SimCoder™

User's Guide

Powersim Inc.

SimCoder User's Guide

Version 2.0

Release 1

March 2010

Copyright © 2008-2010 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

1 SimCoder Overview

1.1 Introduction 1

1.2 Supported Hardware Targets 1

1.3 Elements for Code Generation 1

2 Code Generation - A Step-by-Step Approach

2.1 Overview 3

2.2 System in Continuous Domain 3

2.3 System in Discrete Domain 4

2.4 Code Generation without Hardware Target 5

2.5 Code Generation with Hardware Target 7

2.6 System with Event Control 10

3 Code Generation for Sub-Systems

3.1 Input Restriction 13

3.2 Code Generation 14

3.3 Simulating Sub-System Using Generated Code 15

4 Event Handling

4.1 Basic Concept 17

4.2 Elements for Event Handling 17

4.3 Restrictions on Subcircuits with Events 18

5 SimCoder Libraries

5.1 Elements from Standard PSIM Library 21

5.1.1 Defining Global Parameters in Parameter File 23

5.1.2 Generating Sawtooth Waveform 24

5.2 Event Control Elements 24

5.2.1 Input Event 24

5.2.2 Output Event 25

5.2.3 Default Event 26

5.2.4 Event Connection 26

5.2.5 Flag for Event Block First Entry 26

5.3 Global Variable 27

5.4 Interrupt 28

6 TI F28335 Hardware Target

6.1 PWM Generators 33

6.2 Start PWM and Stop PWM 40

6.3 Trip-Zone and Trip-Zone State 40

6.4 A/D Converter 41

6.5 Digital Input and Digital Output 45

6.6 Up/Down Counter 45

6.7 Encoder and Encoder State 46

6.8 Capture and Capture State 47

6.9 DSP Clock 48

6.10 Hardware Board Configuration 48

6.11 Project Settings and Memory Allocation 49

7 PE-Pro/F28335 Hardware Target

7.1 PWM Generators 53

7.2 Start PWM and Stop PWM 55

7.3 A/D Converter 56

7.4 D/A Converter 57

7.5 Digital Input / Encoder / Trip-Zone 58

7.6 Digital Output / Single PWM 59

8 PE-Expert3 Hardware Target

8.1 PEV Board 61

8.1.1 PWM Generators 61

8.1.2 A/D Converter 64

8.1.3 Digital Input / Capture / Counter 64

8.1.4 Digital Output 66

8.1.5 Encoder 66

8.2 LED Output 66

8.3 PE-Expert3 Runtime Library Functions 67

9 General Hardware Target

9.1 PWM Generators 69

9.2 A/D Converter 71

9.3 D/A Converter 73

9.4 Digital Input and Output 73

9.5 Encoder 74

9.6 Capture 75

Index 77

1

SimCoder Overview

1.1 Introduction

SimCoder[1] is an add-on option of the PSIM software for automatic code generation. Using SimCoder, one can simulate a system in PSIM, and automatically generate C code for specific target DSP (digital signal processor) hardware platform.

Automatic code generation using SimCoder greatly speeds up the design process, and reduces development time and cost.

This manual describes how to use SimCoder.

1.2 Supported Hardware Targets

SimCoder supports the following hardware targets:

- *TI F28335 Hardware Target*:

  With this target, SimCoder can generate code for hardware that uses the floating-point DSP TMSF28335 from Texas Instruments (TI).

- *PE-Pro/F28335 Hardware Target*:

  PE-Pro/F28335 is a DSP development platform made by Myway Co. (www.myway.co.jp). It uses TI's floating-point DSP TMSF28335 and Myway's PE-OS library. With this target, SimCoder can generate code that is ready to run on the PE-Pro/F28335 DSP board.

- *PE-Expert3 Hardware Target*:

  PE-Expert3 is a DSP development platform made by Myway Co. It uses TI's floating-point DSP TMS320C6713 and Myway's PE-OS library. With this target, SimCoder can generate code that is ready to run on the PE-Expert3 DSP hardware.

- *General Hardware Target*:

  With this target, SimCoder can generate code for generic-type hardware platform. After the code is generated, users can add their own code to this code and adopt it for their specific hardware.

Elements for these hardware targets can be found under the menu **Elements** -> **SimCoder for Code Generation**, under the sub-menus **TI F28335 Target**, **PE-Pro/F28335 Target**, **PE-Expert3 Target**, and **General Hardware Target**.

1.3 Elements for Code Generation

All the elements under the menus **Elements** -> **Event Control** and **Elements** -> **SimCoder for Code Generation** are for code generation. Besides, some 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 will appear next to these elements that can be used for code generation.

Also, if the same option box is checked, the image will appear next to the elements of the TI F28335 Target, next to the elements of the PE-Pro/F28335 Target. next to the elements of the PE-Expert3 Target, and next to the elements of the General Hardware Target.

1. SimCoder™ is a trademark of Powersim Inc., and is copyright by Powersim Inc., 2008-2010

For a list of elements in the standard library that can be used for code generation, please refer to Section 5.1.

2

Code Generation - A Step-by-Step Approach

2.1 Overview

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

- Simulate a system in PSIM with the control in continuous domain.
- Simulate the system with the control in discrete domain.
- If there is no hardware target, place the control 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.

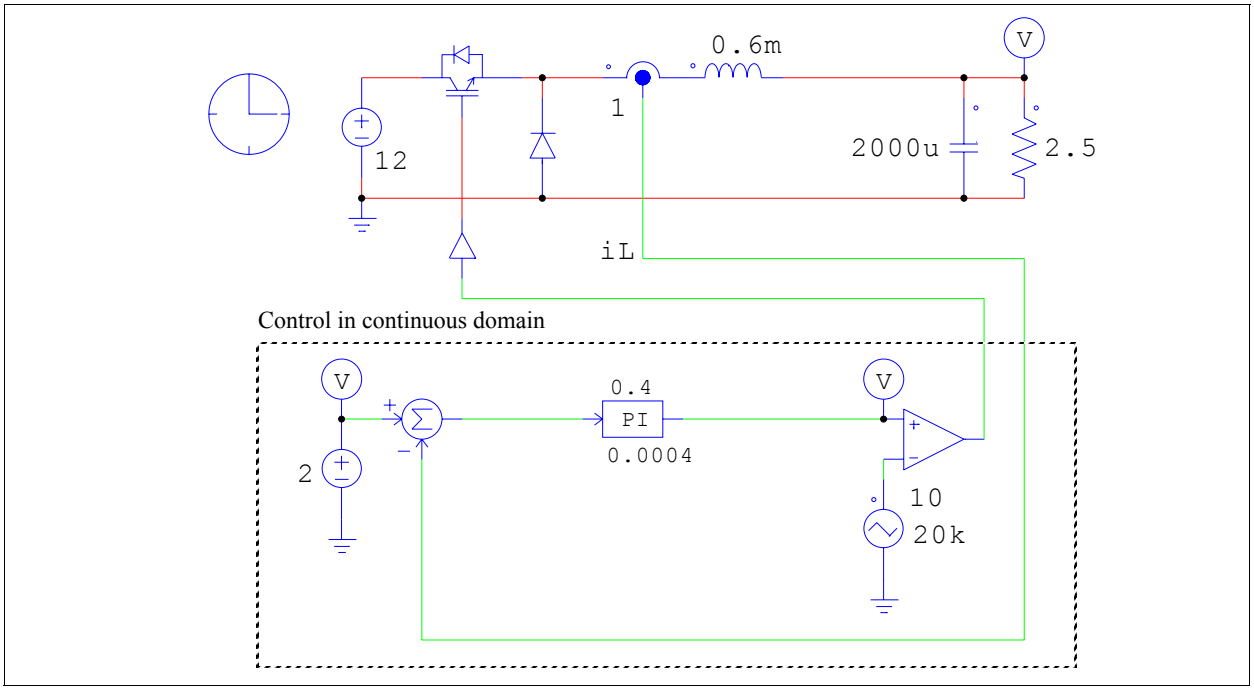
Please note that code can only be generated when control is in discrete domain, not in continuous domain. Therefore, Digital control Module is needed for SimCoder.

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.



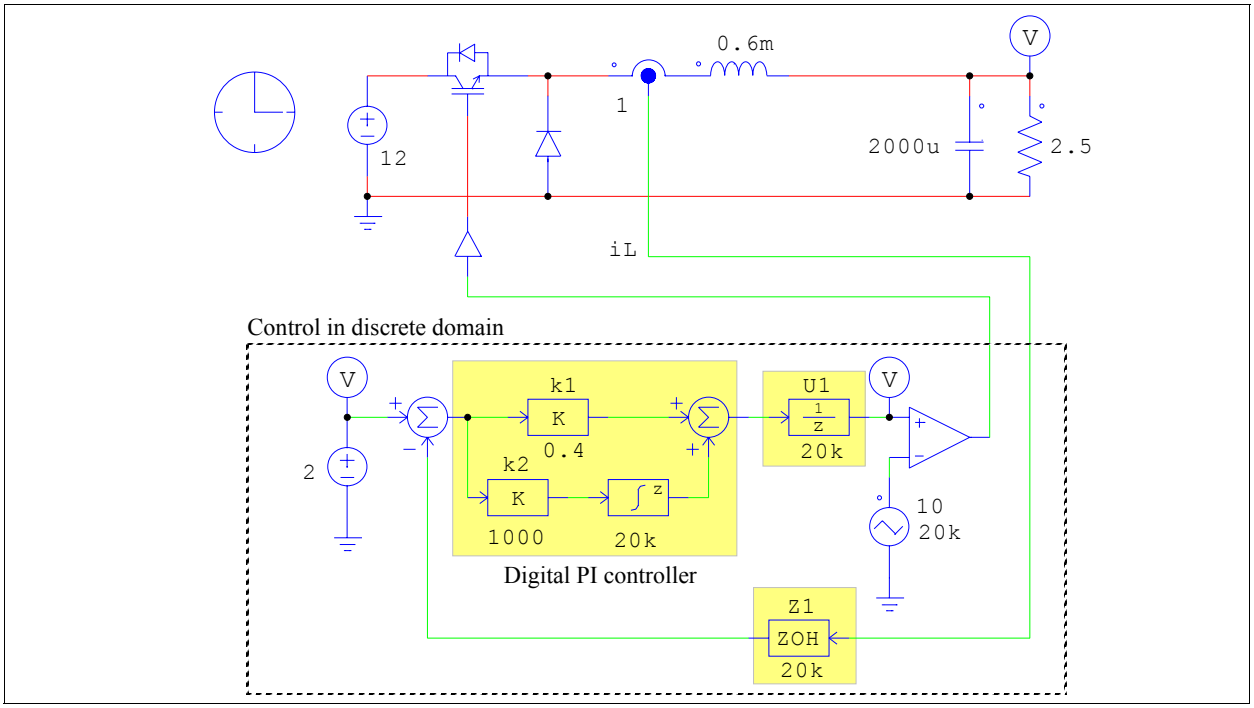
2.3 System in Discrete Domain

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.

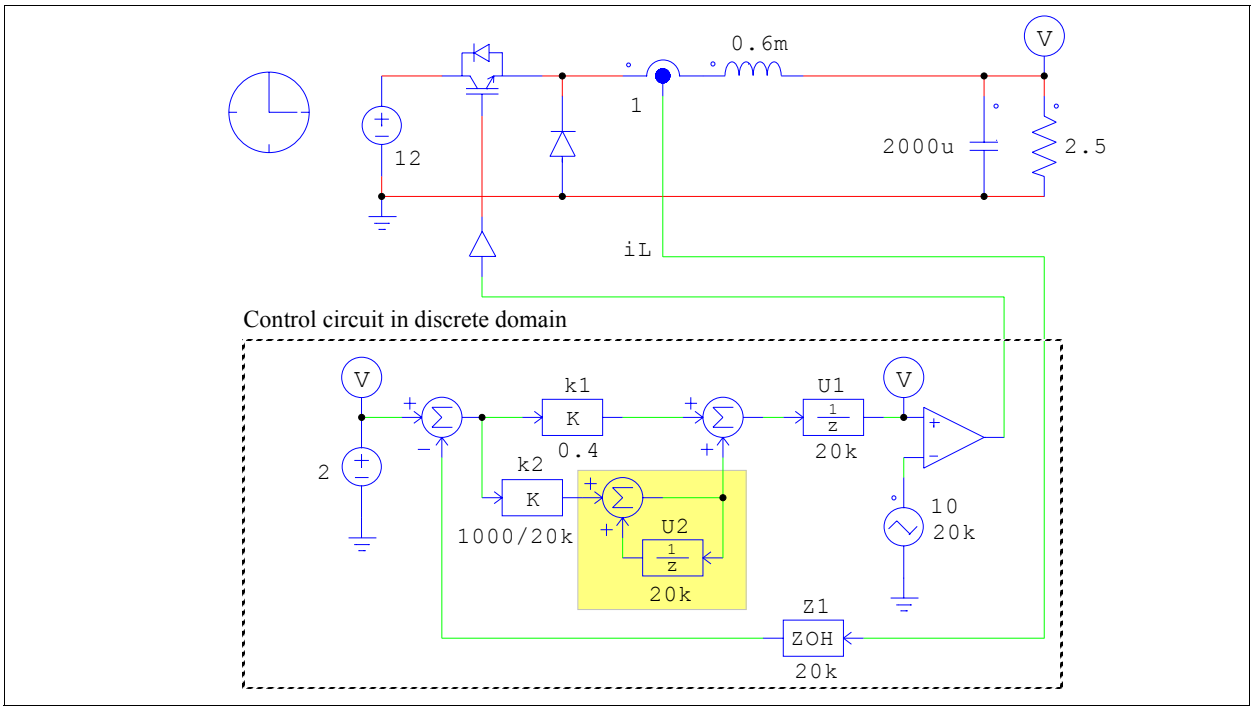
Note that the converted digital controller should result in a stable control loop and desired performance. If the simulation results with the digital control are not stable or not as desired, one needs to go back to the analog control system, re-design the analog controller, and repeat the process.

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:

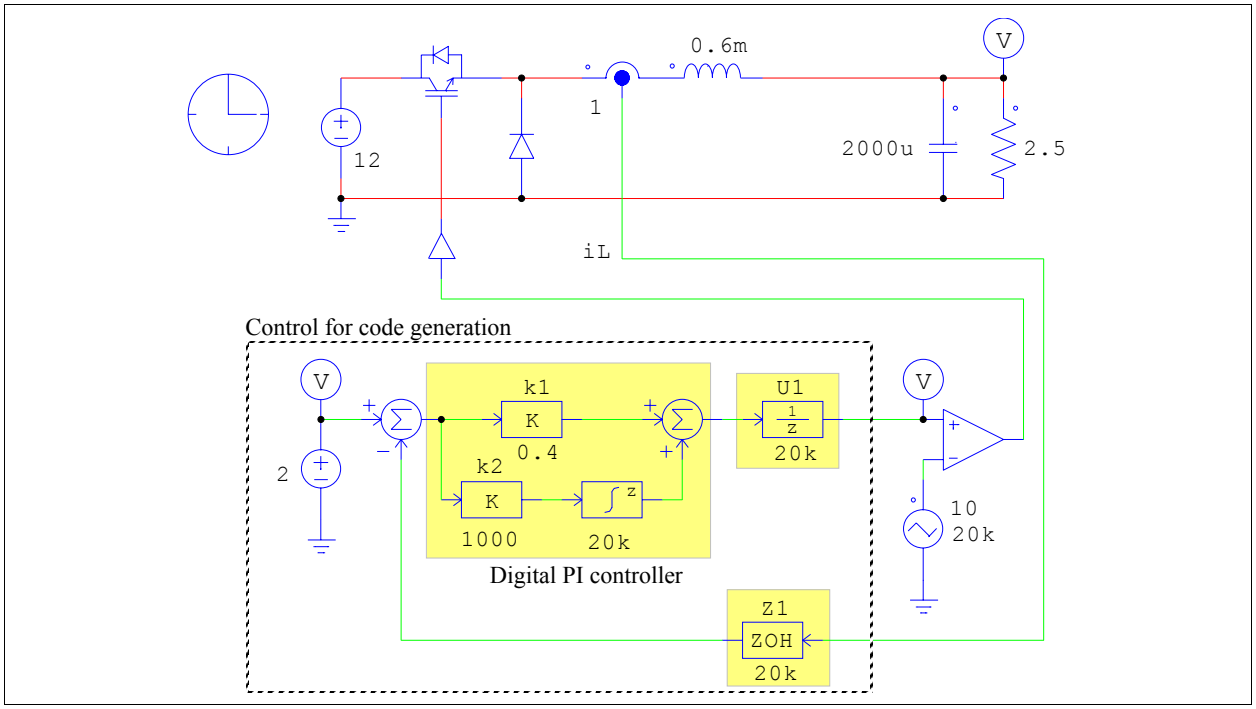


Note that, due to the factor T_s in the equation, the gain of the proportional block k_2 needs to be divided by the sampling frequency of 20kHz. The advantage of this circuit is that it is easier to start or stop the integration of the integrator.

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

2.4 Code Generation without Hardware Target

SimCoder can generate code for a system without hardware target. For the system in Section 2.3, for example, we can generate the code for the control circuit. The system is redrawn as below.

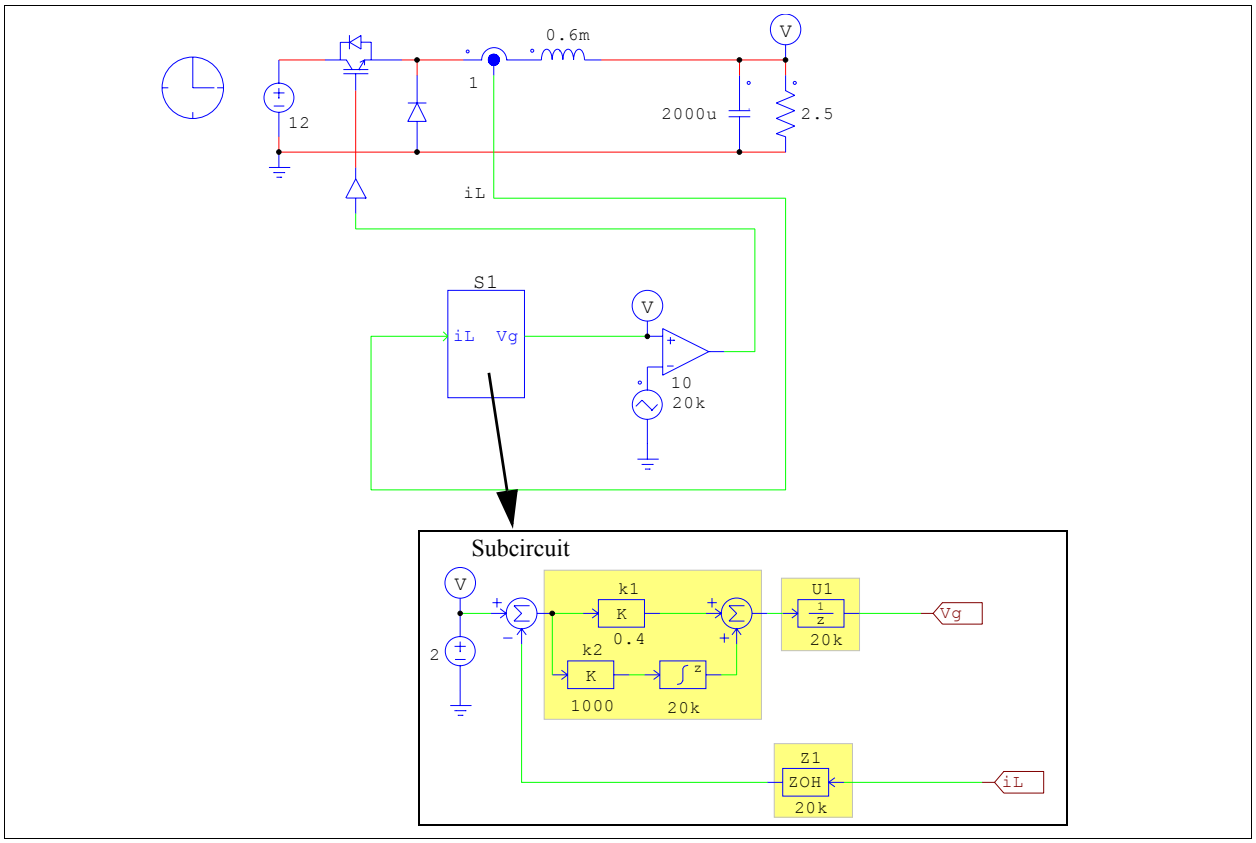


Note that we will generate the code only for the part of the control circuit within the dotted box. The reason why the comparator and the triangular wave source are excluded is that, if they were included, both of them will be treated as having a sampling rate of 20 kHz, and will be calculated only once in a period of 20 kHz, contrary to the fact that they must be calculated at every time step.

The reason for this is because, in the code generation, the sampling rate of every element must be defined. Since the comparator has two inputs, one with the 20 kHz sampling rate, and the other (triangular wave source) undefined. SimCoder will assume that the comparator will have the rate of the input that is defined.

To generate the code for the control circuit in the dotted box, we need to place the circuit in a subcircuit first. To create the subcircuit, select the circuit in the dotted box. Right click, select **Create Subcircuit** from the menu, and define the subcircuit file name.

After adjustment of the port location and the wiring, the circuit and the subcircuit appear as below.



To generate the code for the subcircuit, in the main circuit, right click on top of the subcircuit, choose **Attributes** to bring out the property dialog window. In the **Subcircuit Variables** tab, click on **Generate Code**.

An excerpt of the generated code is shown below. This code can be used in the C block for simulation. For more information on how to use the code for simulation, please refer to Chapter 3.

Users 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.

```
/*********************************************************************
// This code is created by SimCoder Version 1.0
//
// SimCoder is copyright by Powersim Inc., 2008
//
// Date: June 23, 2008 15:10:04
*********************************************************************/

#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 C block for the generated code has the following additional output port(s):
 2 - S1.iref
*/
void _SetVP5(int bRoutine, DefaultType fVal);
void InitInOutArray()
{
#if (1 > 0)
 inAry = new DefaultType[1];
#else
 inAry = NULL;
#endif
#if (2 > 0)
 outAry = new DefaultType[2];
#else
 outAry = NULL;
#endif
}

void FreeInOutArray()
{
 if (inAry != NULL)
  delete[] inAry;
 if (outAry != NULL)
  delete[] outAry;
}
... ... ... ...
```

2.5 Code Generation with Hardware Target

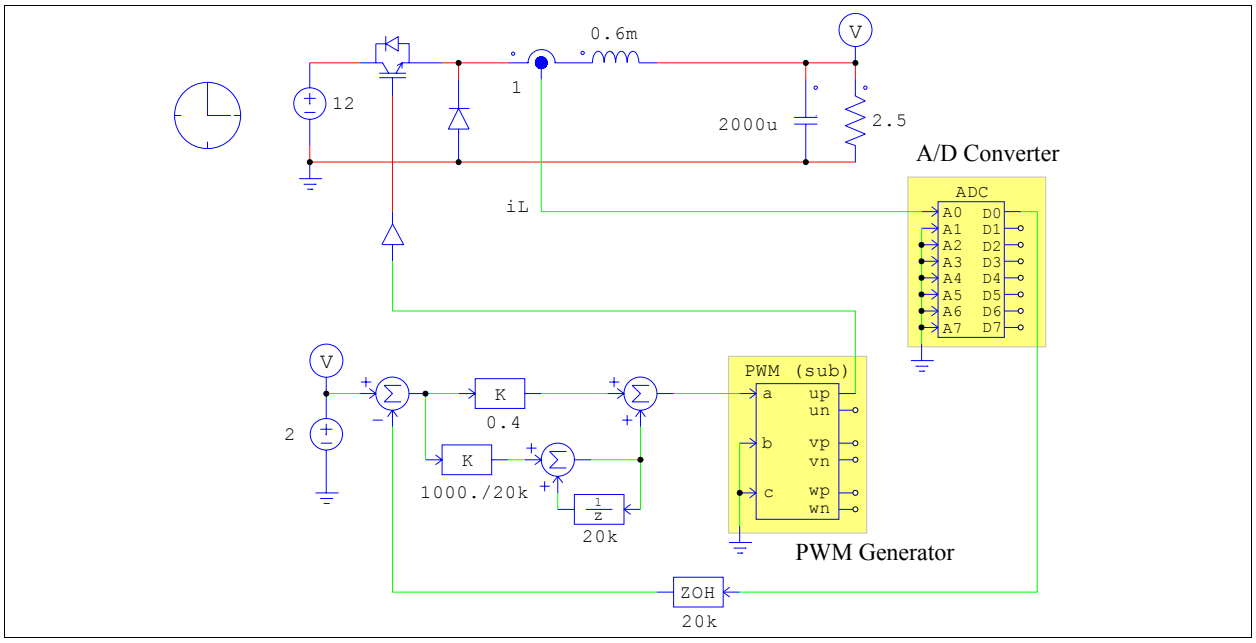
SimCoder can generate code for specific hardware target. Before the code can be generated, the system must be modified to include relevant hardware elements. Also, values may need to be scaled properly to take into account the value ranges of the hardware elements.

In general, changes to the schematic involve the following:

- Adding A/D converters, digital input/output, etc.;
- Replacing the PWM generation circuit with the hardware PWM generator;
- Adding event sequence control if necessary.
- Specify the hardware target in **Simulation Control**.

Below is the circuit after Myway PE-Expert3 hardware elements are added. For better illustration, the hardware

elements are highlighted in yellow.



As compared to the system with digital control, the following changes are made here:

- After the current sensor, an A/D converter is added. Note that one needs to pay attention to the input range of the A/D converter. If the current sensor output is out of the range of the A/D converter, the sensor signal must be scaled accordingly. In this case, the current sensor output is within the A/D converter range of -5V to +5V, and scaling is not needed.

  In this example, the parameter "AD0 Range" of the A/D converter is set to 5 (with the setting of 5, the gain of the A/D output v.s. the input is 1).

- The comparator and the carrier waveform for PWM generation in the previous circuit is replaced by a hardware PWM generator element. Note that since the hardware PWM generator already contains one sampling period delay inherently, the unit delay block U_1 at the input of the comparator in the original circuit is removed.

  For the PWM generator, the parameter "*Carrier Frequency*" is set to 20 kHz, "*Peak Value*" is set to 10V (the same as in the triangular carrier source in the previous circuit), and "*Start PWM at Beginning*" is set to "Start".

- In **Simulation Control**, the parameter "*Hardware Type*" is set to "Myway_PEExpert3", with either "PE-View8" or "PE-View9" selected for PE-View version.

Again, users 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 in the dialog window.

To check the validity of the changes after the hardware elements are added, this system can be simulated. 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 selecting **Simulate** -> **Generate Code** in PSIM. The code generated for Myway's PE-Expert3 hardware is ready to run without any changes.

Below is a listing of the C code for the system above.

```
/*********************************************************************
// This code is created by SimCoder Version 1.0 for Myway PE-Expert3
//
// SimCoder is copyright by Powersim Inc., 2008
//
// Date: March 14, 2008 15:10:04
*********************************************************************/

#include.....<math.h>
#include.....<mwio3.h>
#define.....DEBUG  /* Comment it out if not watching data from PE_VIEW */

interrupt void Task();

float .....fGbliref = 0.0;
float          fGblUDELAY1 = 0.0;

interrupt void Task()
{
        float fVDC2, fPEV_ADC1, fPEV_ADC1_1, fPEV_ADC1_2, fPEV_ADC1_3, fZOH3, fSUM1, fP2;
        float fSUMP3, fUDELAY1, fP1, fSUMP1, fPEV_PWM_11Div0, fPEV_PWM_11Div1, fPEV_PWM_11Div5;
        float fPEV_PWM_11Div2, fPEV_PWM_11Div6, fPEV_PWM_11Div3, fPEV_PWM_11Div7;
        pev_ad_start(0, 0);

        fUDELAY1 = fGblUDELAY1;
        fVDC2 = 2;

#ifdef  _DEBUG
        fGbliref = fVDC2;
#endif
        while (pev_ad_in_st(0, 0));
        pev_ad_in_grp(0, 0, &fPEV_ADC1, &fPEV_ADC1_1, &fPEV_ADC1_2, &fPEV_ADC1_3);

        fZOH3 = fPEV_ADC1;
        fSUM1 = fVDC2 - fZOH3;
        fP2 = fSUM1 * (1000./20000);
        fSUMP3 = fP2 + fUDELAY1;
        fGblUDELAY1 = fSUMP3;
        fP1 = fSUM1 * 0.4;
        fSUMP1 = fP1 + fSUMP3;
        fPEV_PWM_11Div0 = ((0+10)/2.0);
        fPEV_PWM_11Div1 = fSUMP1 - fPEV_PWM_11Div0;
        fPEV_PWM_11Div5 = fPEV_PWM_11Div1 * (2.0/(10));
        fPEV_PWM_11Div2 = 0 - fPEV_PWM_11Div0;
        fPEV_PWM_11Div6 = fPEV_PWM_11Div2 * (2.0/(10));
        fPEV_PWM_11Div3 = 0 - fPEV_PWM_11Div0;
        fPEV_PWM_11Div7 = fPEV_PWM_11Div3 * (2.0/(10));

        pev_inverter_set_uvw(0, 0, fPEV_PWM_11Div5, fPEV_PWM_11Div6, fPEV_PWM_11Div7);

#ifdef  _DEBUG
        watch_data();
#endif
}
```

If defined, values of voltage probes can be watched in PE-View.

The main interrupt service routine for 20 kHz

```
void Initialize(void)          The initialization routine
{
        pev_init(0);
        pev_ad_set_range(0, 0, 5, 1, 1, 1);
        pev_ad_set_range(0, 1, 1, 1, 1, 1);
        pev_inverter_init(0, 0, 20000, (4e-6)*1E9);
        pev_inverter_set_syncint(0, 0.0);
        int5_init_vector(Task);
        int5_enable_int();
        pev_inverter_enable_up_int5(0);
        pev_inverter_set_uvw(0, 0, 0, 0, 0);
        wait(100);
        pev_inverter_stop_pwm(0, 0);
}

void main()                    The main program
{
#ifdef  _DEBUG
        watch_init();
#endif
        int_disable();
        Initialize();
        int_enable();
        while (1) {
        }
}
```

The generated code has the following structure:

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

void Initialize (): This is the initialization routine. It initializes hardware.

Interrupt void Task (): This is the interrupt service routine for 20 kHz. At the end of each 20-kHz cycle, this routine will be called.

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.

Also, for testing and debugging purposes, a debug flag _DEBUG is also created in the code so that voltage probes in the PSIM schematic that connect to the code generation circuit can be displayed in real time in PE-View's scope environment. For example, in this case, the voltage probe connected to the 2-V dc source for the current reference can be displayed in PE-View's scope.

The code and necessary project files are stored in a sub-folder in the same directory as the main schematic file. One can then load the project file into Myway's PE-View environment, compile the code, and upload it onto the DSP hardware.

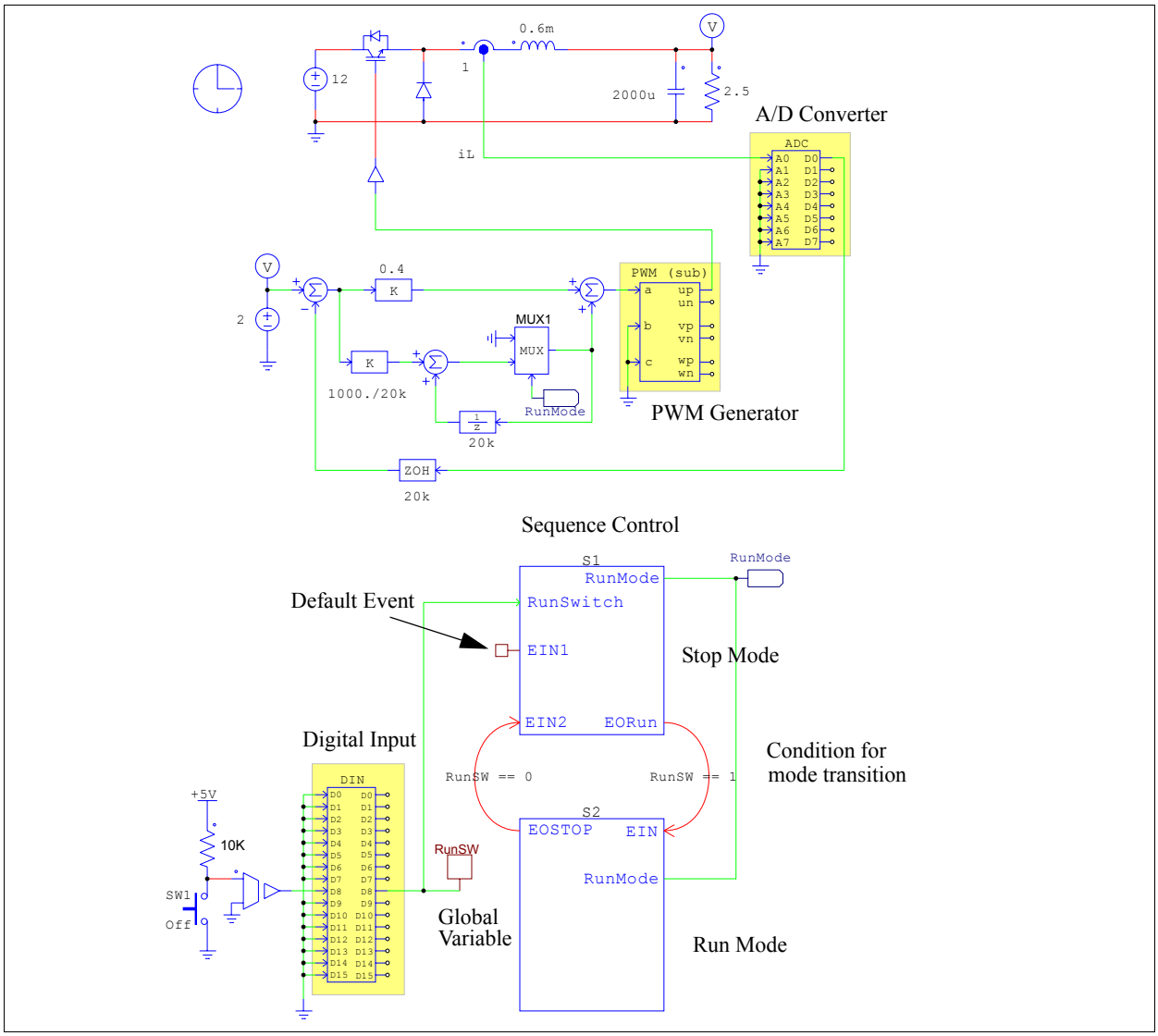
2.6 System with Event Control

Often a system may include event transition. That is, 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 4.

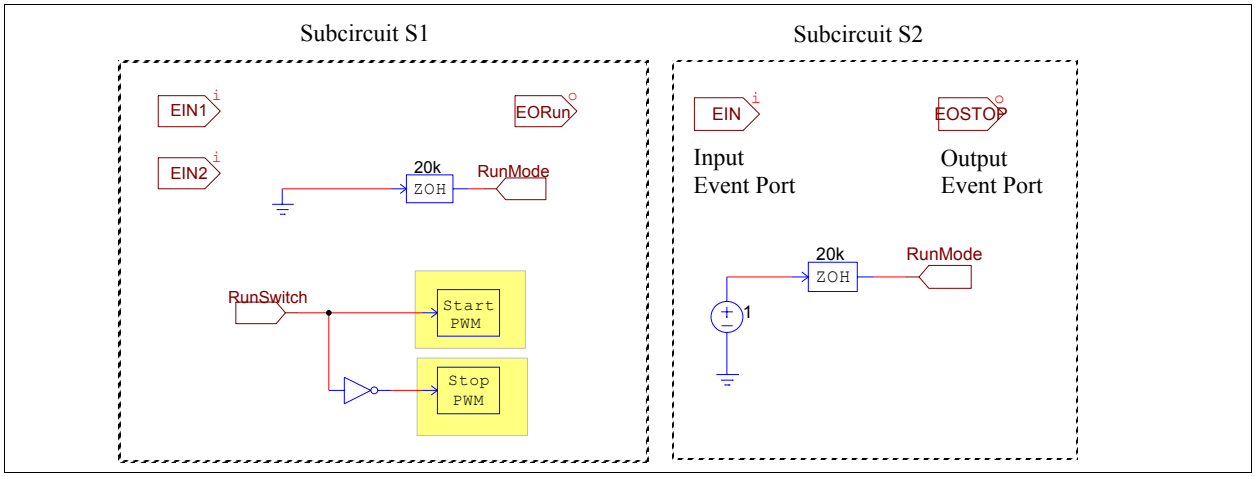
To illustrate how event control works, the system in Section 2.3 is changed with the following considerations:

- A manual switch is added 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 by changing the switch position.
- In the Stop Mode, the integrator should stop integrating, and the integrator output be reset to 0.

The system with the event control is shown below.



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



As compared to the system in Section 2.3, the following changes are made:

- Sequence control is added. This system has 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 *EOSTOP* 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, one can run the simulation to verify that the changes are correct. The results should be the same as the results of the system in Section 2.3.

3

Code Generation for Sub-Systems

3.1 Input Restriction

In PSIM, a sub-system is represented by a subcircuit. SimCoder has the capability to generate code for systems that contain sub-systems, or for sub-systems alone. There are, however, several restrictions on the sub-system for code generation, as described below:

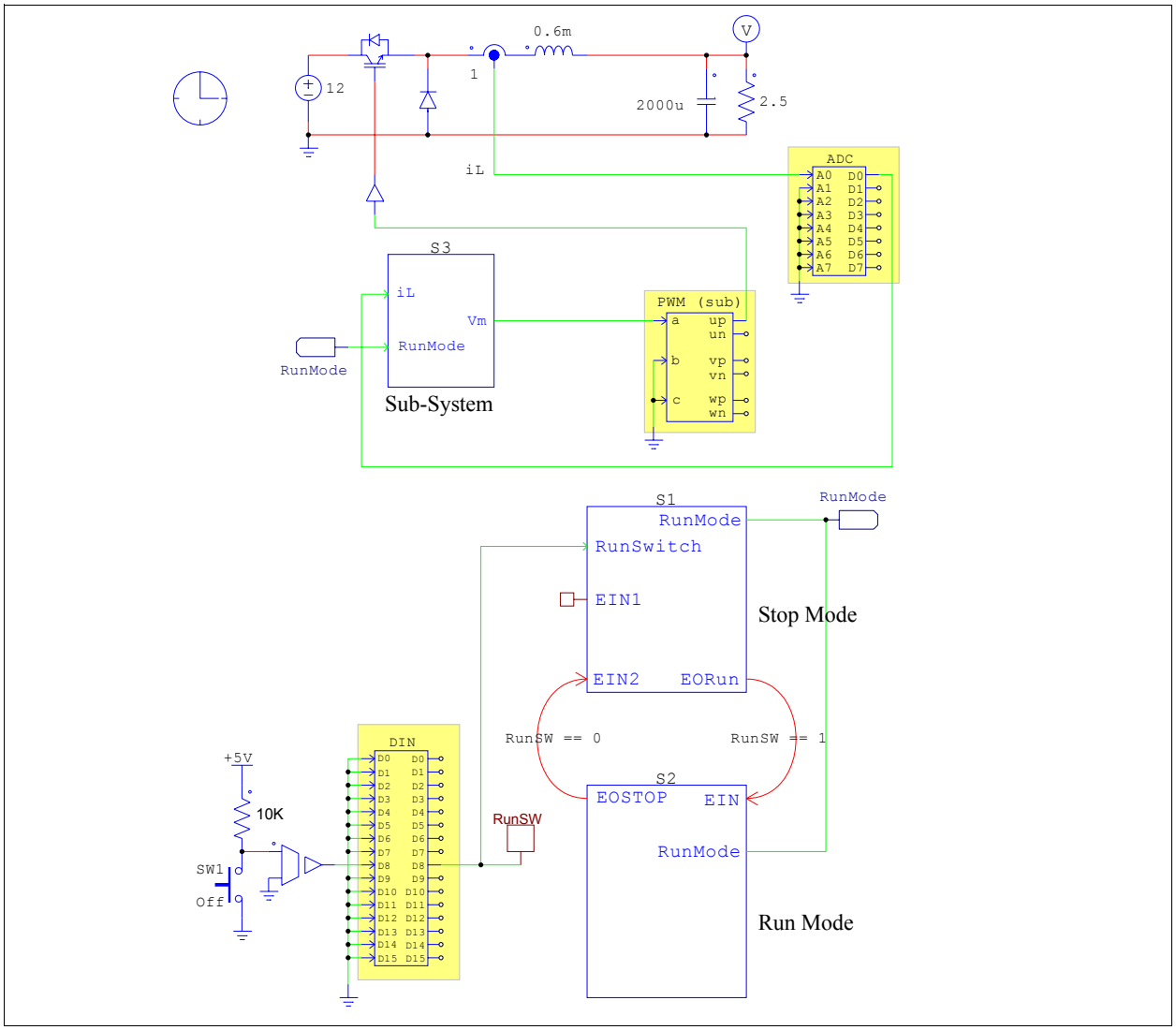
- All the elements in the sub-system must be supported for code generation. To find out if an element can be used for code generation, in PSIM, go to **Options** -> **Settings**, and check the box **Show image next to elements that can be used for code generation**. Any elements that have an image next 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.
- Hardware input/output elements (such as A/D converter, digital input/output, encoder, counter, and PWM generator) as well as hardware interrupt elements can not be placed inside a sub-system. They must be in the top-level main circuit only.
- If an input of the 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.

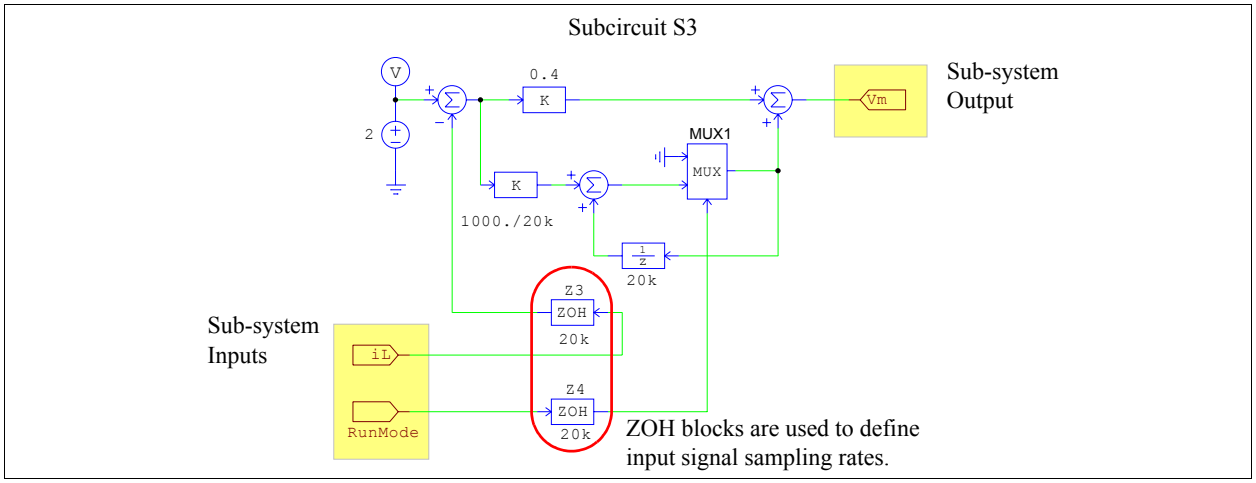
  For example, for a particular signal flow path, the sampling rate outside the sub-system is well defined. But inside the sub-system there are no discrete elements to indicate the sampling rate. In this case, a zero-order-hold block with the same sampling frequency must be connected to this input.

  If the input of the sub-system 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 that a zero-order-hold block be connected to each input that has a sampling rate to explicitly define its sampling rate.

To illustrate how the code is generated for sub-systems, we will generate the code for the current feedback and the PI controller portion of the system in Section 2.5. The system is redrawn as below, with the sub-system of interest as Subcircuit S3.

Subcircuit S3 has two inputs, i_L and *RunMode*, and one output, V_m. The sampling rates of i_L and *RunMode* are both 20 kHz. In the circuit of the Subcircuit S3, both inputs are connected to a zero-order-hold block. The zero-order-hold block Z3 already exists in the original system. However, the block Z4 is newly added.





3.2 Code Generation

To generate the code for a sub-system, from the circuit that calls the subcircuit, right click on top of the subcircuit. Choose **Attributes**. Go to the **Subcircuit Variables** tab, and click on **Generate Code**.

On the same dialog, if the checkbox **Replace subcircuit with generated code for simulation** is checked, the

block diagram inside the subcircuit will be replaced by the generated C code for simulation.

For this example, the generated code is shown below.

This sub-system has only one sampling rate. As a result, the generated code has only one function. The variables *fIn0* and *fIn1* correspond to the two inputs of the subcircuit i_L and *RunMode*, and the variable *fOut0* corresponds to the subcircuit output V_m.

Unlike the generated code for the whole system, the code for the sub-system does not have the main program and the initialization routine.

```
float       fGblS3_iref = 0.0;
float       fGblS3_UDELAY1 = 0.0;

void TaskS3(float fIn0, float fIn1, float *fOut0)
{
            float fS3_VDC2, fS3_Z3, fS3_SUM1, fS3_P1, fS3_P2, fS3_SUMP3, fS3_Z4, fS3_MUX1;
            float fS3_UDELAY1;

            fS3_UDELAY1 = fGblS3_UDELAY1;
            fS3_VDC2 = 2;
            fS3_Z3 = fIn0;
            fS3_SUM1 = fS3_VDC2 - fS3_Z3;
            fS3_P1 = fS3_SUM1 * 0.4;
            fS3_P2 = fS3_SUM1 * (1000./20000);
            fS3_SUMP3 = fS3_P2 + fS3_UDELAY1;
            fS3_Z4 = fIn1;
            fS3_MUX1 = (fS3_Z4 == 0) ? 0 : fS3_SUMP3;
            *fOut0 = fS3_P1 + fS3_MUX1;

#ifdef      _DEBUG
            fGblS3_iref = fS3_VDC2;
#endif
            fGblS3_UDELAY1 = fS3_MUX1;
}
```

3.3 Simulating Sub-System Using Generated Code

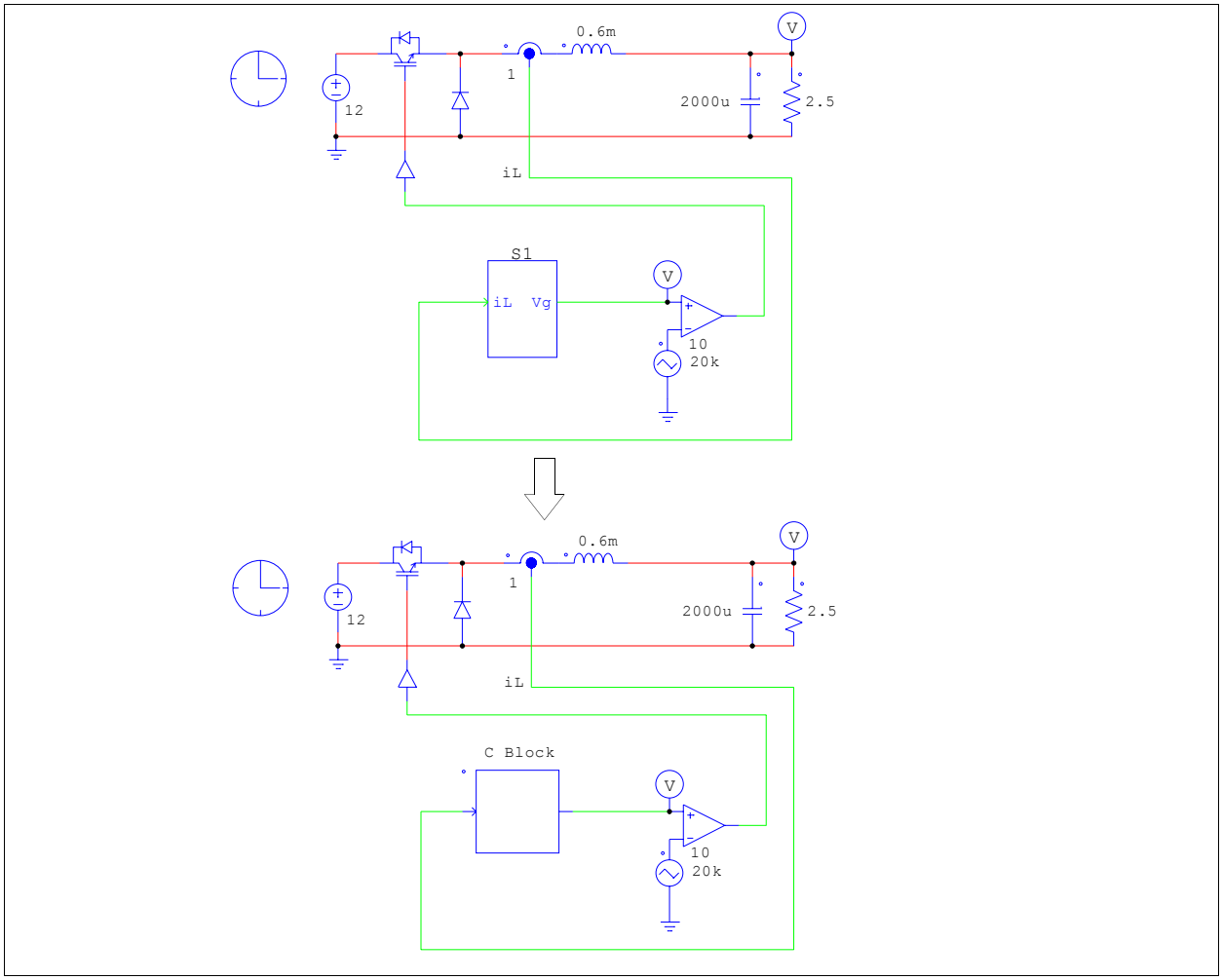
Once code is generated for a sub-system, it is possible to replace the sub-system with a C block (which can be found under **Elements** -> **Other** -> **Function Blocks**) using the generated C code. Note that this is possible only if the sub-system does not contain any hardware elements inside and it is not connected to any hardware elements in the circuit where it is called.

For example, the sub-circuit in the system in Section 2.4 can be replaced by a C block as shown below.

The generated code of the sub-system contains four sections: *RunSimUser* function, *OpenSimUser* function, *CloseSimUser* function; and the rest of the code.

Similarly, the C block input has four sections: *Variable/Function definitions*, *OpenSimUser Fcn*, *RunSimUser Fcn*, and *CloseSimUser Fcn*.

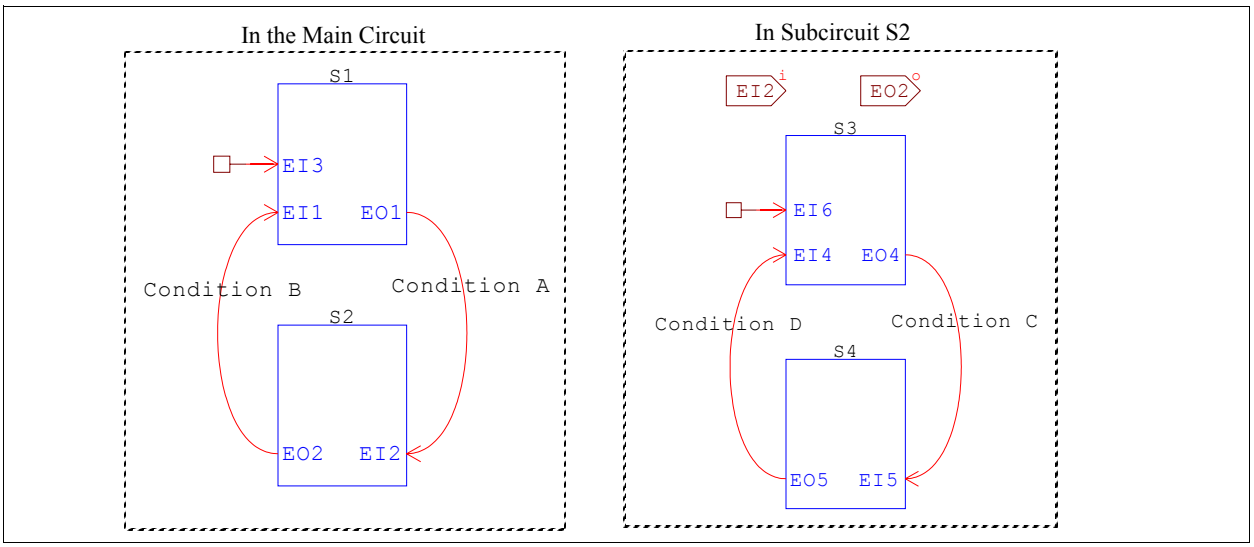
To use the generated code in the C block, simply copy the generated code from the *RunSimUser* function to the *RunSimUser Fcn* section in the C block, from the *OpenSimUser* function to the *OpenSimUser Fcn* section, from the *CloseSimUser* function to the *CloseSimUser Fcn* section, and from the rest of the code to the *Variable/ Function definitions* section. After this, the system with the C block can be simulated.


0.6m
V
12
1
2000u
2.5
iL
S1
iL Vg
V
10
20k
0.6m
V
12
1
2000u
2.5
iL
C Block
V
10
20k

4
Event Handling

4.1 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 of 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.

4.2 Elements for Event Handling

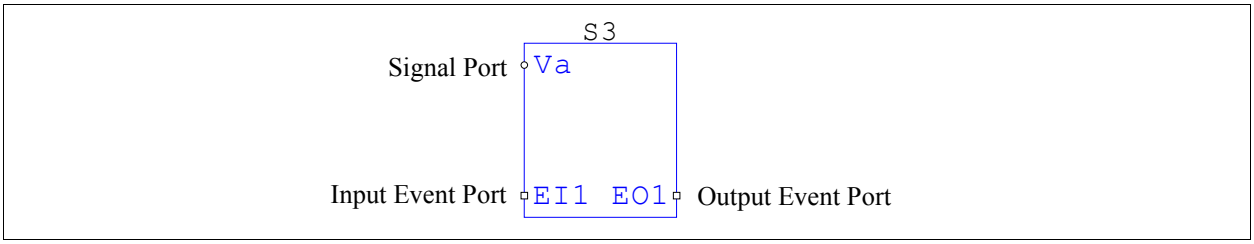
The following elements are used to define events and the state transition:

- 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.

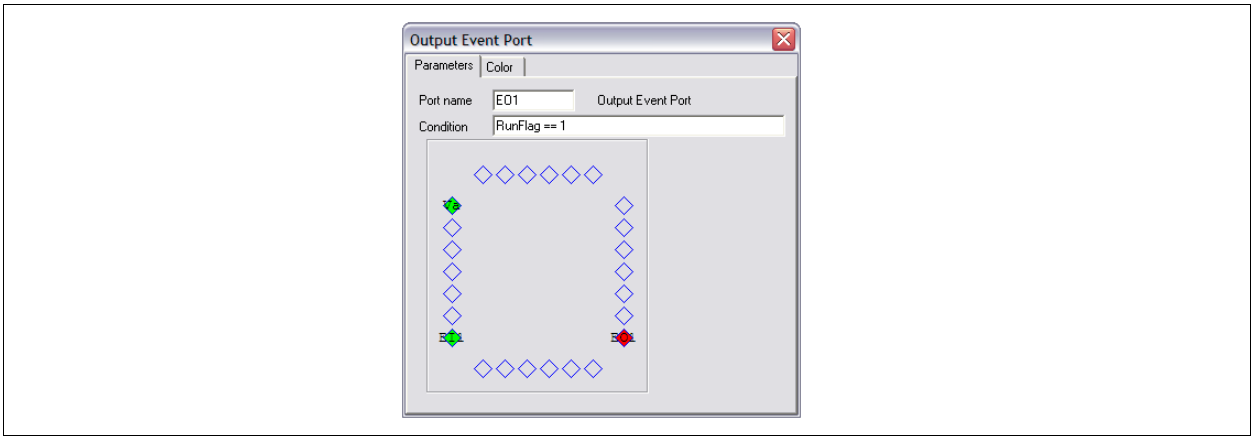
For example, the figure below shows the image of a subcircuit after an input event port and an output event port are placed inside the subcircuit.



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:



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.

4.3 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.

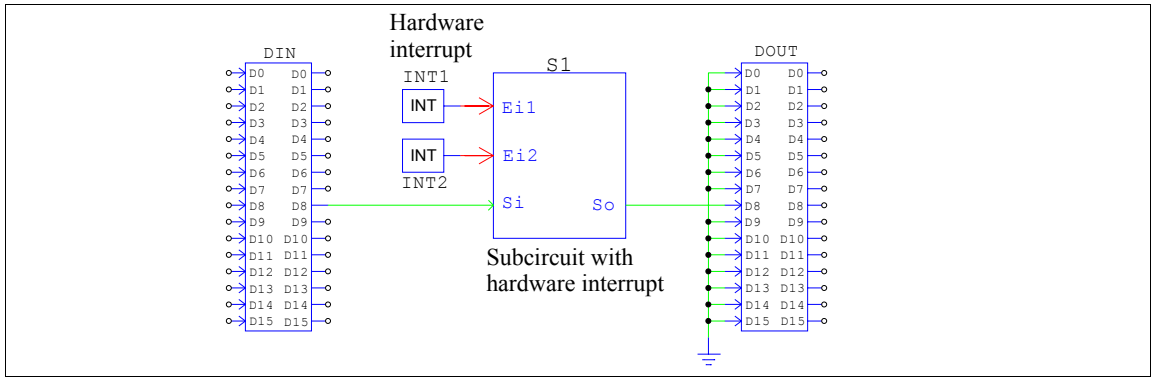
As subcircuits are used to handle events, there are now three types of subcircuits in PSIM:

- *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.

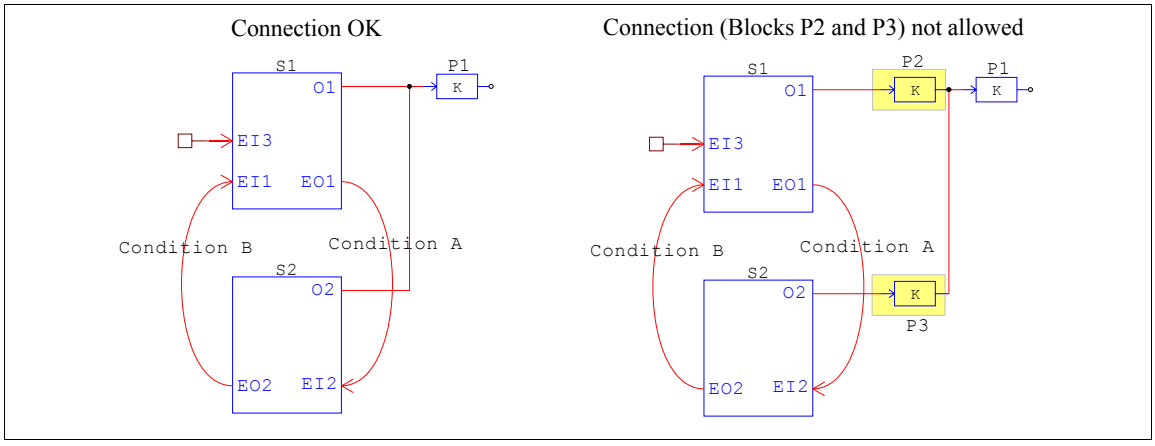
Since a subcircuit with events or with hardware interrupt is involved in the code generation only, the following restrictions are imposed on these two types of subcircuits:

- 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:



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.

5
SimCoder Libraries

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.

Elements that can be used in SimCoder and 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 for Code Generation**.

Elements that are shared by hardware targets include the following:

- The *Interrupt* element under **Elements** -> **SimCoder for Code Generation**.

Elements that are hardware-specific include the following:

- TI F28335 Hardware Target: All the elements under **Elements** -> **SimCoder for Code Generation** -> **TI F28335 Target**.
- PE-Expert3 Hardware Target: All the elements under **Elements** -> **SimCoder for Code Generation** -> **PE-Expert3 Target**.
- General Hardware Target: All the elements under **Elements** -> **SimCoder for Code Generation** -> **General Hardware Target**.

Elements that are independent or common to all hardware targets are described in this Chapter. Elements that are specific to each hardware target are described in Chapter 6 to 8.

5.1 Elements from Standard PSIM Library

Below is a list of elements in the standard library that can be used in SimCoder for code generation:

Under the menu **Elements** -> **Control**:

- Proportional
- Comparator
- Limiter
- Upper Limiter
- Lower Limiter
- Range Limiter
- Summer (+/-)
- Summer (3-input)

Under **Elements** -> **Control** -> **Computational Blocks**:

- Multiplier
- Divider
- Square-root
- Sine
- Sine (in rad.)
- Arcsine
- Cosine
- Cosine (in rad.)

- Tangent
- Arctangent 2
- Exponential (a^x)
- Power (x^a)
- LOG (base e)
- LOG10 (base 10)
- Absolute Value
- Sign Block

Under **Elements** -> **Control** -> **Other Function Blocks**:

- Multiplexer (2-input)
- Multiplexer (4-input)
- Multiplexer (8-input)

Under **Elements** -> **Control** -> **Logic Elements**:

- AND Gate
- AND Gate (3-input)
- OR Gate
- OR Gate (3-input)
- XOR Gate
- NOT Gate
- NAND Gate
- NOR Gate

Under **Elements** -> **Control** -> **Digital Control Module**:

- Zero-Order Hold
- Unit Delay
- Integrator
- Differentiator
- External Resetable Integrator
- Internal Resetable Integrator
- FIR Filter
- FIR Filter (file)
- Digital Filter
- Digital Filter (file)
- z-domain Transfer Function
- Circular Buffer (single-output)

Under **Elements** -> **Control** -> **Other**:

- Parameter File

Under **Elements** -> **Control** -> **Other** -> **Function Blocks**:

- abc-dqo Transformation
- dqo-abc Transformation
- abc-alpha/beta Transformation
- alpha/beta-abc Transformation
- ab-alpha/beta Transformation
- ac-alpha/beta Transformation
- alpha/beta-dq Transformation
- dq-alpha/beta Transformation
- x/y-r/angle Transformation
- r/angle-x/y Transformation
- Math Function
- Math Function (2-input)
- Math Function (3-input)
- Math Function (5-input)
- Math Function (10-input)
- Simplified C Block

Under **Elements** -> **Sources**:

- Constant
- Ground
- Ground (1)
- Ground (2)

Under **Elements** -> **Sources** -> **Voltage**:

- DC
- DC (battery)
- Sawtooth
- Math Function
- Grounded DC (circle)
- Grounded DC (T)

Please note that, for all the math function blocks and the Simplified C Block, variables t (for time) and delt (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.

5.1.1 Defining Global Parameters in Parameter File

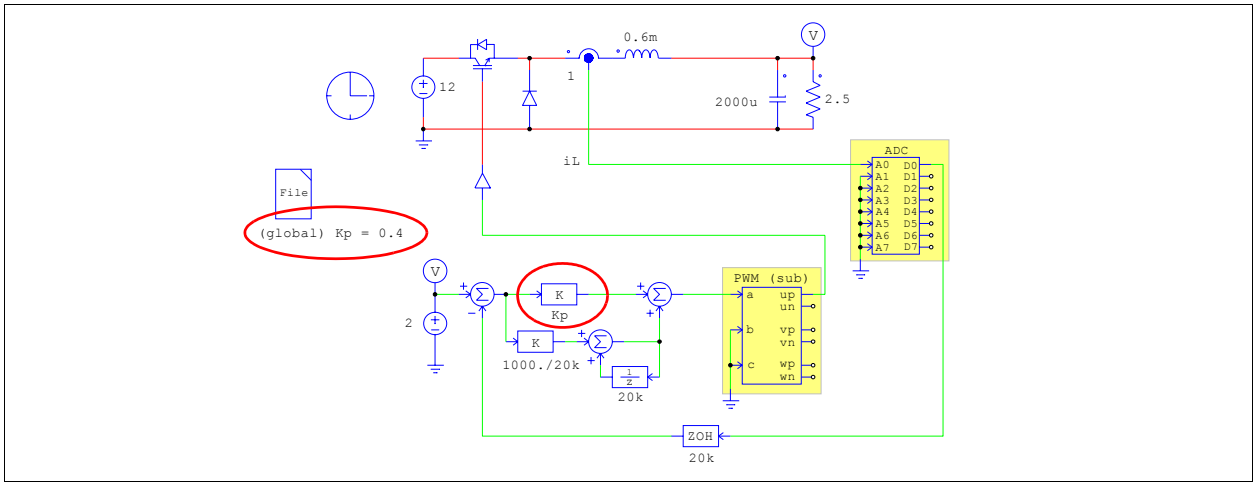
The parameter file element can be used in the same way as before. In SimCoder, it can also be used to define global parameters.

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.

For example, the example below shows a circuit where the gain of the proportional controller is defined as Kp. In the parameter file, the parameter Kp is defined as:

(global) Kp = 0.4



The generated code 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 1.0 for Myway PE-Expert3
//
// SimCoder is copyright by Powersim Inc., 2008
//
// Date: September 10, 2008 10:19:36
*********************************************************************/
interrupt void Task();

float       fGbliref = 0.0;
float       fGblUDELAY1 = 0.0;

float       Kp = 0.4;       The global parameter Kp is defined here.
interrupt void Task()
{
            float fVDC2, fPEV_ADC1, fPEV_ADC1_1, fPEV_ADC1_2, fPEV_ADC1_3, fZOH3, fSUM1, fP2;
            float fSUMP3, fUDELAY1, fP1, fSUMP1, fPEV_PWM_11Div0, fPEV_PWM_11Div1, fPEV_PWM_11Div5;
            float fPEV_PWM_11Div2, fPEV_PWM_11Div6, fPEV_PWM_11Div3, fPEV_PWM_11Div7;

            pev_ad_start(0, 0);

            fUDELAY1 = fGblUDELAY1;
            fVDC2 = 2;

#ifdef      _DEBUG
            fGbliref = fVDC2;
#endif
            while (pev_ad_in_st(0, 0));
            pev_ad_in_grp(0, 0, &fPEV_ADC1, &fPEV_ADC1_1, &fPEV_ADC1_2, &fPEV_ADC1_3);

            fZOH3 = fPEV_ADC1;
            fSUM1 = fVDC2 - fZOH3;
            fP2 = fSUM1 * (1000./20000);
            fSUMP3 = fP2 + fUDELAY1;

            fGblUDELAY1 = fSUMP3;
            fP1 = fSUM1 * Kp;       The parameter Kp is used here.
            fSUMP1 = fP1 + fSUMP3;
            ... ... ...
```

5.1.2 Generating Sawtooth Waveform

A sawtooth waveform can be used in hardware as the system time, 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.

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.

For the General Hardware Target, it assumes that there exists a 32-bit counter in the hardware, incrementing in every 20 ns, to generate the sawtooth waveform.

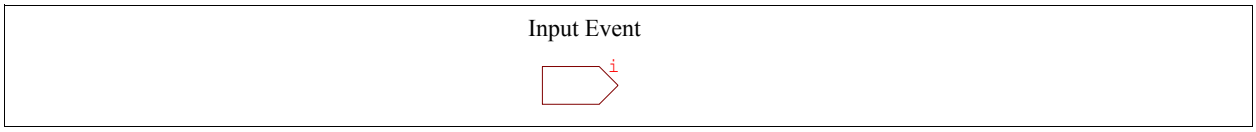
5.2 Event Control Elements

The following elements are used to implement event control.

5.2.1 Input Event

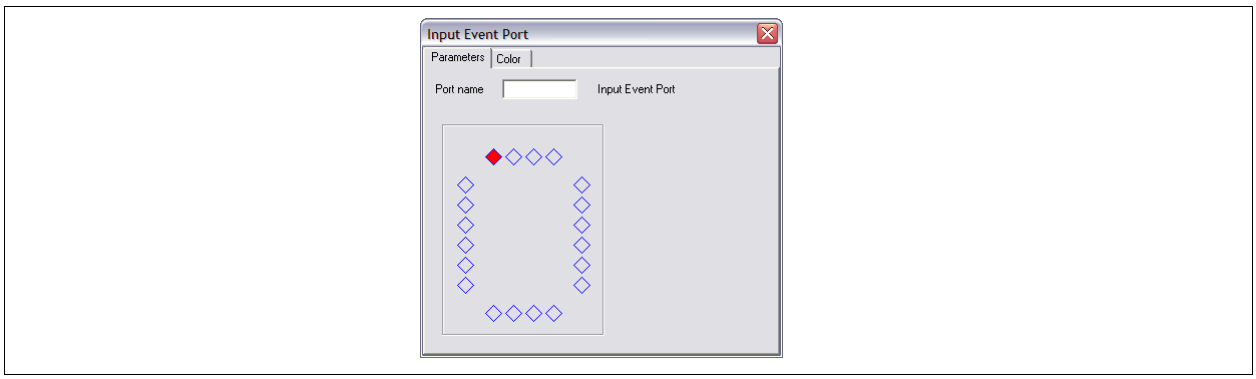
The image of an input event element is shown below.

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:

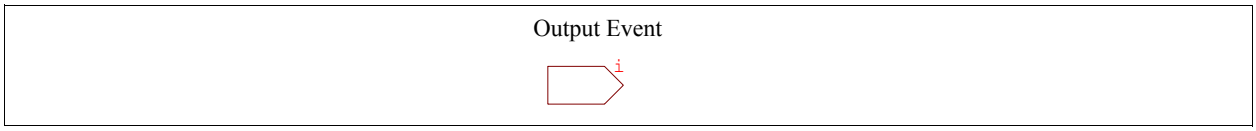


In the main circuit that calls this subcircuit, if there is an event connection wire connecting to this port, when the condition of the event connection is met, the system will transit to this subcircuit through this input event port.

5.2.2 Output Event

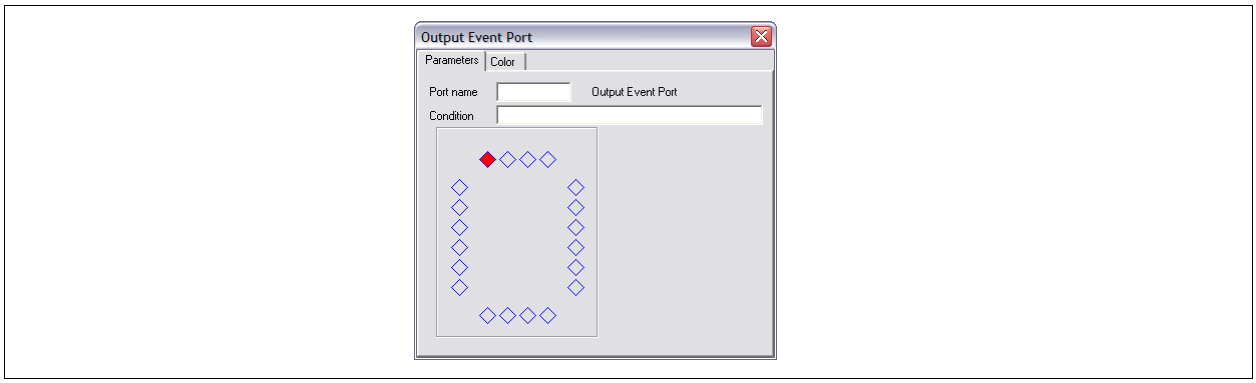
The image of an output event element is shown below.

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:



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:

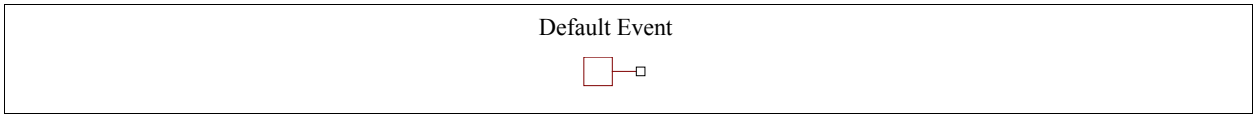
A == 1
A >= B
(A > B) && (C > D)

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

5.2.3 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.

5.2.4 Event Connection

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

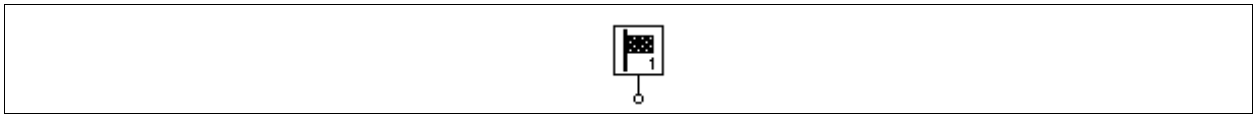
When double clicking on the event connection wire, one can edit the condition statement of the output event port that the event wire connects to.

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".

5.2.5 Flag for Event Block First Entry

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

Image:



Attribute:

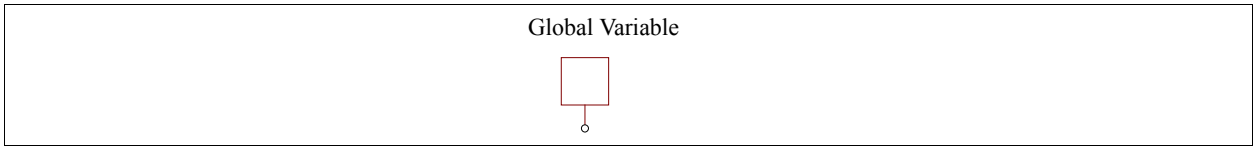
Parameters	Description
Event Subcircuit Block Name	The name of the event subcircuit block that the flag is for.

The flag node is an output node. When the event subcircuit block is entered 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.

5.3 Global Variable

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

Image:



Attributes:

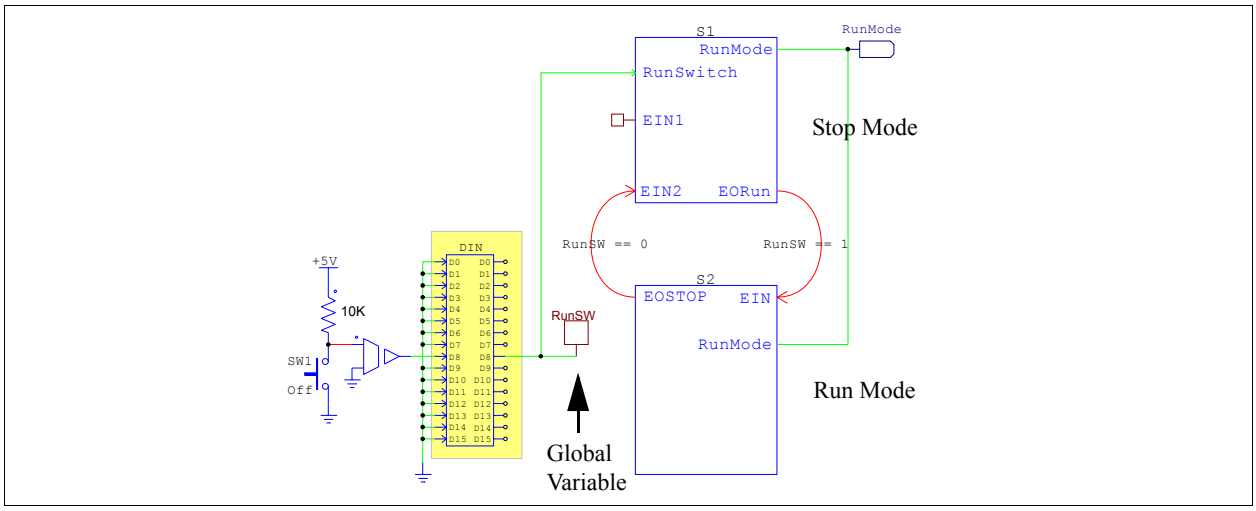
Parameters	Description
Name	The name of the global variable name
Initial Value	The initial value of the global variable

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.



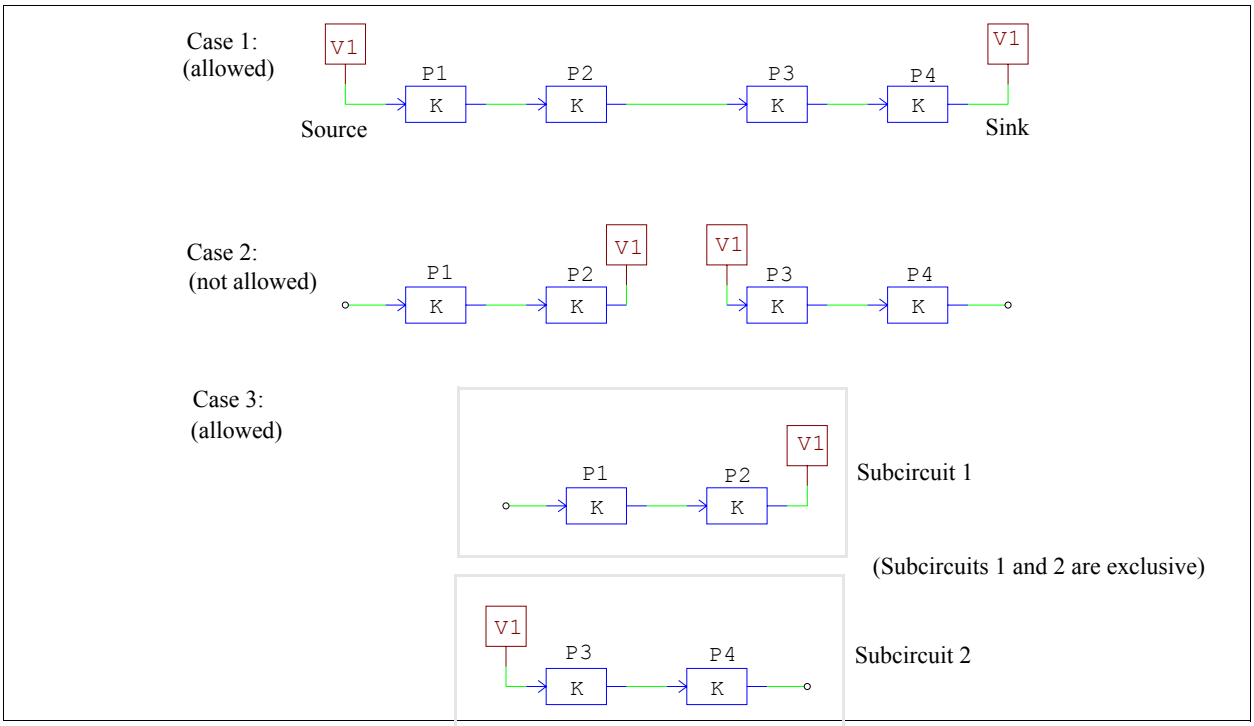
In this example, a global variable, *RunSW*, is connected to the output pin D8 of the digital input. This global variable is then used in the conditional statements between the transition of the two modes of operation.

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 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.

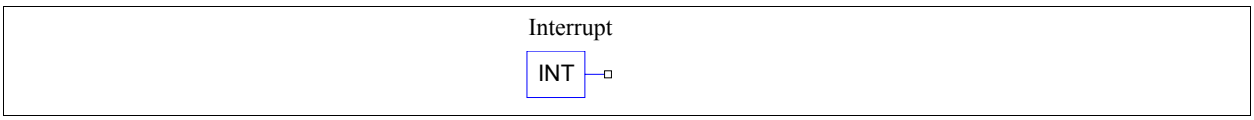
In Case 3, on the other hand, the global variable V1 is used in both Subcircuits 1 and 2. Subcircuit 1 and 2, however, are two exclusive states. That is, the system will run either Subcircuit 1, or Subcircuit 2, but not both. The use of the global variables is allowed in this case.

5.4 Interrupt

In a hardware target, elements such as digital input, encoder, capture, and PWM generators (for TI F28335 only) 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.

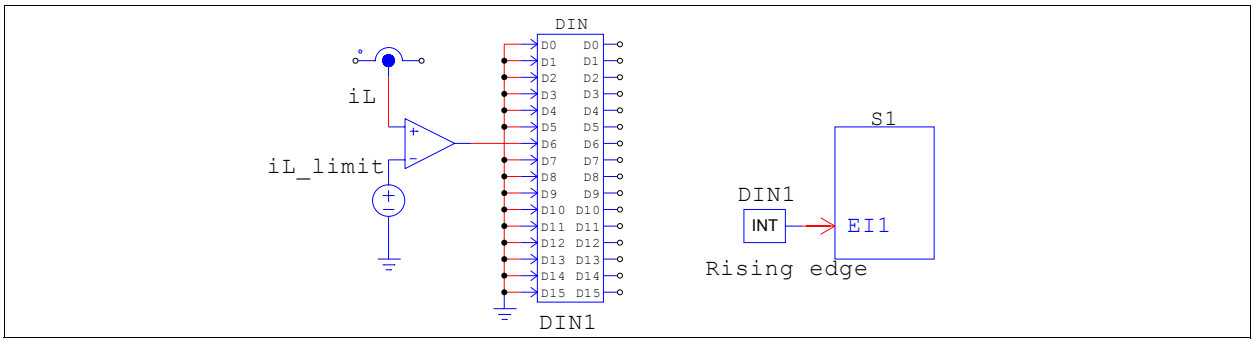
Image:



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.



In this circuit, the current i_L is measured and compared with the limit i_{L_limit}. If the current i_L exceeds the limit, it will generate a pulse which is sent to Input D6 of the digital input DIN1. The interrupt block in the circuit is defined to have the "Device Name" as "DIN1", and the "Edge Detection Type" as "Rising edge". The pulse will then generate a hardware interrupt and the operation will transit to Subcircuit S1 through the input event port E I1.

6

TI F28335 Hardware Target

With the TI F28335 Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F28335 floating-point DSP.

The TI F28335 Hardware Target will work with all F28335 packages. The figures below show the pin assignments of the F28335 DSP in the low-profile flatpack (LQFP) package. The main functions implemented in the TI F28335 Hardware Target are marked in color in the figures.

The TI F28335 Hardware Target library includes the following function blocks:

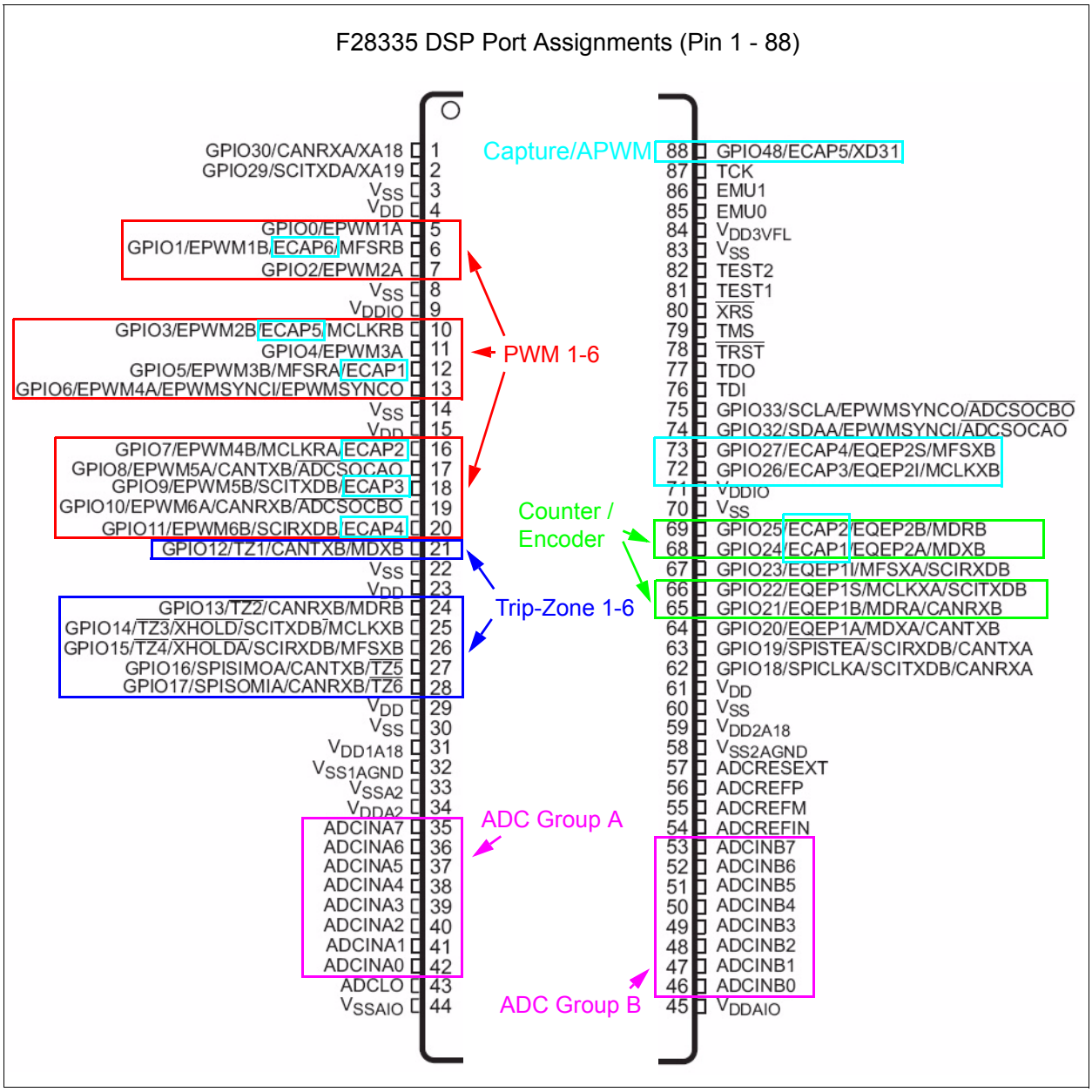
- 3-phase, 2-phase, 1-phase, and single PWM generators
- Start/Stop functions for PWM generators
- Trip-zone and trip-zone state blocks
- A/D converter
- Digital input
- Digital output
- Up/Down counter
- Encoder and encoder state blocks
- Capture and capture state blocks
- DSP clock block
- Hardware board configuration block

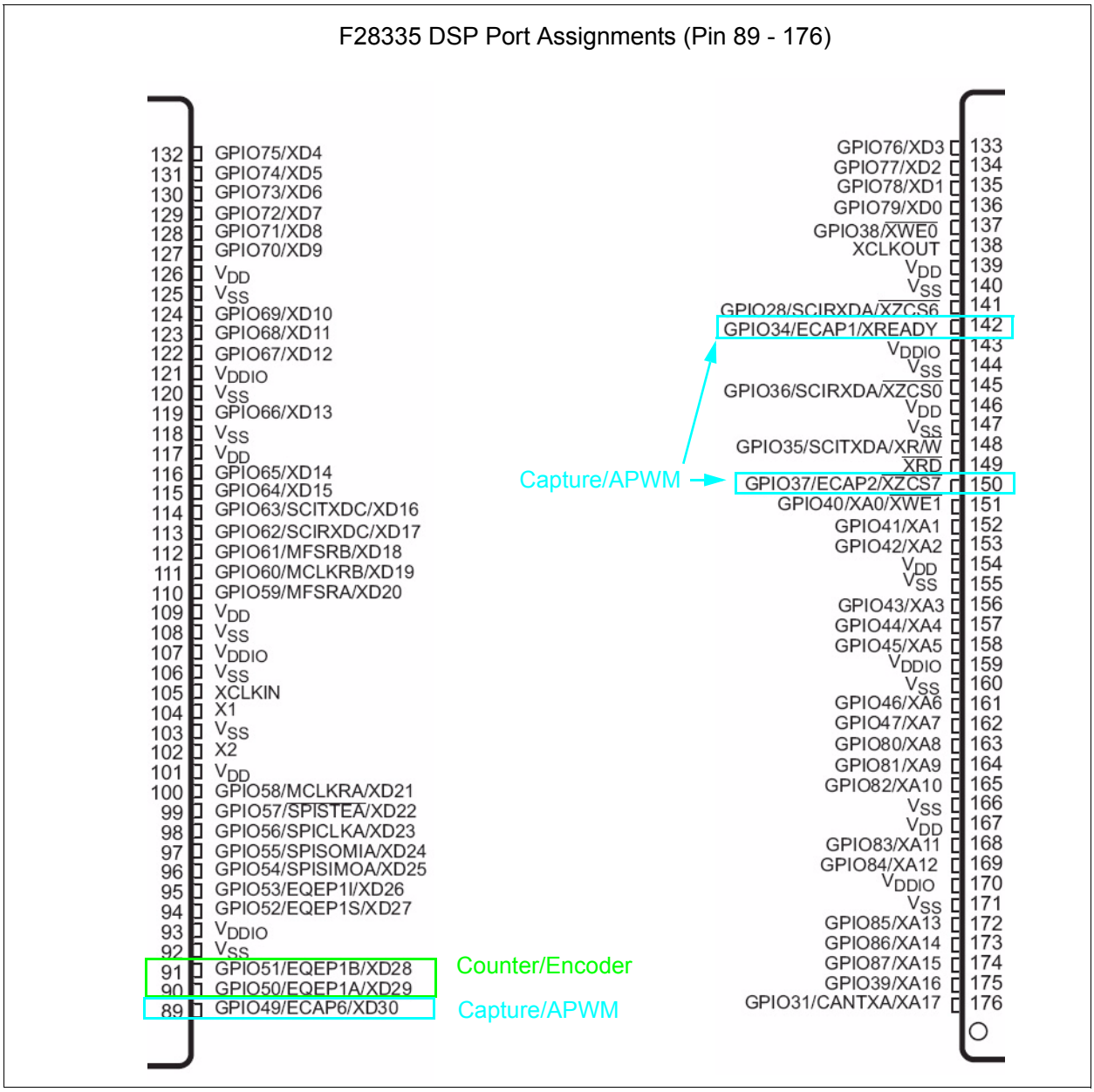
When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed, If there are more than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

In TI 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 TI F28335 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
PWM 1-6
Trip-Zone 1-6
ADC Group A
ADC Group B
Capture/APWM
Counter / Encoder
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



6.1 PWM Generators

The F28335 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 the two outputs complementary to each other.

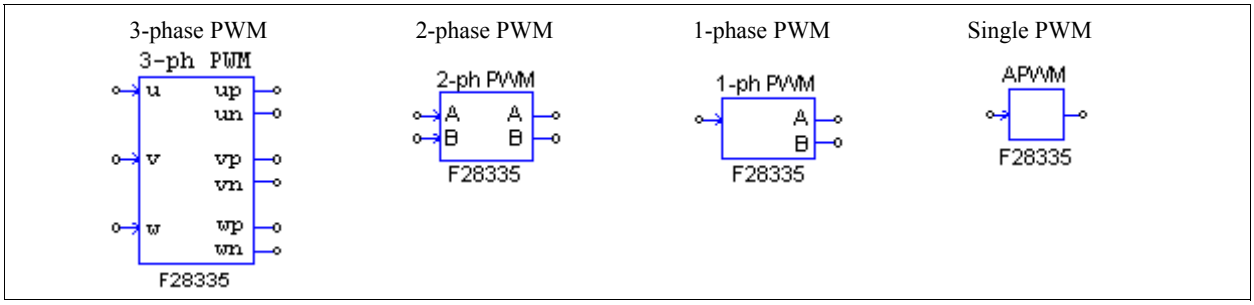
These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 single PWM generators (referred to as APWM in TI's datasheet) 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:



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.

For 3-phase and 1-phase PWM generators:

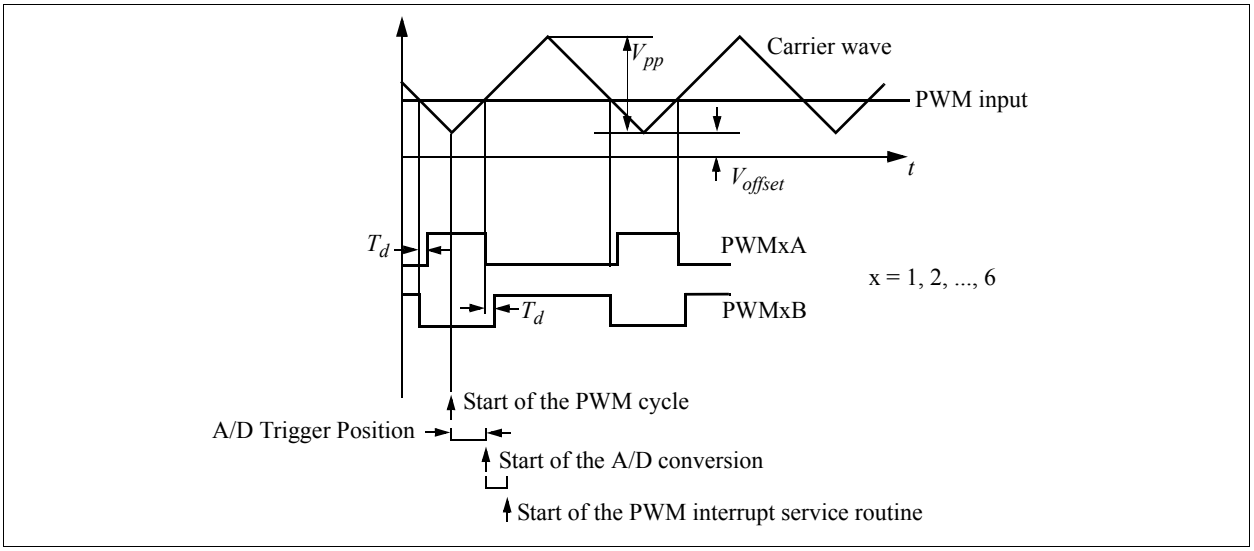
Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. For 3-phase PWM generators, 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. For 1-phase PWM generators, it can be PWM 1 to PWM 6.
Output Mode	The output mode of the PWM generator (for 1-phase PWM generators only). 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	Type of the carrier wave. It can be either "Triangular wave" or "Sawtooth wave".

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 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 *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 the 3-phase inputs u, v, and w (for 3-phase PWM generator only)
Initial Input Value	Initial value of the input (for 1-phase PWM generator only)
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.

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.

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



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.

For 2-phase PWM generators:

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 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.

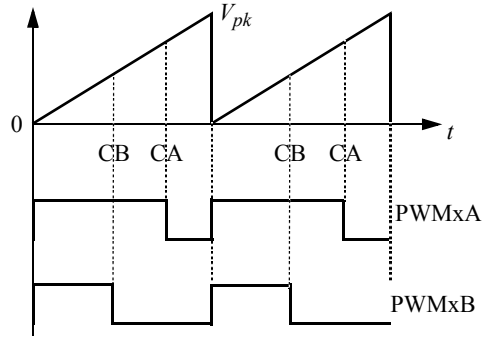
Use Trip-Zone *i*	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where *i* ranges from 1 to 6. It can be one of the following: - *Disable Trip-Zone i*: Disable the i_{th} trip-zone signal. - *One shot*: The PWM generator uses the trip-zone signal in the one-shot mode. Once 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 shows the waveforms of operation 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.

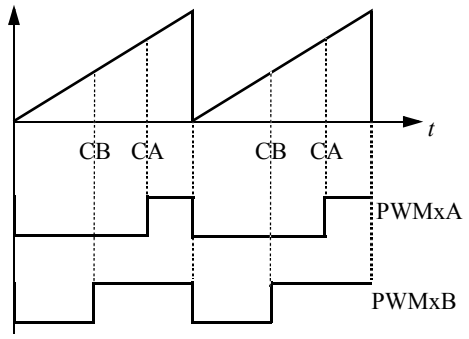
Mode 1:



Operation Mode 2:

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

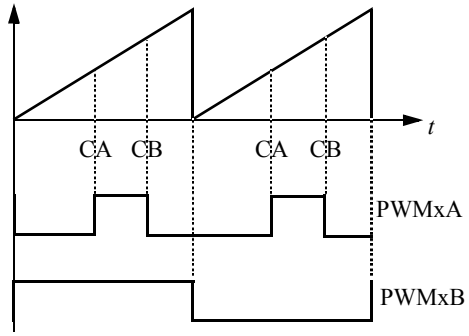
Mode 2:



Operation Mode 3:

The figure below shows the waveforms of operation mode 3. In this mode, 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.

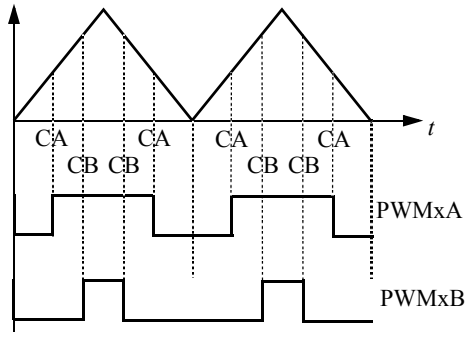
Mode 3:



Operation Mode 4:

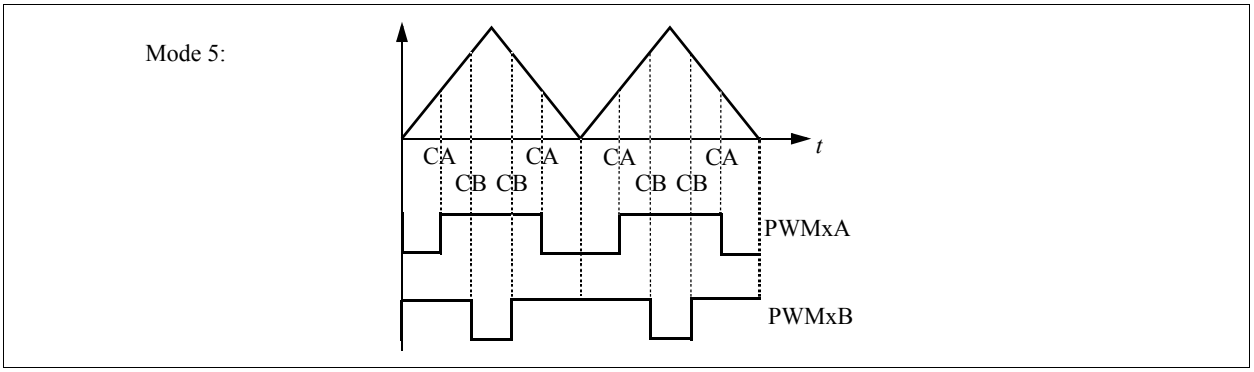
The figure below shows the waveforms of operation mode 4. In this mode, the carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

Mode 4:



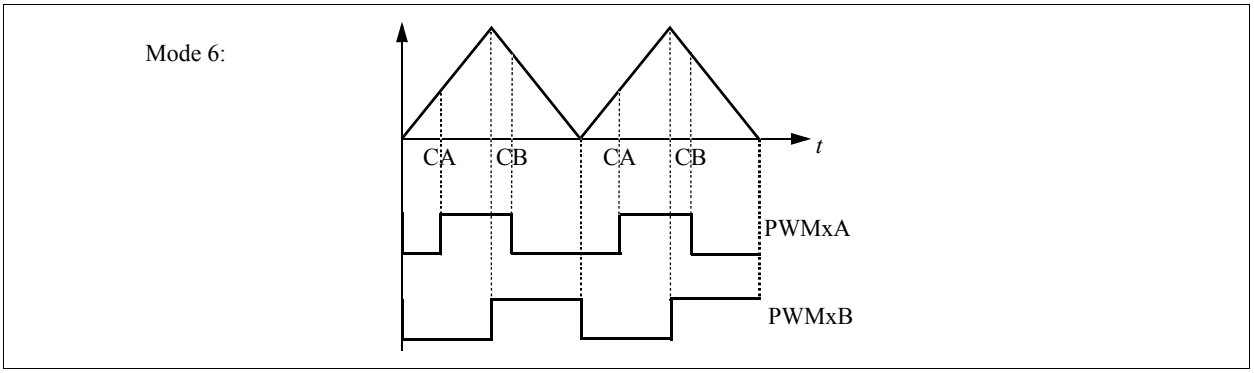
Operation Mode 5:

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



Operation Mode 6:

The figure below shows the waveforms of operation 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.



For single PWM generators (APWM):

Attributes:

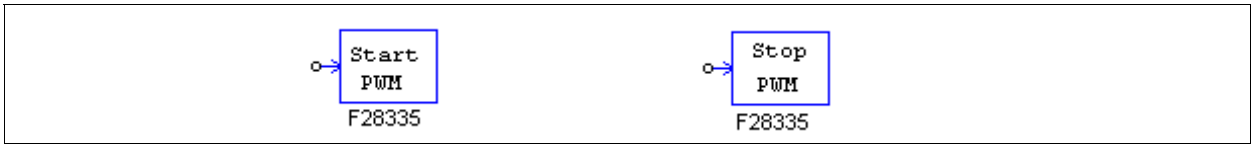
Parameters	Description
PWM Source	The single PWM generator shares the same resource as captures. The PWM source can be one of the six APWM's in 14 designated GPIO ports, as listed below: - *APWM 1* (*GPIO5, GPIO24, GPIO34*) - *APWM 2* (*GPIO7*, *GPIO25*, *GPIO37*) - *APWM 3* (*GPIO9*, *GPIO26*) - *APWM 4* (*GPIO11*, *GPIO27*) - *APWM 5* (*GPIO3*, *GPIO48*) - *APWM 6* (*GPIO1*, *GPIO49*)
PWM Frequency	Frequency of the PWM generator, in Hz
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
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.

As noted before, the single PWM generator is limited in functionality. It can not trigger the A/D converter and can not use the trip-zone signal.

6.2 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.

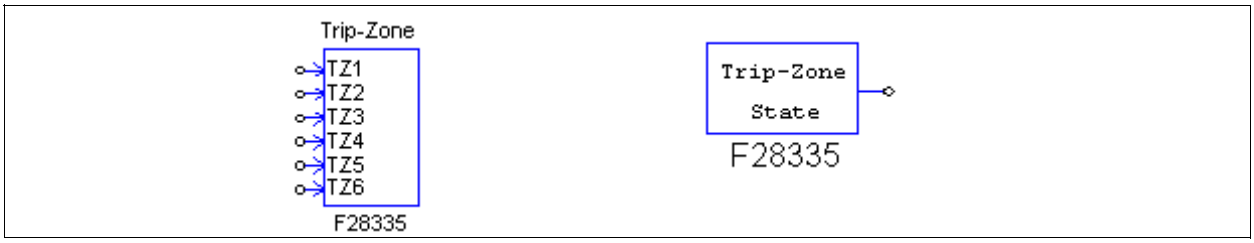
6.3 Trip-Zone and Trip-Zone State

The F28335 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.

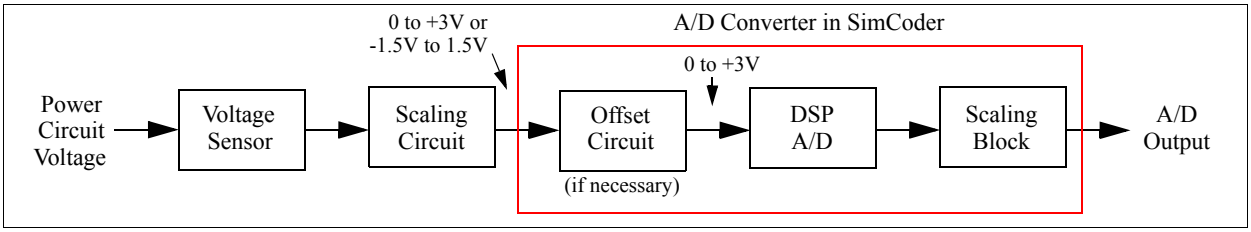
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle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

6.4 A/D Converter

The F28335 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.

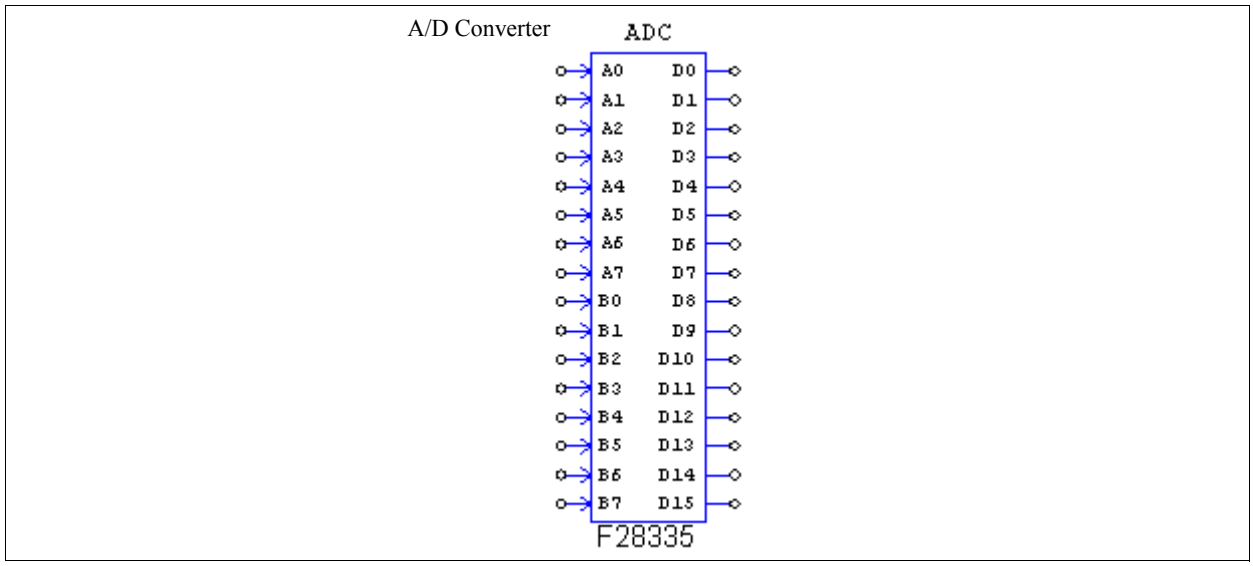
Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. Take a power voltage as an example. A power circuit voltage, which can be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale 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 the offset circuit, the DSP A/D converter, and the scaling block.

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 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 Gain	Gain k 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, or it can be triggered by a PWM generator.

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.

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 mode is dc, the maximum input voltage be scaled to +3V; and when the input mode is ac, the maximum peak voltage be scaled to +1.5V.

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:

$$V_{i_max_s_c} = 1.5 * 2 = 3 \text{ V}$$

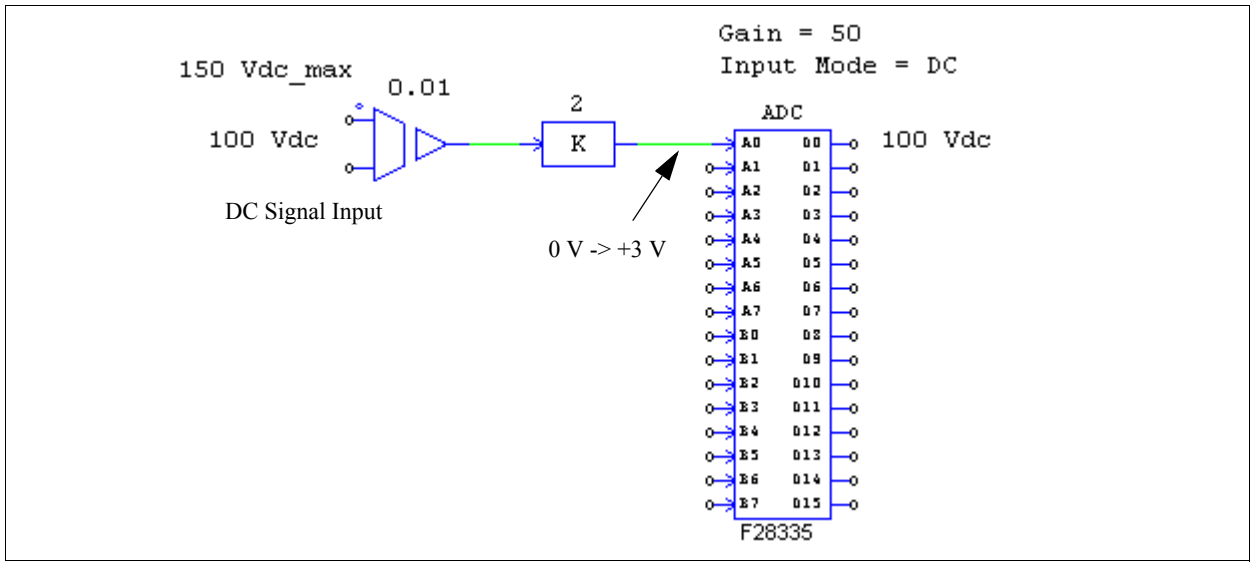
$$V_{i_s_c} = 1 * 2 = 2 \text{ V}$$

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:

$$V_{o_max} = 50 * 3 = 150 \text{ V}$$

$$V_o = 50 * 2 = 100 \text{ V}$$

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 a ac quantity, and the range is as follows:

$$V_{i_max} = +/- 75 \text{ V}$$

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:

$$V_i = +/- 50 \text{ V}$$

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

$$V_{i_max_s} = +/- 0.75 \text{ V}$$

$$V_{i_s} = +/- 0.5 \text{ V}$$

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:

$$V_{i_max_s_c} = +/- 1.5 \text{ V}$$

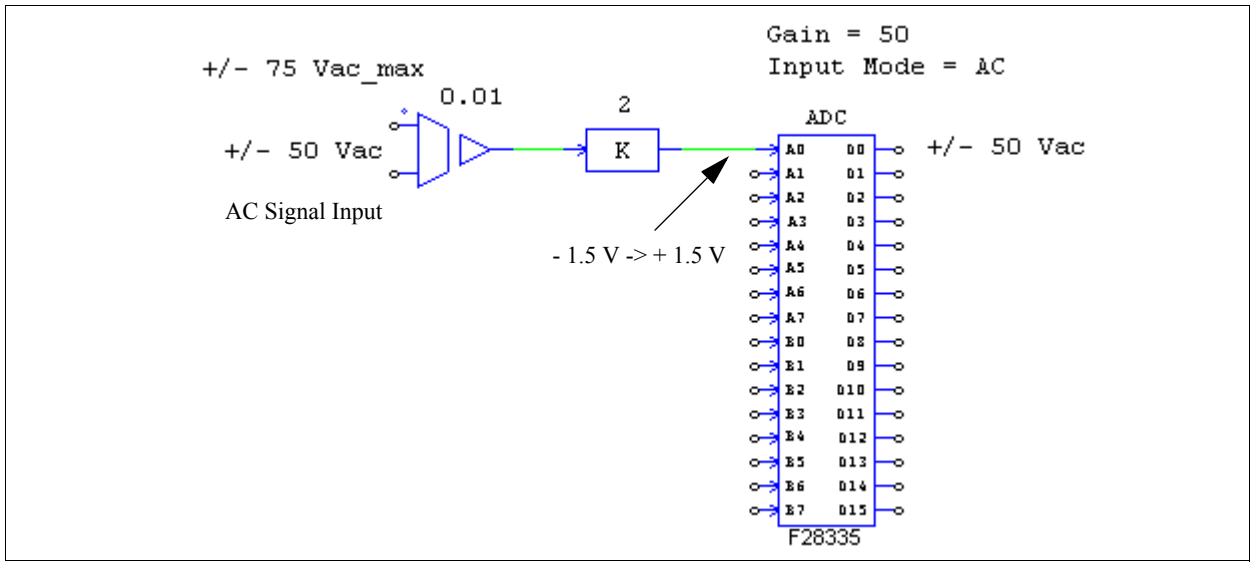
$$V_{i_s_c} = +/- 1 \text{ V}$$

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:

$$V_{o_max} = +/- 75 \text{ V}$$

$$V_{o} = +/- 50 \text{ V}$$

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.

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

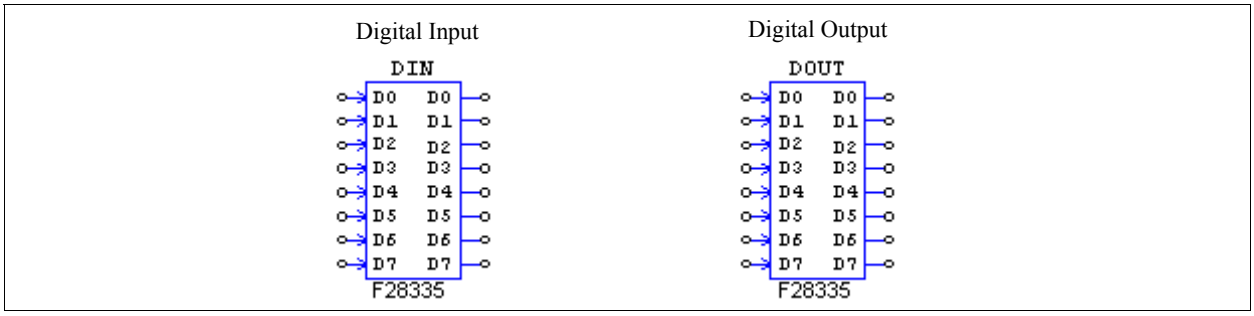
- The A/D converter can be triggered by only one PWM generator. That is, if there are multiple PWM generators, only one can be set to trigger the A/D converter, and the rest should be set not to trigger the A/D converter.
- It is not permitted to have the A/D converter triggered by one PWM generator, 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.

6.5 Digital Input and Digital Output

The F28335 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.

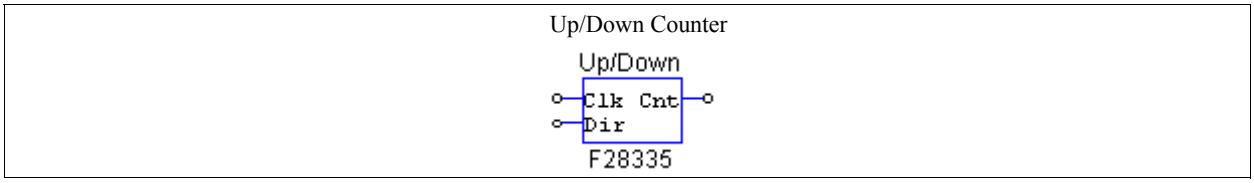
In the F28335 DSP, up to 7 external interrupt sources can be defined from ports GPIO0 to GPIO63 (specifically, up to 2 interrupt sources from Port GPIO0 to GPIO31, and up to 5 interrupt sources from Port GPIO32 to GPIO63). The priority of external interrupts in Port GPIO0 to GPIO31 is higher than the priority of the interrupt in Port GPIO32 to GPIO63.

6.6 Up/Down Counter

The F28335 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	Define 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 correponds 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.

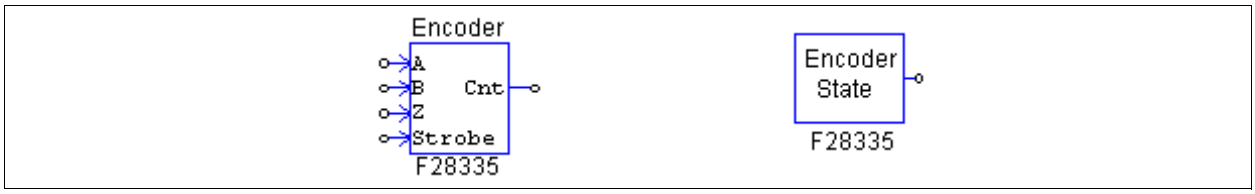
Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

6.7 Encoder and Encoder State

The F28335 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 Encoder State block is used to indicate which input signal (either index signal or strobe signal) generates the interrupt.

Image:



Attributes for Encoder:

Parameters	Description
Encoder Source	Define source of the encoder. It can be one of the following: - *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.
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.
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 following: - *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.

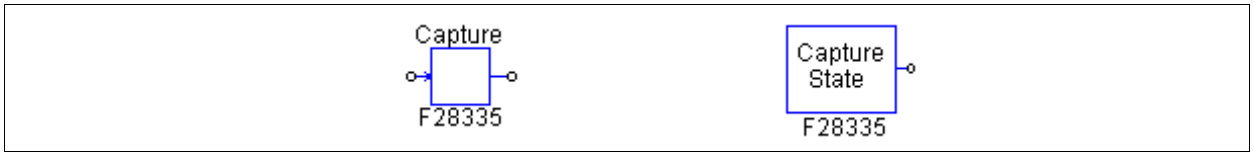
The output of the encoder gives the counter value.

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.

6.8 Capture and Capture State

The F28335 DSP provides 6 captures. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Image:



Attributes for Capture:

Parameters	Description
Capture Source	Define the 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	Select the event filter prescale. the input signal is divided by the selected prescale.
Timer Mode	Indicate the capture counter timer mode. It can be either *Absolute time* or *Time difference*.

Attributes for Capture State:

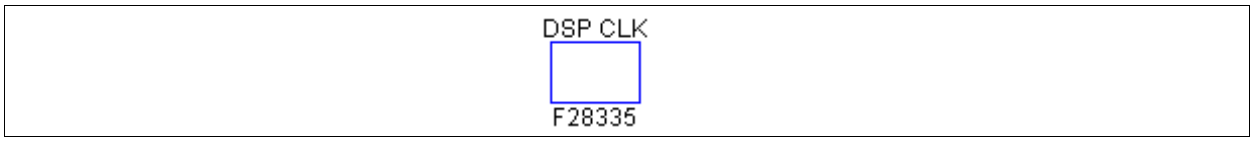
Parameters	Description
Capture Source	Define the source of the capture. It can be one of the 6 captures: *Capture 1*, *Capture 2*, ..., *Capture 6*.

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

6.9 DSP Clock

The DSP Clock block defines the external clock frequency and the speed of the F28335 DSP.

Image:



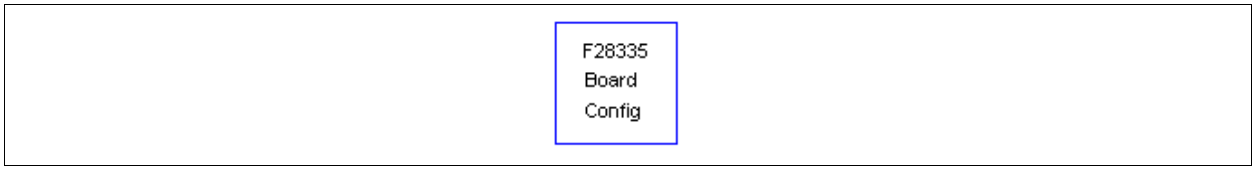
Attributes:

Parameters	Description
External Clock (MHz)	Define the 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)	Define the 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.

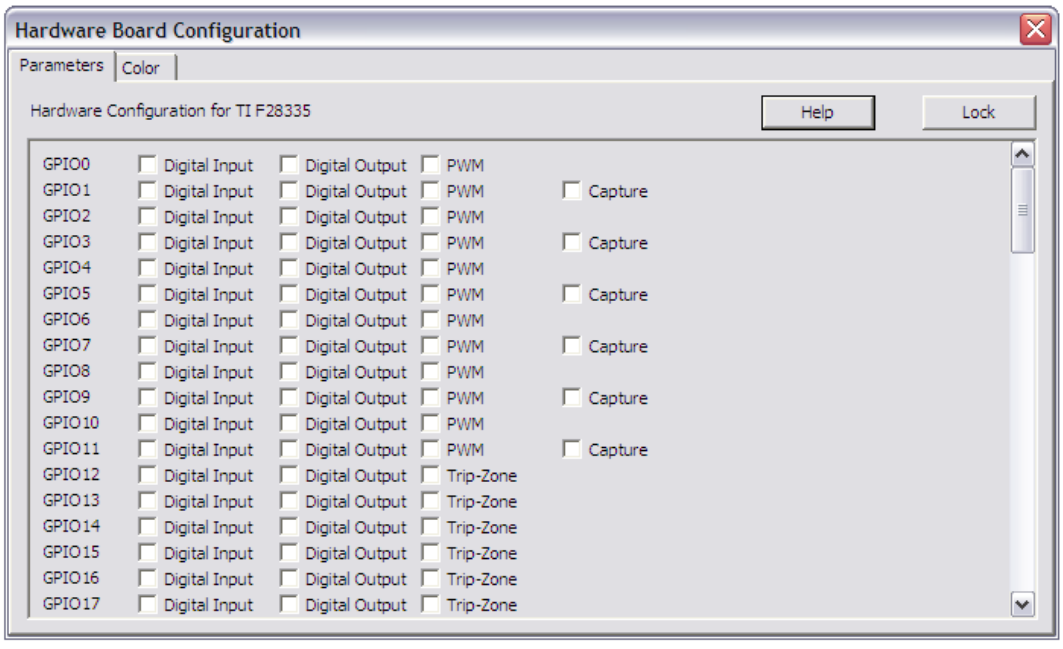
6.10 Hardware Board Configuration

The F28335 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 Board Configuration block provides a way to configure SimCoder for a particular DSP board.

Image:



The dialog window of the block is shown below:



For each GPIO port, the functions that are possible for this port are listed, with a checkbox in front of each 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 checkboxes should be left unchecked. If in the circuit Port GPIO1 is used as "Digital Input", SimCoder will report an error.

6.11 Project Settings and Memory Allocation

When generating the code for the TI F28335 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 F28335 library
- passwords.asm:	File for specifying the DSP code password
- test.pjt:	Project file for Code Composer Studio
- DSP2833x_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

Besides, the project also needs the following files:

- PS_bios.lib:	SimCoder F28335 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.

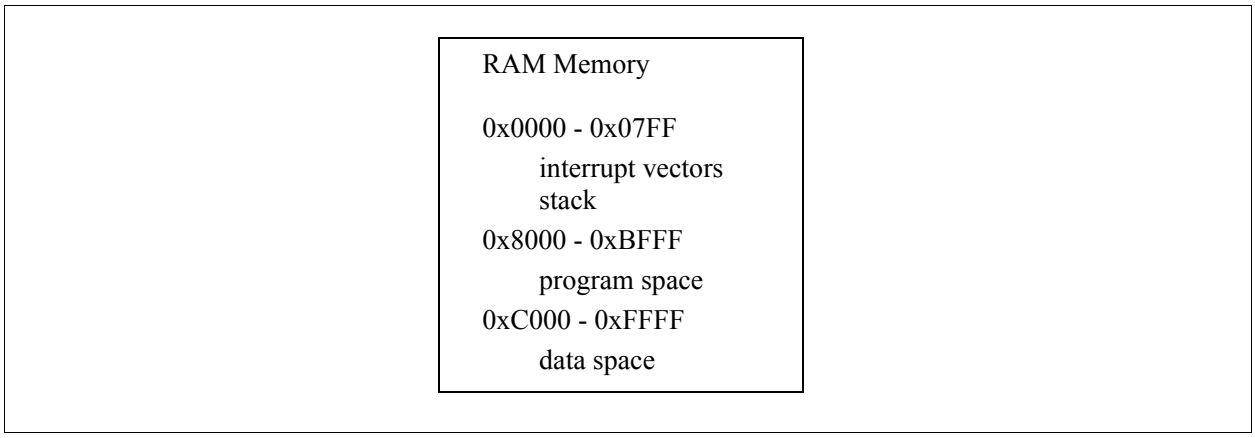
The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release

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.

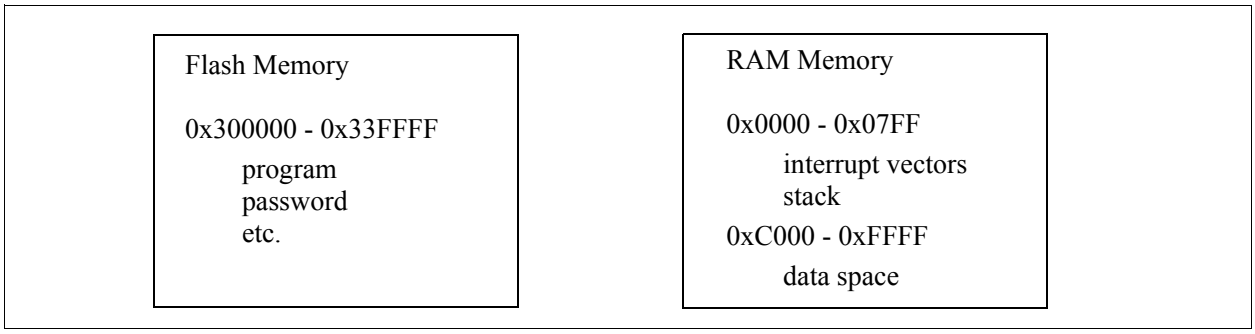
Memory Allocation:

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:



With the Flash Release setting:



For RAM Debug and RAM release, SimCoder predefines the program space from 0x8000 to 0xBFFF, and the data space from 0xC000 to 0xFFFF in the RAM memory. That is, half of the memory (i.e. 0x4000 words) is allocated to the program space, and the other half to the data space. If either the required program space or the data space exceeds 0x4000 words, however, the memory allocation can be changed manually by modifying the linker file. The modified linker file willl not be overwritten by subsequent code generation since SimCoder will not overwrite the linker file if it detects that it already exists in the file folder.

The following examples show how to change the program or data space.

If the required data space exceeds 0x4000 words, an error message "Error: Run placement fails for object ".ebss"" will be reported. For example, to increase the data space from 0x4000 to 0x4900 words, modify the linker file F28335_RAM_Lnk.cmd or F28335_Flash_RAM_Lnk.cmd in the following way (changes are marked in red):

```
... ... ...
PAGE 0: /*Program Memory*/
... ... ...
RAML0:     : origin = 0x008000, length = 0x003700
... ... ...
PAGE 1: /*Data Memory*/
... ... ...
RAML4:     : origin = 0x00B700, length = 0x004900
... ... ...
```

Similarly, if the required program space exceeds 0x4000 words, an error message "Error: Run placement fails

for object ".text"" will be reported. For example, to increase the program space from 0x4000 to 0x5900 words, modify the linker file F28335_RAM_Lnk.cmd or F28335_Flash_RAM_Lnk.cmd in the following way (changes are marked in red):

```
... ... ...
PAGE 0: /*Program Memory*/
... ... ...
RAML0:      : origin = 0x008000, length = 0x005900
... ... ...
PAGE 1: /*Data Memory*/
... ... ...
RAML4:      : origin = 0x00D900, length = 0x002700
... ... ...
```

If the combined program and data space exceeds 0x8000 words, one must use the Flash Release setting.

7

PE-Pro/F28335 Hardware Target

With the PE-Pro/F28335 Hardware Target, SimCoder can generate code that is ready to run on 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.

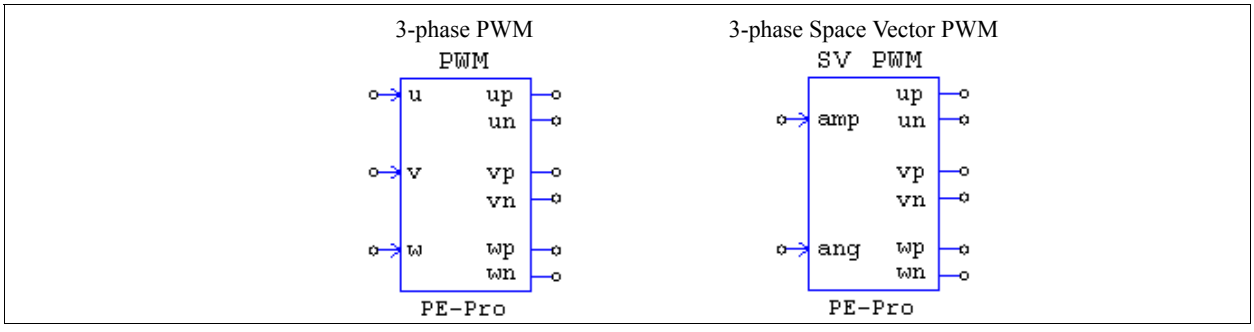
The definitions of the elements in the PE-Pro/F28335 Hardware Target library are described in this Chapter.

7.1 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:



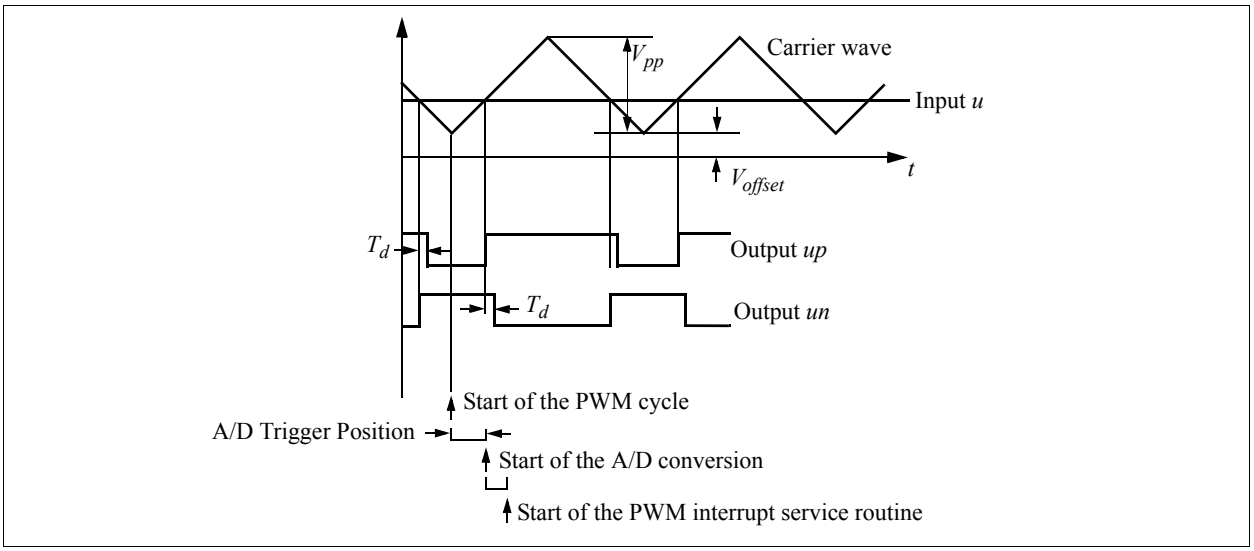
Attributes:

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.

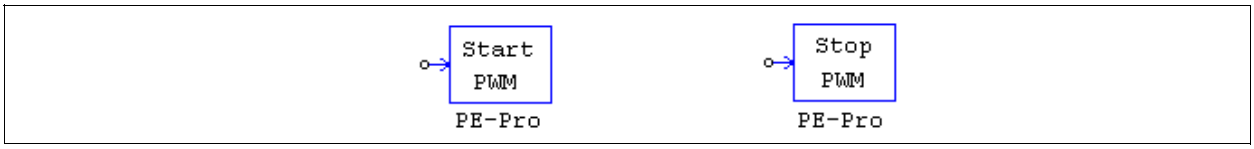
Interrupt can be generated by the PWM generator in two ways:

- *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.

7.2 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:



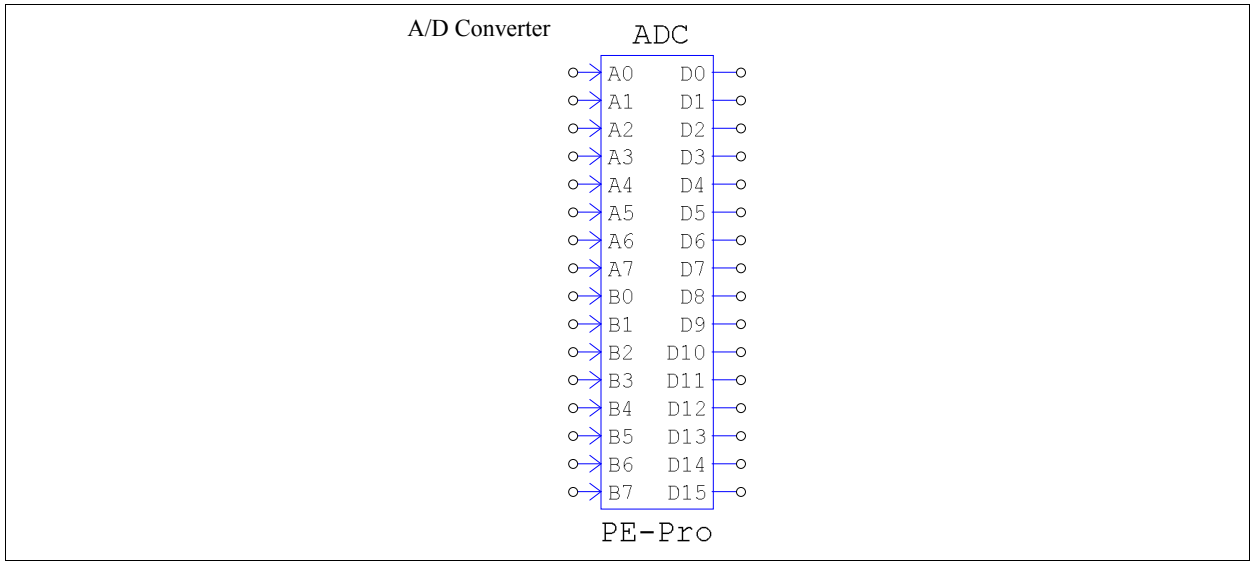
Attributes:

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*

7.3 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:

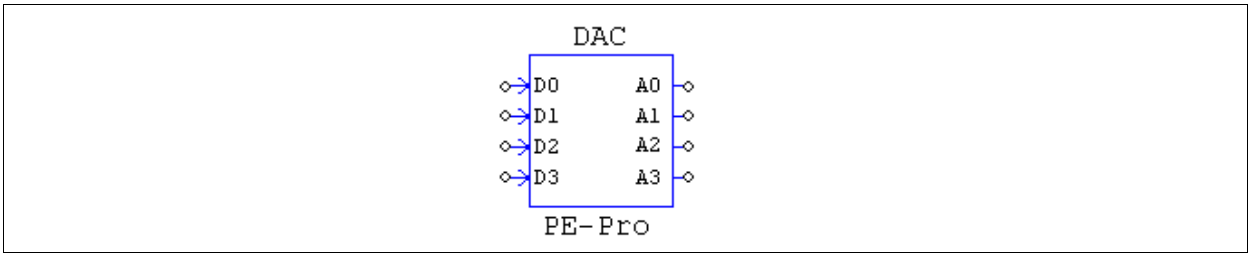
- The A/D converter can be triggered by one PWM generator only.
- It is not permitted to have the A/D converter triggered by one PWM generator, but some of the signals in this group are also used in a circuit that has a different sampling rate than the frequency of the PWM generator.

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

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

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

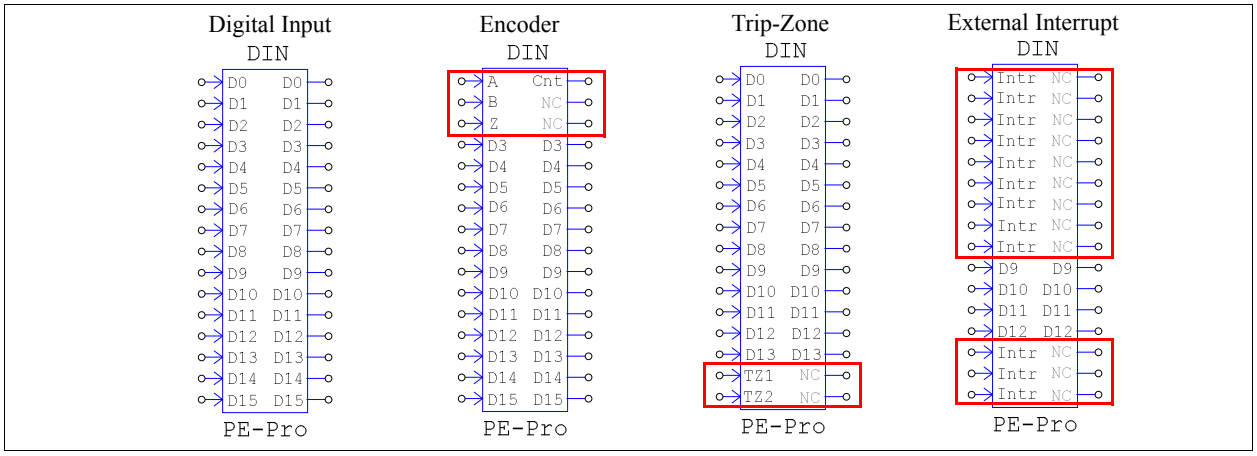
where V_i is the value at the input of the D/A converter. For example, in the ac mode, if $V_{range} = 20$, and $V_i = 10$,

then $V_o = 3.75$. In the dc mode, if $V_{range} = 20$, and $V_i = 10$, then $V_o = 2.5$.

7.5 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.

As Encoder:

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.

As Trip-Zone:

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 0 to 1, it will trigger PWM Generator 123, and when the input signal of trip-zone 2changes 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.

As External Interrupt:

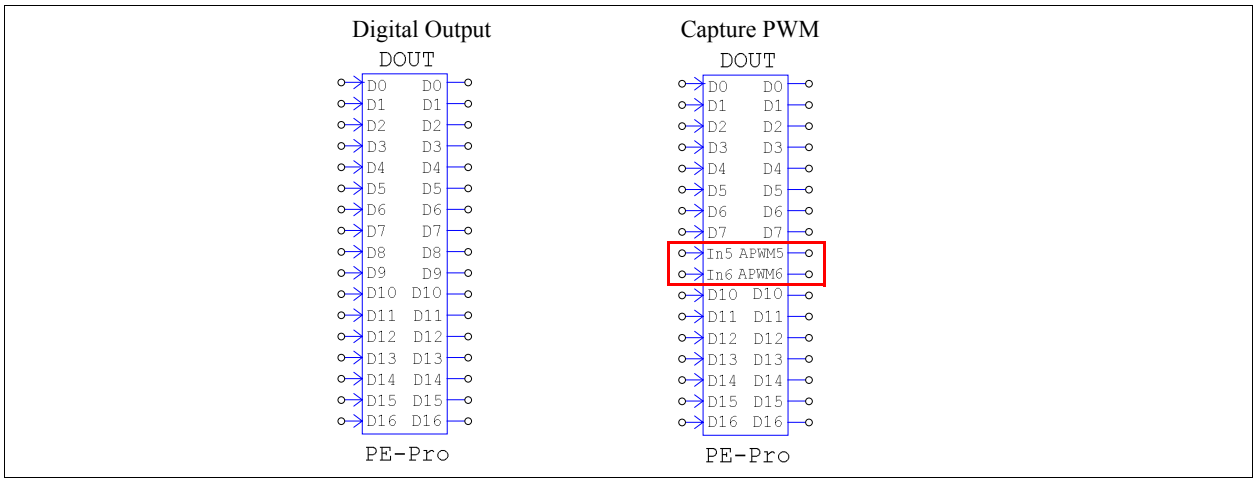
When a channel is defined as the input of the external interrupt, when the input changes from 0 to 1, an interrupt will be generated.

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.

7.6 Digital Output / Single PWM

The combo element of Digital Output / Single PWM can be configured as either Digital Output or Single PWM (also called APWM in TI's datasheet) 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:



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 single PWM 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

With the PE-Expert3 Hardware Target, SimCoder can generate the code that is ready to run on the Myway PE-Expert3 DSP development platform 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.

8.1 PEV Board

The PEV board contains the following functions and hardware elements:

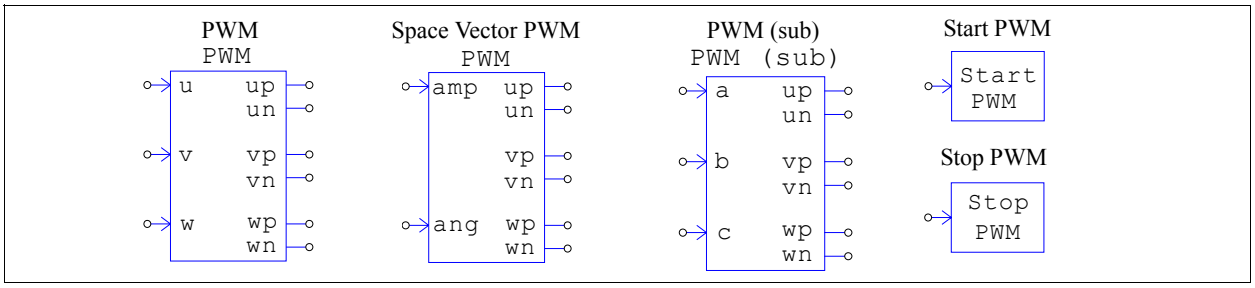
- PWM generators and Start/Stop PWM functions
- A/D converter
- Digital input
- Digital output
- Up/Down counter
- Encoder
- Capture

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.

8.1.1 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):**

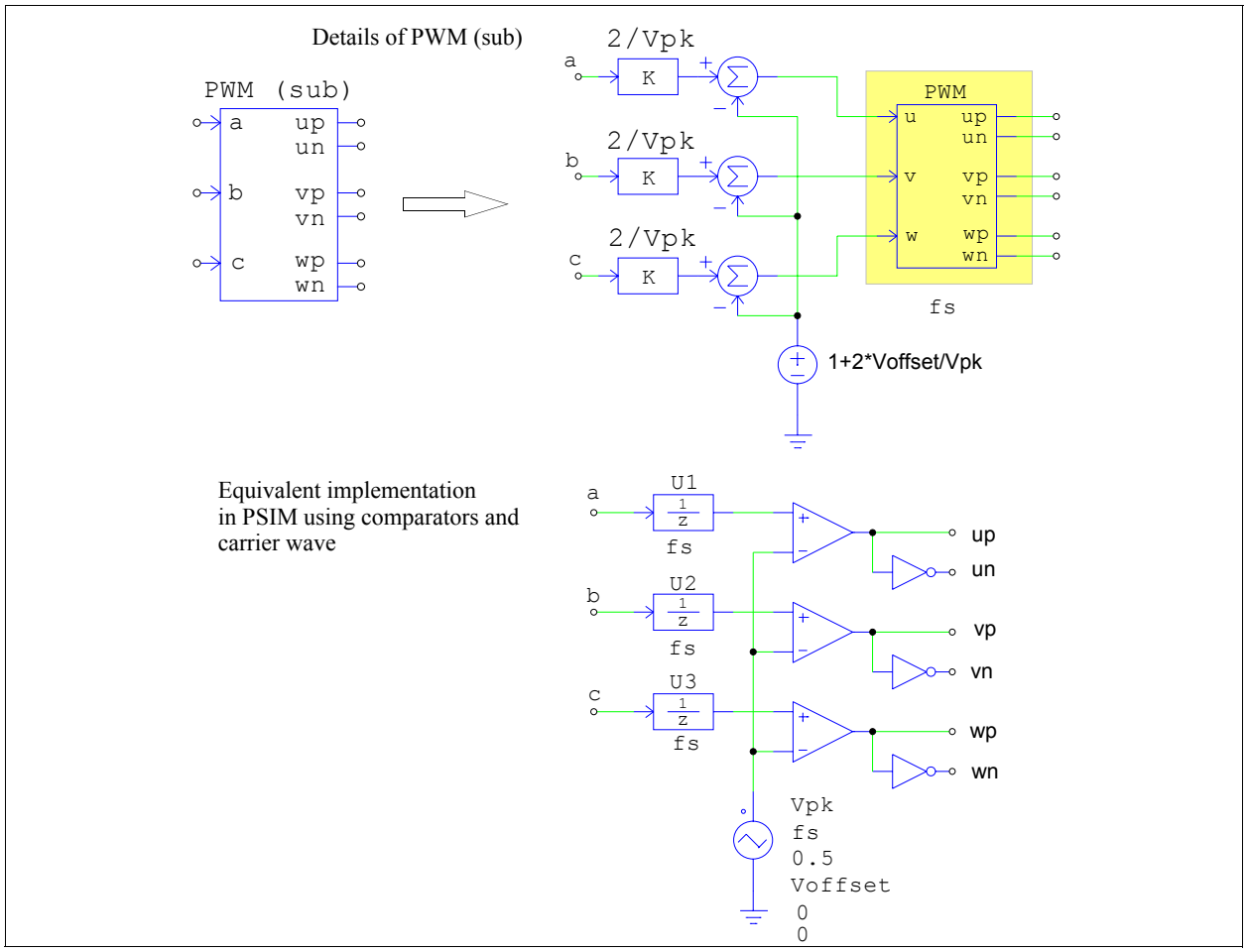
Parameters	Description
Board No.	The board number of the PEV board that contains the PWM generator.
Channel No.	The channel number of the PWM generator. It can be either 0 or 1.
Dead Time	The dead time for the PWM generator, in sec.
Carrier Frequency	Frequency of the PWM generator, in Hz
Peak Value	Peak-to-peak 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.

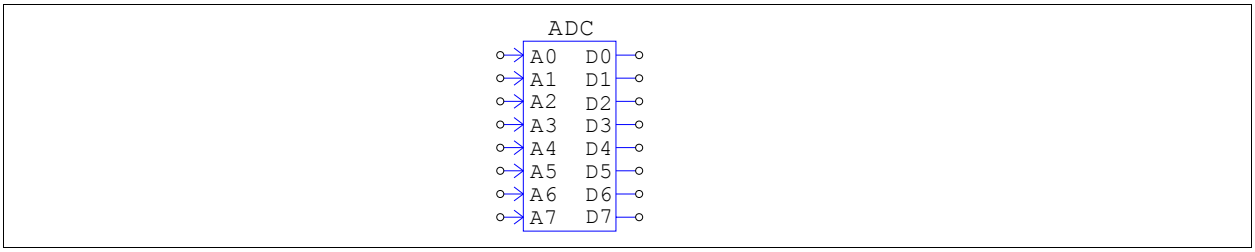
To start PWM, apply a signal of 1V to the input of the **Start PWM** element. To stop PWM, apply a signal of 1V to the input of the **Stop PWM** element.

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

8.1.2 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 $V_o = 2 * 20 / 5 = 8$.

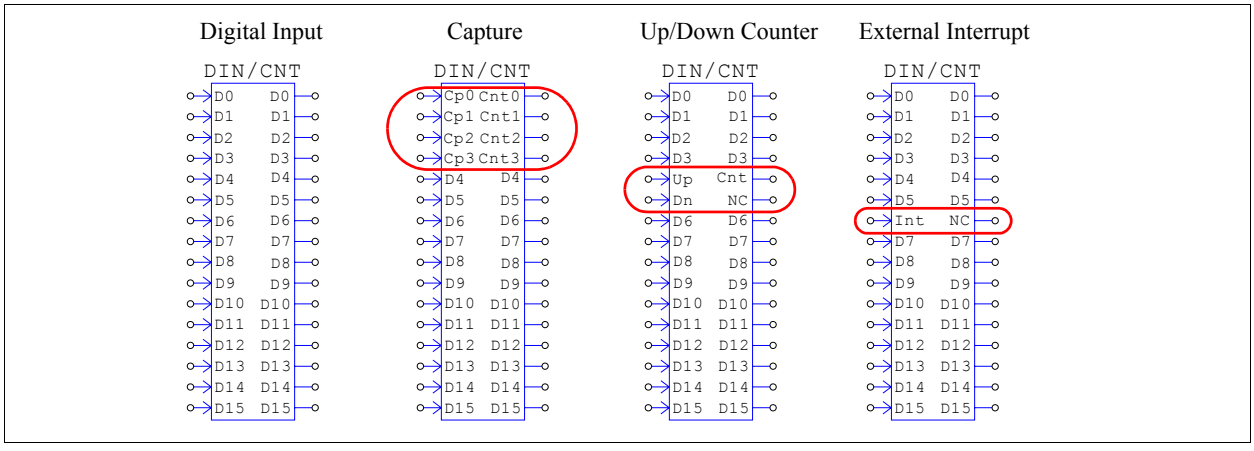
8.1.3 Digital Input / Capture / Counter

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.

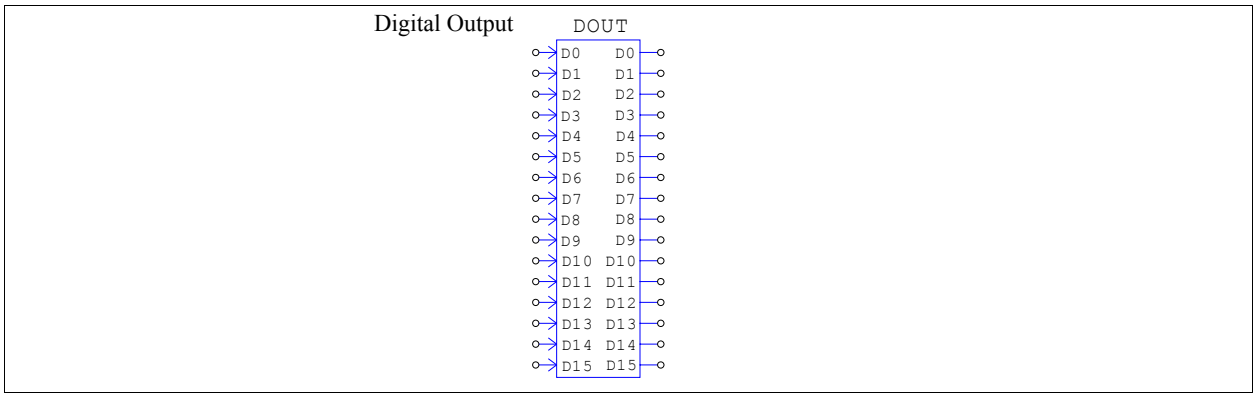
As an External Interrupt:

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.

8.1.4 Digital Output

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

Image:



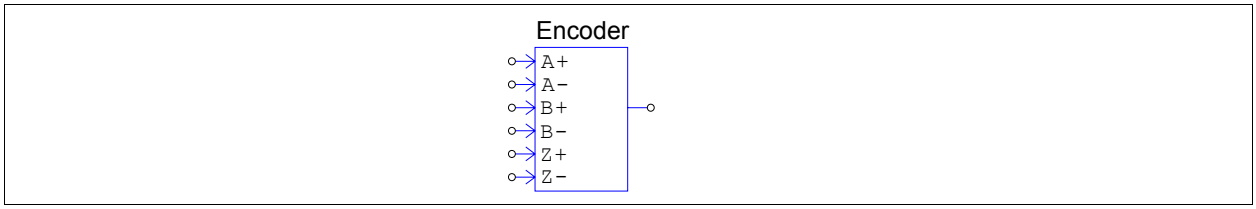
Attributes:

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

8.1.5 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-.

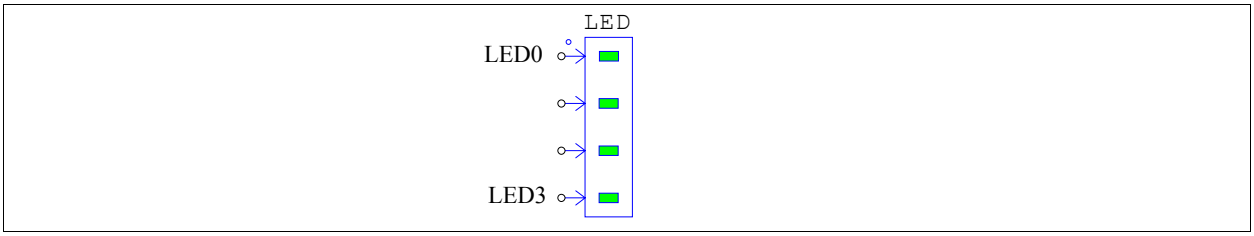
8.2 LED Output

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.

8.3 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

General Hardware Target

With the General Hardware Target, SimCoder can generate code that provides a template code so that users can adopt the code for their own hardware. Please note that the code is not to run on an actual hardware since the generated code does not include the code that interfaces with the hardware (such as PWM generation, A/D conversion, etc.), and users need to write this part of the code themselves.

A general hardware contains the following hardware elements and functions:

- PWM generators and Start/Stop PWM functions
- A/D and D/A converters
- Digital input and output
- Encoder
- Capture

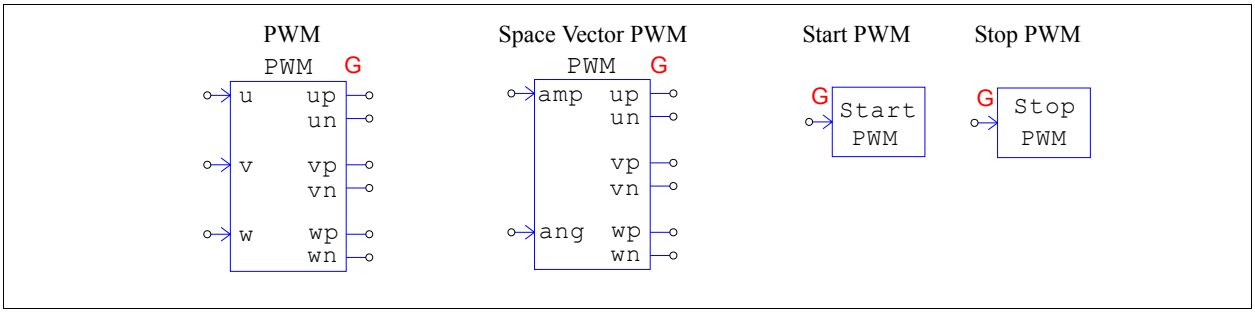
Among them, digital input, encoder, and capture can also generate hardware interrupts. An interrupt must be associated with an interrupt service routine (a subcircuit that represents the interript service routine) through the interrupt block as described in Section 5.4 of this Manual.

Please note that hardware input/output elements, including PWM generators, A/D and D/A converters, digital input/output, 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 and the Interrupt element, however, can be placed in either the main circuit or subcircuits.

9.1 PWM Generators

There are four elements associated with PWM generation: PWM generator, Space Vector PWM generator, Start PWM function, and Stop PWM function.

Images:



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

Parameters	Description
Device ID	An integer number used to identify the device that contains the PWM generator.
Dead Time	The dead time for the PWM generator, in sec.
Sampling Rate	The sampling rate, or switching frequency, of the PWM generator, in Hz
Type of Carrier Wave	Type of carrier wave. It can be either "Triangular wave" or "Sawtooth wave". Note that this parameter is for the **PWM** generator only.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it set to "Do not start", one needs to start the PWM using the "Start PWM" function.

The PWM generator generates sinusoidal PWM signals for a three-phase system. The inputs "u", "v", and "w" are for three-phase input modulation signals, and 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 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, and the range is from $-\pi$ to π.

The letter "G" in the images denotes the fact that the element is part of the general hardware.

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

The following functions are associated with these elements in the generated code. Note that these functions are hardware specific and have not been implemented yet. Users need to write the code for them.

For **PWM**:

The following functions are called during initialization:

- **GeneralPwmInit** ({device ID}, {PWM frequency}, {dead time}):
  [This function initializes the PWM frequency and dead time of the PWM generator.]
- **GeneralPwmIntrVector** ({interrupt function name})
  [This function defines the interrupt function associated with this PWM generator.]
- **SetGeneralPwmUvw** ({device ID}, {initial value of input U}, {initial value of input V}, {initial value of input W})
  [This function sets the initial values of Inputs U, V, and W of the PWM generator.]

The following function is called to set the input values for PWM:

- **SetGeneralPwmUvw** ({device ID}, {input U value}, {input V value}, {input W value})
  [This function sets the values of Inputs U, V, and W of the PWM generator.]

For **Space Vector PWM**:

The following functions are called during initialization:

- **GeneralPwmInit** ({device ID}, {PWM frequency}, {dead time}):
  [This function initializes the PWM frequency and dead time of the PWM generator.]
- **GeneralPwmIntrVector** ({interrupt function name})
  [This function defines the interrupt function associated with this PWM generator.]
- **SetGeneralPwmRa** ({device ID}, {initial value of the amplitude input R}, {initial value of the angle input A})
  [This function sets the initial values of the amplitude and angle of the space vector PWM generator.]

The following function is called to set the input values for PWM:

- **SetGeneralPwmRa** ({device ID}, {amplitude input R}, {angle input A})
  [This function sets the values of the amplitude and angle of the space vector PWM generator.]

For **Start PWM**:

The following function is called for this element when the input is high:

- **StartGeneralPwm** ({device ID}, 1):
  [This function starts the PWM generator.]

For **Stop PWM**:

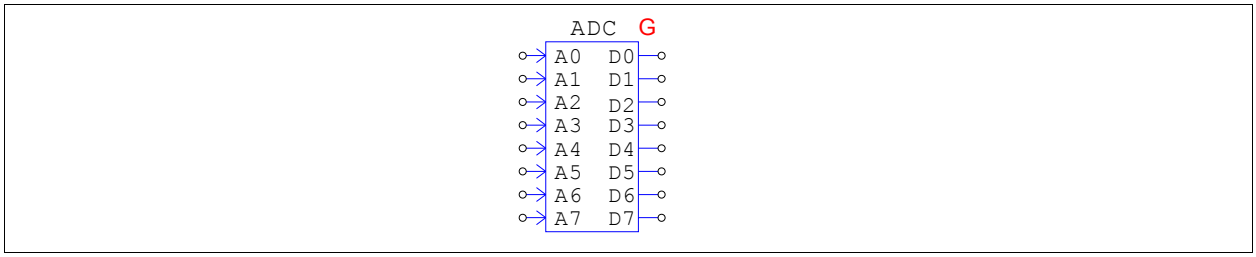
The following function is called for this element when the input is high:

- **StopGeneralPwm** ({device ID}, 0):
  [This function stops the PWM generator.]

9.2 A/D Converter

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

Image:



Attributes:

Parameters	Description
Device ID	An integer number used to identify the A/D converter.
ADC Mode	The operation mode of the A/D converter. It can be one of the five modes: - *ADC always ready*: In this mode, the A/D converter data is always ready to fetch. - *1 sampling buffer*: The converter has only one buffer and does conversion of one input channel at a time. - *2 sampling buffers*: The converter has 2 buffers, and does conversion of 2 input channels at a time. Channels A0 and A1 are read together as a group, so are Channels A2 and A3, A4 and A5, and A6 and A7. - *4 sampling buffers*: The converter has 4 buffers, and does conversion of 4 input channels at a time. Channels A0, A1, A2, and A3 are read together as a group, so are Channels A4, A5, A6, and A7. - *8 sampling buffers*: The converter has 8 buffers, and does conversion of all 8 input channels at once. Channels A0, A1, A2, A3, A4, A5, A6, and A7 are read together as a group.
Ch*i* Gain	The gain K_i of the i_{th} channel of the A/D converter.

The input range of the A/D converter is not limited. The output of the A/D converter is scaled based on the following:

$$V_o = V_i * K_i$$

The following functions are associated with the A/D converter in the generated code. Note that these functions are hardware specific and have not been implemented yet. Users need to write the code for them.

When the ADC Mode is in the "*ADC always ready*" mode:

The following function is called during initialization:

- **GeneralAdcInit0** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):
  [This function initializes the gain of each A/D channel.]

The following function is called to read the A/D converter values:

- **GeneralAdcRead** ({device ID}, {ch0 value}, {ch1 value}, {ch2 value}, {ch3 value}, {ch4 value}, {ch5 value}, {ch6 value}, {ch7 value})
  [This function reads the values of all channels.]

When the ADC Mode is in the "*1 sampling buffer*" mode:

The following function is called during initialization:

- **GeneralAdcInit1** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):
  [This function initializes the gain of each A/D channel.]

The following functions are called to read the A/D converter values:

- **GeneralAdcStart1** ({device ID}, {channel number})
  [This function starts the A/D converter for the specified input channel. The channel number can be from 0 to 7.]

- **GeneralAdcWait1** ({device ID}, {channel number})
  [This function waits until the A/D value is ready to be read.]

- **GeneralAdcRead1** ({device ID}, {channel output})
  [This function reads the value of the specified input channel.]

When the ADC Mode is in the "*2 sampling buffers*" mode:

The following function is called during initialization:

- **GeneralAdcInit2** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):
  [This function initializes the gain of each A/D channel.]

The following functions are called to read the A/D converter values:

- **GeneralAdcStart2** ({device ID}, {group number})
  [This function starts the A/D converter for the group of the input channels. The group number can be from 0 to 3.]

- **GeneralAdcWait2** ({device ID}, {group number})
  [This function waits until values of the group of the input channels are ready to be read.]

- **GeneralAdcRead2** ({device ID}, {output_a}, {output_b})
  [This function reads the two input channel values of the specified group. For example, when the group number is 0, output_a and output_b are the values of Channels 0 and 1.]

When the ADC Mode is in the "*4 sampling buffers*" mode:

The following function is called during initialization:

- **GeneralAdcInit4** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):
  [This function initializes the gain of each A/D channel.]

The following functions are called to read the A/D converter values:

- **GeneralAdcStart4** ({device ID}, {group number})
  [This function starts the A/D converter for the group of the input channels. The group number can be either 0 or 1.]

- **GeneralAdcWait4** ({device ID}, {group number})
  [This function waits until values of the group of the input channels are ready to be read. The group number can be either 0 to 1.]

- **GeneralAdcRead4** ({device ID}, {output_a}, {output_b}, {output_c}, {output_d})
  [This function reads the four input channel values of the specified group. For example, when the group number is 0, output_a, output_b, output_c, and output_d are the values of Channels 0 through 3.]

When the ADC Mode is in the "*8 sampling buffers*" mode:

The following function is called during initialization:

- **GeneralAdcInit8** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[This function initializes the gain of each A/D channel.]

The following functions are called to read the A/D converter values:

- **GeneralAdcStart8** ({device ID})
  [This function starts the A/D converter.]

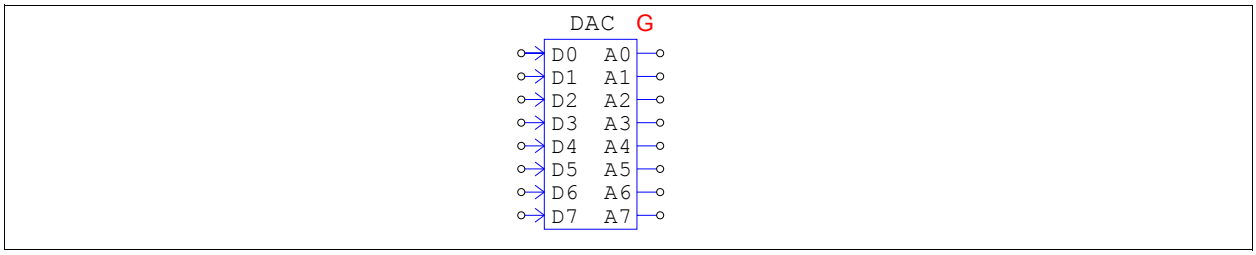
- **GeneralAdcWait8** ({device ID})
  [This function waits until values of the A/D converter are ready to be read.]

- **GeneralAdcRead8** ({device ID}, {output_0}, {output_1}, {output_2}, {output_3}, {output_4}, {output_5}, {output_6}, {output_7})
  [This function reads the eight input channel values of the A/D converter. The variable output_*i* refers to the value of the i_{th} input channel.]

9.3 D/A Converter

A D/A converter converts a digital signal into an analog signal.

Image:



Attributes:

Parameters	Description
Device ID	An integer number used to identify the D/A converter.
Ch*i* Gain	The gain K_i of the i_{th} channel of the D/A converter.

The output of the D/A converter is scaled based on the following:

$$V_o = V_i * K_i$$

Note that these functions are hardware specific and have not been implemented yet. Users need to write the code for them.

The following function is called during initialization:

- **GeneralDacInit** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):
  [This function initializes the gain of each D/A channel.]

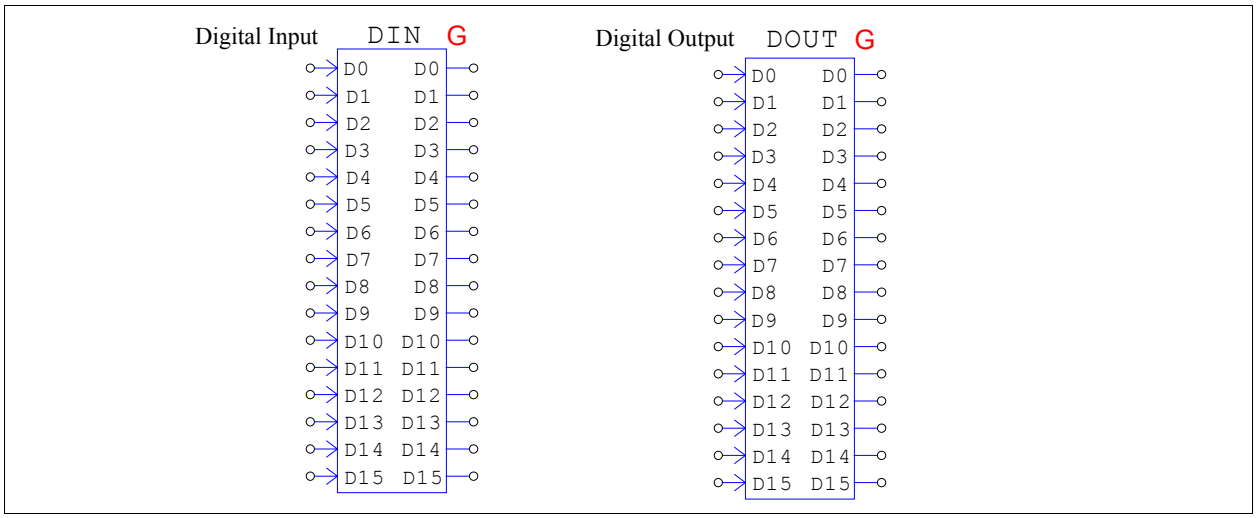
The following function is called to read the D/A converter values:

- **SetGeneralDacValue** ({device ID}, {channel number}, {value})
  [This function sets the output of the D/A converter.]

9.4 Digital Input and Output

The digital input and digital output elements are defined as below.

Images:



Attributes:

Parameters	Description
Device ID	An integer number used to identify the element.

The following functions are called for the digital input and digital output.

For *Digital Input*:

- **GetGeneralDinValue** ({device ID}):
  [This function reads the digital inputs.]

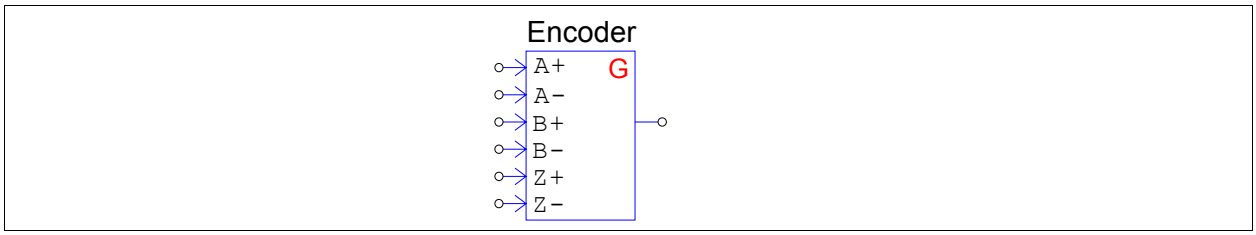
For *Digital Output*:

- **SetGeneralDoutValue** ({device ID}):
  [This function sets the digital outputs.]

9.5 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
Device ID	An integer number used to identify the encoder.
Use Z Signal	It can be either "True" or "False". When it is set to "True", the encoder operates in the ABZ 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-.

The following function is called during initialization:

- **SetEncoderMode** ({device ID}, {mode flag}):
  [This function sets the operating mode of the encoder.]

- **ClearEncoderCount** ({device ID}):
  [This function clears the counter of the encoder.]

- **EnableEncoderIntr** ({device ID}):
  [This function enables the encoder interrupt.]

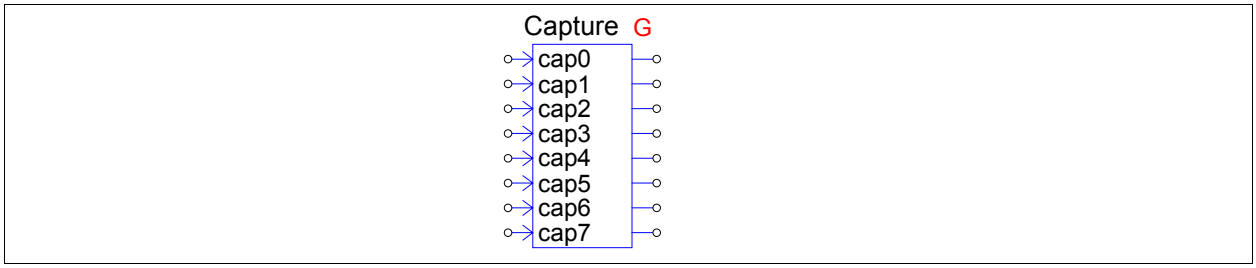
The following function is called to read the encoder value:

- **GetGeneralEncoderValue** ({device ID})
  [This function reads the encoder value.]

9.6 Capture

A capture can capture the counter value of an encoder or general-purpose timer.

Images:



Attributes:

Parameters	Description
Device ID	An integer number used to identify the capture.
Ch*i* Counter Source	The name of the counter source. It can be either "GP_TIMER" for general-purpose timer, or the name of an encoder.

The capture element has 8 inputs. When an input changes from low to high (from 0 to 1), this element will capture the counter value of the source, and output it through the output port.

The following function is called during initialization:

- **SetGeneralCaptureMode** ({device ID}, {channel number}, {}, {}):
  [This function sets the operating mode of the capture.]

The following function is called to read the encoder value:

- **GetGeneralCaptureCount** ({device ID})
  [This function reads the capture counter value.]

Index

A

APWM 34, 39, 53, 56, 59, 60

B

Backward Euler 4
Bilinear method 4

C

C block 6, 15
  simplified 22, 23
capture 29, 61, 64, 65, 69, 75
capture PWM 53
capture state 31, 47
CloseSimUser 15
Code Composer Studio 49
code generation 1, 3, 10, 19, 21
  for sub-system 13, 14
  with hardware target 7
  without hardware target 5
connection
  event 18, 25, 26
converter
  A/D 7, 8, 13, 31, 61, 64, 69, 71
  D/A 69
counter 13, 24, 31, 61, 64, 65

D

D/A 73
digital input 12, 13, 29, 31, 61, 64, 69, 73
digital output 31, 61, 66, 73
DSP clock 31, 48

E

encoder 13, 31, 61
encoder state 31, 46, 47
event
  default 17, 26
  input 17, 18, 19, 24, 25, 26
  output 17, 25, 26

F

F28335 31
file
  parameter 18, 22, 23
flash RAM release 49, 50
flash release 49, 50, 51

H

hardware 7
  F28335 31
  general 69
  PE-Pro/F28335 53

I

interrupt 29, 66, 69, 75
  hardware 28

L

LED 66, 67
library
  PE-Expert3 runtime 67
  SimCoder 13, 21
  standard PSIM 1, 21

M

memory allocation 49, 50

O

OpenSimUser 15

P

parameter
  global 23
PE-Expert3 7, 8, 24, 61, 67
PE-Pro/F28335 1, 53, 54, 55, 56, 57, 59
PE-View 8, 10, 61, 66
port
  bi-directional 13
  input event 17, 18, 19, 25, 26
  input signal 13
  output event 17, 18, 19, 25
  output signal 13
  signal 18
  uni-directional 13
project setting 49
PWM 61, 62
  start 61, 63, 69, 70
  stop 61, 69
PWM (sub) generator 61, 62, 63
PWM generator 69
  space vector 61, 62, 69

R

RAM 50
RAM debug 49, 50
RAM release 49, 50
RunSimUser 15

S

sawtooth waveform 24
sequence control 7, 11
simulation control 6, 7, 8

subcircuit
    regular 18
    with events 18, 19
    with hardware interrupt 19
system
    in continuous domain 3
    in discrete domain 4
    with event control 10

T

TI F28335 1, 21, 28, 31, 33, 40, 41, 45, 46, 47, 48, 49, 54
trip-zone 31, 33, 34, 35, 37, 40, 41, 54
trip-zone state 31, 40, 41

U

unit delay 8, 22, 63

V

variable
    global 17, 18, 26, 27

Z

zero-order hold 13, 22