIO20 and 21 is used. - *Encoder 1* (*GPIO50, 51*): Encoder 1 at Port GPIO50 and 51 is used. - *Encoder 2* (*GPIO24, 25*): Encoder 2 at Port GPIO24 and 25 is used.

Attributes for Encoder Index/Strobe Pos:

Parameters	Description
Encoder Source	Define which encoder generates the interrupt. It can be one of the followings, must be the same Encoder used in the same schematic: - *Encoder 1* (*GPIO20*, *21*): Encoder 1 at Port GPIO20 and 21 is used. - *Encoder 1* (*GPIO50*, *51*): Encoder 1 at Port GPIO50 and 51 is used. - *Encoder 2* (*GPIO24*, *25*): Encoder 2 at Port GPIO24 and 25 is used.
Latch Position	Specify the kept counter type, choose from the followings: - IndexPos, if the setting "Use Z Siganl" is not "No" in Encoder - StrobePos, if the setting "Use Strobe Siganl" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - The first latched position, or - The current latched position

5.11 Capture and Capture State

The F2833x DSP provides 6 captures. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Images:

Capture
F28335
Capture
State
F28335

Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture. There are in total 6 captures that use 14 designated GPIO ports, as listed below: - *Capture 1* (*GPIO5, GPIO24, GPIO34*) - *Capture 2* (*GPIO7, GPIO25, GPIO37*) - *Capture 3* (*GPIO9, GPIO26*) - *Capture 4* (*GPIO11, GPIO27*) - *Capture 5* (*GPIO3, GPIO48*) - *Capture 6* (*GPIO1, GPIO49*)
Event Filter	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either *Absolute time* or *Time difference*.

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It can be one of the 6 captures: *Capture 1*, *Capture 2*, ..., *Capture 6*.

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

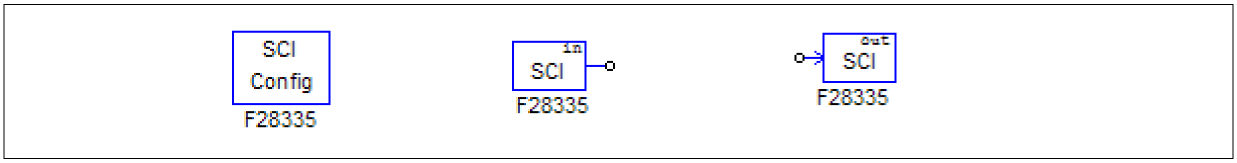
5.12 Serial Communication Interface (SCI)

The F2833x DSP provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

For more detailed descriptions on SCI and the monitoring function, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring in F2833x Target.pdf*".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

Images:



5.12.1 SCI Configuration

The SCI Configuration block defines the SCI port, communication speed, parity check type, and data buffer size.

Attributes:

Parameters	Description
SCI Port	Define the SCI port. There are 7 sets of GPIO ports that can be used for SCI, as listed below: - *SCIA* (*GPIO28*, *GPIO29*) - *SCIA* (*GPIO35*, *GPIO36*) - *SCIB* (*GPIO9*, *GPIO11*) - *SCIB* (*GPIO14*, *GPIO15*) - *SCIB* (*GPIO18*, *GPIO19*) - *SCIB* (*GPIO22*, *GPIO23*) - *SCIC* (*GPIO62*, *GPIO63*)
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either *None*, *Odd*, or *Even*.
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

For more information on how to select the buffer size, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring in F2833x Target.pdf*".

5.12.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

In the schematic, the SCI input behaviors as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

5.12.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when then DSP Oscilloscope is in the *continuous* mode. When it is in the *snap-shot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaviors as a voltage probe.

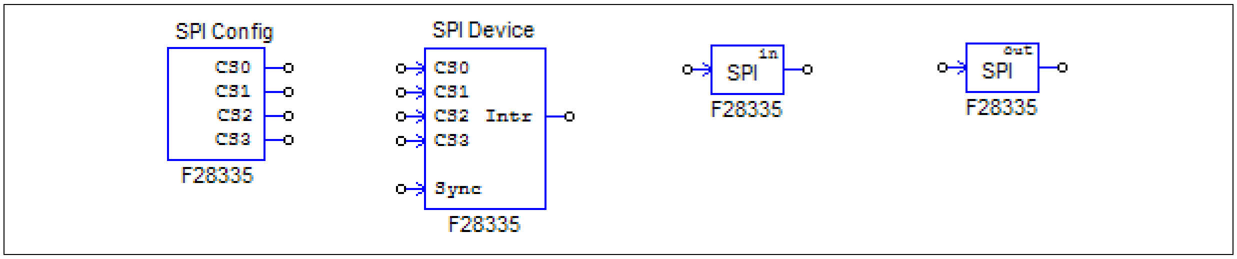
5.13 Serial Peripheral Interface (SPI)

The F2833x DSP provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F833x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring in F2833x.pdf*".

Four SCI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

Image:



5.13.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Attributes:

Parameters	Description
SPI Port	Define the SPI port. The SPI port can be either *GPIO16-19* or *GPIO54-57*.
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/ Output elements.

5.13.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.
Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: - *Rising edge without delay*: The clock is normally low, and data is latched at the clock rising edge. - *Rising edge with delay*: The clock is normally low, and data is latched at the clock rising edge with delay. - *Falling edge without delay*: The clock is normally high, and data is latched at the clock falling edge. - *Falling edge with delay*: The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.

Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either *Rising edge* or *Falling edge*.
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: - *No hardware interrupt* - Rising edge - Falling edge
Interrupt Timing	Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following: - *No interrupt*: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - *Multiple interrupt in series*: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - *One-time interrupt*: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

5.13.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: *ElementName* = {*Xn*[*MSB..LSB*]} where - *ElementName* is the name of the SPI input device. If it is the current SPI input device, use *y* instead. - {} means that the item in the bracket repeats multiple times. - *Xn* is the n_{th} word received from the SPI input device, and *n* start from 0. - *MSB..LSB* defines the position of the significant bits in the word.
Input Range	Specify the parameter Vmax that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to Vmax. If the device conversion mode is AC, the input ranges from -Vmax/2 to Vmax/2.
Scale Factor	Output scale factor Kscale. If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, y=x1[3..0]x2[7..0], means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \dfrac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data\_\mathrm{Length}}}$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \dfrac{(Result - 2^{Data\_\mathrm{Length-1}}) \cdot V_{max} \cdot K_{scale}}{2^{Data\_\mathrm{Length-1}}}$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

5.13.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale}. If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max}. If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either *DC* or *AC*. Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: *ElementName* = {*Xn*[*MSB..LSB*]} where - *ElementName* is the name of the SPI output device. If it is the current SPI output device, use *y* instead. - {} means that the item in the bracket repeats multiple times. - *Xn* is the n_{th} word sent to the SPI output device, and *n* start from 0. - *MSB..LSB* defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, y=x1[3..0]x2[7..0], means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \dfrac{Result \cdot K_{scale} \cdot 2^{Data\_\text{Length}}}{V_{max}}$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = 2^{Data\_\text{Length}} + \dfrac{Result \cdot K_{scale} \cdot 2^{Data\_\text{Length-1}}}{V_{max}}$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

5.14 Project Settings and Memory Allocation

When generating the code for the F2833x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio development environment where the code will be compiled, linked, and uploaded to the DSP.

At the present, the Code Composer Studio version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2833x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- DSP28335_Headers_nonBIOS.cmd:Peripheral register linker command file
- F28335_FLASH_Lnk.cmd:Flash memory linker command file
- F28335_FLASH_RAM_Lnk.cmd:Flash RAM memory linker command file
- F28335_RAM_Lnk.cmd:RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F28335. For example, if the target hardware is F28334, the file names will be *F28334 FLASH Lnk.cmd*, *F28334 FLASH RAM Lnk.cmd*, and *F28334RAM Lnk.cmd* accordingly.

Besides, the project also needs the following files:

- PS_bios.lib: SimCoder F2833x library, located in the PSIM folder
- C28x_FPU_FastRTS_beta1.lib:TI fast floating-point library, located in the PSIM \lib sub-folder

These two files "PS_bios.lib" and "C28x_FPU_FastRTS_beta1.lib" will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pjt file (in this example, "test.c" and "test.pjt") will be created. If you have made changed manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When the RAM Debug or RAM Release setting is selected, Code Composer Studio uses the linker command file F28335_RAM_Lnk.cmd to allocate the program and data space.

When the Flash Release setting is selected, Code Composer Studio uses the linker command file F28335_FLASH_Lnk.cmd to allocate the program and data space.

When the Flash RAM Release setting is selected, Code Composer Studio uses the linker command file F28335_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory
0x0000 - 0x07FF (2K)
interrupt vectors
stack
0x8000 - 0xFFFF (32K*)
program and data space

With the Flash Release setting:

RAM Memory	**Flash Memory**
0x0000 - 0x07FF (2K)	0x300000 - 0x33FFFF (256K**)
interrupt vectors	program
stack	password
0x8000 - 0xFFFF (32K*)	etc.
data space	

Notes:

***** The RAM memory predefined by SimCoder for program and data space is:

- For F28335, F28334: from 0x8000 to 0xFFFF (32K)
- For F28332, from 0x8000 to 0xDFFF (16K)
- If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.

** The flash memory predefined by SimCoder for program space is:

- For F28335: from 0x300000 to 0x33FFFF (256K)
- For F28334: from 0x320000 to 0x33FFFF (128K)
- For F28332: from 0x330000 to 0x33FFFF (64K)

6

F2803x Hardware Target

6.1 Overview

With the F2803x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2803x fixed-point DSP.

The TI F2803x Hardware Target will work with all F2803x packages. The figures in the next three pages show the pin assignments of the F2803x DSP in different packages.

The TI F2803x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Start/Stop functions for PWM generators
- Trip-one and trip-zone state
- A/D converter
- Comparator input, output, and DAC
- Digital input and output
- Up/Down counter
- Encoder and encoder state
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- DSP clock
- Hardware configuration

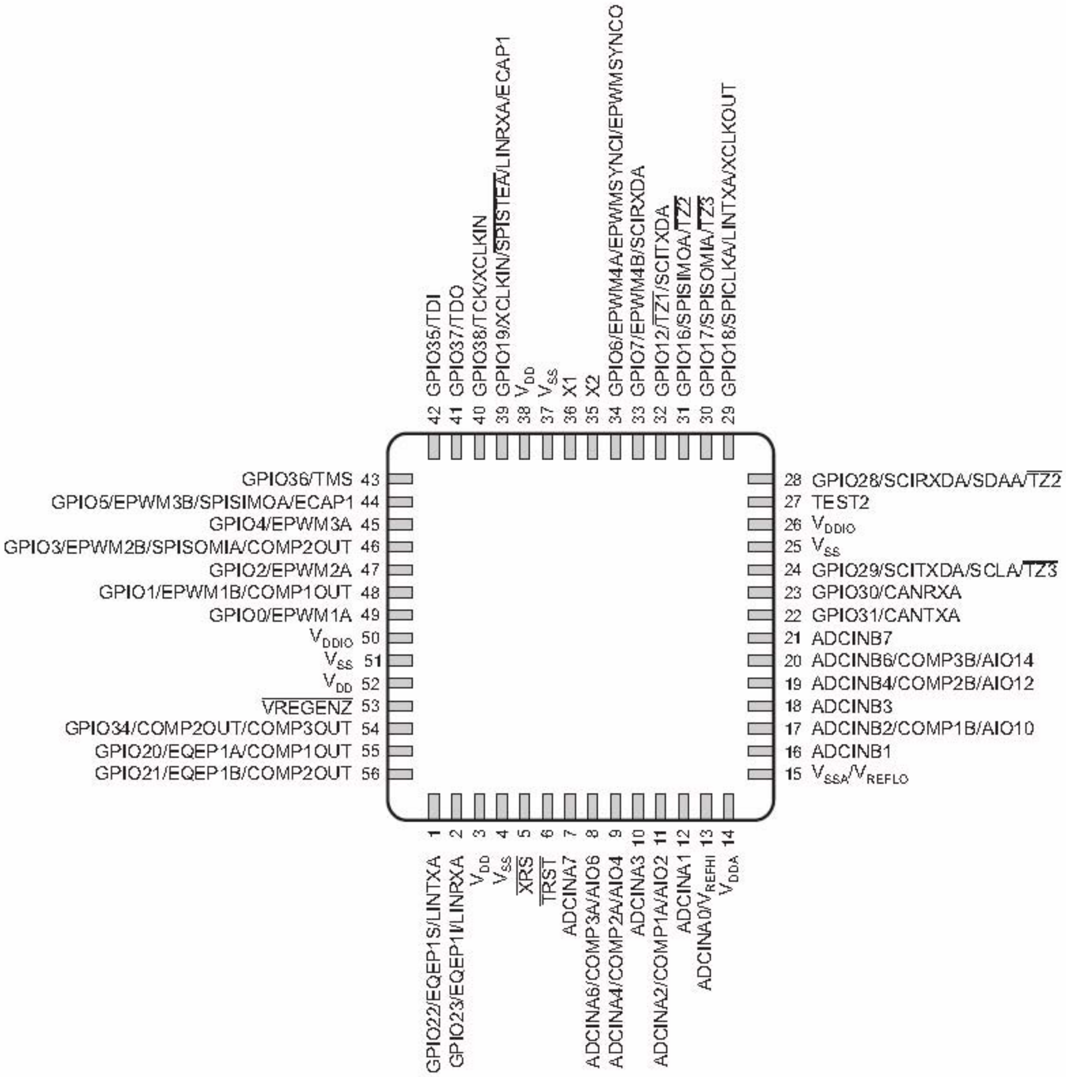
When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed, If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In TI F2803x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

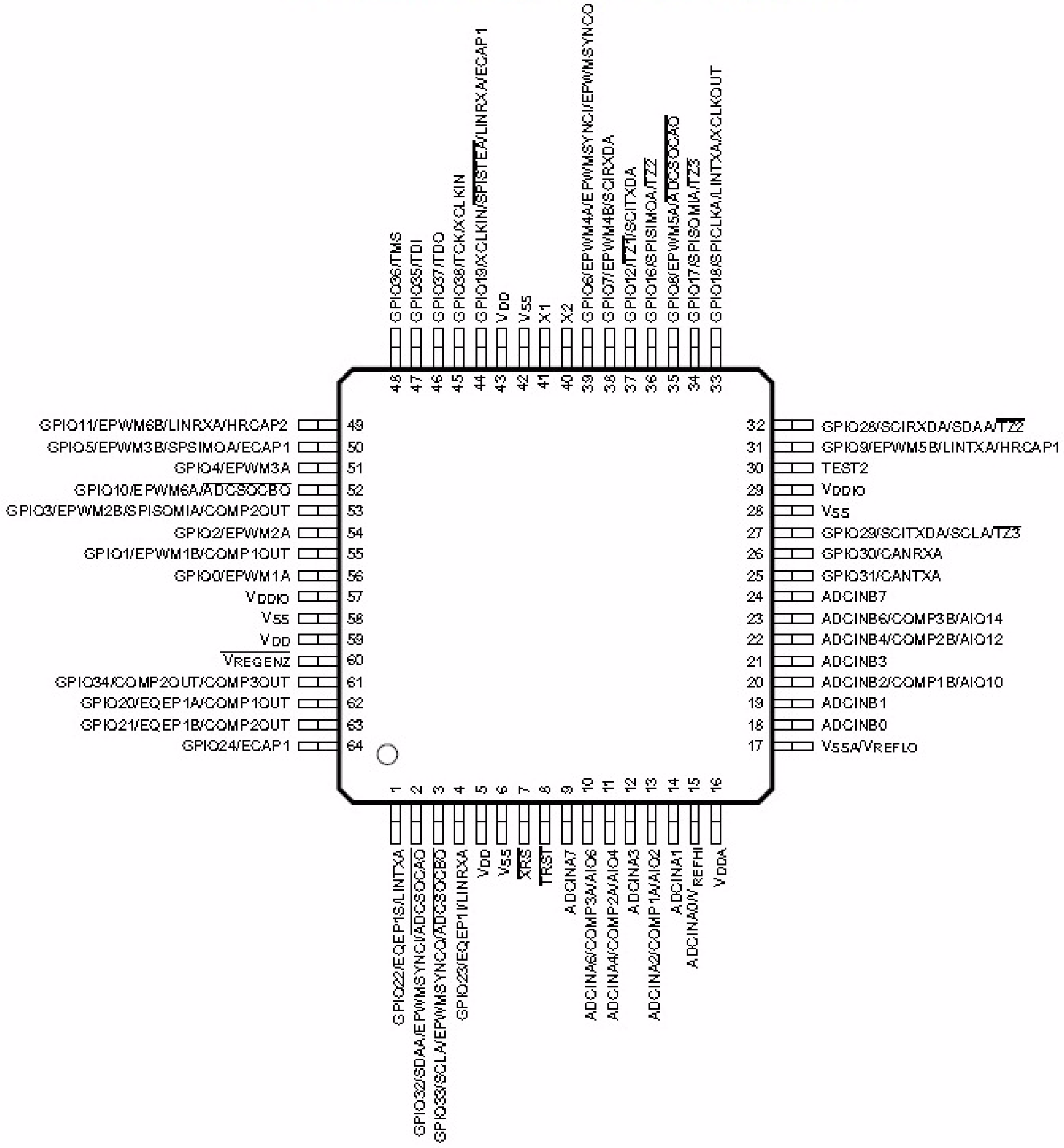
In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

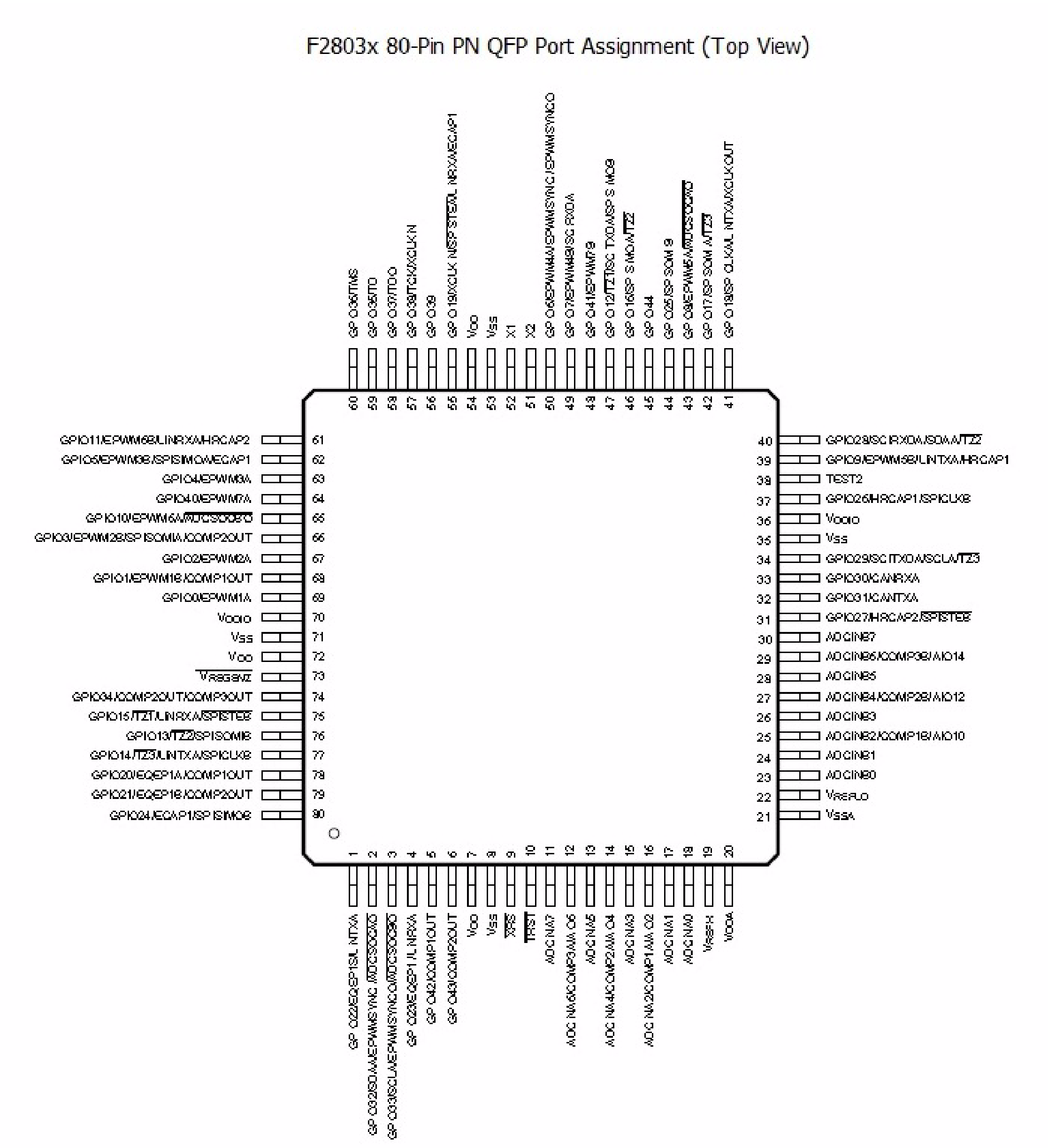
The definitions of the elements in the TI F2803x Hardware Target library are described in this Chapter.

F2803x 56-Pin RSH VQFN Port Assignment (Top View)



F2803x 64-Pin PAG TQFP Port Assignment (Top View)





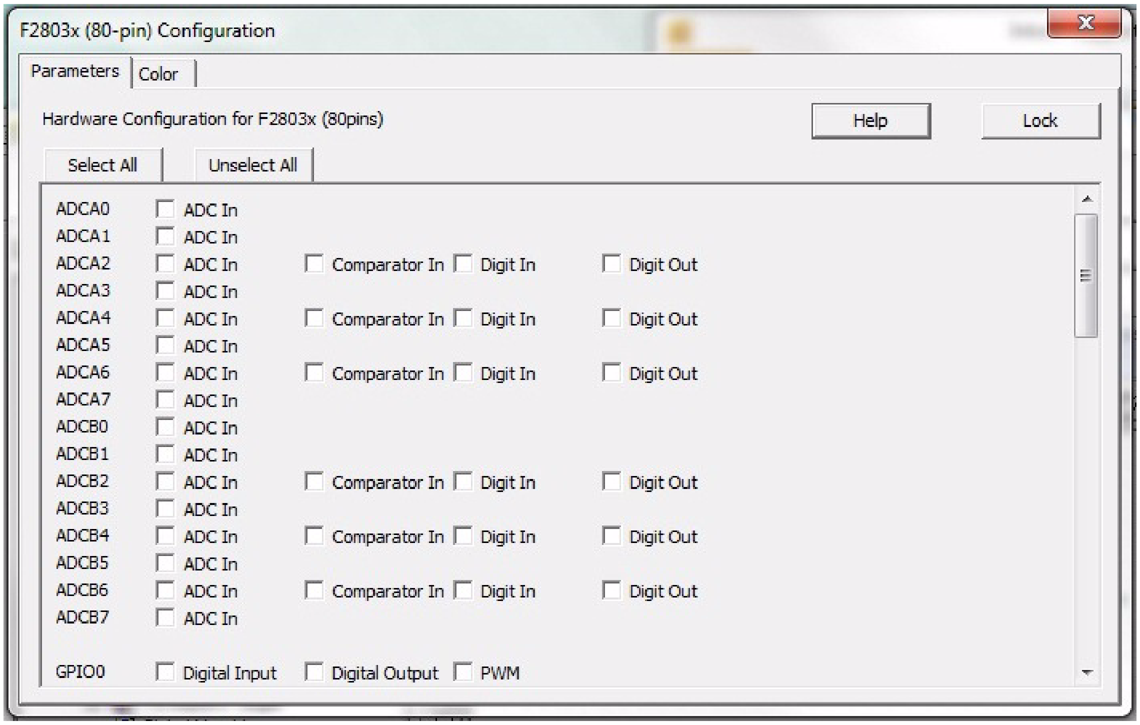
6.2 Hardware Configuration

The F2803x DSP provides a 16 channel analog inputs and up to 45 individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO1 can be used either as a digital input, a digital output, or a PWM output. Some of the 16 A/D channels can also be defined as either digital input, digital output, or comparator input. Therefore, user must assign the functions to the GPIO ports correctly according to the PSIM circuit schematic.

Image:

Hardware
Config
(80-pin)
F2803x

The dialog window of the block is shown below:



F2803x (80-pin) Configuration

Parameters | Color

Hardware Configuration for F2803x (80pins)

Help | Lock

Select All | Unselect All

ADCA0	□ ADC In			
ADCA1	□ ADC In			
ADCA2	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCA3	□ ADC In			
ADCA4	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCA5	□ ADC In			
ADCA6	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCA7	□ ADC In			
ADCB0	□ ADC In			
ADCB1	□ ADC In			
ADCB2	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCB3	□ ADC In			
ADCB4	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCB5	□ ADC In			
ADCB6	□ ADC In	□ Comparator In	□ Digit In	□ Digit Out
ADCB7	□ ADC In			
GPIO0	□ Digital Input	□ Digital Output	□ PWM	

The Hardware Configuration block is for user to specify the I/O ports of the F2803x hardware. Every port to be used must be assigned correctly. The ports not in use can be left unchecked.

For each GPIO port, a check box is provided for each of its available function. When a box is checked, the GPIO port is configured for that particular function. For example, if the checkbox for "Digital Input" is checked for port GPIO1, this port is configured as a digital input, and hence, cannot be used for any other functions. If it is used as a PWM output in the PSIM circuit schematic, an error message will be generated.

6.3 DSP Clock

The DSP Configuration block defines the external clock frequency and the speed of the F2803x DSP, as well as the program space size.

Image:

DSP
Clock
F2803x

Attributes:

Parameters	Description
DSP Clock Source	There are five ways of providing system clock to F2803x. They are as follows: - Internal oscillator 1 - Internal oscillator 2 - External oscillator - External clock (GPIO19) - External clock (GPIO38)
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 10 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 60 MHz.

If a DSP Configuration block is not used in a circuit, the default values of the DSP Configuration block are used.

6.4 PWM Generators

The F2803x DSP contains 7 sets of PWM modules. Each set of PWM module has two output ports: PWM 1 (GPIO0 and GPIO1), PWM 2 (GPIO2 and GPIO3), PWM 3 (GPIO4 and GPIO5), PWM 4 (GPIO6 and GPIO7), PWM 5 (GPIO8 and GPIO9), PWM 6 (GPIO10 and GPIO11), and PWM 7 (GPIO40 and GPIO41).

The two outputs of each PWM module usually are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM module is set in one of a few special operation modes.

In SimCoder, these 7 PWM's can be used in the following ways:

- Two 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);
- Seven 2-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, and 7, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, and 7, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, 4, 5, and 6, with two outputs complementary to each other.

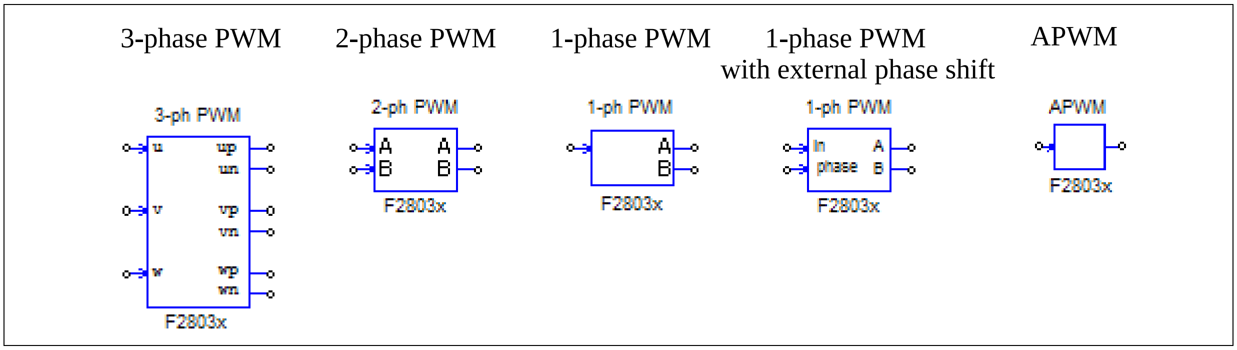
These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 APWM generators that use the same resources as the captures. These PWM generators have restricted functionality as compared to the 6 PWM generators (PWM 1 to 6) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

The images and parameters of the PWM generators are shown below.

Images:



6.4.1 3-Phase PWM Generator

In the 3-phase PWM generator image, "*u*", "*v*", and "*w*" refer to the three phases (alternatively they are called Phase "*a*", "*b*", and "*c*"). The letter "*p*" refers to the positive output, and "*n*" refers to the negative output. For example, for 3-phase PWM 123, "*up*" is PWM1A, and "*un*" is PWM1B.

Attributes:

Parameters	**Description**
PWM Source	Source of the PWM generator. It can be either "3-phase PWM 123" that uses PWM 1 to 3, or "3-phase PWM 456" that uses PWM 4 to 6.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Triangular* (*start low*): Triangular wave, and the initial PWM output state is low. - *Triangular* (*start high*): Triangular wave, and the initial output state is high. - *Sawtooth* (*start low*): Sawtooth wave, and the initial output state is low. - *Sawtooth* (*start high*): Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC*: PWM will trigger A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.

Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
DC Trip Source	Specify the digital compare (DC) trip sources. Each PWM channel may have up to two DC trip sources. For PWM channel A, the two sources are DCAH and DCAL. For PWM channel B, the two sources are DCBH and DCBL. Each trip source can be one of the followings - Do not use: PWM doesn't use this signal. - Trip-zone 1: PWM uses trip-zone 1 as digital compare input signal. - Trip-zone 2: PWM uses trip-zone 2 as digital compare input signal. - Trip-zone 3: PWM uses trip-zone 3 as digital compare input signal. - Comparator 1: PWM uses Comparator 1 as digital compare input signal. - Comparator 2: PWM uses Comparator 2 as digital compare input signal. - Comparator 3: PWM uses Comparator 3 as digital compare input signal.
DC Trip Type	Define how to use the DC trip signal. PWM may uses it as an one-shot signal or as a cycle-by-cycle signal. The active level of the DC trip signal can be selected from the followings. - *Do not use*: PWM doesn't use this signal. - *Trip source 1 (or 2) is low*: PWM is tripped if the source signal is low. - *Trip source 1 (or 2) is high*: PWM is tripped if the source signal is high. - *Trip source 1 low & source 2 high*: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - High impedance: The PWM outputs are in high impedance. - PWM A high & B low: The PWM positive output is high, and the negative output is low. - PWM A low & B high: The PWM positive output is low, and the negative output is high. - No action: No action is taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u, v, w	Initial value of 3-phase inputs *u*, *v*, and *w*
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

6.4.2 1-Phase PWM Generator

The attributes of 1-phase PWM generators with internal phase shift are the same as the ones with the external phase shift. The difference is that the 1-phase PWM with external phase shift has one extra input for the phase shift (labeled as "phase"). The phase shift unit is in degree, the same as the 1-phase PWM without external phase shift.

Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 6. With phase shift, it can be PWM 2 to PWM 6.
Output Mode	The output mode of the PWM generator. It can be one of the following: - *Use PWM A&B*: Both PWM outputs A and B are used, and they are complementary. - *Use PWM A*: Only PWM output A is used. - *Use PWM B*: Only PWM output B is used.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Triangular* (*start low*): Triangular wave, and the initial PWM output state is low. - *Triangular* (*start high*): Triangular wave, and the initial output state is high. - *Sawtooth* (*start low*): Sawtooth wave, and the initial output state is low. - *Sawtooth* (*start high*): Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC Group A*: PWM will trigger Group A of the A/D converter. - *Trigger ADC Group B*: PWM will trigger Group B of the A/D converter. - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

DC Trip Source	Specify the digital compare (DC) trip sources. Each PWM channel may have up to two DC trip sources. For PWM channel A, the two sources are DCAH and DCAL. For PWM channel B, the two sources are DCBH and DCBL. Each trip source can be one of the followings - Do not use: PWM doesn't use this signal. - Trip-zone 1: PWM uses trip-zone 1 as digital compare input signal. - Trip-zone 2: PWM uses trip-zone 2 as digital compare input signal. - Trip-zone 3: PWM uses trip-zone 3 as digital compare input signal. - Comparator 1: PWM uses Comparator 1 as digital compare input signal. - Comparator 2: PWM uses Comparator 2 as digital compare input signal. - Comparator 3: PWM uses Comparator 3 as digital compare input signal.
DC Trip Type	Define how to use the DC trip signal. PWM may uses it as an one-shot signal or as a cycle-by-cycle signal. The active level of the DC trip signal can be selected from the followings. - *Do not use*: PWM doesn't use this signal. - *Trip source 1 (or 2) is low*: PWM is tripped if the source signal is low. - *Trip source 1 (or 2) is high*: PWM is tripped if the source signal is high. - *Trip source 1 low & source 2 high*: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - *High impedance*: The PWM outputs are in high impedance. - *PWM A high & B low*: The PWM positive output is high, and the negative output is low. - *PWM A low & B high*: The PWM positive output is low, and the negative output is high. - *No action*: No action is taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. (for 1-phase PWM generator without external phase shift input)
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

Phase Shift

A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. For F2803x DPS, the PWM phase shift must follow the rules described below.

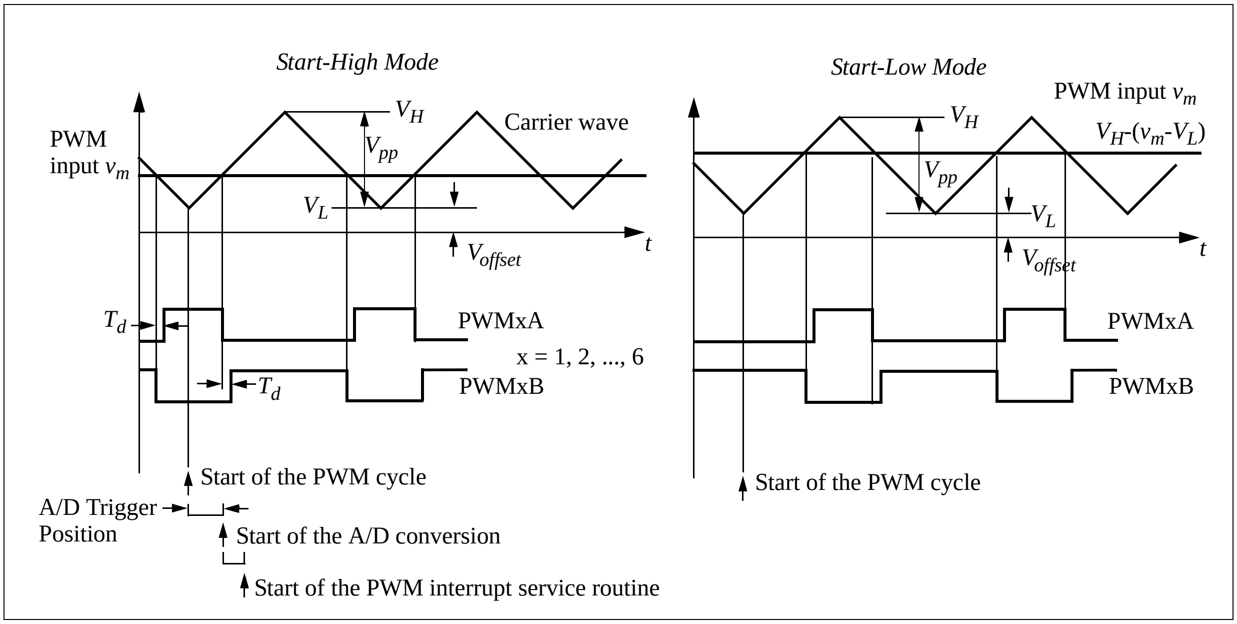
- Any PWM can be phase shifted only if the previous PWM is also set as phase shift. For example, PWM1, 2 and 3 is a correct series for phase shift, but PWM 1, 3, 4 is not a correct series for phase shift.
- The reference PWM and the PWM being shifted must be consecutive in the series. That is, it is not permitted to use PWM 1 as the reference, and phase shift PWM 3, or PWM 5 or 6.

The phase shift value is in degrees. When the value is -30^{o}, the output will be shifted to the right (lagging) by 30^{o} of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30^{o}. When the phase value is 30^{o}, the output will be shifted to the left (leading) by 30^{o}.

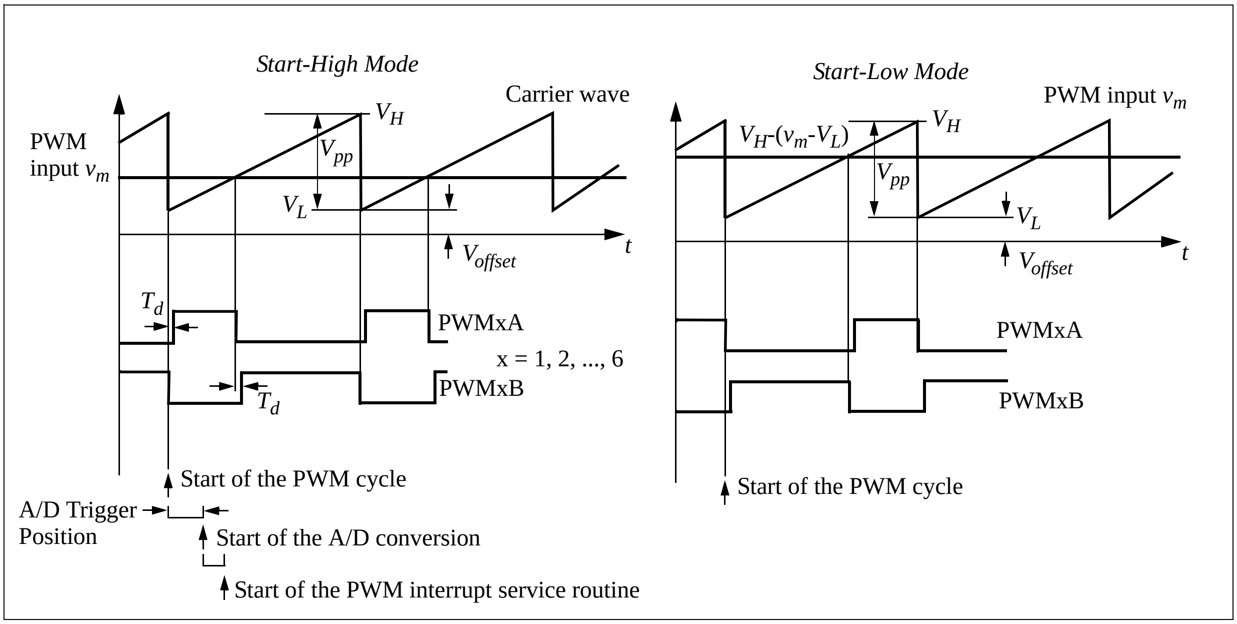
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below.

The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m, and the lowest value of the carrier wave is V_L and the highest value is V_H. In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, V_L=0, and V_H=1. If v_m=0.2, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value V_H-(v_m-V_L). For example, for a carrier wave from 0 to 1, V_L=0, and V_H=1. If v_m=0.2, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

Note: In the start-low mode, the PWM input v_m is converted to V_H-(v_m-V_L) internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with V_L=0 and V_H=1, or for a triangular wave with V_L= -V_H, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

6.4.3 2-Phase PWM Generator

Attributes:

Parameters	**Description**
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 6.
Mode Type	The operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The scaling factor between the PWM frequency and the sampling frequency. It can be 1, 2, or 3. That is, the PWM frequency (the frequency of the PWM output signals used to control switches) can be multiples of the sampling frequency. For example, if the sampling frequency is 50 kHz and the scaling factor is 2, it means that the PWM frequency is 100 kHz. Switches will switch at 100 kHz, but the gating signals are updated once per two switching cycles at 50 kHz.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter. - *Trigger ADC Group A*: PWM will trigger Group A of the A/D converter. - *Trigger ADC Group B*: PWM will trigger Group B of the A/D converter. - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180^o position of the PWM cycle.
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - *Cycle by cycle*: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

DC Trip Source	Specify the digital compare (DC) trip sources. Each PWM channel may have up to two DC trip sources. For PWM channel A, the two sources are DCAH and DCAL. For PWM channel B, the two sources are DCBH and DCBL. Each trip source can be one of the followings - Do not use: PWM doesn't use this signal. - Trip-zone 1: PWM uses trip-zone 1 as digital compare input signal. - Trip-zone 2: PWM uses trip-zone 2 as digital compare input signal. - Trip-zone 3: PWM uses trip-zone 3 as digital compare input signal. - Comparator 1: PWM uses Comparator 1 as digital compare input signal. - Comparator 2: PWM uses Comparator 2 as digital compare input signal. - Comparator 3: PWM uses Comparator 3 as digital compare input signal.
DC Trip Type	Define how to use the DC trip signal. PWM may uses it as an one-shot signal or as a cycle-by-cycle signal. The active level of the DC trip signal can be selected from the followings. - *Do not use*: PWM doesn't use this signal. - *Trip source 1 (or 2) is low*: PWM is tripped if the source signal is low. - *Trip source 1 (or 2) is high*: PWM is tripped if the source signal is high. - *Trip source 1 low & source 2 high*: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - *High impedance*: The PWM outputs are in high impedance. - *PWM A high & B low*: The positive output of the PWM is high, and the negative output is low. - *PWM A low & B high*: The positive output of the PWM is low, and the negative output is high. - *No action*: No action is taken.
Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

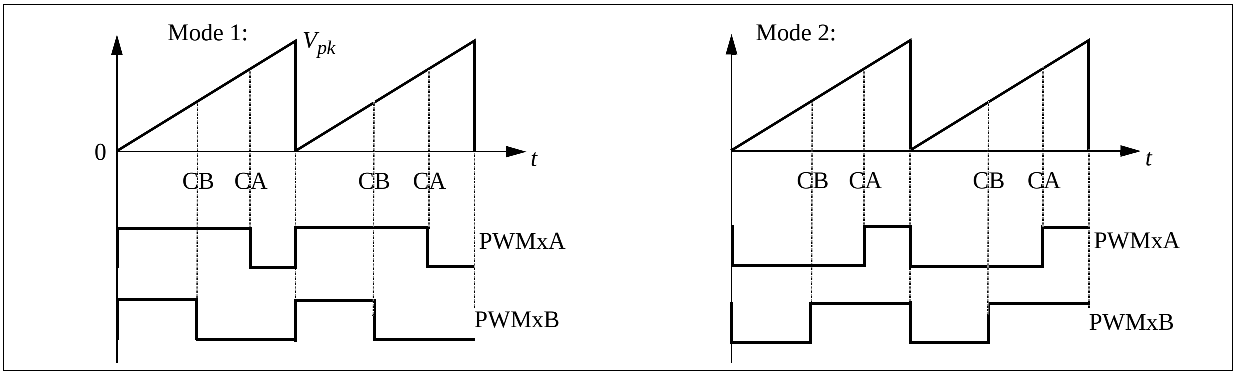
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk}, and there is no dc offset.

<u>Operation Mode 1:</u>

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

<u>Operation Mode 2:</u>

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

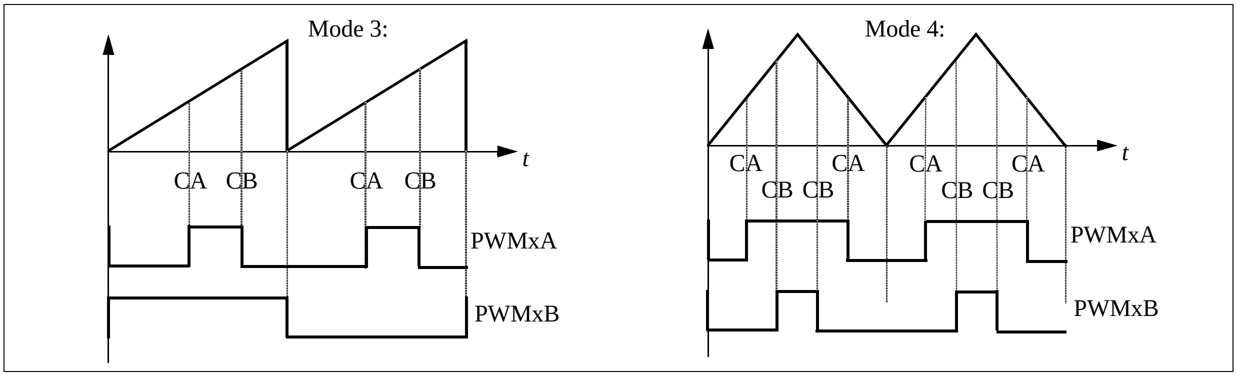


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

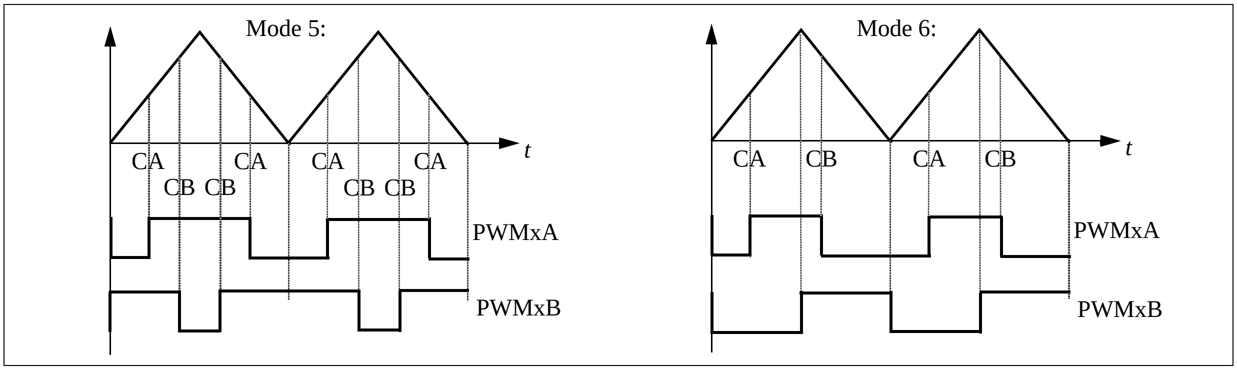


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

The figure below on the right shows the waveforms of Mode 6. In this mode, Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



6.4.4 APWM Generator:

Attributes:

Parameters	Description
PWM Source	APWM generators share the same resource as captures. The PWM source can be from one of the three designated GPIO ports: - *APWM 1* (*GPIO5, GPIO24, GPIO34*)
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - *Sawtooth* (*start low*): Sawtooth wave, with the PWM output in the low state initially. - *Sawtooth* (*start high*): Sawtooth wave, with the PWM output in the high state initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: - *Output low*: The PWM output will be set to low. - *Output high*: The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

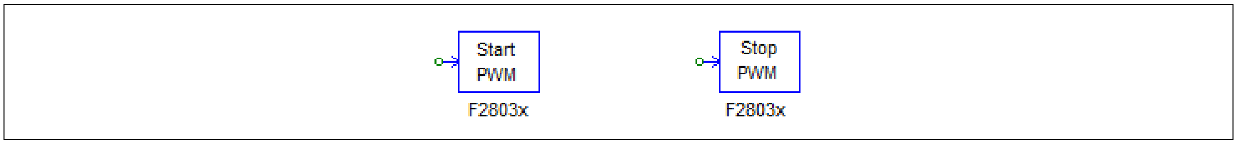
Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The phase shifting rules are also the same as the 1-phase PWM generators.

As noted before, the APWM generators has reduced number of functions than 1-phase PWM generators. It can not trigger the A/D converter and can not use the trip-zone signal.

6.5 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Images:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, 3-phase PWM 123 and PWM 456, and Capture 1.

6.6 Trip-Zone and Trip-Zone State

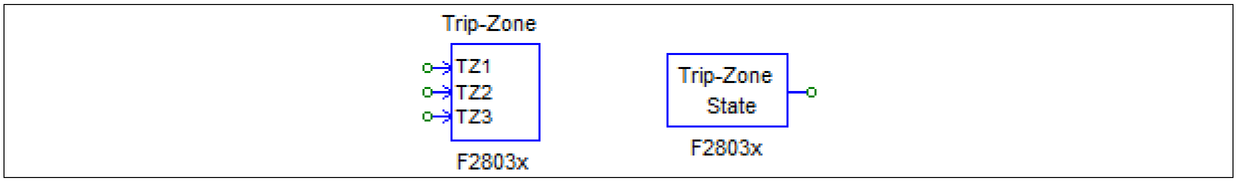
F2803x DSP contains 6 trip-zone input signals, 3 from GPIO ports and 3 from Comparators. Comparators can only be used in Digital Compare to trip PWM

Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0). The trip-zone signals through Digital Compare trigger a trip action in specified level (either high or low)

Images:



Attributes for Trip-Zone:

Parameters	Description
Use Trip-Zone i	Specify if this Trip-Zone i is used.
GPIO Port for Trip-Zone i	Specify a designated GPIO port as Trip-Zone input signal. - GPIO port for trip-zone 1: select either GPIO12 or GPIO13 - GPIO port for trip-zone 2: select either GPIO13, GPIO16, or GPIO18 - GPIO port for trip-zone 3: select either GPIO14, GPIO17, or GPIO19
Use Comparator i	Specify if comparator 1, 2, or 3 is used as Trip-Zone input signal i

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, and 3-phase PWM 123 and PWM 456.

The trip-zone interrupt can be generated in either one-shot mode of cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

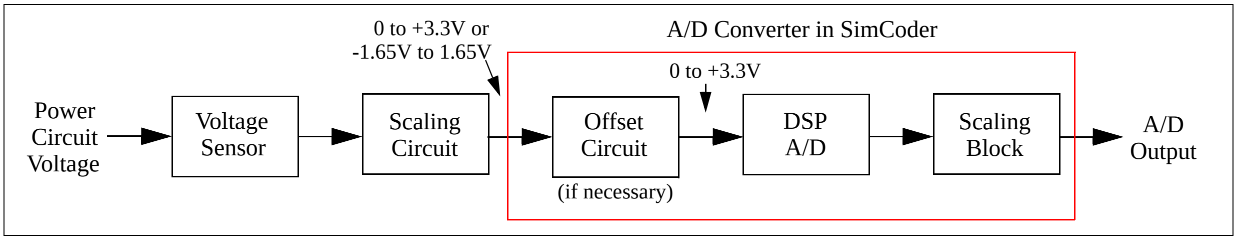
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

6.7 A/D Converter

The F2803x DSP provides a 16-channel 12-bit A/D converter.

Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the range of 0V and +3.3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.

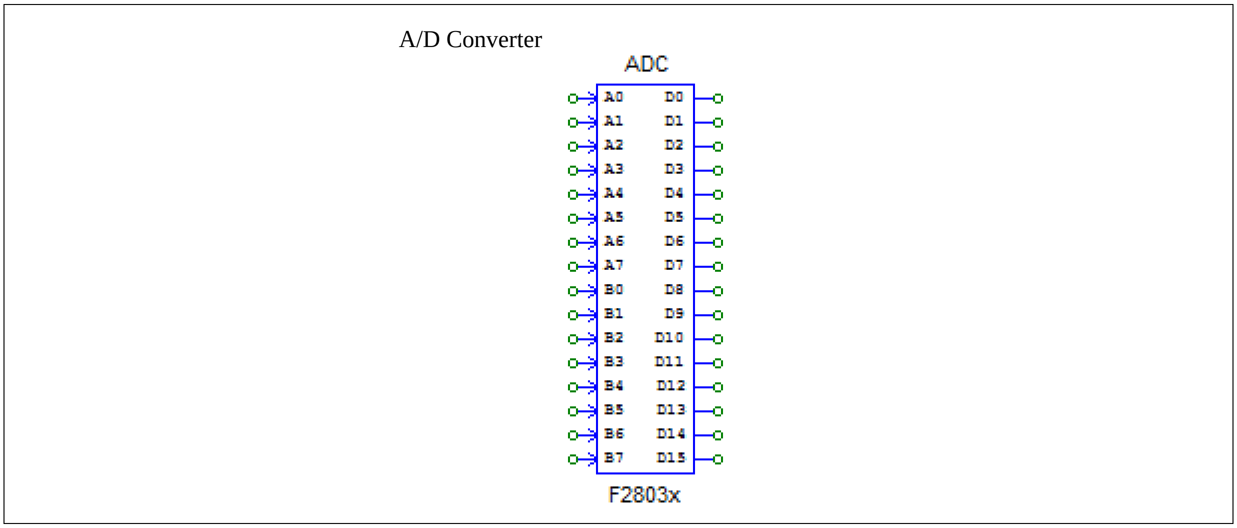


As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications. It will be further explained in Section 6.5.3.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.3V. SimCoder's A/D converter for F2833x DSP provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

The image and the parameters of the A/D converter in the SimCoder library are described below. In the following description, "A/D converter" refers to the A/D converter in the SimCoder library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	**Description**
Ch A_i *or* B_i Mode	Input mode of the A/D converter channel A_i or B_i, where *i* is from 0 to 7. The input mode can be one of the following: - *AC*: The input range is considered from -1.65V to +1.65V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3.3V range. - *DC*: The input is a dc value, and the range is from 0 to +3.3V.
Ch A_i *or* B_i Gain	Gain *k* of the A/D converter channel A_i (or B_i), where *i* is from 0 to 7.

Trigger Source:

The A/D converter for F2803x can be triggered from multiple sources. Multiple A/D channels may share the same trigger source. Each A/D Channel can be triggered by

- One of the PWM generators,
- Timer1 or by Timer2.
- More than one trigger source.

In a schematic, if a A/D channel is not associated with a PWM, user should insert a ZOH block at the output of the A/D channel so that the SimCoder will select the timer as trigger source.

It is not permitted to have a A/D converter channel triggered by one source, but its output signal is used in a circuit section that has a different sampling rate.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input port of the A/D converter.

Input Offset and Scaling:

The input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input mode is DC, the maximum input voltage be scaled to +3.3V; and when the input mode is AC, the peak voltage be scaled to +/-1.65V.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

$$V_{i\_min} = 0 \text{ V}, V_{i\_max} = 150 \text{ V}$$

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3V. Assume that the actual value of the voltage at a certain point is:

$$V_i = 100 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i\_max\_s} = 150 *0.01 = 1.5 \text{ V}$$

$$V_{i\_s} = 100 *0.01 = 1 \text{ V}$$

To utilize the full range of the DSP, a conditioning circuit with a gain of 2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: 0.01*2 = 0.02. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$V_{i\_max\_s\_c}$ = 1.5 * 2 = 3 V

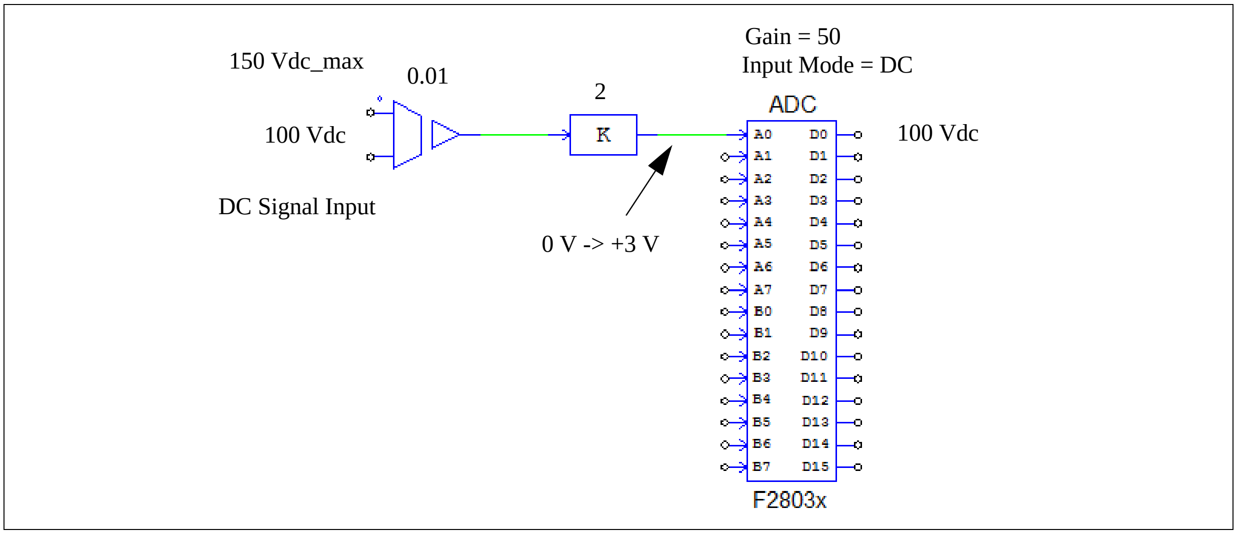
$V_{i\_s\_c}$ = 1 * 2 = 2 V

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

V_{o_max} = 50 * 3 = 150 V

V_o = 50 * 2 = 100 V

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.



Please note that, in this example, if the gain of the proportional block is changed from 2 to 1, and the A/D gain is changed from 50 to 100, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3V.

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

V_{i_max} = +/- 75 V

The input mode of the A/D converter will be set to ac, and the input range is from -1.5V to +1.5V. Assume that the actual value of the voltage has a peak value of:

V_i = +/- 50 V

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$V_{i\_max\_s}$ = +/- 0.75 V

V_{i_s} = +/- 0.5 V

Since the A/D converter input range is from -1.5V to +1.5V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2 is needed (i.e. 1.5/0.75 = 2). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$V_{i\_max\_s\_c}$ = +/- 1.5 V

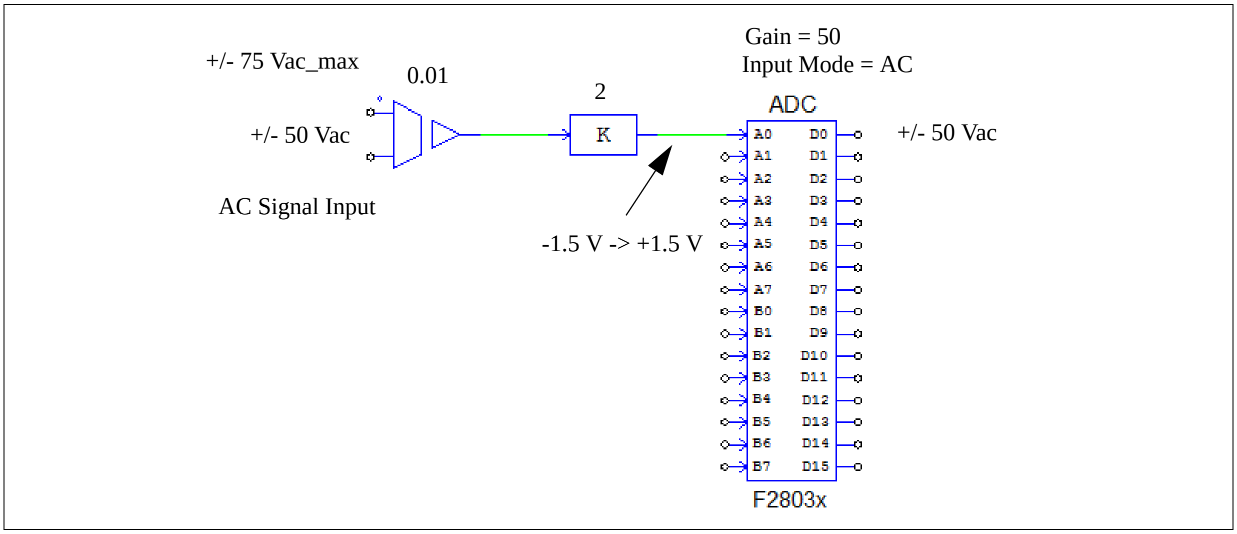
$V_{i\_s\_c}$ = +/- 1 V

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

V_{o_max} = +/- 75 V

V_o = +/- 50 V

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.

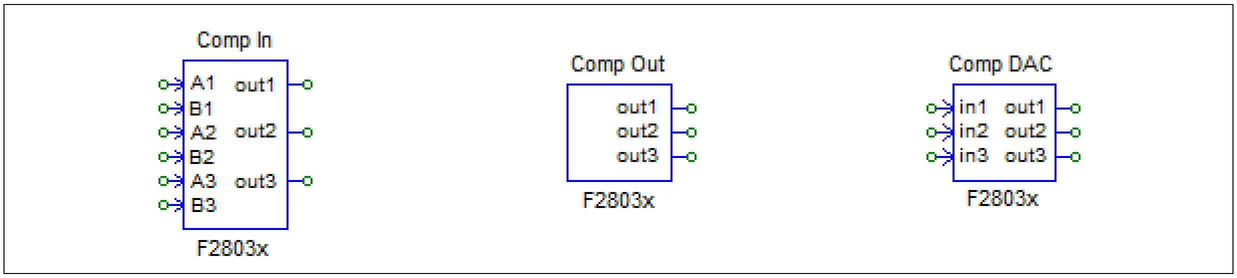


Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to AC, the input range is from -1.65V and +1.65V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal must be scaled and offset so that the range is within 0V to +3.3V required by the DSP A/D converter.

6.8 Comparator

The F2803x DSP support three comparator modules. Each comparator block can accommodate two external analog inputs, or use one external analog input and use the internal DAC reference for the other input. The comparator output can be sent to the PWM trip-zone and to the GPIO output.

Images:



6.8.1 Comparator Input

In F2803x, there are 3 Comparators.The 3 pairs of inputs share the same ADC/AIO ports with ADC input channels.

- Comparator 1 input A: from port ADCA2 which is also AIO2
- Comparator 1 input B: from port ADCB2 which is also AIO10
- Comparator 2 input A: from port ADCA4 which is also AIO4
- Comparator 2 input B: from port ADCB4 which is also AIO12
- Comparator 3 input A: from port ADCA6 which is also AIO6
- Comparator 3 input B: from port ADCB6 which is also AIO14

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is used as a A/D converter channel or AIO in the same PSIM circuit schematic.

Attributes

Parameters	**Description**
Comparator *i*	Define how the comparator output to be used. It can be one of the followings: - Do not use - As a normal comparator - As a Trip-Zone signal
Output Logic *i*	Indicate the output logic. High when A > B High when A < B
Compare Method *i*	Select the source to be compared with the input-A of the comparator. - Compare to Input-B - Compare to Constant Value - Compare to DAC output
Constant Value *i*	Valid when choosing 'Compare to Constant Value' in Compare Method *i*

For any comparator, its input-A is always from external analog input. If input-B is another external analog input, the corresponding ADC/AIO port must be defined as a comparator input in the F2803x Configuration block. If the method is to compare to a constant value, input-B can be left unconnected. The corresponding ADC/AIO port can be used for other functions. If the method is to compare to DAC output, input-B should be connected to a Comparator DAC output. The corresponding ADC/AIO port can be used for other functions.

6.8.2 Comparator Output

The output of the comparator can be used as PWM trip-zone signal, as well as the GPIO output.

Attributes:

Parameters	**Description**
Comparator *i* Output	Port position of the comparator output i, where i ranges from 1 to 3. If the comparator is in use: - The output port of Comparator A can be GPIO1, GPIO20, or GPIO42. - The output port of Comparator B can be GPIO3, GPIO21, GPIO34, or GPIO43. - The output port of Comparator C can be GPIO34.

If the comparator output is enabled, be sure its corresponding comparator input channel is used in the schematic. Otherwise an error message will be reported.

6.8.3 Comparator DAC

Each comparator block in F2803x contains a 10-bit DAC reference that can supply the inverter input (B side input) of the comparator.

Attributes:

Parameters	**Description**
DAC *i* Range	Indicates the input signal range of comparator DAC i. The lower limit is 0.

The DAC output is scaled as 0 to 3.3V. The output of DAC can be calculated as following:

DAC output = DAC input * 3.3 / DAC range

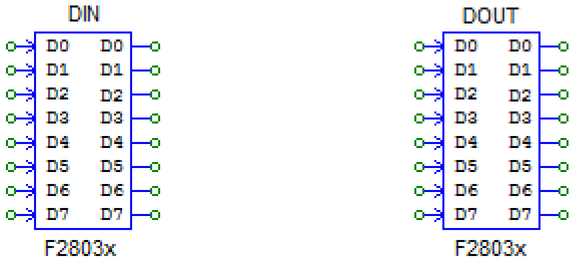
When using Comparator DAC, the corresponding Comparator Input must be enabled in the F2803x Hardware Configuration block or an error message will be reported.

6.9 Digital Input and Digital Output

The F2803x DSP support 45 general-purpose-input-output (GPIO 0 to 44)) ports that can be configured as either digital inputs or digital outputs. In addition, there are 6 analog-input-output ports (AIO 2, 4, 6, 10, 12, and 14) which also can be used as digital inputs and outputs.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/output blocks can be used in the same schematic.

Images:



Attributes for Digital Input:

Parameters	**Description**
Port Position for Input *i*	The port position of the Input *i*, where *i* is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	**Description**
Port Position for Output *i*	The port position of the Output *i*, where *i* is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.

Note that each GPIO and AIO port can be used for one function only. If an IO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

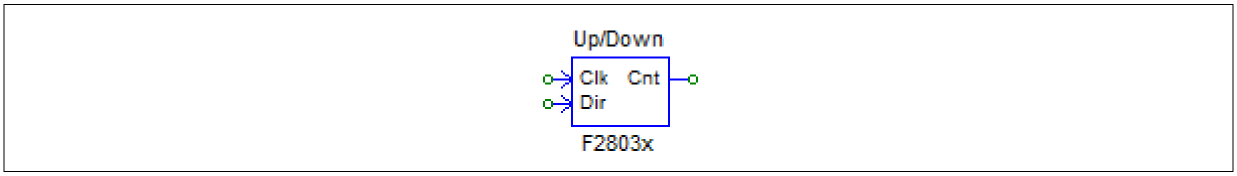
F2803x DSP supports 3 masked external interrupts (XINT1 to XINT3). There are no dedicated pins for the external interrupts. XINT1, XINT2, and Xint3 interrupts can accept inputs from GPIO0 to PGPIO31 pins.

6.10 Up/Down Counter

The F2803x DSP supports one up/down counter. The input ports are:

- GPIO20 for clock
- GPIO21 for direction

Image:



Attributes:

In the image, "Clk" refers to the input clock signal, and "Dir" refers to the signal that defines the counting direction. When the "Dir" input is 1, the counter counts forward, and when the input is 0, the counter counts backward.

The output of the up/down counter gives the counter value.

Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and in an encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

6.11 Encoder and Encoder State

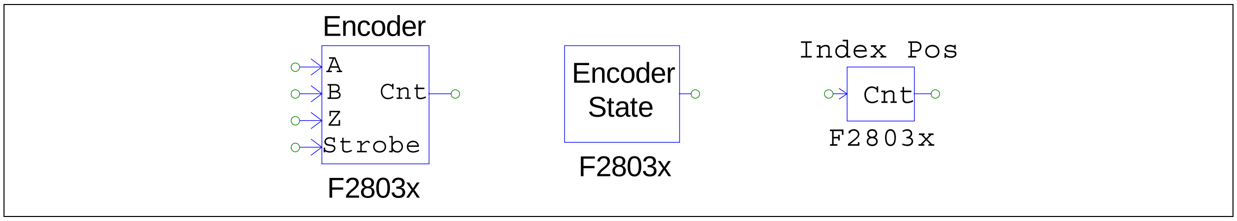
The F2803x DSP supports an **Encoder** module. The GPIO ports used by encoder are:

- GPIO20 for clock input
- GPIO21 for direction input
- GPIO22 for Z (or index signal) input
- GPIO23 for strobe signal input

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder’s intial position. When the input of this block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Images:



Attributes for Encoder:

Parameters	**Description**
Use Z Signal	Define if the encoder uses the Z (or index) signal.
Use Strobe Signal	Define if the encoder uses the strobe signal.

Counting Direction	The counting direction - Forward: the encoder counts up. - Reverse: the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

The **Attributes** for Encoder Index/Strobe Pos:

Parameters	**Description**
Latch Position	Specify the kept counter type, choose from the followings: - IndexPos, if the setting "Use Z Siganl" is not "No" in Encoder - StrobePos, if the setting "Use Strobe Siganl" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - The first latched position, or - The current latched position

6.12 Capture and Capture State

The F2803x DSP contains an enhanced capture module. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Images:

Capture
F2803x
Capture State
F2803x

Attributes for Capture:

Parameters	**Description**
Capture Source	Source of the capture may come from one of 3 GPIO ports, as listed below: - Capture 1 (GPIO5) - Capture 1 (GPIO19) - Capture 1 (GPIO24)
Event Filter	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either *Absolute time* or *Time difference*.

Attributes for Capture State:

Parameters	**Description**
Capture Source	Source of the capture. It has only one source *Capture1*.

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

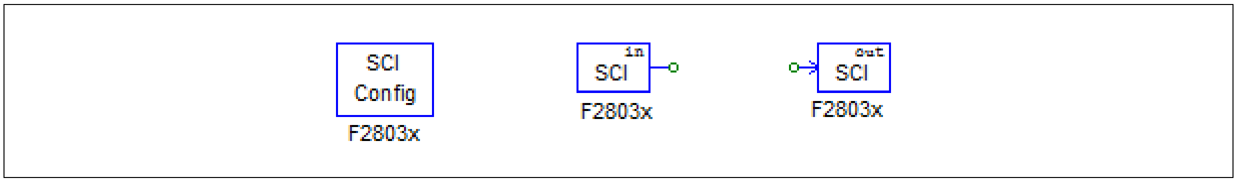
6.13 Serial Communication Interface (SCI)

The F2803x DSP provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

For more detailed descriptions on SCI and the monitoring function, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring in TI F28335 Target.pdf*".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

Images:



6.13.1 SCI Configuration

The SCI Configuration block defines the SCI port, the communication speed, the parity check type, and the data buffer size.

Attributes:

Parameters	Description
SCI Port	Define the SCI port. There are 7 sets of GPIO ports that can be used for SCI, as listed below: - *SCIA* (*GPIO28*, *GPIO29*) - *SCIA* (*GPIO7*, *GPIO12*)
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either *None*, *Odd*, or *Even*.
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

For more information on how to select the buffer size, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring in TI F28335 Target.pdf*".

6.13.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

In the schematic, the SCI input behaviors as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

6.13.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when then DSP Oscilloscope is in the *continuous* mode. When it is in the *snap-shot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaviors as a voltage probe.

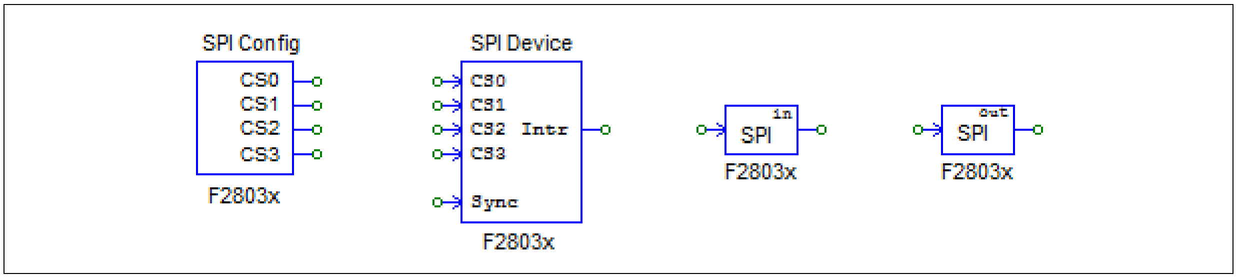
6.14 Serial Peripheral Interface (SPI)

The F2803x DSP provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F803x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for Real-Time Monitoring in TI F28335.pdf*".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

Images:



6.14.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Attributes:

Parameters	Description
SPI Port	Define the SPI port from the options: - SPIA (GPIO 16-19) - SPIA (GPIO 3, 5, 18, 19) - SPIB (GPIO 12-15) - SPIB (GPIO 24-27)
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/ Output elements.

6.14.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.
Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: - *Rising edge without delay*: The clock is normally low, and data is latched at the clock rising edge. - *Rising edge with delay*: The clock is normally low, and data is latched at the clock rising edge with delay. - *Falling edge without delay*: The clock is normally high, and data is latched at the clock falling edge. - *Falling edge with delay*: The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either *Rising edge* or *Falling edge*.
SPI Initial Command	The SPI command that initializes the SPI device.

Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: - *No hardware interrupt* - *Rising edge* - *Falling edge*
Interrupt Timing	Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following: - *No interrupt*: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - *Multiple interrupt in series*: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - *One-time interrupt*: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

6.14.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: *ElementName* = {*Xn*[*MSB..LSB*]} where - *ElementName* is the name of the SPI input device. If it is the current SPI input device, use *y* instead. - {} means that the item in the bracket repeats multiple times. - *Xn* is the n_{th} word received from the SPI input device, and *n* start from 0. - *MSB..LSB* defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max}. If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale}. If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, y=x1[3..0]x2[7..0], means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \dfrac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data\_\text{Length}}}$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \dfrac{(Result - 2^{Data\_\text{Length-1}}) \cdot V_{max} \cdot K_{scale}}{2^{Data\_\text{Length-1}}}$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

6.14.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale}. If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max}. If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either *DC* or *AC*. Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: *ElementName* = {*Xn*[*MSB..LSB*]} where - *ElementName* is the name of the SPI output device. If it is the current SPI output device, use *y* instead. - {} means that the item in the bracket repeats multiple times. - *Xn* is the n_{th} word sent to the SPI output device, and *n* start from 0. - *MSB..LSB* defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, y=x1[3..0]x2[7..0], means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $$Output = \frac{Result \cdot K_{scale} \cdot 2^{Data\_\mathrm{Length}}}{V_{max}}$$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $$Output = 2^{Data\_\mathrm{Length}} + \frac{Result \cdot K_{scale} \cdot 2^{Data\_\mathrm{Length-1}}}{V_{max}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

6.15 Project Settings and Memory Allocation

When generating the code for the TI F2803x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio development environment, so that the code can be compiled, linked, and uploaded to the DSP.

At the present, the Code Composer Studio version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2803x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- DSP2803x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F28035_FLASH_Lnk.cmd: Flash memory linker command file
- F28035_FLASH_RAM_Lnk.cmd:Flash RAM memory linker command file
- F28035_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F28035. For example, if the target hardware is F28034, the file names will be *F28034 FLASH Lnk.cmd*, *F28034 FLASH RAM Lnk.cmd*, and *F28034RAM Lnk.cmd* accordingly.

Besides, the project also needs the following files:

- PsBiosRamF03xFixpt.lib:SimCoder F2803x library, located in the PSIM folder
- PsBiosRomF03xFixpt.lib:SimCoder F2803x library, located in the PSIM folder
- IQmath.lib: TI's IQmath.lib, located in the PSIM /lib folder
- 2803x_IQmath_BootROMsymbols.libIQmath symbols library, located in the PSIM /lib folder

PsBiosRamF03xFixpt.lib, PsBiosRomF03xFixpt.lib, 2803x_IQmath_BootROMsymbols.lib and IQmath.lib will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pjt file (in this example, "test.c" and "test.pjt") will be created. If you have made changed manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When the RAM Debug or RAM Release setting is selected, Code Composer Studio uses the linker command file F2803x_RAM_Lnk.cmd to allocate the program and data space.

When the Flash Release setting is selected, Code Composer Studio uses the linker command file F2803x_FLASH_Lnk.cmd to allocate the program and data space.

When the Flash RAM Release setting is selected, Code Composer Studio uses the linker command file F2803x_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory
0x0000 - 0x07FF (2K)
interrupt vectors
stack
0x8000 - 0x9FFF (8K*)
program and data space

With the Flash Release setting:

RAM Memory	Flash Memory
0x0000 - 0x07FF (2K)	0x3E8000 - 0x3F7FFF (64K**)
interrupt vectors	program
stack	password
0x8000 - 0x9FFF (8K*)	etc.
data space	

Notes:

* The RAM memory predefined by SimCoder for program and data space is:

- For F28035, F28034, F28032, and F28032: from 0x8000 to 0x9FFF (8K)
- For F28031: from 0x8000 to 0x97FF (6K)
- For F28030: from 0x8000 to 0x8FFF (4K)
- If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.

** The flash memory predefined by SimCoder for program space is:

- For F28035 and F28034: from 0x3E8000 to 0x3F7FFF (64K)
- For F28033, F28032, and F28031: from 0x3F0000 to 0x3F7FFF (32K)
- For F28030: from 0x3F4000 to 0x3F7FFF (16K)

7

PE-Pro/F28335 Hardware Target

7.1 Overview

With the PE-Pro/F28335 Hardware Target, SimCoder can generate code for the PE-Pro/F28335 DSP hardware board, made by Myway Co. (www.myway.co.jp), that is based on Texas Instruments' F28335 floating-point DSP. The generated code uses Myway's PE-OS library, and it requires Myway's PE-View software to compile and upload to the DSP.

One major advantage of the PE-View environment is that it allows users to view waveforms of variables inside the DSP and change variable values in real time, thus making debugging and parameter adjustment very easy.

The PE-Pro/F28335 Hardware Target library includes the following function blocks:

- 3-phase regular and space vector PWM generators
- Start/Stop functions for PWM generators
- A/D converter
- Combo element of digital input / encoder / trip-zone
- Combo element of digital output / capture PWM

When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed, If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In PE-Pro/F28335, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 5.4 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the PE-Pro/F28335 Hardware Target library are described in this Chapter.

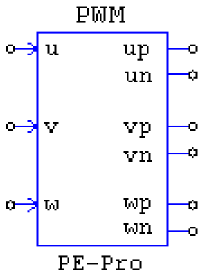
7.2 PWM Generators

Two types of PWM generators are provided in PE-Pro/F28335: 3-phase PWM generators (PWM 123 and PWM 456), and single capture PWM generator, also called APWM (APWM 5 and APWM 6)). The 3-phase PWM generator is described in this section, and the single capture PWM generator is described in Section 7.6.

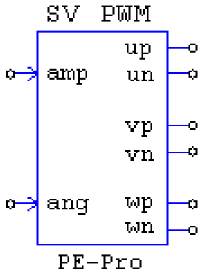
There are two kinds of 3-phase PWM generators: regular PWM generator, and space vector PWM generator. Their images and parameters are described below.

Images:

3-phase PWM



3-phase Space Vector PWM



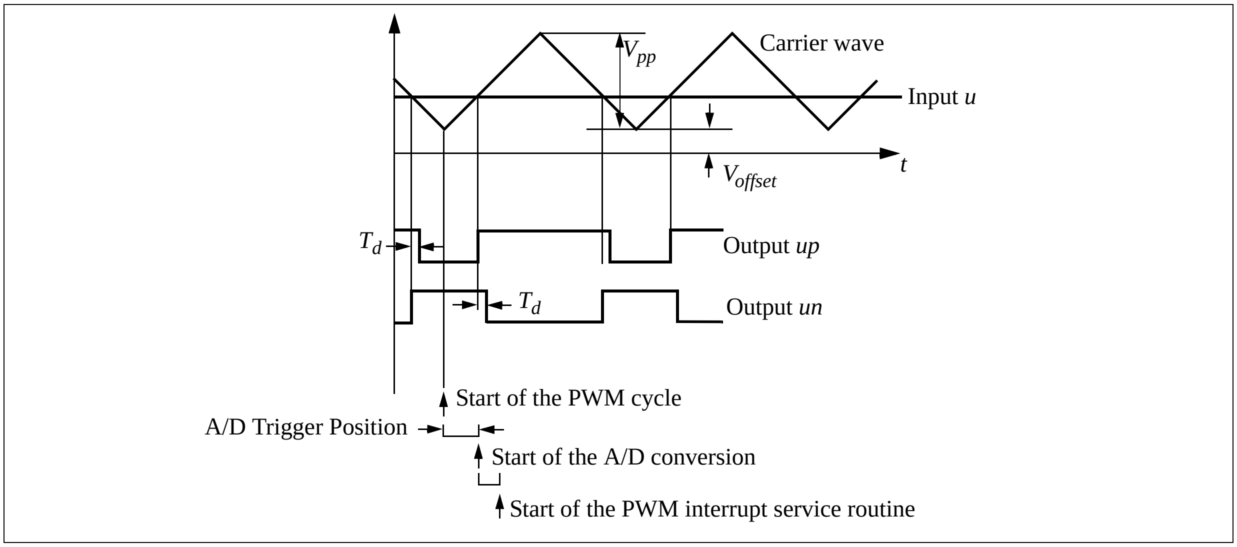
Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one of the following: - *3-phase PWM 123*: It consists of PWM 1, PWM 2, and PWM 3; - *3-phase PWM 456*: It consists of PWM 4, PWM 5, and PWM 6.
Dead Time	The dead time T_d for the PWM generator, in sec.
PWM Frequency	Frequency of the PWM generator, in Hz. For the PE-Pro/F28335 DSP board, the frequency must be no less than 1145 Hz.
Trigger ADC	The option whether for the PWM generator to trigger the A/D converter. It can be one of the following: - *Do not trigger ADC*: PWM does not trigger the A/D converter; - *Trigger ADC Group A&B*: PWM will trigger both Group A and B of the A/D converter (Channel 1 to 16).
ADC Trigger Position	The A/D trigger position ranges from 0 to 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 1, the A/D converter is triggered at the end of the PWM cycle.
Use Trip-Zone	Define whether the PWM generator uses the trip-zone signal (Trip-zone 1 and Trip-zone 2). It can be one of the following: - *Disable*: Disable the trip-zone signal; - *Enable*: The generator PWM 123 will use the trip-zone signal 1, and the generator PWM 456 will use the trip-zone signal 2, in the one-shot mode. Once triggered, the PWM must be started manually.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u / v / w *or* Initial Amplitude / Angle	Initial value of the 3-phase inputs *u*, *v*, and *w* (for regular PWM generator), or the amplitude and angle (for space vector PWM)
Start PWM at Beginning	It can be set to either *Start* or *Do not start*. When it is set to *Start*, PWM will start right from the beginning. If it is set to *Do not start*, one needs to start PWM using the "Start PWM" function.

In the image, "*u*", "*v*", and "*w*" refer to the three phases (alternatively they are called Phase "*a*", "*b*", and "*c*"). The letter "*p*" refers to the positive output, and "*n*" refers to the negative output. In the 3-phase space vector PWM generator image, "amp" refers to the amplitude input, and "ang" refers to the angle input.

For the regular PWM Generator, the PWM carrier waveform is triangular. The input and output waveforms of the regular PWM generator are shown below. In the figure, the outputs *up* and *un* correspond to the *up* and *un* terminals in the PWM generator image, and are the signals at the PE-Pro/F28335 DSP board. Note that when the input *u* is greater than the carrier wave, the output *up* is low. This is opposite to the definition of the PWM generator for the TI/F28335 Target. This is because the DSP output is inverted on the PE-Pro/F28335 DSP board before it is sent to the board output. One needs to be careful with this difference when working with both the TI/F28335 Target and PE-Pro/F28335 Target.

The space vector PWM generator generates PWM signals for a three-phase system based on space vector PWM technique. The input "amp" is for the space vector amplitude, and the range is from 0 to 1. The input "ang" is for the space vector phase angle in rad., and the range is from -2π to 4π.



The figure above also shows how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is complete, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

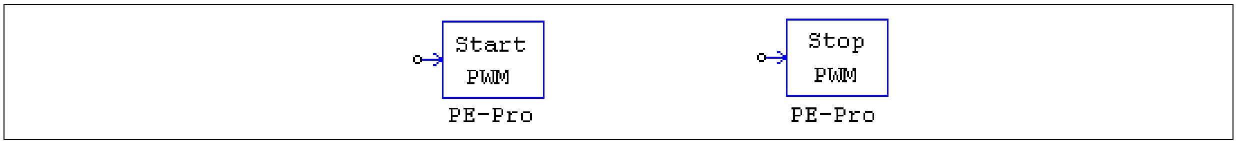
Interrupt can be generated by the PWM generator in two ways:

- *Periodic Interrupt*: The interrupt frequency is equal to the PWM frequency. It can be generated in the following ways:
  - If *Trigger ADC* is not selected, interrupt will be generated by the PWM generator at the beginning of the PWM carrier wave.
  - If *Trigger ADC* is selected, the PWM generator will trigger the A/D converter to start the conversion. After the A/D conversion is complete, interrupt will be generated, as shown in the figure above.
- *Trip-Zone Interrupt*: There are two trip-zone signals in PE-Pro/F28335. The generator PWM 123 uses trip-zone 1, and the generator PWM 456 uses trip-zone 2. When the trip-zone 1 signal changes from 0 to 1, a trip-zone interrupt will be generated in PWM 123. Similarly, when the trip-zone 2 signal changes from 1 to 0, a trip-zone interrupt will be generated in PWM 456. Before entering the trip-zone interrupt, all PWM outputs will be set to high impedance.

7.3 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Images:



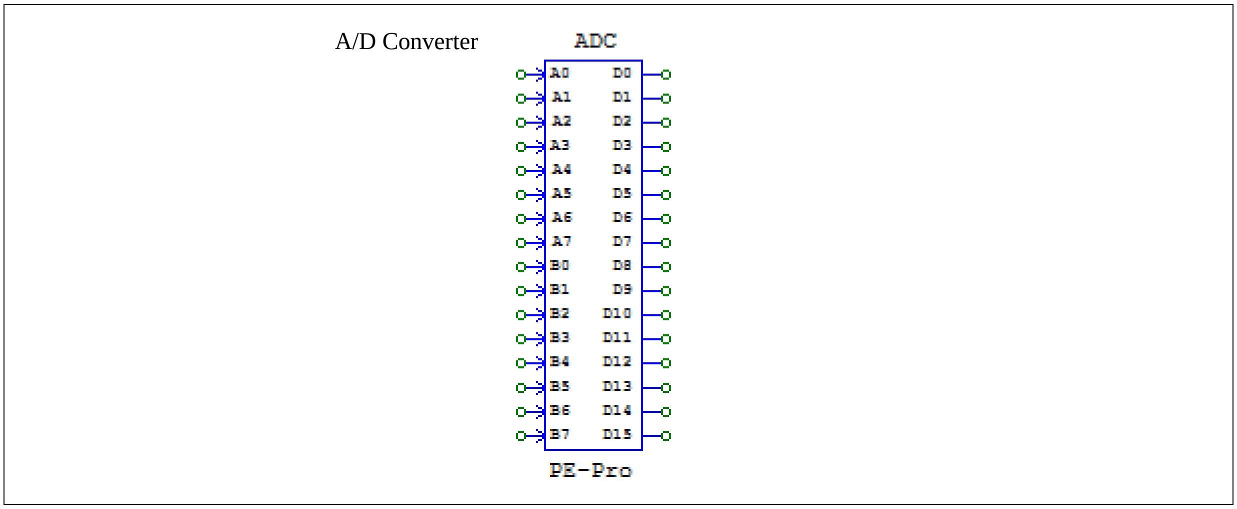
Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be one of the following: - *3-phase PWM 123* - *3-phase PWM 456* - *APWM 5* - *APWM 6*

7.4 A/D Converter

PE-Pro/F28335 provides a 12-bit 16-channel A/D converter. It is divided into two groups: Group A and Group B. The image and the parameters of the A/D converter are described below.

Image:



Attributes:

Parameters	Description
ADC Mode	Define the A/D converter mode of operation. It can be one of the following: - *Continuous*: The A/D converter performs the conversion continuously. When the converter value is read, the result of the last conversion is read. - *Start-stop* (*16-channel*): It is also called "one-cycle scan mode" in the Myway PE-OS Manual. In this mode, the A/D converter only performs the conversion after it is triggered by the PWM generator or by software.
Ch A_i *or* B_i Mode	Input mode of the A/D converter channel A_i or B_i, where *i* is from 0 to 7. The input mode can be one of the following: - *AC*: The input is an ac value, and the range is from -1.5V to +1.5V. - *DC*: The input is a dc value, and the range is from 0 to +3V.
Ch A_i *or* B_i Output Range	The output range V_{range} of the A/D converter channel A_i or B_i, where *i* is from 0 to 7.

The A/D converter can perform conversion autonomously when it is set to the "Continuous" mode. When it is set to the "Start-stop" mode, if a PWM generator is set to trigger the A/D, the conversion will occur when it is triggered by the PWM generator. If the PWM generator is set not to trigger the A/D, the conversion will occur in the software at the beginning of the PWM generator interrupt service routine.

In the dc mode, the input range of an A/D converter channel is from 0 to +3V, and the output range is from 0 to V_{range}. In the ac mode, the input range of a channel is from -1.5V to +1.5V, and the output range is from $-V_{range}$ to V_{range}.

The output is scaled based on the following:

In the dc mode: $V_o = V_i * V_{range} / 3$

In the ac mode: $V_o = V_i * V_{range} / 1.5$

where V_i is the value at the input port of the A/D converter. For example, in the dc mode, if V_{range} = 100, and V_i = 3, then V_o = 100. In the ac mode, if V_{range} = 100, and V_i = 1.5, then V_o = 100.

Note that the input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Note the following restrictions in using PWM generator triggered A/D converter:

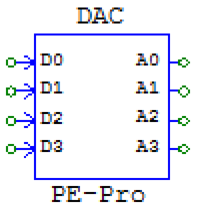
- The A/D converter can be triggered by one PWM generator only.
- It is not permitted to have the A/D converter triggered by one PWM generator, but some of the signals in this group are also used in a circuit that has a different sampling rate than the frequency of the PWM generator.

In these situations, it is recommended that the A/D converter be set to the "Continuous" mode.

7.5 D/A Converter

PE-Pro/F28335 provides a 12-bit 4-channel D/A converter.The image and the parameters of the converter are described below.

Image:



Attributes:

Parameters	**Description**
Ch D_i Mode	Input mode of the D/A converter channel D_i, where *i* is from 0 to 3. The input mode can be one of the following: - *AC*: The input is an ac value, and the range is from $-V_{range}$ to $+V_{range}$. - *DC*: The input is a dc value, and the range is from 0 to $+V_{range}$.
Ch D_i Input Range	The input range V_{range} of the D/A converter channel D_i, where *i* is from 0 to 3.

The D/A converter performs the D/A conversion. The output range is from 0 to +5V. In the dc mode, the input channel range is from 0 to $+V_{range}$. In the ac mode, the input channel range is from $-V_{range}$ to $+V_{range}$.

The output is scaled based on the following:

In ac mode: $V_o = V_i * 2.5 / V_{range} + 2.5$

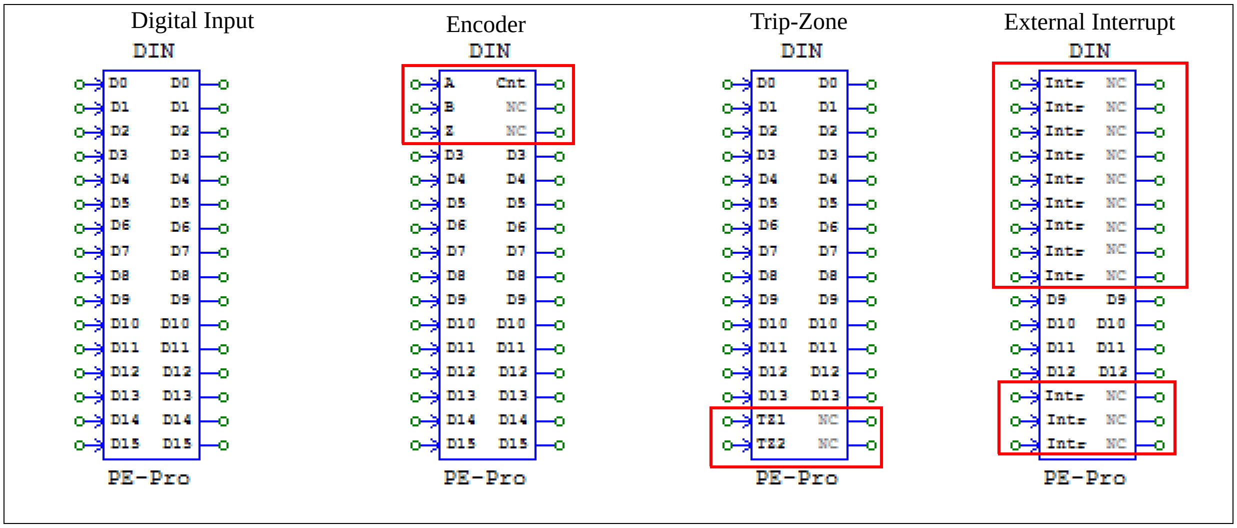
In dc mode: $V_o = V_i * 5 / V_{range}$

where V_i is the value at the input of the D/A converter. For example, in the ac mode, if $V_{range} = 20$, and $V_i = 10$, then $V_o = 3.75$. In the dc mode, if $V_{range} = 20$, and $V_i = 10$, then $V_o = 2.5$.

7.6 Digital Input / Encoder / Trip-Zone

The combo element of Digital Input / Encoder / Trip-Zone can be configured as digital input, encoder, trip-zone, and external interrupt. The images in each configuration and the parameters are shown below.

Images:



Attributes:

Parameters	Description
Ch D*i* Mode	The mode of the i_{th} channel, where *i* ranges from 0 to 15. It can be one of the following: - *Digital input*: For all the channels; - *Encoder*: For Channel Din 0; - *Trip-zone 1* or *2*: For Channel 14 and 15; - External interrupt: For Channel 1 to 8, and 13 to 15.
Use Z Signal	Define if the encoder in Channel Din 0-1-2 uses the Z (or index) signal
Counting Direction	For the encoder in Channel Din 0-1-2, the counting direction can be either *Forward* or *Reverse*. When it is set to *Forward*, the encoder counts up. Otherwise, the encoder counts down.
Encoder Resolution	The resolution of the encoder in Channel Din 0-1-2. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

<u>As Encoder:</u>

Channel Din 0, 1, and 2 can be configured as inputs of an encoder, where Din 0 is for input A, Din 1 is for input B, and Din 2 is for the input of the Z (index) signal. The output, labelled at the Channel Din 0 output as "Cnt", gives the encoder counter value.

<u>As Trip-Zone:</u>

Channel Din 14 can be configured as the input for trip-zone 1, and Din 15 as the input for trip-zone 2. When the input signal of trip-zone 1 changes from 1 to 0, it will trigger PWM Generator 123, and when the input signal of trip-zone 2 changes from 1 to 0, it will trigger PWM Generator 456.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

<u>As External Interrupt:</u>

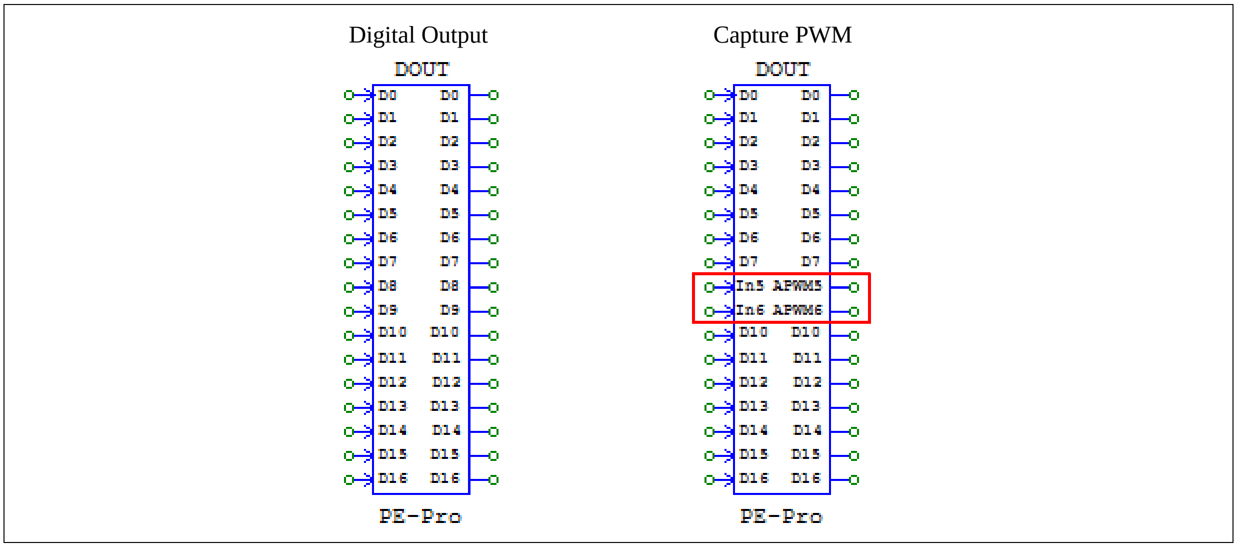
When a channel is defined as the input of the external interrupt, when the input changes from 0 to 1, an interrupt will be generated.

In PE-Pro/F28335, there is a limitation that up to 7 channels can be set as external interrupts. Up to 2 channels can be set as external interrupt in Din 3, 4, 5, 6, 14, and 15; and up to 5 channels can be set as external interrupt in Din 0, 1, 2, 7, 8, and 13.

7.7 Digital Output / APWM

The combo element of Digital Output / APWM can be configured as either Digital Output or APWM in Channel 8 and 9. Channel 8 can be configured as APWM 5 (GPIO48), and Channel 9 can be configured as APWM 6 (GPIO49). The images in each configuration and the parameters are shown below.

Image:



Digital Output

Capture PWM

DOUT
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16
PE-Pro
DOUT
D0 D1 D2 D3 D4 D5 D6 D7
In5 APWM5
In6 APWM6
D10 D11 D12 D13 D14 D15 D16
PE-Pro

Attributes:

Parameters	**Description**
Ch D8 Mode	The mode of Channel D8. It can be either *Digital Output* or *APWM 5.*
APWM5 Frequency	Frequency of the PWM generator APWM 5, in Hz
APWM5 Peak-to-Peak Value	Peak-to-peak value of the carrier wave for APWM 5
APWM5 Offset Value	DC offset value of the carrier wave for APWM 5
APWM5 Initial Value	Initial input value of APWM 5
Start APWM5 at Beginning	It can be set to *Start* or *Do not start*. When it is set to *Start*, APWM 5 will start from the beginning. If it is set to *Do not start*, one needs to start APWM 5 using the "Start PWM" function.
Ch D9 Mode	The mode of Channel D9. It can be either *Digital Output* or *APWM 6.*

APWM6 Frequency	Frequency of the PWM generator APWM 6, in Hz
APWM6 Peak-to-Peak Value	Peak-to-peak value of the carrier wave for APWM 6
APWM6 Offset Value	DC offset value of the carrier wave for APWM 6
APWM6 Initial Value	Initial input value of APWM 6
Start APWM6 at Beginning	It can be set to *Start* or *Do not start*. When it is set to *Start*, APWM 6 will start from the beginning. If it is set to *Do not start*, one needs to start APWM 6 using the "Start PWM" function.

The APWM generators APWM 5 and APWM 6 are limited in functionality. They can generate interrupt, but can not trigger the A/D converter, and can not use the trip-zone signals.

8

PE-Expert3 Hardware Target

8.1 Overview

With the PE-Expert3 Hardware Target, SimCoder can generate the code for the PE-Expert3 DSP development platform from Myway with the following boards:

- DSP Board MWPE3-C6713 (for PE-View8) and MWPE3-C6173A (for PE-View9)
- PEV Board

When generating the code for a system that has multiple sampling rates, SimCoder will use the PWM generator interrupts for the PWM sampling rates. It will then first use the Timer 0 interrupt, and then Timer 1 interrupt if needed, for other sampling rates in the control system. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In the PE-Expert3 system, digital input, encoder, and capture can also generate hardware interrupts. An interrupt must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine) through the interrupt block as described in Section 5.4 of this Manual. In PE-Expert3, since interrupts generated by digital input, encoder, and capture are handled by the same interrupt service routine, all the interrupt blocks must connect to the same subcircuit block.

The hardware functions and elements of the PE-Expert3 Hardware Target are described in the sections below.

8.2 PEV Board

The PEV board contains the following functions and hardware elements:

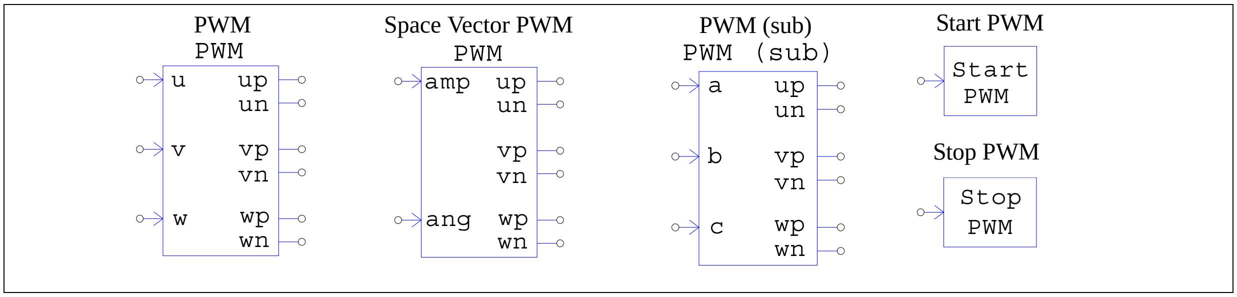
- PWM generators and Start/Stop PWM functions
- A/D converter
- Digital input
- Digital output
- Up/Down counter
- Encoder
- Capture

Please note that hardware input/output elements, including PWM generators, A/D converter, digital input/output, up/down counter, encoder, and capture, can not be placed inside a subcircuit. They must be in the top-level main circuit only. The Start/Stop PWM function elements, however, can be placed in either the main circuit or subcircuits.

8.2.1 PWM Generators

There are five elements associated with PWM generation: **PWM (sub)** generator, **PWM** generator, **Space Vector PWM** generator, **Start PWM** function, and **Stop PWM** function. The **PWM (sub)** element is a built-in module based on the **PWM** element.

Images:



Attributes for **PWM** and **Space Vector PWM:**

Parameters	**Description**
Board No.	The board number of the PEV board that contains the PWM generator.
Channel No.	The channel number of the PWM generator. It can be either 0 or 1.
Dead Time	The dead time for the PWM generator, in sec.
Carrier Frequency	Frequency of the PWM generator, in Hz
Start PWM at Beginning	It can be set to either *Start* or *Do not start*. When it is set to *Start*, PWM will start right from the beginning. If it is set to *Do not start*, one needs to start PWM using the "Start PWM" function.

Attributes for **PWM (sub):**

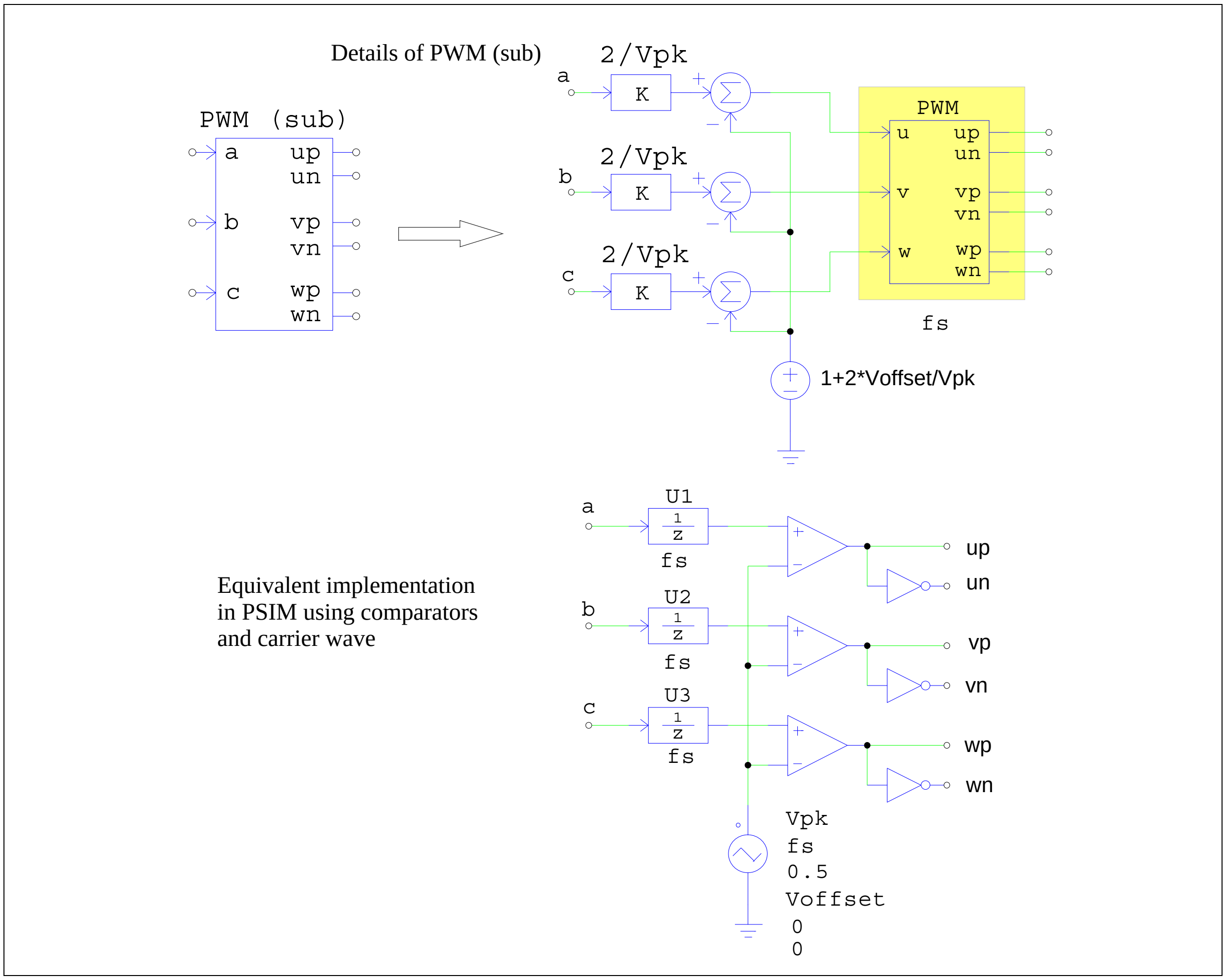
Parameters	**Description**
Board No.	The board number of the PEV board that contains the PWM generator.
Channel No.	The channel number of the PWM generator. It can be either 0 or 1.
Dead Time	The dead time for the PWM generator, in sec.
Carrier Frequency	Frequency of the PWM generator, in Hz
Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Start PWM at Beginning	It can be set to either *Start* or *Do not start*. When it is set to *Start*, PWM will start right from the beginning. If it is set to *Do not start*, one needs to start PWM using the "Start PWM" function.

The element **PWM** generates sinusoidal PWM signals for a three-phase system. The inputs "u", "v", and "w" are for three-phase input modulation signals. The input ranges are between -1 to 1. That is, when the input is -1, the duty cycle is 0, and when the input is 1, the duty cycle is 1. With the input at 0, the duty cycle is 0.5. The carrier wave is a triangular wave with 0.5 duty cycle (the intervals of the rising slope and the falling slope are equal).

The element **Space Vector PWM** generates PWM signals for a three-phase system based on space vector PWM technique. The input "amp" is for the space vector amplitude, and the range is from 0 to 1. The input "ang" is for the space vector phase angle, and the range is from -2π to 4π.

Because the input range of the **PWM** generator is between -1 and 1, while in PSIM simulation, normally PWM signals are generated by comparing a modulation signal with a carrier signal, and the modulation signal range may not be from -1 to 1, scaling may be needed before the modulation signal is sent to the PWM generator.

In order to make it easier to switch from the comparator-based PWM to the hardware PWM generator element, the **PWM (sub)** element is provided. It is a built-in block consisting of the **PWM** element and the scaling circuit, as shown below.



The circuit on the top right shows the details of the **PWM (sub)** element. It consists of a scaling circuit and the hardware **PWM** element. With the scaling, ranges of the inputs *a*, *b*, and *c* are no longer limited to -1 and 1. Similar to the definition of a carrier voltage source, the peak-to-peak value and the dc offset can be defined directly.

The circuit on the bottom right shows the equivalent circuit of the **PWM (sub)** element, implemented in PSIM using comparators and a triangular carrier voltage source. The carrier source parameters are:

V_peak_to_peak:	Vpk
Frequency:	fs
Duty Cycle:	0.5
DC Offset:	Voffset
Tstart:	0
Phase Delay:	0

Note the inclusion of three unit delay blocks U1, U2, and U3, as they are used to model the one-cycle delay effect existing in the hardware **PWM** element. Also, the carrier wave duty cycle is fixed at 0.5, as the carrier wave in the hardware **PWM** element is of triangular type.

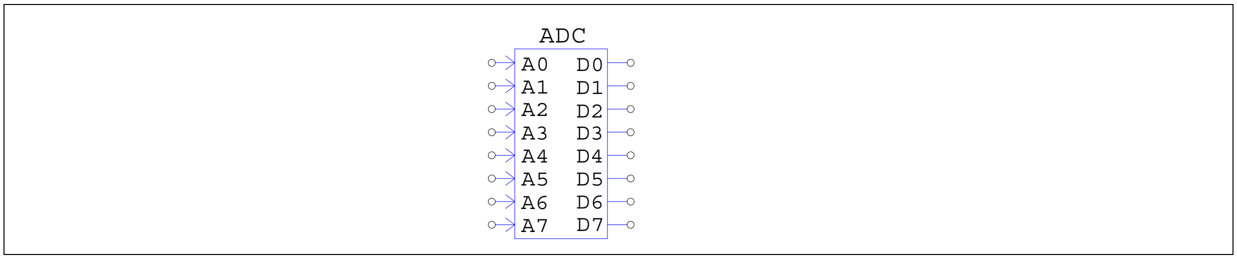
To start PWM, apply a signal of 1V to the input of the **Start PWM** element. To stop PWM, apply a signal of 1V to the input of the **Stop PWM** element.

Please note that when the **PWM** and **PWM (sub)** generators are simulated, the dead time is ignored and is not considered in the simulation.

8.2.2 A/D Converter

An A/D converter converts an analog signal into a digital signal that DSP can process.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the A/D converter.
Ch A*i* Output Range	The output range V_{range} of the i_{th} A/D channel.

The input range of the A/D converter is from -5V to +5V, and the output range is from -V_{range} to V_{range}. The output is scaled based on the following:

$$V_o = V_i * V_{range} / 5$$

For example, if $V_i = 2$, and $V_{range} = 20$, then $V_o = 2 * 20 / 5 = 8$.

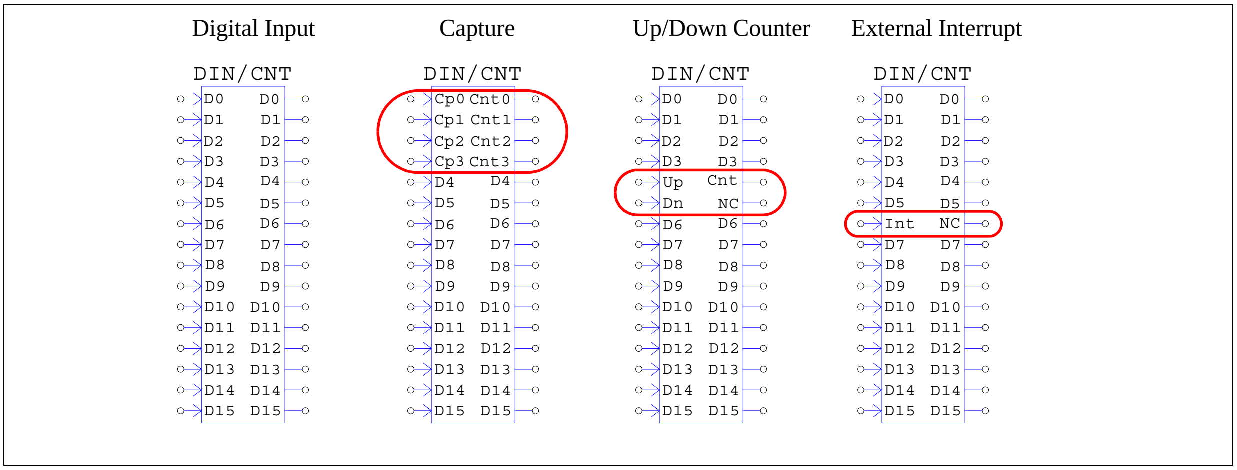
8.2.3 Digital Input / Capture / Counter

The hardware provides a 16-pin digital input element. Note that Pins D0 through D3 are shared with the capture element, Pins D4 and D5 are shared with the up/down counter element, and Pin D6 can be used for input of external interrupt.

Because of the shared pins, a combo element is provided to represent digital input, capture, and counter. Input/output pins are assigned to different functions depending on the definition.

The images and attributes of the combo element in different functions are shown below.

Images:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the element.
Input/Capture *i*	The index *i* changes from 0 to 3, corresponding to Inputs D0 to D3 respectively. The parameter can be defined as one of the following: - *Digital Input i*: Input pin D_i will be a digital input. - *Capture i*: Input pin D_i will be the input of Capture *i*, and the output pin D_i will be the counter output of Capture *i*. The captions of the input/output pins will be changed to Cap_i and Cnt_i.
Counter Source *i*	The index *i* changes from 0 to 3, corresponding to Inputs D0 to D3 respectively. The parameter The name of the counter source. The parameter can be either "GP_TIMER" for general-purpose timer, or the name of the encoder.
Input 4 and 5 / Counter	It can be defined as one of the following: - *Digital Input 4 and 5*: Input pin D4 and D5 will be digital inputs. - *Counter*: Input pin D4 and D5 will be the inputs of the up/down counter, and the output pin D4 will be the counter output. In the counter mode, the output pin D5 is not used. The captions of the input pin D4 will be changed to *Up* (for up counter input) and pin D5 to *Dn* (for down counter input). The caption of the output pin D5 will be changed to *Cnt* (for counter output), and pin D6 to *NC* (for not connected).
Counter Mode	When Inputs D4 and D5 are defined as counter inputs, the counter mode can be either *Up/down* or *Direction/pulse*.
Input 6 / External Interrupt	It can be one of the following: - *Digital Input 6*: Input pin 6 will be a digital input. - *External Interrupt*: This pin will be the input of the external interrupt. The caption of the input will be changed to *Int* (for interrupt) and the output to *NC* (for not connected).

As a Capture:

The capture element has 4 inputs. When an input changes from low to high (from 0 to 1), it will capture the counter value of the source, and output it through the output pin. The counter source can be either the general-purpose timer (which is the 32-bit free-run counter on the PEV Board), or the encoder.

As a Counter:

The counter has two modes of operations: up/down mode and direction/pulse mode. When the counter is in the "Up/down" mode, the counter will count up when there is a pulse at the "up" input, and will count down when there is a pulse at the "dn" input.

When the counter is in the "Direction/pulse" mode, and when there is a pulse at the "pulse" input, the counter will count up when the "direc" input is 0, and will count down when the "direc" input is 1.

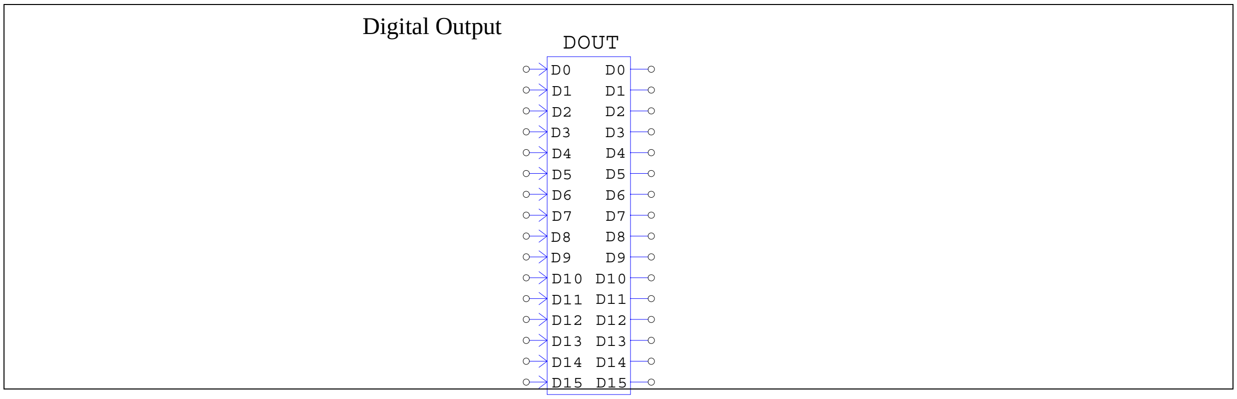
As an External Interrupt:

When Input pin D6 is defined as the input of the external interrupt, when the input changes from 0 to 1, an interrupt will be generated.

8.2.4 Digital Output

The images and attributes of the digital output element are shown below.

Image:



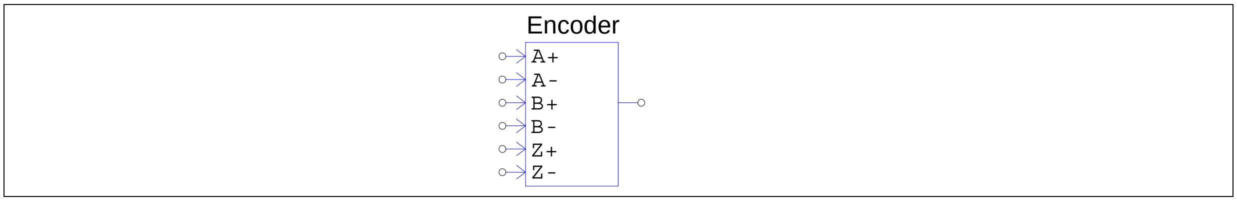
Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the element.

8.2.5 Encoder

An encoder is used for position measurement in a motor drive system. It can operate in either "Open Collector" or "Differential Mode" mode.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the encoder.
Encoder Mode	The encoder mode can be either "Open Collector" or "Differential Mode".
Counting Direction	It can be either "Forward" or "Reverse". When it is set to "Forward", the encoder counts up, and when set to "Reverse", the encoder counts down.

The output of the encoder output gives the counter value. Also, an interrupt can be generated by the input signal Z+ and Z-.

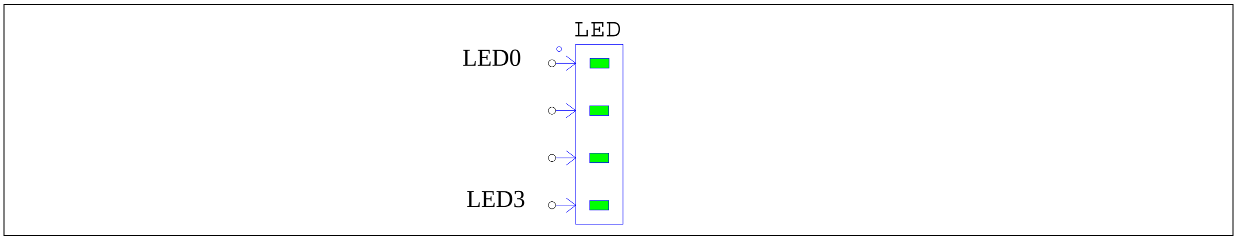
8.3 LED Output

The LED output element is available for the DSP Board MWPE3-C6713A (with PE-View9) only. If the software environment for PE-Expert3 is set to PE-View8, this element will be ignored.

There are four light-emitting diodes (LED) on the DSP Board MWPE3-C6713A: LED0, LED1, LED2, and LE D3.

When the input level of a LED is higher than 0.5, the LED will be on. Otherwise, it will be off.

Image:



In the diagram, the input with the dot corresponds to the LED0.

Note that this element is for hardware implementation only, and it will be ignored in the simulation. To display the LED value in the simulation, connect a voltage probe to the input node.

8.4 PE-Expert3 Runtime Library Functions

PE-Expert3 provides a runtime library for high-speed calculation. When the code is generated, whenever possible, functions from the PE-Expert3 runtime library are used.

The table below shows the PSIM elements and the corresponding PE-Expert3 runtime library functions.

PSIM Elements	PE-Expert3 Runtime Library Functions
Sine or Sine (in rad.)	mwsin (float x)
Cosine or Cosine (in rad.)	mwcos (float x)
Tangent Inverse	mwarctan2 (float y, float x)
Square-root	mwsqrt (float x)
abc-alpha/beta Transformation	uvw2ab (float u, float v, float w, float *a, float *b)
ab-alpha/beta Transformation	uv2ab (float u, float v, float *a, float *b)
ac-alpha/beta Transformation	uw2ab (float u, float w, float *a, float *b)
alpha/beta-abc Transformation	ab2uvw (float a, float b, float *u, float *v, float *w)
alpha/beta-dq Transformation	ab2dq (float a, float b, float *d, float *q)
dq-alpha/beta Transformation	dq2ab (float d, float q, float *a, float *b)
xy-r/angle Transformation	xy2ra (float y, float x, float *a, float *b)
r/angle-xy Transformation	ra2zy (float r, float a, float *x, float *y)

9

TI DMC Library

9.1 Overview

The SimCoder library supports the TI Digital Motor Control (DMC) library versions V4.0, V4.1 and V4.2. User may select the version in the **Simulation Control** under the **SimCoder** tab. Note that only one version can be used at a time. Once a version is selected, elements that are not supported in that version will be disabled.

SimCoder's TI DMC Library contains the function blocks listed in the table below. A brief description of these blocks are given in this Chapter. For more detailed description, please refer to the document from Texas Instruments.

Element Name	Description	Version Available		
ACI_FE	Flux estimator of the 3-phase induction motor	4.0	4.1	4.2
ACI_SE	Speed estimator of the 3-phase induction motor	4.0	4.1	4.2
ANGLE_MATH	Wrap angle within 0 and 1.0 (0 to 2π) or -0.5 and 0.5 (-π to +π)			4.2
CLARK	Clarke transformation	4.0	4.1	4.2
COMTN_TRIG	Commutation trigger generator	4.0	4.1	4.2
CUR_MOD	Current model	4.0	4.1	4.2
IMPULSE	Impulse generator	4.0	4.1	4.2
IPARK	Inverse Park transformation	4.0	4.1	4.2
PARK	Park transformation	4.0	4.1	4.2
PHASE_VOLT	Phase voltage reconstruction	4.0	4.1	4.2
PI	PI controller with anti-windup correction	4.0	4.1	4.2
PI_POS	PI controller with anti-windup and position error wrapper		4.1	4.2
PI_POS_REG4	PI controller with anti-windup and position error wrapper, proportional gain separated.			4.2
PI_REG4	PI controller with anti-windup correction, proportional gain separated.			4.2
PID_GRANDO	PID controller grando	4.0	4.1	4.2
PID_REG3	Digital PID controller with anti-windup	4.0	4.1	4.2
RAMP_GEN	Generate a ramp with adjustable gain	4.0	4.1	4.2
RMP_CNTL	Ramp up/down to targeted value and flag when output equals input.	4.0	4.1	4.2
RMP2_CNTL	Ramp up/down to targeted value	4.0	4.1	4.2
RMP3_CNTL	Ramp down to targeted value and flag when output equals input	4.0	4.1	4.2
SMOPOS	Sliding-mode rotor position observer of PMSM	4.0	4.1	4.2

SPEED_EST	Speed calculator based on rotor angle without direction information	4.0	4.1	4.2
SPEED_FR	Speed calculator based on rotor angle from encoder sensor	4.0	4.1	4.2
SPEED_PRD	Speed calculator based on period measurement	4.0	4.1	4.2
SVGEN	Space vector generator with quadrature control	4.0	4.1	4.2
SVGEN_COMM	Space vector generator using common mode voltage	4.0	4.1	4.2
SVGEN_DPWM	Space vector generator with DPWM approach	4.0	4.1	4.2
SVGEN_MF	Space vector generator using magnitude and frequency	4.0	4.1	4.2
VHZ_PROFILE	Volt/Hertz profile for ac induction motor	4.0	4.1	4.2

9.2 ACI_FE: Flux Estimator of 3-phase Induction Motors

This block implements the flux estimator with the rotor flux angle for the 3-phase induction motor based upon the integral of back emf's (voltage model) approach. To reduce the errors due to pure integrator and stator resistance measurement, the compensated voltages produced by PI compensators are introduced. Therefore, this flux estimator can be operating over a wide range of speed, even at very low speed.

The ACI_FE module requires eight constants according to the machine parameters, base quantities, mechanical parameters, and sampling period. These eight constants are computed by the macro ACI_FE_CONST.

SimCoder combined ACI_FE and ACI_FE_CONST functions into one block to simplifies the process. When this element is in a circuit schematic, SimCoder copies ACI_FE macro to the generated project folder and the ACI_FE_CONST sub-module in ACI_FE's initial data structure definition.

The overall of the flux estimator is shown below. The rotor flux linkages in the stationary reference frame are mainly computed by means of the integral of back emf's in the voltage model. By introducing the compensated voltages generated by PI compensators, the errors associated with pure integrator and stator resistance measurement can be taken care.

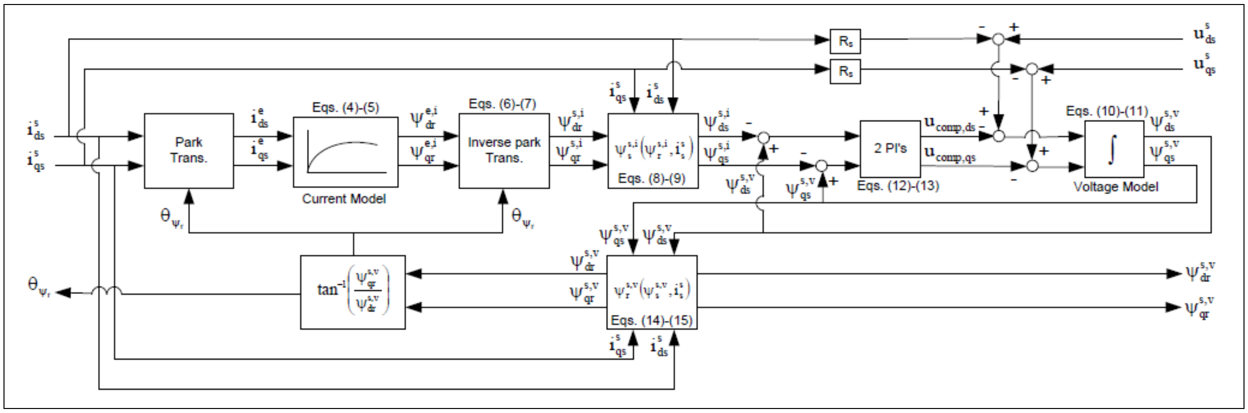
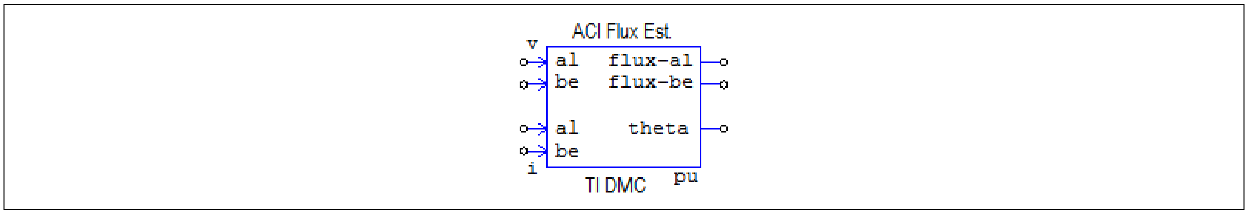


Image:



Attributes:

Parameters	Description
Stator Resistance Rs	Motor stator resistance, in ohm
Stator Inductance Ls	Motor stator inductance, in H
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Magnetizing Inductance Lm	Motor magnetizing inductance, in H
Base Phase Voltage Vb	Motor base phase voltage, in volt
Base Phase Current Ib	Motor base phase current, in ampere
PI Proportional Gain Kp	PI controller proportional gain
PI Integral Gain Ki	PI controller integral gain
Sampling Frequency	System sampling frequency

Input and Output Signals:

Name	I/O	Description
ud	Input	Stationary d-axis stator voltage
uq	Input	Stationary q-axis stator voltage
id	Input	Stationary d-axis stator current
iq	Input	Stationary q-axis stator current
flux-d	Output	Stationary d-axis estimated rotor flux
flux-q	Output	Stationary q-axis estimated rotor flux
theta	Output	Rotor flux angle, in degrees, range 0 to 360

9.3 ACI_SE: Speed Estimator of 3-phase Induction Motors

This block implements a speed estimator of the 3-phase motor based upon its mathematics model. The estimator’s accuracy relies heavily on knowledge of critical motor parameters.

The speed estimator of Induction motor module requires four constants according to the machine parameters, base quantities, mechanical parameters, and sampling period. These four constants are computed by the macro ACI_SE_CONST.

SimCoder combined ACISE and ACISE_CONST in this block to simplify the process. When this element is in a circuit schematic, SimCoder copies ACISE macro to the generated project folder and implemented ACISE _CONST sub-module in ACI_SE's initial data structure definition.

The open-loop speed estimator is derived basing on the mathematics equations of induction motor in the stationary reference frame. The precise values of machine parameters are unavoidably required, otherwise the steady-state speed error may happen.

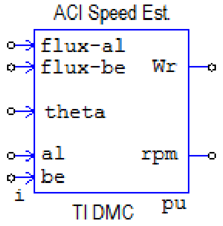
Three equations are mainly employed to compute the estimated speed in per unit:

The rotor speed in per unit: $$\omega_{r,pu} = \omega_{e,pu} - K_1\left(\frac{\lambda^s_{dr,pu} i^s_{qs,pu} - \lambda^s_{qr,pu} i^s_{ds,pu}}{\left(\lambda^s_{r,pu}\right)^2}\right) \quad \text{pu}$$

The synchronous speed in per unit: $$\omega_{e,pu}(k) = K_2\left(\theta_{\lambda_r,pu}(k) - \theta_{\lambda_r,pu}(k-1)\right) \quad \text{pu}$$

The estimated speed in per unit: $$\hat{\omega}_{e,pu}(k) = K_3\hat{\omega}_{e,pu}(k-1) + K_4\omega_{e,pu}(k) \quad \text{pu}$$

Image:



Attributes:

Parameters	Description
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Filter Cut-off Frequency fc	Cut-off frequency of the low pass filter, in Hz
Base Electrical Frequency fb	Base electrical frequency, in Hz
Base Speed in rpm	Base motor speed in rpm
Sampling Frequency	System sampling frequency

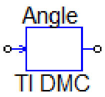
Input and Output Signals:

Name	I/O	Description
flux-d	Input	Stationary d-axis estimated rotor flux
flux-q	Input	Stationary q-axis estimated rotor flux
theta	Input	Rotor flux angle, in degrees, range 0 to 360
id	Input	Stationary d-axis stator current
iq	Input	Stationary q-axis stator current
Wr	Output	Estimated motor speed in per unit
rpm	Output	Estimated motor speed in rpm

9.4 ANGLE_MATH: Angle Wrap

This block wraps angle within 0 and 1.0, or wraps angle within -0.5 and +0.5.

Image:



Attributes:

Parameters	Description
Angle Range	Define the range of angle output. 0 to +1.0: wrap angle within 0 and 2π (1.0) -0.5 to +0.5: wrap angle within -π (-0.5) and +π (+0.5)

Input and Output Signals:

Name	I/O	Description
	Input	Raw angle input
	Output	Wrapped angle output

9.5 CLARKE: Clarke Transformation

This block implements the Clarke transformation from balanced 3-phase quantities into balanced 2-phase quadrature quantities.

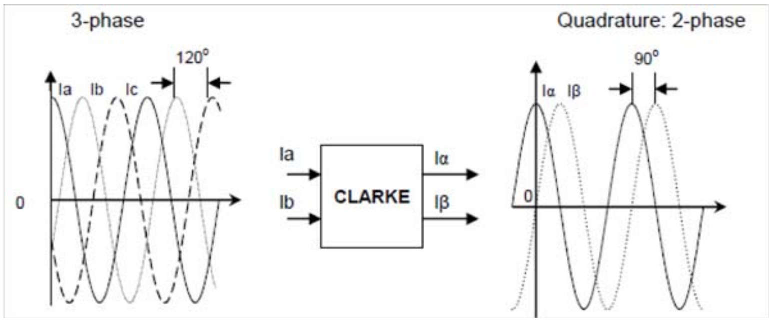
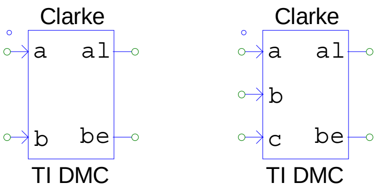


Image:



Attributes:

Parameters	Description
Number of Input Phase	Define the number of input phases. - 2-Phase: Inputs are Phases a and b; - 3-phase: Inputs are Phases a, b, and c.

Input and Output Signals:

Name	I/O	Description
a	Input	Phase A component of the balanced 3-phase quantities.
b	Input	Phase B component of the balanced 3-phase quantities.
al	Output	Direct axis (d) component of the transformed signal.
be	Output	Quadrature axis (q) component of the transformed signal.

The transformation equations are given below:

If the number of input phase is selected as 2-Phase, the transformation equations are:

$$v_\alpha = v_a$$

$$v_\beta = \frac{2v_b + v_a}{\sqrt{3}}$$

If the number of input phase is selected as 3-Phase, the transformation equations are:

$$v_\alpha = v_a$$

$$v_\beta = \frac{v_b - v_c}{\sqrt{3}}$$

9.6 COMNT_TRIG: Commutation Trigger Generator for BLDC Motors

This block determines the back emf zero crossing points of a 3-phase BLDC motor based on motor phase voltage measurements and then generates the commutation trigger points for the 3-phase power inverter switches.

The figure below shows the 3-phase power inverter topology used to drive a 3-phase BLDC motor. In this arrangement, the motor and inverter operation is characterized by a two phase ON operation. This means that two of the three phases are always energized, while the third phase is turned off.

The bold arrows on the wires indicate the Direct Current flowing through two motor stator phases. For sensorless control of BLDC drives it is necessary to determine the zero crossing points of the three Bemf voltages and then generate the commutation trigger points for the associated 3-ph power inverter switches.

The COMTN_TRIG module computes the neutral voltage, The back emf zero-crossing point, and the time delay corresponding to the commutation delay angle.

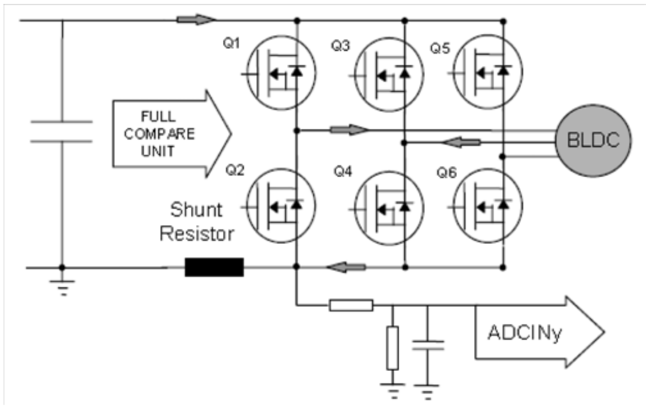
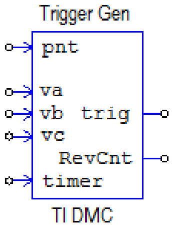


Image:



Attributes:

Parameters	Description
Noise Windows Delta	Noise windows delta
Noise Windows Threshold	Noise windows dynamic threshold

Input and Output Signals:

Name	I/O	Description
pnt	Input	Commutation state pointer
va	Input	Motor phase a voltage referenced to ground
vb	Input	Motor phase b voltage referenced to ground
vc	Input	Motor phase c voltage referenced to ground
time	Input	Virtual timer for commutation delay angle calculation
trig	Output	Commutation trigger output, (0 or 0x7FFF)

9.7 CUR_MOD: Current Model

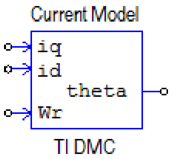
This block calculates the rotor flux position from motor current measurements after Park transformation.

With the asynchronous drive, the mechanical rotor angular speed is not equal to the rotor flux angular speed. This implies that the necessary rotor flux position cannot be detected directly by the mechanical position sensor used with the asynchronous motor (QEP or tachometer). The current model module is usually added to the generic structure in the regulation block diagram to perform a current and speed closed loop for a three phases ACI motor in FOC control.

This module requires three constants according to machine parameters, base quantities, mechanical parameters, and sampling period. These constants are computed by the macro CUR_MOD_CONST.

SimCoder combines the functions of CUR_MOD and CUR-MOD_CONST to simplify the process. When this element in used in a circuit schematic, SimCoder copies CUR_MOD macro into the generated project code folder and implements CUR_MOD_CONST sub module in CUR_MOD's initial data structure definition.

Image:



Attributes:

Parameters	Description
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Base Electrical Frequency	Base electrical frequency in Hz
Sampling Frequency	System sampling frequency

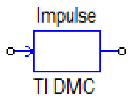
Input and Output Signals:

Name	I/O	Description
iq	Input	Synchronous rotating q-axis current
id	Input	Synchronous rotating d-axis current
Wr	Input	Motor rotor electrical angular velocity
theta	Output	Motor rotor flux angle, in degree, range 0 to 360

9.8 IMPULSE: Impulse Generator

This block generates a periodic impulse.

Image:

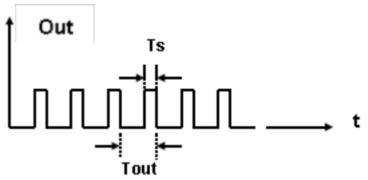


Input and Output Signals:

Name	I/O	Description
	Input	Period of output in number of sampling period
	Output	Impulse output.

If sampling time period is Ts and input = N, then, the output period Tout = N times sampling period.

Out = 0x7FFF for one sampling period at the end of Tout.
Out = 0 for the rest of the time.



9.9 IPARK: Inverse Park Transformation

This block implements the transformation vectors in orthogonal rotating reference frame into two phase orthogonal stationary frame, as shown below:

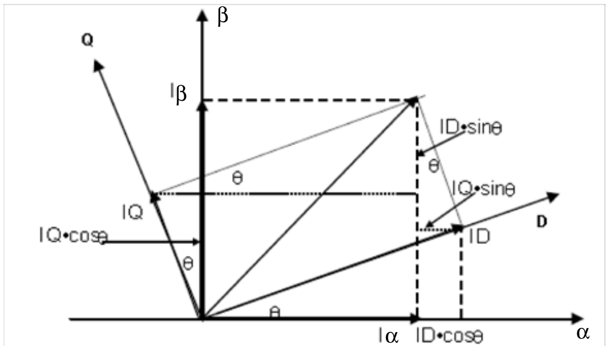
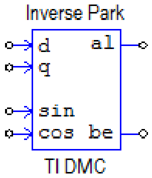


Image:



Input and Output Signals:

Name	I/O	Description
d	Input	Rotating d-axis stator variable
q	Input	Rotating q-axis stator variable
sin	Input	SIN of the phase angle between stationary and rotating frame
cos	Input	COS of the phase angle between stationary and rotating frame
al	Output	Stationary d-axis stator variable
be	Output	Stationary q-axis stator variable

The transformation equations are:

$$v_\alpha = v_d \cdot \cos\theta - v_q \cdot \sin\theta$$

$$v_\beta = v_d \cdot \sin\theta + v_q \cdot \cos\theta$$

9.10 PARK: Park Transformation

This block implements the transformation which converts vectors in balanced 2-phase orthogonal stationary system into orthogonal rotating reverence frame, as shown below.

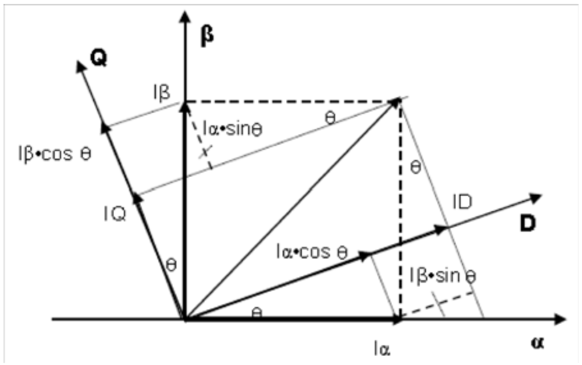
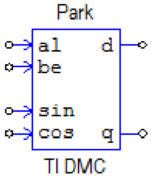


Image:



Input and Output Signals:

Name	I/O	Description
al	Input	Stationary d-axis stator variable v_α
be	Input	Stationary q-axis stator variable v_β
sin	Input	SIN of the phase angle between stationary and rotating frame
cos	Input	COS of the phase angle between stationary and rotating frame
d	Output	Rotating d-axis stator variable v_d
q	Output	Rotating q-axis stator variable v_q

The transformation equations are:

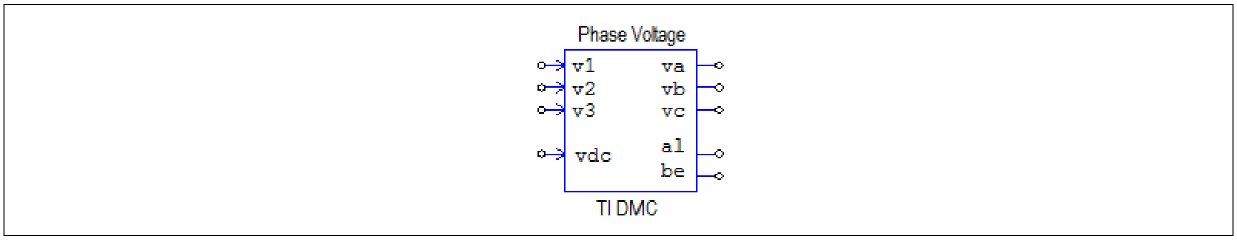
$$v_d = v_\alpha \cdot \cos\theta + v_\beta \cdot \sin\theta$$

$$v_q = v_\alpha \cdot \sin\theta + v_\beta \cdot \cos\theta$$

9.11 PHASE_VOLT: Phase Voltage Reconstruction

This element calculates three phase voltages impressing to the 3-phase electric motor (i.e., induction or synchronous motor) by using the conventional voltage-source inverter. Three phase voltages can be reconstructed from the DC-bus voltage and three switching functions of the upper power switching devices in the inverter. In addition, this software module also includes the clarke transformation changing from three phase voltages into two stationary dq-axis phase voltages.

Image:



Attributes:

Parameters	Description
Out-of-Phase Adjustment	Specify if the input phase signals are out of phase with respect of output. - Yes if 3-phase input signals are from the lower switching functions. - No if 3-phase input signals are from upper switching functions

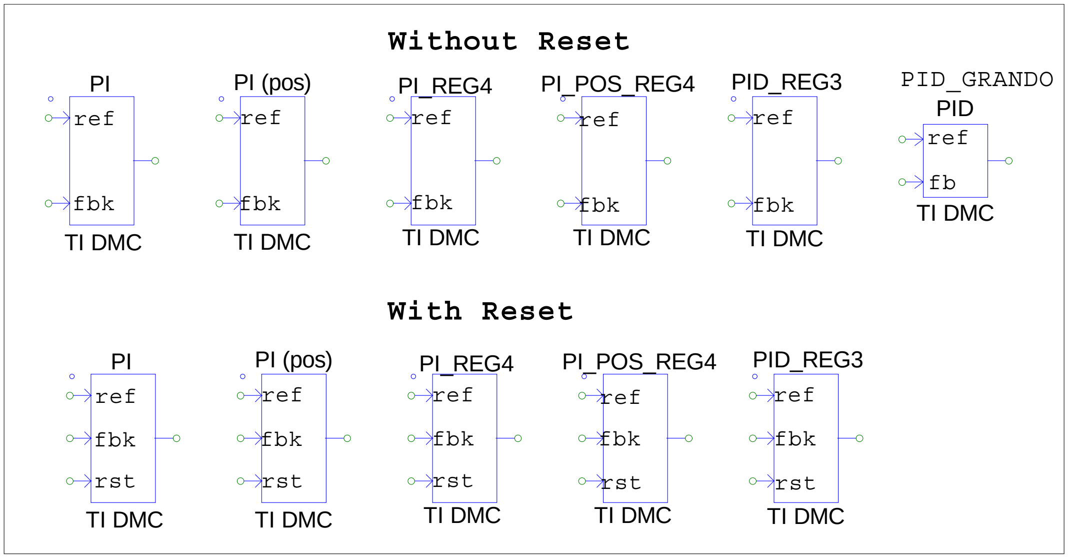
Input and Output Signals:

Name	I/O	Description
v1, v2, v3	Inputs	Modulation voltage phase A, B, and C for inverter upper switching devices.
vdc	Input	DC bus voltage
va, vb, vc	Outputs	Line-to-neutral voltage for phase A, B, and C
al	Output	Stationary d-axis phase voltage
be	Output	Stationary q-axis phase voltage

9.12 PID Controllers

SimCoder implemented all the TI DMC PID controller algorithm into different function blocks. The images and the input and output signals of these PID controllers are show below. All controllers except PID_GRANDO allow user to select with or without reset. If *With Reset* is selected, an extra input *rst* will be added to the block. The integrator output will be clamped to zero when the *rst* input is logically high.

Images:



Input and Output Signals:

Name	I/O	Description
ref	Input	Reference set point
fb	Input	Feedback
rst	Input	When set to logic high (1), the integral output is clamped to zero.
	Output	Controller output

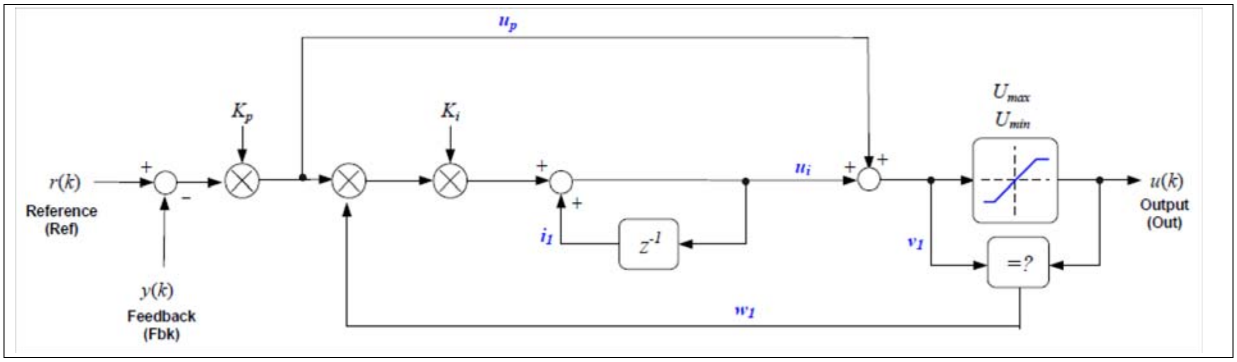
The functions of these controllers are described in the subsections.

9.12.1 PI: PI Controller with Anti-Windup

This block implements a simple 32-bit digital PI controller with anti-windup correction. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.

The PI controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

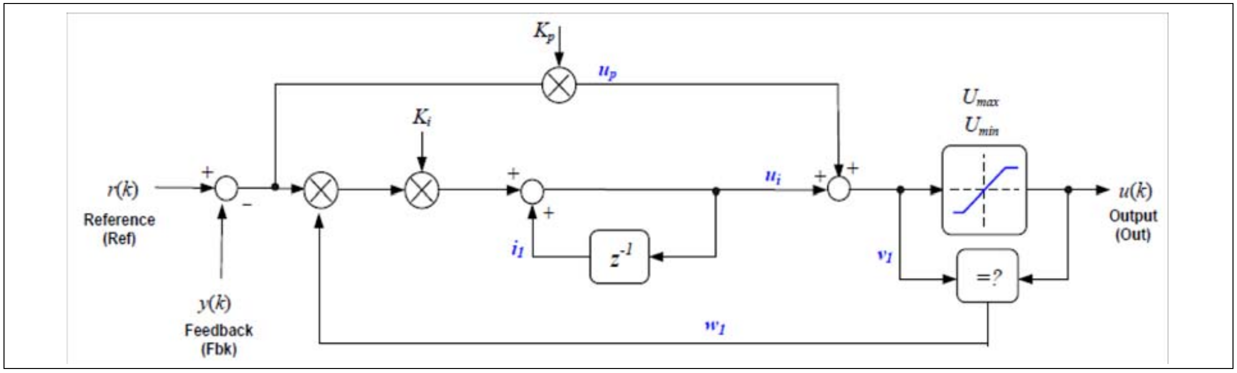
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit Umax
Minimum Output	Minimum output limit Umin

9.12.2 PI_REG4: PI Controller with Anti-Windup

This module implements a simple 32-bit digital PI controller with anti-windup correction. Functionally, it is similar to PI module described above, the difference is in the path of P control such that Kp can be set to zero. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.

The PI_POS controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

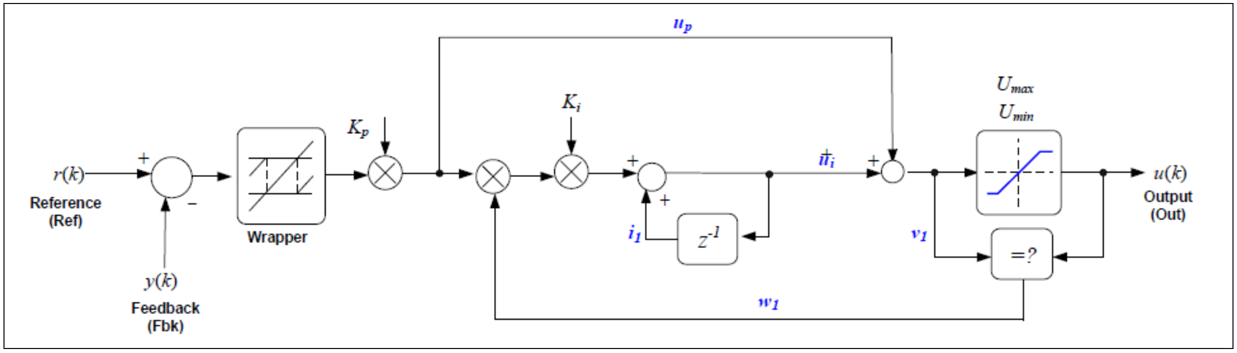
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

9.12.3 PI _POS: PI Controller with Position Error Wrapper

This block implements a simple 32-bit digital PI controller with anti-windup correction, and also with a position error wrapper. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.
- Position error wrap around to fit within -π and +π.

The PI_POS controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

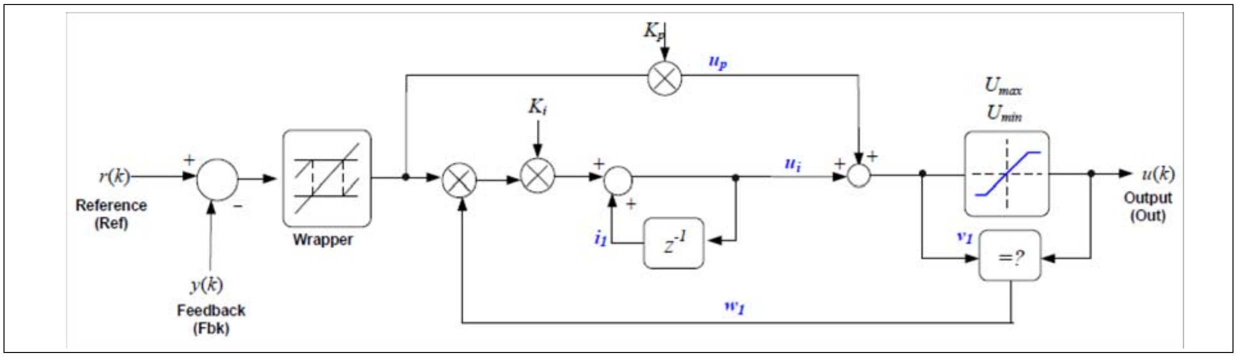
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

9.12.4 PI _POS_REG4: PI Controller with Position Error Wrapper

This module implements a generic, simple 32-bit digital PI controller with anti-windup correction, exactly same as in the previous section on PI_POS controller but treated as in PI_REG4. This block implements a simple 32-bit digital PI controller with anti-windup correction, and also with a position error wrapper. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.
- Position error wrap around to fit within -π and +π.

The PI_POS_REG4 controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



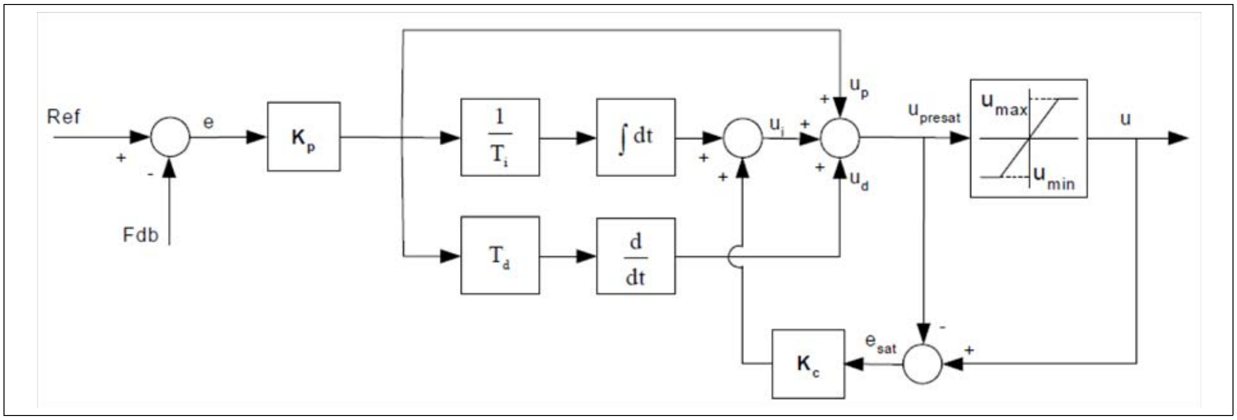
Attributes:

Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

9.12.5 PID_REG3: PID Controller with Anti-Windup

This block implements a 32-bit digital PID controller with anti-windup correction. It can be used for PI or PD controller as well. In this digital PID controller, the differential equation is transformed to the difference equation by means of the backward approximation.

The block diagram of this conventional PID controller with anti-windup correction is shown below.



Attributes:

Parameters	**Description**
Proportional Loop Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Derivative Gain Kd	Derivative gain
Integral Correction Gain Kc	Integral correction gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

9.12.6 PID_GRANDO: PID Controller

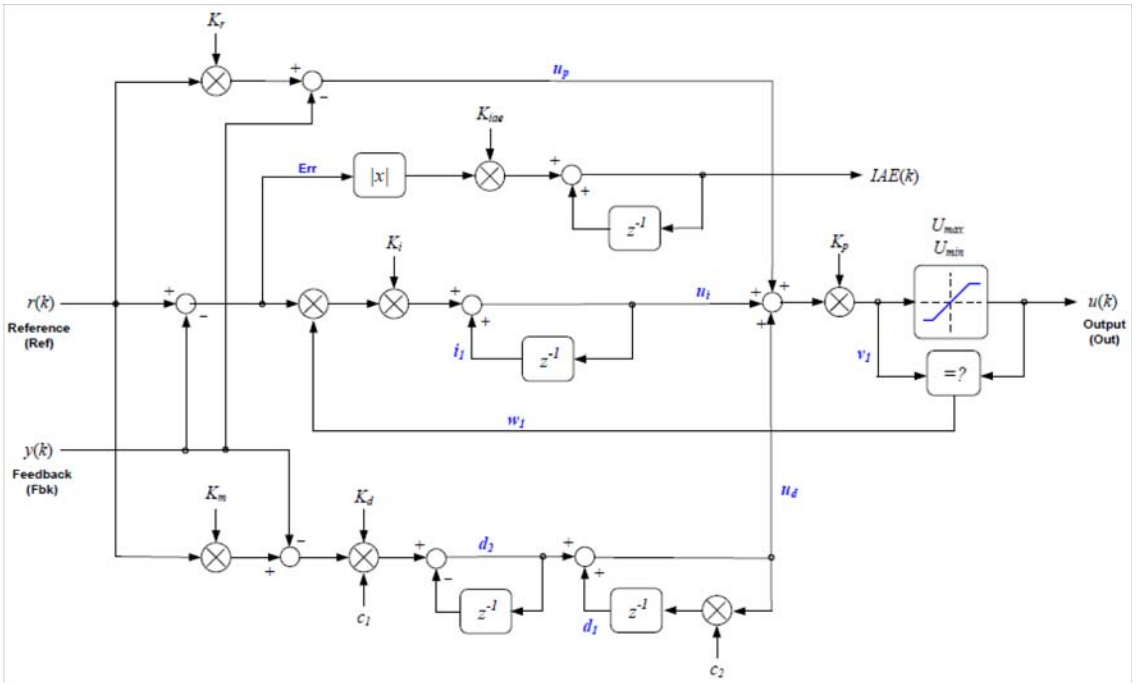
This block implements a basic summing junction and PID control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Independent reference weighting on derivative path
- Anti-windup integrator reset.
- Programmable derivative filter
- Transient performance measurement

PID Grando is an example of a PID structure often called "standard" form, in which proportional gain is applied after the three controller paths have been summed. This contrasts with the "parallel" PID form, in which P, I, and D gains are applied in separate paths. All input, output and internal data is in IQ24 fixed-point format.

The Grando controller includes a saturation block to limit the range of the control effort, u(k). If the output saturates, the integrator is disabled to prevent a phenomenon known as "wind-up". In cases where saturation may occur in other parts of the control loop, user code should disable integral action by temporarily setting the integrator gain (Ki) to zero when saturation occurs, and restoring it once saturation has been cleared.

The block diagram of the internal controller structure is shown below.



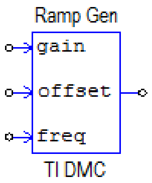
Attributes:

Parameters	**Description**
Proportional Ref Weight Kr	Reference weighting on proportional path
Proportional Loop Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Derivative Gain Kd	Derivative gain
Derivative Ref. Weight Km	Derivative reference weighting
Cut-off Frequency fc	Cut-off frequency for the first order filter on derivative path
Maximum Output Umax	Maximum output limit
Minimum Output Umin	Minimum output limit
Sampling Frequency	System sampling frequency, in Hz

9.13 RAMP_GEN: Ramp Generator

This block generates ramp output of adjustable gain, frequency, and dc offset.

Image:



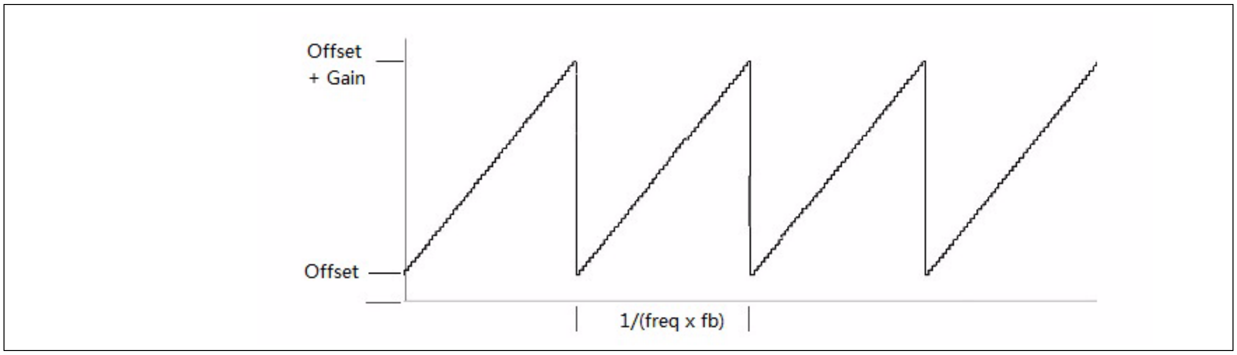
Attributes:

Parameters	**Description**
Base Frequency fb	Base frequency, in Hz
Sampling Frequency	Sampling frequency, in Hz

Input and Output Signals:

Name	**I/O**	**Description**
gain	Input	Ramp gain, in the range of 0 to 2.
0ffset	Input	Ramp offset, in the range of -1 to +1. The value of offset+gain must be less than 1.0.
freq	Input	Ramp frequency ratio. The ramp output frequency is freq*fb
	Output	Ramp signal output

The ramp output waveform is shown below.



The output ramp frequency is the multiple of the base frequency.

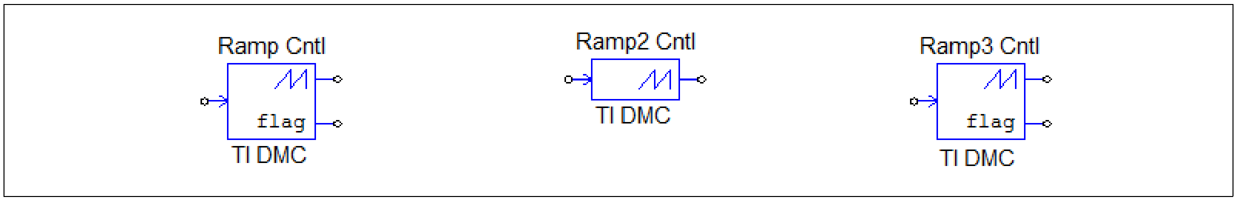
The output range is limited from -1 to +1. Therefore, the range of *Offset* is from -1 to +1 and the range of *Gain* is from 0 to 2. The value of *Offset+Gain* must be less than 1.0.

9.14 Ramp Control

There are three different ramp control blocks corresponding to the three ramp macros in the TI DMC library:

- RMP_CNTL: Implements a ramp up and down, with an output to flag when output equals input.
- RMP2CNTL: Implements a ramp up and down.
- RMP3CNTL: Implements a ramp down, with an output to flag when output equals input.

Images:



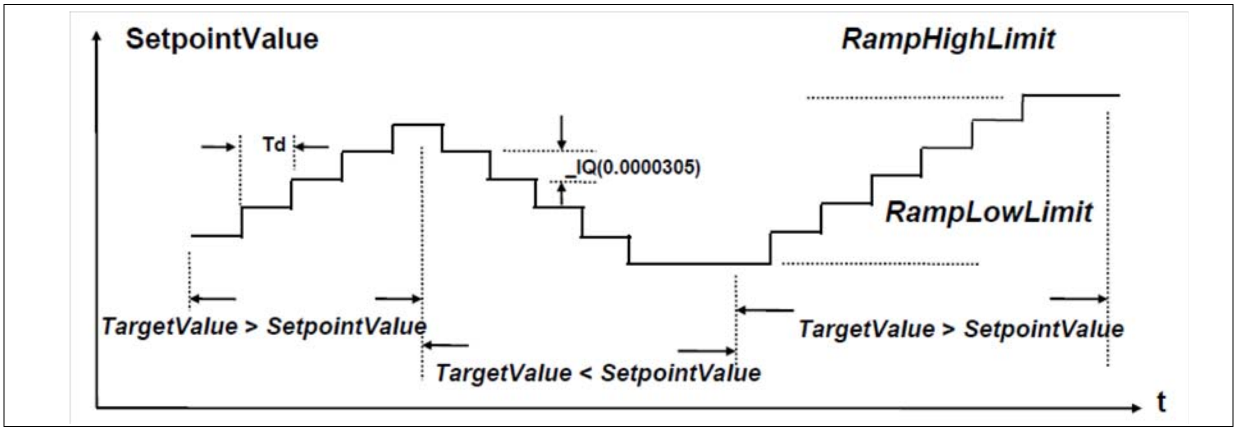
Input and Output Signals:

Name	I/O	Description
	Input	Target input value
	Output	Target output
flag	Output	Flag to indicate output equals input (not available for RMP2CNTL).

9.14.1 RMP_CNTL: Ramp Control

This block implements a ramp up and ramp down function with a flag to indicate when the output variable equals the input variable

- The ramp step = IQ(0.0000305).
- For input > output: output increments one ramp step after delay if it is blow the Maximum Limit.
- For input < output: output decrements one ramp step after delay if it is above the Minimum Limit.
- For input = output: the flag is set to 7FFFFFFFh.


SetpointValue
RampHighLimit
Td
_IQ(0.0000305)
RampLowLimit
TargetValue > SetpointValue
TargetValue < SetpointValue
TargetValue > SetpointValue
t

Example:

SetpointValue = 0 (initial value), TargetValue = 1000 (user specified), RampDelayMax = 500 (user specified), and sampling loop time period Ts = 0.000025 sec.

This means that the time delay for each ramp step is Td = 500x0.000025 = 0.0125 sec. Therefore, the total ramp time will be Tramp = 1000x0.0125 = 12.5 sec.

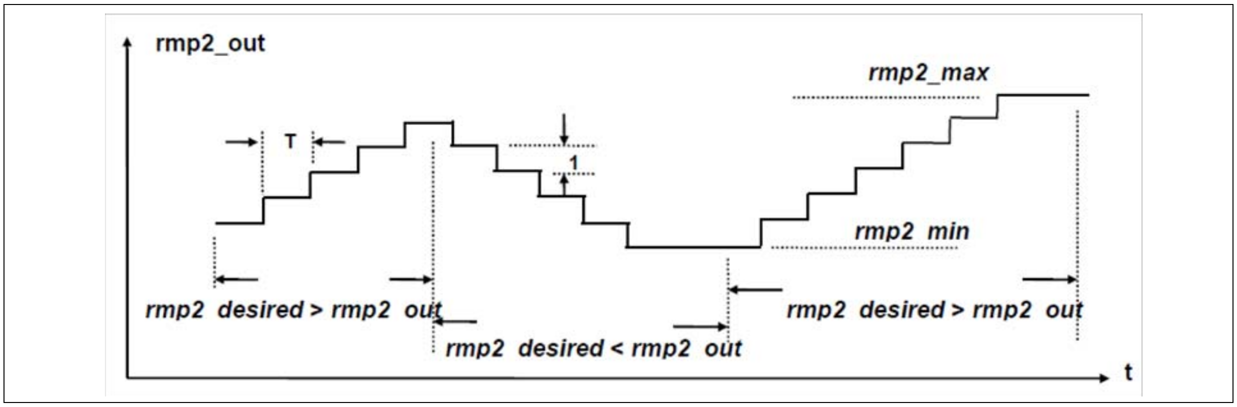
Attributes:

Parameters	**Description**
Max Delay Rate	Delay rate for each ramp step, in number of sampling time period
Minimum Output	Minimum output limit
Maximum Output	Maximum output limit

9.14.2 RMP2CNTL: Ramp 2 Control

This block implements a ramp up and ramp down function. The output follows the input.

- The ramp step = 1.
- If input > output: output increments one ramp step after delay if it is below the maximum output.
- If input < output: output decrements one ramp step after delay if it is above the minimum output.



Example:

Out = 0 (initial value), DesiredInput = 1000 (user specified), Ramp2Delay = 500 (user specified), and sampling loop time period Ts = 0.000025 sec.

This means that the time delay for each ramp step is Td = 500x0.000025 = 0.0125 sec. Therefore, the total ramp time will be Tramp = 1000x0.0125 = 12.5 sec.

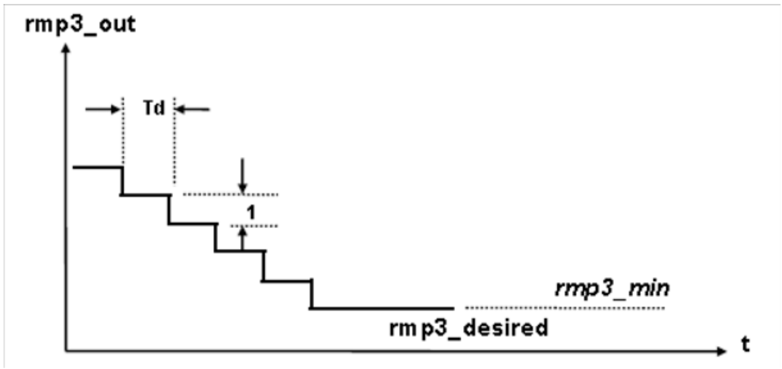
Attributes:

Parameters	Description
Delay in # of Period	The delay rate for each ramp step, in number of sampling time period.
Minimum Output	Minimum output limit
Maximum Output	Maximum output limit
Initial Output Value	Initial output value

9.14.3 RMP3CNTL: Ramp 3 Control

This block implements a ramp down function.

- The ramp step = 1.
- Output decrements one ramp step after delay if it is above the Minimum Limit.
- If output = input: the flag is set to 7FFFh.



Example:

Out=500(initial value), DesiredInput=20(user specified),

Ramp3Delay=100(user specified), sampling loop time period Ts=0.000025 Sec.

This means that the time delay for each ramp step is Td=100x0.000025=0.0025 Sec. Therefore, the total ramp down time will be Tramp=(500-20)x0.0025 Sec=1.2 Sec

Attributes:

Parameters	Description
Delay in # of Period	The delay rate for each ramp step, in number of sampling time period.
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit
Initial Output Value	Initial output value

9.15 SOMPOS: Sliding-Mode Rotor Position Observer

This block implements a rotor position estimation algorithm for permanent-magnet synchronous motor (PMSM) based on sliding-mod observer (SMO).

The sliding-mode rotor position observer of PMSM module requires two constants (Fsmopos and Gsmopos) to be input basing on the machine parameters, base quantities, mechanical parameters, and sampling period. These two constants are computed by the macro SMOPOS_CONST.

SimCoder combines SMOPOS and SMOPOS_CONST in this block to simplify the process. When a SMOPOS element is in a circuit schematic, SimCoder copies SMOPOS macro to the generated project folder and implements SMOPOS_CONST sub-module in SMOPOS's initial data structure definition.

The figure below is an illustration of the coordinate frames and voltage and current vectors of PMSM, with *a*, *b* and *c* being the phase axes, α and β being a fixed Cartesian coordinate frame aligned with phase a, and *d* and *q* being a rotating Cartesian coordinate frame aligned with rotor flux. *vs*, *is* and *es* are the motor phase voltage, current and back emf vectors (each with two coordinate entries). All vectors are expressed in α-β coordinate frame for the purpose of this discussion. The α-β frame expressions are obtained by applying Clarke transformation to their corresponding three phase representations.

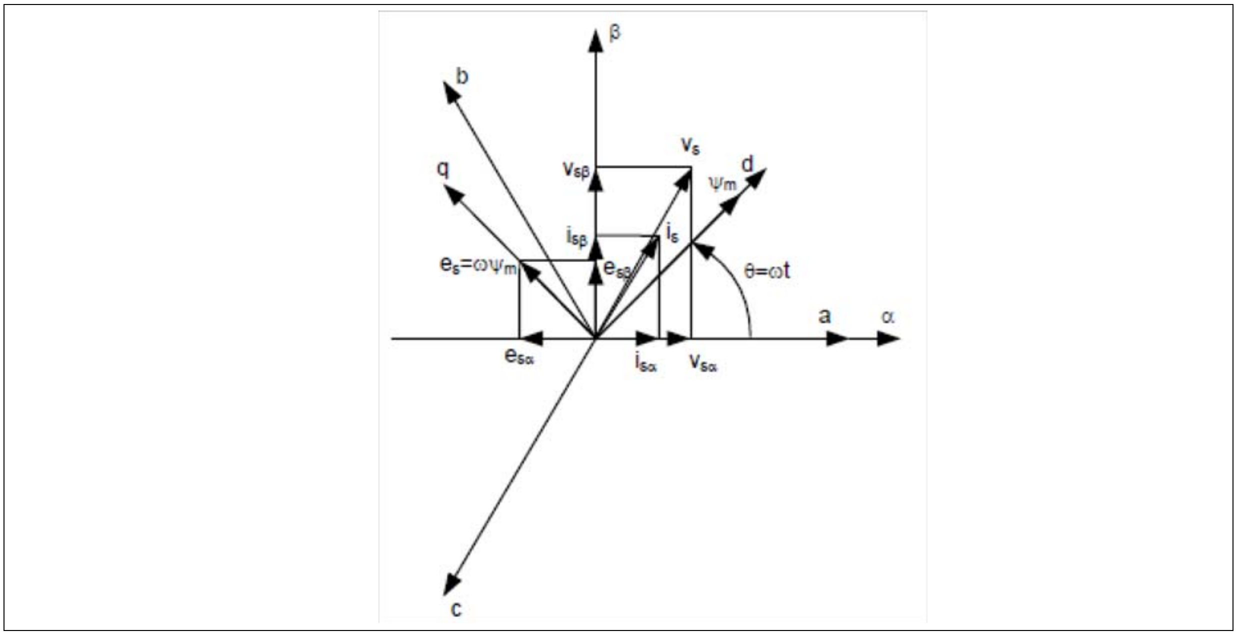
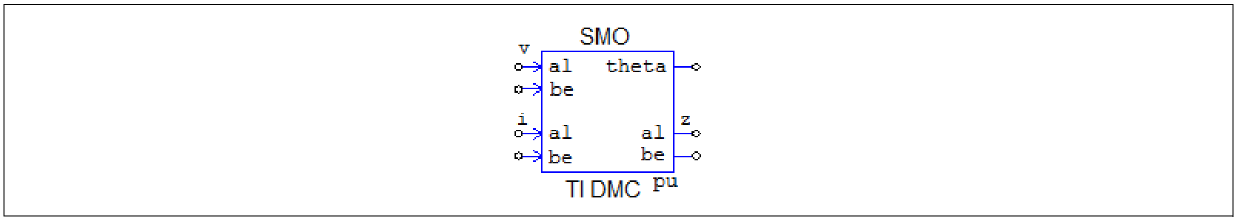


Image:



Attributes:

Parameters	**Description**
Sliding-Mode Control Gain	Sliding-mode control gain (Kslide).
Sliding-Mode Filter Gain	Sliding -mode filter gain (Kslf).
Stator Resistance Rs	Stator resistance, in ohm.
Stator inductance Ls	Stator inductance, in H.
Base Phase Current Ib	Base phase current, in Amp.
Base Phase Voltage Vb	Base phase voltage, in volt.
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

All input and output signals are in per unit.

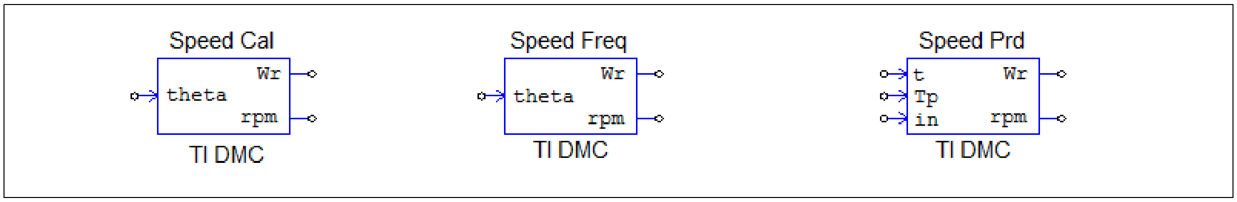
Name	I/O	Description
v-al	Input	Stationary alpha-axis stator voltage
v-be	Input	Stationary beta-axis stator voltage
i-al	Input	Stationary alpha-axis stator current
i-be	Input	Stationary beta-axis stator current
theta	Output	Compensated rotor position angle
z-al	Output	Stationary alpha-axis stator sliding control
z-be	Output	Stationary beta-axis stator sliding control

9.16 Speed Calculators

SimCoder built three different Speed Calculator blocks corresponding to the three macros in the TI DMC library:

- SPEED_EST: calculates the motor speed based on the estimated rotor position when the rotation direction information is not available.
- SPEED_FR: calculates the motor speed based on a rotor position measurement from QEP sensor.
- SPEED_PRD: calculates the motor speed based on a signal's period measurement.

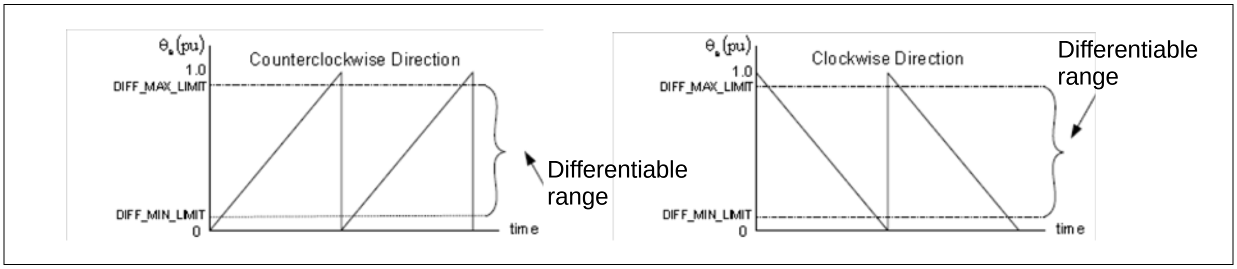
Images:



9.16.1 SPEED_EST: Speed Calculator

This block calculates the motor speed based on the estimated rotor position angle when the rotation direction information is not available. A first-order low-pass filter is used at the output variable.

The typical waveforms of the electrical rotor position angle θ in both directions are shown below. Assuming the direction of rotation is not available. To take care the discontinuity of angle from 360 to 0 degree (CCW) or from 0 to 360 degree (CW), the differentiator is simply operated only within the differentiable range as seen in this Figure. This differentiable range does not significantly lose the information to compute the estimated speed. In the figure below, θ = 1.0 pu = 360 degree.



Attributes:

Parameters	Description
Filter Cut-off Frequency fc	The cut-off frequency of the low-pass filter at the output variable, in Hz
Base Frequency fb	Base frequency, in Hz
Bask Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

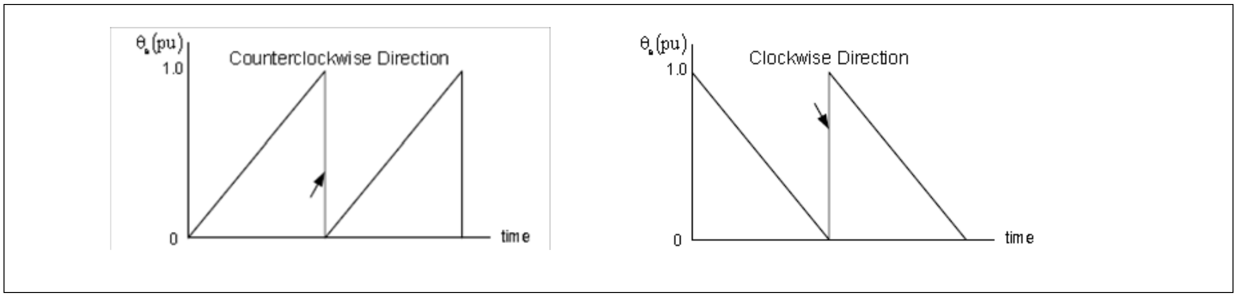
Input and Output Signals:

Name	I/O	Description
theta	Input	Estimated rotor position angle, in degrees from 0 to 360.
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

9.16.2 SPEED_FR: Speed Calculator with QEP Sensor

This block calculates the motor speed based on the rotor position measurement from QEP sensor. A first-order low-pass filter is used at the output variable.

The typical waveforms of the electrical rotor position angle q in both directions can be shown as in the figure below. Assuming the direction of rotation is not available, speed is estimated based on differentiation of angular values between successive iterations. To take care of the discontinuity of angle from 360 to 0 degrees (CCW) or from 0 to 360 degrees (CW), an error roll over to fit the difference numerically within -180 and +180 degrees is performed.



Attributes:

Parameters	Description
Filter Cut-off Frequency fc	The cut-off frequency of the low-pass filter at the output variable, in Hz
Base Frequency fb	Base frequency, in Hz
Bask Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

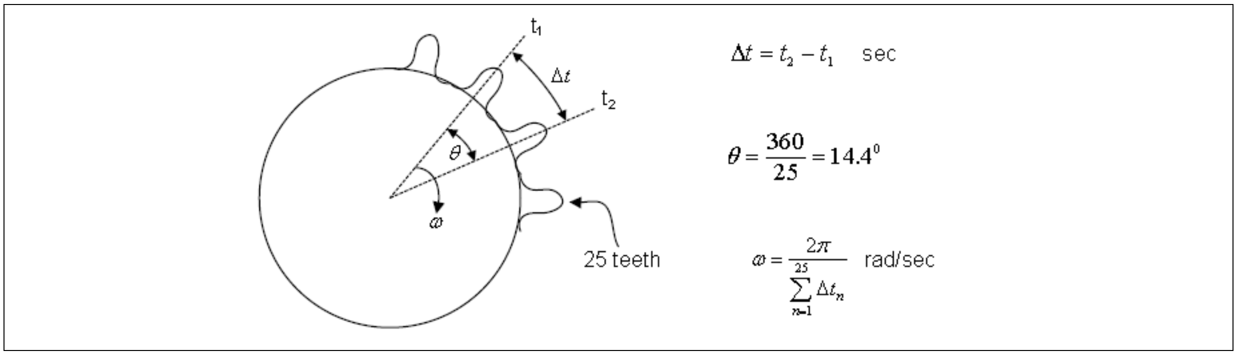
Name	I/O	Description
theta	Input	Estimated rotor position angle, in degrees from 0 to 360.
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

9.16.3 SPEED_PRD: Speed Calculator with Period Measurement

This block calculates the motor speed based on the measurement of a signal's period. Such a signal, for which the period is measured, can be the periodic output pulses from a motor speed sensor such as a shaft sprocket together with a Hall effect gear tooth sensor.

A low cost shaft sprocket with n teeth and a Hall effect gear tooth sensor is used to measure the motor speed. The figure below shows the physical details associated with the sprocket. The Hall effect sensor outputs a square wave pulse every time a tooth rotates within its proximity. The resultant pulse rate is n pulses per revolution. The Hall effect sensor output is fed directly to the Capture input pin. The capture unit will capture (the value of it.s base timer counter) on either the rising or the falling edges (whichever is specified) of the Hall effect sensor output. The captured value is passed to this s/w module through the variable called TimeStamp.

In this module, every time a new input TimeStamp becomes available it is compared with the previous TimeStamp. Thus, the tooth-to-tooth period (t2-t1) value is calculated. In order to reduce jitter or period fluctuation, an average of the most recent n period measurements can be performed each time a new pulse is detected.


t_1
Δt
t_2
θ
ω
25 teeth
$\Delta t = t_2 - t_1$ sec
$\theta = \frac{360}{25} = 14.4^0$
$\omega = \frac{2\pi}{\sum_{n=1}^{25} \Delta t_n}$ rad/sec

Attributes:

Parameters	Description
# of Sprocket Teeth	The number of sprocket teeth, an integer number
Bask Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

Name	I/O	Description
t	Input	Time stamp for a capture event.
Tp	Input	Event period between time stamps.
in	Input	Input selection. 0 for time stamp; 1 for event period
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

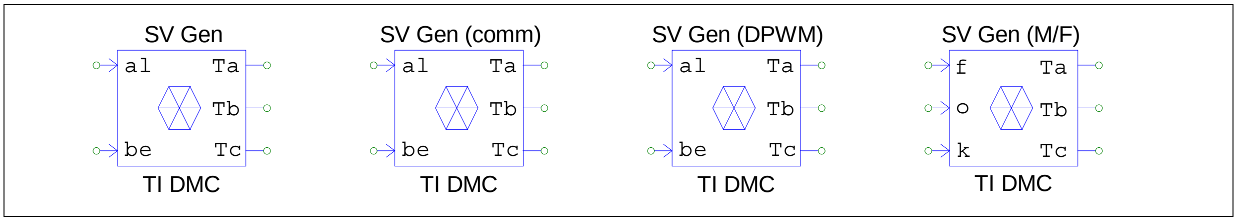
9.17 Space Vector Generators

Space vector generators calculates the appropriate duty ratios needed to generate a given stator voltage using space vector PWM technique. SimCoder built different space vector blocks corresponding to the two macros in the TI DMC library:

- **SVGEN**: the stator reference DPW voltage is calculated from its alpha-beta components.
- **SVGEN_COMM**: The stator reference voltage is calculated using common mode voltage.

- **SVGEN_DPWM**: Different than regular SVGEN, this modulation technique keeps one of the three switches off during the entire 120 degrees to minimize switching losses.
- **SVGEN_MF**: The stator reference voltage is calculated from its magnitude and frequency.

Images:



Input and Output Signals:

Name	I/O	Description
al	Input	Reference alpha-axis phase voltage.
be	Input	Reference beta-axis phase voltage.
Ta	Output	Duty ratio reference for phase-a switching function.
Tb	Output	Duty ratio reference for phase-b switching function.
Tc	Output	Duty ratio reference for phase-c switching function.

For Blocks SVGEN, SVGEN_COMM, and SVGEN_DPWM: There are no parameters for these blocks.

For Block SVGEN_MF:

Attributes:

Parameters	Description
Base Frequency	Base frequency, in Hz
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

Name	I/O	Description
f	Input	Reference frequency, in Hz
o	Input	Reference offset voltage, in volt
k	Input	Reference gain voltage
Ta	Output	Duty ratio reference for phase-a switching function.
Tb	Output	Duty ratio reference for phase-b switching function.
Tc	Output	Duty ratio reference for phase-v switching function.

9.18 VHZ_PROFILE: Volt/Hertz Profile for AC Induction Motors

This block generates an output voltage for a specific input frequency according to the specific volt/hertz profile. This is used for variable speed implementation of AC induction motor drives.

The relationship between the output voltage and the input frequency is illustrated below.

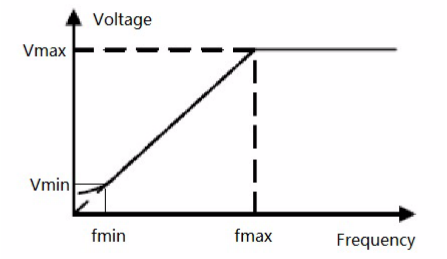
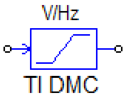


Image:



Attributes:

Parameters	Description
Low Frequency (pu)	Low frequency f_{min}, in per unit
High Frequency (pu)	High frequency f_{rated} at the rated voltage V_{rated}, in per unit
Maximum Frequency (pu)	Maximum frequency, in per unit
Rated Voltage (pu)	Rated voltage V_{rated}, in per unit
Low Freq. Voltage (pu)	Voltage at the low frequency f_{min}, in per unit

Input and Output Signals:

Name	I/O	Description
	Input	Frequency
	Output	Voltage

Index