Digital Control Loop Designer SDK User Guide

z-Domain Configuration Window

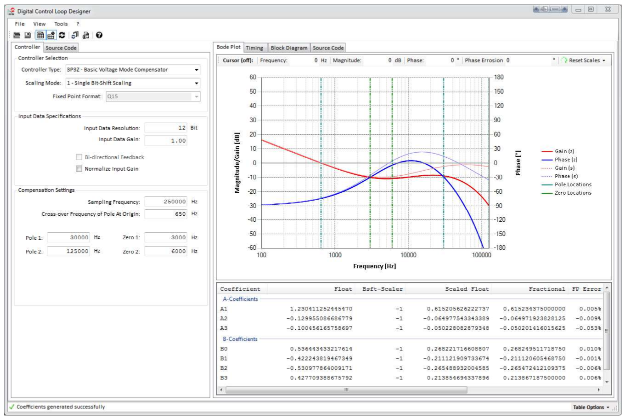
INTRODUCTION

The Digital Control Loop Designer SDK is a Software Development Kit (SDK) consisting of individual tools covering system definition, system modeling, code generation, control system fine tuning and real-time debugging of fully digital control systems for Switched-Mode Power Supplies (SMPS) for dsPIC® Digital Signal Controllers (DSC).

This user guide section is meant to be a quick start guide to the z-Domain Controller Configuration tool (DCLD.exe), which can be used to select and configure discrete time domain control systems, tailor their features to the specific controller device used and generate control libraries with a generic application programming interface (API) to allow fast and seamless integration of the generated source code in custom firmware projects.

PLEASE NOTE

This software is still in a preliminary, experimental state with limited support. All features and functions are subject to change at any time without further rnotice. Please always refer to the most recent readme.txt file to get updates on features and functions and to review the release history.



Controller Selection

Controller Type: 3P3Z - Basic Voltage Mode Compensator

Scaling Mode: 1 - Single Bit-Shift Scaling

Fixed Point Format: Q15

Input Data Specifications

Input Data Resolution: 12 Bit

Input Data Gain: 1.00

Bi-directional Feedback

Normalize Input Gain

Compensation Settings

Sampling Frequency: 250000 Hz

Cross-over Frequency of Pole At Origin: 650 Hz

Pole 1: 30000 Hz | Zero 1: 3000 Hz

Pole 2: 125000 Hz | Zero 2: 6000 Hz

Cursor (off): Frequency: 0 Hz | Magnitude: 0 dB | Phase: 0 ° | Phase Errosion 0 °

Coefficient	Float	Bsft-Scaler	Scaled Float	Fractional	FP Error
A-Coefficients					
A1	1.230411252445470	-1	0.615205626222737	0.615234375000000	0.005%
A2	-0.129955086686779	-1	-0.064977543343389	-0.064971923828125	-0.009%
A3	-0.100456165758697	-1	-0.050228082879348	-0.050201416015625	-0.053%
B-Coefficients					
B0	0.536443433217614	-1	0.268221716608807	0.268249511718750	0.010%
B1	-0.422243819467349	-1	-0.211121909733674	-0.211120605468750	-0.001%
B2	-0.530977864009171	-1	-0.265488932004585	-0.265472412109375	-0.006%
B3	0.427709388675792	-1	0.213854694337896	0.213867187500000	0.006%

Coefficients generated successfully

Figure 1: z-Domain Configuration Window of the Digital Control Loop Designer SDK

Recommended Literature:

Data sheets and reference manuals are available on http://www.microchip.com.

- dsPIC33EP128GS806 data sheet
- dsPIC33CH128MP506 data sheet
- dsPIC33CK256MP506 data sheet
-

Technical Specifications:

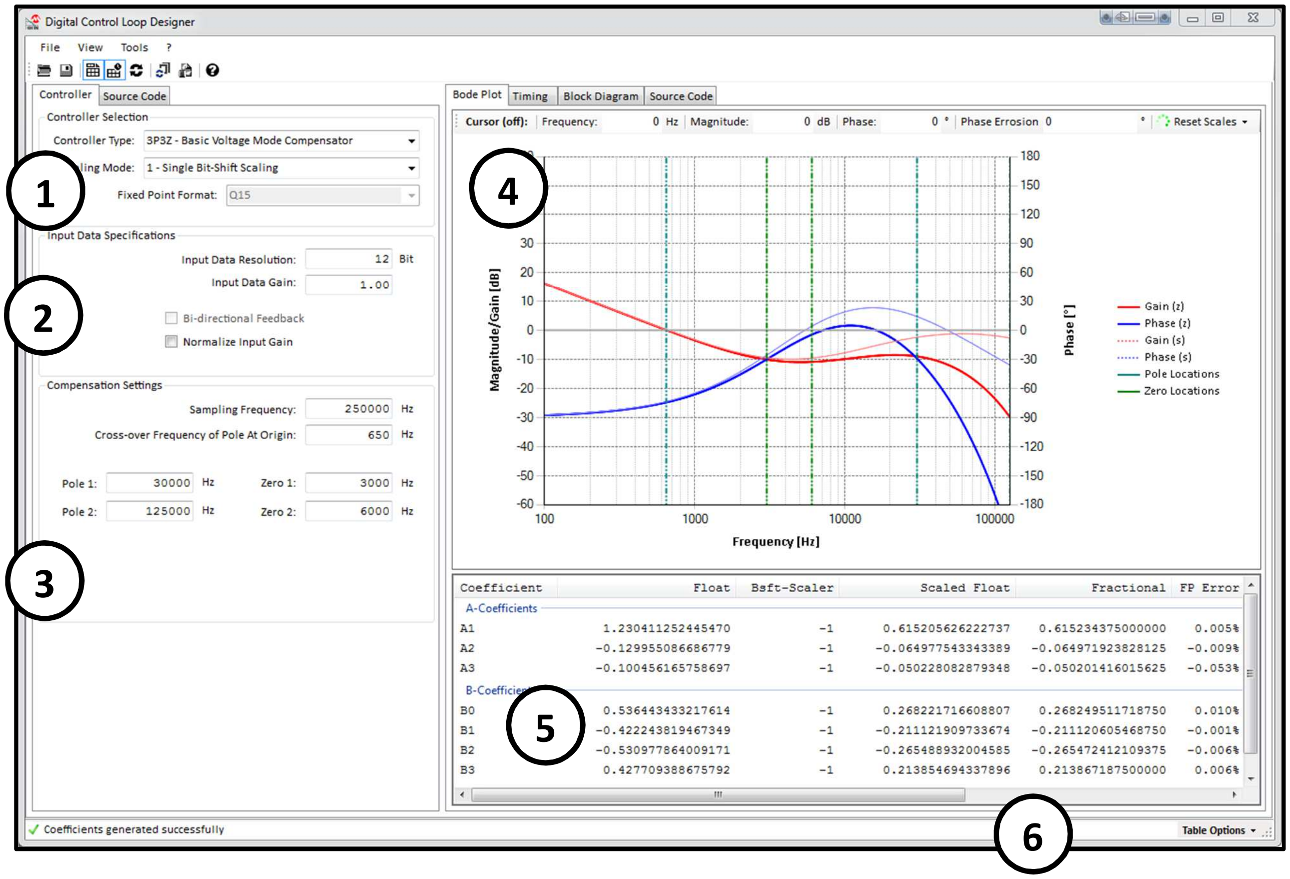
Minimum System Requirements:

- Microsoft Windows® 7 32-bit Operating System
- 1 GB RAM
- 24 MB of free hard drive space

(this page was left blank intentionally)

TABLE OF CONTENTS

1.0 SOFTWARE OVERVIEW



Controller Selection

Controller Type: 3P3Z - Basic Voltage Mode Compensator

Scaling Mode: 1 - Single Bit-Shift Scaling

Fixed Point Format: Q15

Input Data Specifications

Input Data Resolution: 12 Bit

Input Data Gain: 1.00

☐ Bi-directional Feedback

☐ Normalize Input Gain

Compensation Settings

Sampling Frequency: 250000 Hz

Cross-over Frequency of Pole At Origin: 650 Hz

Pole 1: 30000 Hz Zero 1: 3000 Hz

Pole 2: 125000 Hz Zero 2: 6000 Hz

Coefficient	Float	Bsft-Scaler	Scaled Float	Fractional	FP Error
A-Coefficients					
A1	1.230411252445470	-1	0.615205626222737	0.615234375000000	0.005%
A2	-0.129955086686779	-1	-0.064977543343389	-0.064971923828125	-0.009%
A3	-0.100456165758697	-1	-0.050228082879348	-0.050201416015625	-0.053%
B-Coefficients					
B0	0.536443433217614	-1	0.268221716608807	0.268249511718750	0.010%
B1	-0.422243819467349	-1	-0.211121909733674	-0.211120605468750	-0.001%
B2	-0.530977864009171	-1	-0.265488932004585	-0.265472412109375	-0.006%
B3	0.427709388675792	-1	0.213854694337896	0.213867187500000	0.006%

✓ Coefficients generated successfully

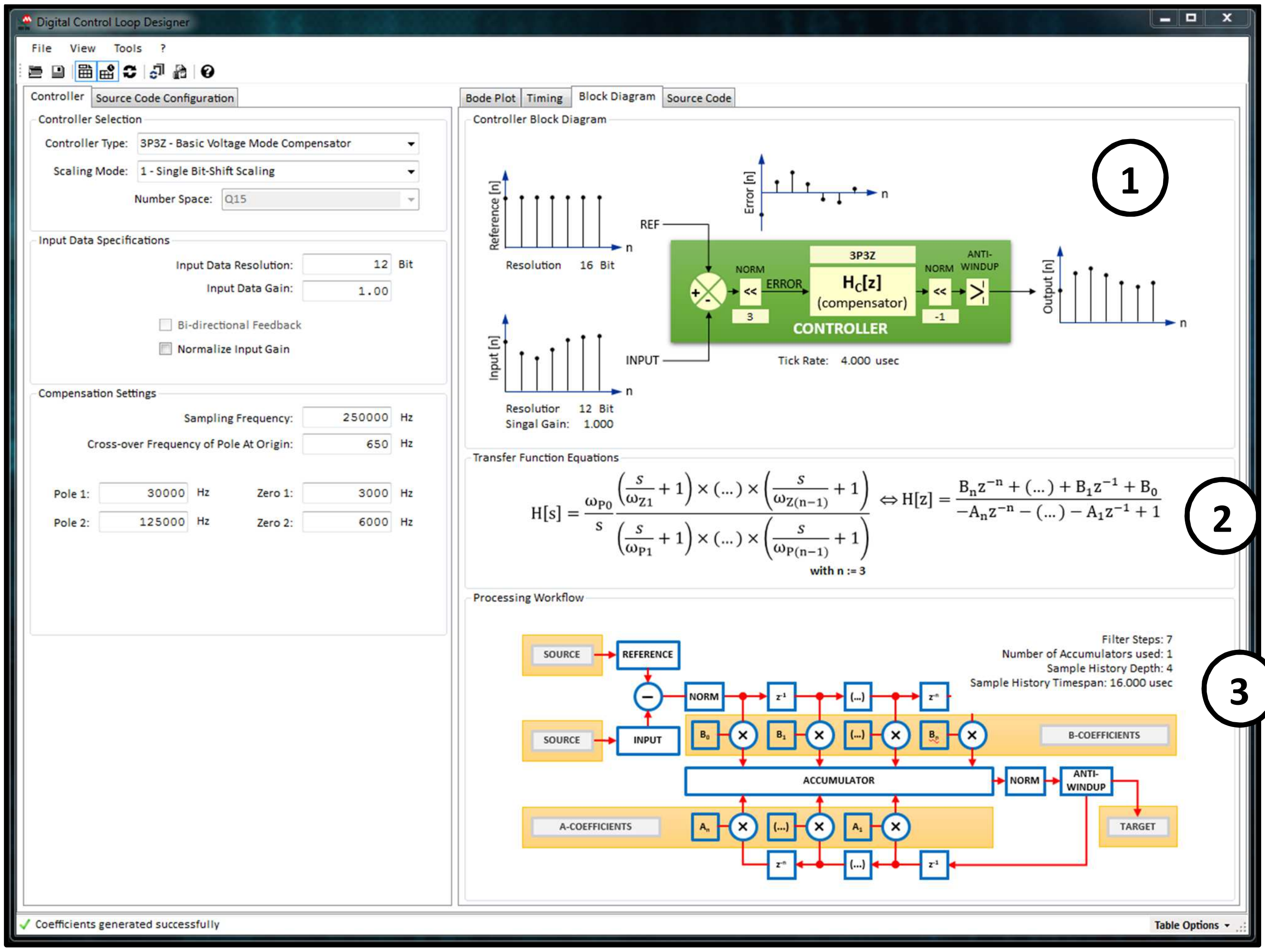
Figure 2: Digital Control Loop Designer z-Domain Controller Configuration Main Window

TABLE 1: MAIN WINDOW DESCRIPTION

No	Description
1	Controller Order and Number Format Selection
2	Input Data Specification
3	Compensator Configuration
4	Frequency Response (Bode Plot) of s-Domain and z-Domain transfer function
5	Digital Filter Coefficients derivation transcript with accuracy analysis of final values
6	Status bar indicating background activity and output messages

The z-Domain Controller configuration window is ordered into a left configuration plane and a right plane showing the results based on recent settings. Both planes are separated in individual sub-planes (tabs) offering access to settings of individual, functional blocks.

The default view starts with the controller selection and frequency domain configuration on the left and the Bode plot graph of the transfer function on the right. Below the Bode plot a data table shows the derivation transcript of the calculation result. This table is also used to display warnings of the number accuracy analyzer.



Controller Selection

Controller Type: 3P3Z - Basic Voltage Mode Compensator

Scaling Mode: 1 - Single Bit-Shift Scaling

Number Space: Q15

Input Data Specifications

Input Data Resolution: 12 Bit

Input Data Gain: 1.00

☐ Bi-directional Feedback

☐ Normalize Input Gain

Compensation Settings

Sampling Frequency: 250000 Hz

Cross-over Frequency of Pole At Origin: 650 Hz

Pole 1:	30000	Hz	Zero 1:	3000	Hz
Pole 2:	125000	Hz	Zero 2:	6000	Hz

Resolution 16 Bit

Tick Rate: 4.000 usec

Resolution 12 Bit
Singal Gain: 1.000

Transfer Function Equations

$$H[s] = \frac{\omega_{P0}}{s}\frac{\left(\frac{s}{\omega_{Z1}}+1\right)\times(\ldots)\times\left(\frac{s}{\omega_{Z(n-1)}}+1\right)}{\left(\frac{s}{\omega_{P1}}+1\right)\times(\ldots)\times\left(\frac{s}{\omega_{P(n-1)}}+1\right)} \Leftrightarrow H[z] = \frac{B_n z^{-n} + (\ldots) + B_1 z^{-1} + B_0}{-A_n z^{-n} - (\ldots) - A_1 z^{-1} + 1}$$

with n := 3

Filter Steps: 7
Number of Accumulators used: 1
Sample History Depth: 4
Sample History Timespan: 16.000 usec

✓ Coefficients generated successfully

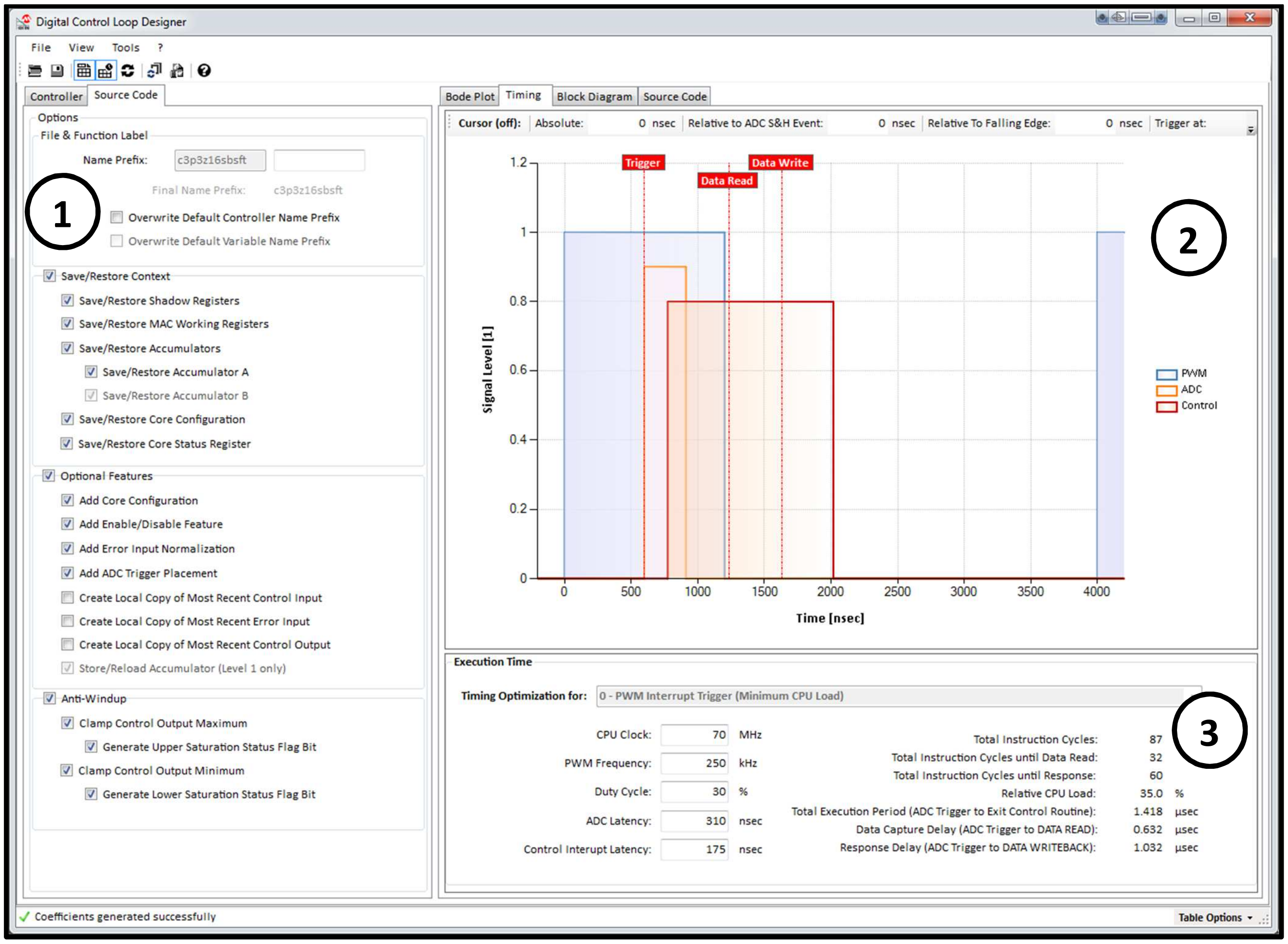
Table Options

Figure 3: Block Diagram Overview

TABLE 2: BLOCK DIAGRAM OVERVIEW DESCRIPTION

No	Description
1	Controller block diagram with discrete time domain signal waveforms
2	Generic format of s- and z-domain compensator transfer function equivalents
3	Firmware module implementation block diagram and flow-chart

The block diagram overview shows three different block diagrams of the system which is configured by this tool, including system description block diagram, mathematical equations used and the firmware implementation and flow chart of the generated code modules.



Options

File & Function Label

Name Prefix: c3p3z16sbsft

Final Name Prefix: c3p3z16sbsft

- ☐ Overwrite Default Controller Name Prefix
- ☐ Overwrite Default Variable Name Prefix

☑ Save/Restore Context

- ☑ Save/Restore Shadow Registers
- ☑ Save/Restore MAC Working Registers
- ☑ Save/Restore Accumulators
  - ☑ Save/Restore Accumulator A
  - ☑ Save/Restore Accumulator B
- ☑ Save/Restore Core Configuration
- ☑ Save/Restore Core Status Register

☑ Optional Features

- ☑ Add Core Configuration
- ☑ Add Enable/Disable Feature
- ☑ Add Error Input Normalization
- ☑ Add ADC Trigger Placement
- ☐ Create Local Copy of Most Recent Control Input
- ☐ Create Local Copy of Most Recent Error Input
- ☐ Create Local Copy of Most Recent Control Output
- ☑ Store/Reload Accumulator (Level 1 only)

☑ Anti-Windup

- ☑ Clamp Control Output Maximum
  - ☑ Generate Upper Saturation Status Flag Bit
- ☑ Clamp Control Output Minimum
  - ☑ Generate Lower Saturation Status Flag Bit

Cursor (off): | Absolute: 0 nsec | Relative to ADC S&H Event: 0 nsec | Relative To Falling Edge: 0 nsec | Trigger at:

Execution Time

Timing Optimization for: 0 - PWM Interrupt Trigger (Minimum CPU Load)

CPU Clock:	70	MHz
PWM Frequency:	250	kHz
Duty Cycle:	30	%
ADC Latency:	310	nsec
Control Interupt Latency:	175	nsec

Total Instruction Cycles:	87	
Total Instruction Cycles until Data Read:	32	
Total Instruction Cycles until Response:	60	
Relative CPU Load:	35.0	%
Total Execution Period (ADC Trigger to Exit Control Routine):	1.418	µsec
Data Capture Delay (ADC Trigger to DATA READ):	0.632	µsec
Response Delay (ADC Trigger to DATA WRITEBACK):	1.032	µsec

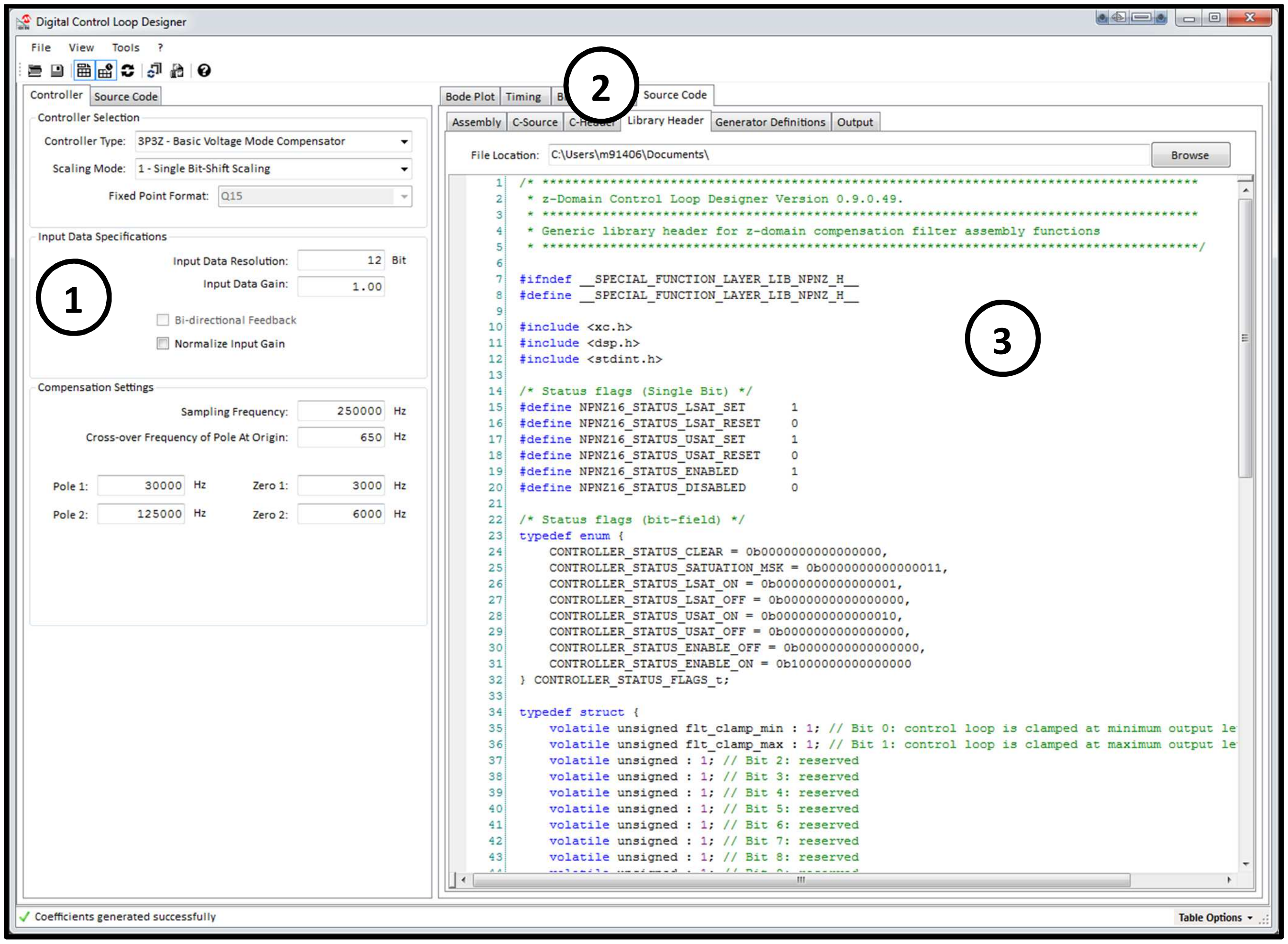
✓ Coefficients generated successfully

Figure 4: Code Generator Configuration and Timing Diagram

TABLE 3: TIMING DIAGRAM OVERVIEW DESCRIPTION

No	Description
1	Code generator configuration option catalog
2	Timing diagram of control loop execution vs. SMPS switching waveform
3	Timing calculation output and target device parameter configuration

Each controller selection and code generation option is impacting the execution time of the control system and thus on the total CPU workload. This view allows the control block analysis and optimization with regards to device speed, core architecture, peripheral features and external physical parameters like switching period and duty ratio.



Controller Selection

Controller Type: 3P3Z - Basic Voltage Mode Compensator

Scaling Mode: 1 - Single Bit-Shift Scaling

Fixed Point Format: Q15

Input Data Specifications

Input Data Resolution: 12 Bit

Input Data Gain: 1.00

Bi-directional Feedback

Normalize Input Gain

Compensation Settings

Sampling Frequency: 250000 Hz

Cross-over Frequency of Pole At Origin: 650 Hz

Pole 1: 30000 Hz   Zero 1: 3000 Hz

Pole 2: 125000 Hz   Zero 2: 6000 Hz

File Location: C:\Users\m91406\Documents\

```
/* *****************************************************************************
 * z-Domain Control Loop Designer Version 0.9.0.49.
 * *****************************************************************************
 * Generic library header for z-domain compensation filter assembly functions
 * *****************************************************************************/

#ifndef __SPECIAL_FUNCTION_LAYER_LIB_NPNZ_H__
#define __SPECIAL_FUNCTION_LAYER_LIB_NPNZ_H__

#include <xc.h>
#include <dsp.h>
#include <stdint.h>

/* Status flags (Single Bit) */
#define NPNZ16_STATUS_LSAT_SET      1
#define NPNZ16_STATUS_LSAT_RESET    0
#define NPNZ16_STATUS_USAT_SET      1
#define NPNZ16_STATUS_USAT_RESET    0
#define NPNZ16_STATUS_ENABLED       1
#define NPNZ16_STATUS_DISABLED      0

/* Status flags (bit-field) */
typedef enum {
    CONTROLLER_STATUS_CLEAR = 0b0000000000000000,
    CONTROLLER_STATUS_SATUATION_MSK = 0b0000000000000011,
    CONTROLLER_STATUS_LSAT_ON = 0b0000000000000001,
    CONTROLLER_STATUS_LSAT_OFF = 0b0000000000000000,
    CONTROLLER_STATUS_USAT_ON = 0b0000000000000010,
    CONTROLLER_STATUS_USAT_OFF = 0b0000000000000000,
    CONTROLLER_STATUS_ENABLE_OFF = 0b0000000000000000,
    CONTROLLER_STATUS_ENABLE_ON = 0b1000000000000000
} CONTROLLER_STATUS_FLAGS_t;

typedef struct {
    volatile unsigned flt_clamp_min : 1; // Bit 0: control loop is clamped at minimum output le
    volatile unsigned flt_clamp_max : 1; // Bit 1: control loop is clamped at maximum output le
    volatile unsigned : 1; // Bit 2: reserved
    volatile unsigned : 1; // Bit 3: reserved
    volatile unsigned : 1; // Bit 4: reserved
    volatile unsigned : 1; // Bit 5: reserved
    volatile unsigned : 1; // Bit 6: reserved
    volatile unsigned : 1; // Bit 7: reserved
    volatile unsigned : 1; // Bit 8: reserved
```

Coefficients generated successfully

Figure 5: Code Generator Output View

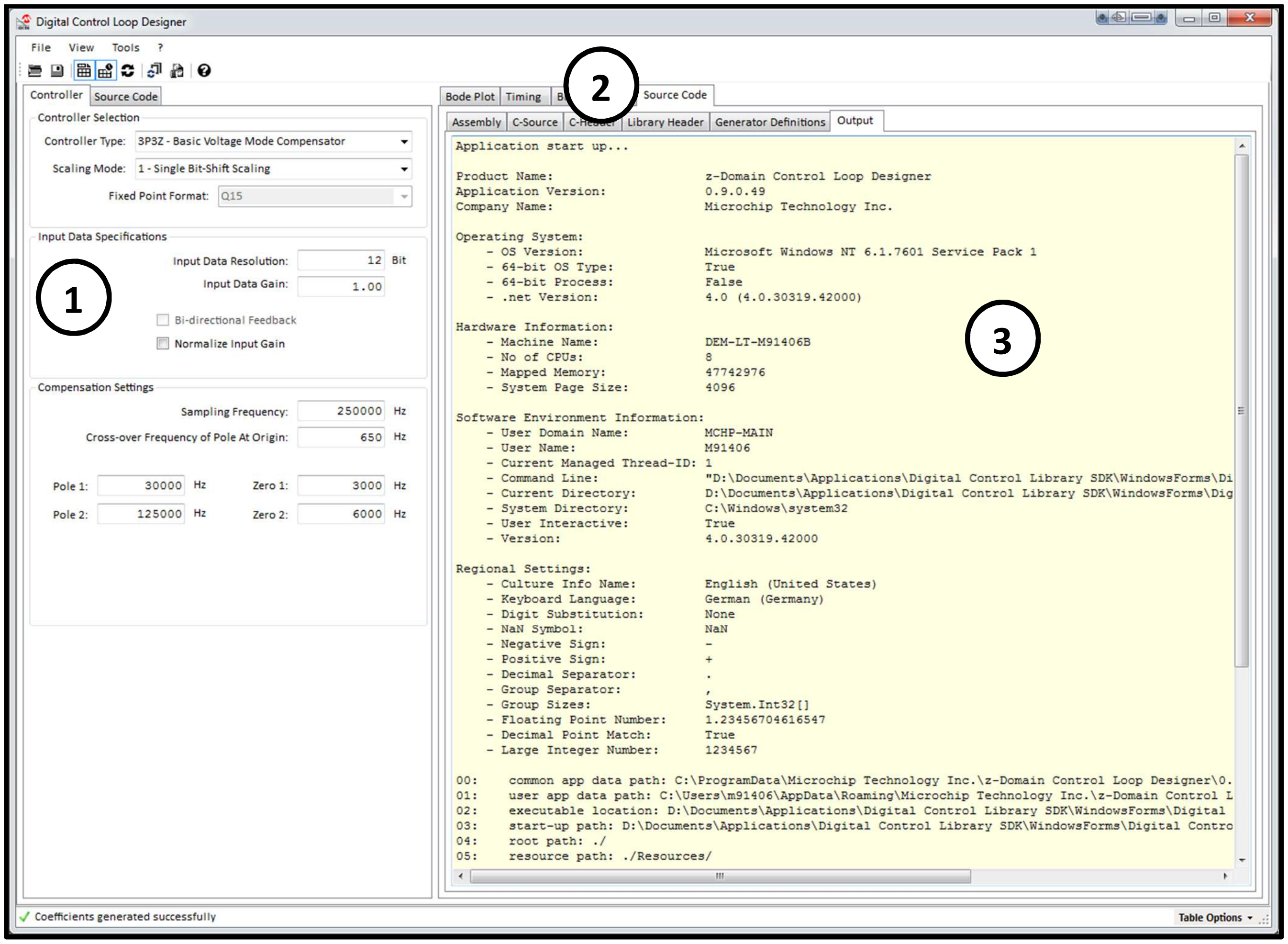
TABLE 4: CODE GENERATOR OUTPUT VIEW DESCRIPTION

No	Description
1	Control loop / Code generator configuration option catalog
2	Source code output tab controls (access to output windows of assembly and c-code modules)
3	Source code output window

Once controller and code generation has been configured, the code generator output is displayed in individual output windows for assembly and C-code modules, where the code can be reviewed and edited[1][2].

[1] user code changes may get overwritten by the generator without warning (see Code Generator Settings)

[2] code editor has no compiler support



Controller Selection
Controller Type: 3P3Z - Basic Voltage Mode Compensator
Scaling Mode: 1 - Single Bit-Shift Scaling
Fixed Point Format: Q15

Input Data Specifications
Input Data Resolution: 12 Bit
Input Data Gain: 1.00
Bi-directional Feedback
Normalize Input Gain

Compensation Settings
Sampling Frequency: 250000 Hz
Cross-over Frequency of Pole At Origin: 650 Hz
Pole 1: 30000 Hz   Zero 1: 3000 Hz
Pole 2: 125000 Hz   Zero 2: 6000 Hz

```
Application start up...

Product Name:                  z-Domain Control Loop Designer
Application Version:           0.9.0.49
Company Name:                  Microchip Technology Inc.

Operating System:
    - OS Version:              Microsoft Windows NT 6.1.7601 Service Pack 1
    - 64-bit OS Type:          True
    - 64-bit Process:          False
    - .net Version:            4.0 (4.0.30319.42000)

Hardware Information:
    - Machine Name:            DEM-LT-M91406B
    - No of CPUs:              8
    - Mapped Memory:           47742976
    - System Page Size:        4096

Software Environment Information:
    - User Domain Name:        MCHP-MAIN
    - User Name:               M91406
    - Current Managed Thread-ID: 1
    - Command Line:            "D:\Documents\Applications\Digital Control Library SDK\WindowsForms\Di
    - Current Directory:       D:\Documents\Applications\Digital Control Library SDK\WindowsForms\Dig
    - System Directory:        C:\Windows\system32
    - User Interactive:        True
    - Version:                 4.0.30319.42000

Regional Settings:
    - Culture Info Name:       English (United States)
    - Keyboard Language:       German (Germany)
    - Digit Substitution:      None
    - NaN Symbol:              NaN
    - Negative Sign:           -
    - Positive Sign:           +
    - Decimal Separator:       .
    - Group Separator:         ,
    - Group Sizes:             System.Int32[]
    - Floating Point Number:   1.23456704616547
    - Decimal Point Match:     True
    - Large Integer Number:    1234567

00:    common app data path: C:\ProgramData\Microchip Technology Inc.\z-Domain Control Loop Designer\0.
01:    user app data path: C:\Users\m91406\AppData\Roaming\Microchip Technology Inc.\z-Domain Control L
02:    executable location: D:\Documents\Applications\Digital Control Library SDK\WindowsForms\Digital
03:    start-up path: D:\Documents\Applications\Digital Control Library SDK\WindowsForms\Digital Contro
04:    root path: ./
05:    resource path: ./Resources/
```

✓ Coefficients generated successfully

Figure 6: Output Window

TABLE 5: OUTPUT WINDOW DESCRIPTION

No	Description
1	Control loop / Code generator configuration option catalog
2	Source code output tab controls (access to output windows of assembly and c-code modules)
3	Output Window view

The output window is an additional software debugging tool helping to verify proper and reliable output results and offers additional information for troubleshooting software and platform issues. It lists system information, folder settings, user inputs and internal messages during execution.

2.0 CONTROLLER CONFIGURATION

Controller Selection

After application start the main window is starting with the controller selection. All digital controllers supported by this tool are based on discrete time domain transfer functions, which have been derived from continuous time domain transfer function prototypes using the bi-linear transform (BLT).

The continuous time domain prototype transfer functions are based on conventional type I, II, III compensation circuits used in SMPS control systems the industry since the early 1980s. All higher order transfer functions used to create this tool, are mathematically up-scaled versions of the same control approach.

These controllers consist of lead-lag compensators of the n-th order multiplied with a simple integrator term incorporating the gain influence over frequency of the pole at the origin. The only difference between these transfer functions is the order of the lead-lag compensator term, which may include no pole/zero pair at all (1st order) or up to n pole/zero pairs (n-th order).

Why using s-Domain Prototype Filters?

Creating discrete time domain controllers would not necessarily require the deviation of their transfer function from a continuous time domain prototype filter. However, this deviation path allows to establish relations between continuous and discrete time domain control systems, which allows on one side to use the identical design techniques and tools for both systems while still being able to consider, incorporate or compensate for the differences between them.

The z-Domain Configuration Window of the Digital Control Loop Designer SDK takes pole and zero frequency locations to characterize the compensation filter and to adjust the total control loop gain and can therefore directly be applied to frequency domain system models, merging continuous and discrete time domain blocks.

Selecting the Right Controller

The selection of the right controller exclusively depends on the power supply plant and control mode used. In any case the controller selection depends on the number of poles and zeros required within the controller to compensate for poles and zeros of the power supply plant. The controller selection offers six orders:

- **1P1Z:**
  1st order with integrator and no pole/zero pair
- **2P2Z:**
  2nd order with integrator and one pole/zero pair
- **3P3Z:**
  3nd order with integrator and two pole/zero pairs
- **4P4Z:**
  4nd order with integrator and three pole/zero pairs
- **5P5Z:**
  5nd order with integrator and four pole/zero pairs
- **6P6Z:**
  6nd order with integrator and five pole/zero pairs

Every pole and zero location can be set by either moving the respective pole or zero indicator in the Bode plot using the mouse pointer or edit the frequency in the respective entry boxes below the controller selection.

Scaling Mode Selection

Each controller design is a tradeoff between performance and accuracy. With increased number space the result accuracy can be enhanced. Enhanced accuracy, however, requires more CPU cycles for the computation. To solve these tradeoffs easily, the z-Domain Configuration Window offers four different scaling mode for each controller type:

- **Single Bit-Shift Scaling**
  Highest performance is achieved by directly utilizing the fixed-point DSP core of the dsPIC® DSC by scaling all filter coefficients with the very same scaling factor. The factor scaling is implemented by shifting the number bit code to

the right (divide by power of 2) or left (multiply by power of 2). This scaling method is sufficient for a wide variety of applications with standard topologies.

- **Single Bit-Shift with Output Factor Scaling**
  Occasionally coefficients with single fixed scalers may be affected by accuracy limitations, which, in the worst case, could corrupt the convolution process of the digital filter and negatively affect the error integration.

  In this scaling mode one additional factor is added and all coefficients are rescaled to minimize the number error.
- **Dual Bit-Shift Scaling**
  The single bit-shifting more with output factor scaling may come short, when coefficients of filter terms A and B vary significantly in size. For these conditions the Dual Bit Shift Mode was introduced, which applies two different scalers for A and B term coefficients. The performance impact is very similar to the Single Bit-Shift with Output Factor Scaling.
- **Fast Floating Point Coefficient Scaling**
  Fast floating point numbers have re-ordered binary encoding. This number format is different from conventional IEEE 754 floating point numbers to optimize the computation process on fixed-point DSP cores. This number format computation is the most accurate but also most intensive in terms of CPU cycles.

The recommended process of determining the best scaling mode is to start from single bit-shifting, observing the coefficient output window below the Bode plot. Should one or more coefficients exceed 0.5% error, it will be marked in yellow (warning level), if the inaccuracy exceeds 1.0%, it will be marked red (error level).

Should any of these warning appear, increase the scaling option until all warning disappear. Observe the timing diagram on the right, to keep track on the CPU load and overall timing alignment.

Input Data Specification

This section is used to normalize the input data to the computation engine. The input data range needs to meet the number format of controller to prevent gain mismatches between model and desired control output. This usually strictly refers to the number bit resolution.

Additionally, the Input Data Specification offers three additional options accounting for physical signal scaling:

- **Input Data Gain**
  Assuming the input data is coming from an analog-to-digital converter (ADC) reading a pre-conditioned analog signal, the gain of the signal conditioning circuit can be entered here (e.g. reading voltage from a voltage divider).
  As a result, the Bode plot graph on the right will show the impact on the frequency response of the controller (gains < 1 will drop the gain, gains > 1will increase the gain)
- **Input Signal Gain Compensation**
  In case the input data gain is different from 1, this option will automatically increase/decrease the gain of the controller to compensate for the physical signal gain. If this option is not selected, gain variations have to be compensated by manual adjustments of the Cross-over frequency of the pole at the origin.
- **Input Data Offsets in Bi-Directional Control Systems**
  By selecting this option the input data is assumed to have a zero-point offset. As this offset value usually changes during runtime or needs to be calibrated under specific conditions, this offset value needs to be specified *in user code* to allow positive and negative input data values.

3.0 TIMING ANALYSIS

In addition to the frequency response chart (Bode plot) a Timing graph is provided, showing the switching period and on-time of the nominal switching signal (blue), the ADC conversion time (orange) and the control loop execution (red) based on the most recent settings.

The switching waveform, ADC timing parameters and CPU speed can be entered underneath the Timing graph. The graph shows the position of data acquisition events, control loop call events, READ and WRITE BACK events along the switching period.

Controller selection, Scaling Mode selection and Code Generation options influence the timing alignment. In high-speed systems this graph can help to analyze the estimated CPU load added by the recently configured controller and help to find the right tradeoff between performance and accuracy.

For more precise analysis the cursor can be used to measure relative time between events by moving/placing the cursor cross within the timing graph.

Additional presets allow to quickly position data acquisition triggers at common trigger points within the switching period (e.g. at 50% on-time, 50% off-time or at the beginning of the switching cycle.

It's also possible to place the trigger at any point by dragging the trigger cursor to the left/right ti fine-tune trigger offsets.

4.0 BLOCK DIAGRAM VIEW

The block diagram view is static and will not reflect details of the most recent settings. Its purpose is to reflect the system level covered by the z-Domain Configuration Window.

The first view shows a high-level block diagram of the controller with its discrete time domain input signals and processing path.

The second section shows the continuous time domain transfer function equation and the resulting discrete time domain transfer function equation as reference.

The third section shows the controller implementation block diagram and flow chart.

5.0 SOURCE CODE VIEW

The Source Code View covers multiple sub windows for every generated code module. The generated control library source code provides four different files:

- **Optimized Assembly Code**
  All runtime functions are generated as optimized assembly routines. These routines read data from and write data to a data structure (`cNPNZ16b_t`), which holds all parameters and pointers to Special Function Registers (SFRs) used by the library. This data is loaded into the data structure by the C-domain initialization code. Depending on code generator options selected, additional information will be written to the data structure, from which C-domain application code can gain access (e.g. status bits, most recent calculation results, etc.)
- **C-Source File**
  The C-source file contains the static set of filter coefficients and the data structure initialization function of this individual controller.

PLEASE NOTE
The C source initialization routine only initializes the digital filter coefficients and number scaling settings. Controller/system-specific parameters like anti-windup thresholds, source and target registers as well as ADC trigger offsets must be set in user code!

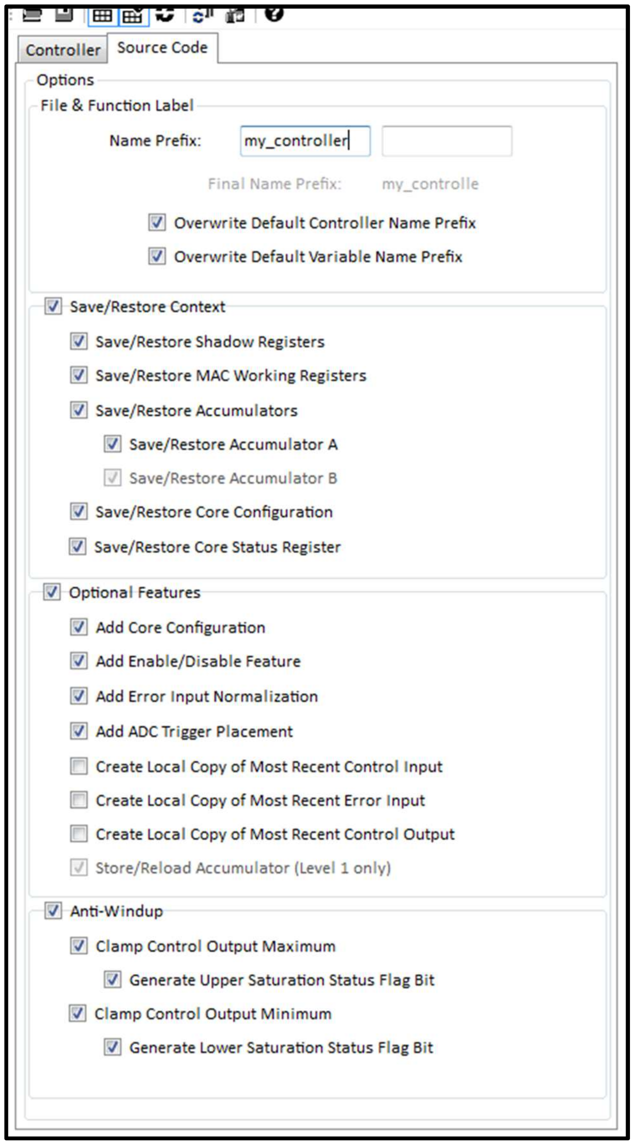
- **C-Header File**
  The C-header file holds all public variable and function declarations of this individual controller, which need to be accessible from application code.
- **C-Library Header File**
  The library header is generic and would only have to be added once, even if the control system is

based on multiple, individual controllers. This file also holds the declaration of the `cNPNZ16b_t` data structure. This data structure is the sole access point to configure, control and monitor the controller block.

PLEASE NOTE

The `cNPNZ16b_t` data structure declared in the library header file may be different for different scaling modes and code generator options. When controllers with fundamentally different configuration sare used, conflicts may occur.

6.0 CODE GENERATOR OPTIONS



Controller | Source Code

Options

File & Function Label

Name Prefix: my_controller

Final Name Prefix: my_controlle

- [✓] Overwrite Default Controller Name Prefix
- [✓] Overwrite Default Variable Name Prefix

[✓] Save/Restore Context

- [✓] Save/Restore Shadow Registers
- [✓] Save/Restore MAC Working Registers
- [✓] Save/Restore Accumulators
  - [✓] Save/Restore Accumulator A
  - [✓] Save/Restore Accumulator B
- [✓] Save/Restore Core Configuration
- [✓] Save/Restore Core Status Register

[✓] Optional Features

- [✓] Add Core Configuration
- [✓] Add Enable/Disable Feature
- [✓] Add Error Input Normalization
- [✓] Add ADC Trigger Placement
- [ ] Create Local Copy of Most Recent Control Input
- [ ] Create Local Copy of Most Recent Error Input
- [ ] Create Local Copy of Most Recent Control Output
- [✓] Store/Reload Accumulator (Level 1 only)

[✓] Anti-Windup

- [✓] Clamp Control Output Maximum
  - [✓] Generate Upper Saturation Status Flag Bit
- [✓] Clamp Control Output Minimum
  - [✓] Generate Lower Saturation Status Flag Bit

Figure 7: Code Generator Options

The code generator offers several options helping to tailor the code implementation, features and overall timing to application specific needs.

FILE & FUNCTION LABEL

Variable, control loop object and file names are using the control loop label specified here. The automatically generated label may not be unique or may not reflect the function the control loop serves within the application code.

This label can be changed here to enhance the flexibility and compatibility with user application code.

- **Overwrite Default Controller Name Prefix**
  this option will replace the automatically generated label used for file names and function calls.
- **Overwrite Default variable Name Prefix**
  When selected, this option will also replace the name prefix of all global variables, tying them more closely to the controller block configured by this application
  (mandatory when building multi-loop systems to prevent naming conflicts)

Please observe the C-source file generator output to see how name prefixes change.

Save/Restore Context

Across device families there are different features which can be utilized to reduce interrupt latency by handling context save and restore processes in silicon or software. For example, dsPIC33FJ DSCs only have one set of shadow registers where working registers (WREGs) need to be pushed to and popped from individually. dsPIC33EP, in comparison, offer additional sets of alternative working registers where the recent context can be swapped over to in a single CPU cycle. dsPIC33CH and dsPIC33CK have even further extended features including DSP accumulators and the core configuration register.

Depending on which feature is used in which application code context, one may or may not have to save and restore certain registers and settings. By selecting specific options from this option list, code for saving and restoring will be added/removed from the generated assembly code file.

PLEASE NOTE

All these settings need to be configured individually using dedicated SFRs and configuration bits. These configuration steps are not included by this code generator

- **Shadow Registers**
  covering working registers WREG to WREG 3 only. These registers usually hold function parameters like the start address of the `cNPNZ16b_t` data structure.
- **MAC Working Registers**
  covering working registers WREG4 through WREG10 used for the filter computation.
- **Accumulators**
  if the DSP core is used by other application code modules, it may be that the DSP accumulator contents need to be kept available.
  As the generated controller library will always start with a cleared accumulator and leave the result in the accumulator until the next computation step, it will not be affected by other code modules changing the content of the accumulators. In return, however, these control libraries will override any contents of accumulators which have been stored previously.
- **Core Status Register**
  The core status register may hold information about active calculation status bits any may therefore be affected by computations run by the control library. If any additional application code module may rely on this information, this register needs to be saved and restored.
- **Core Configuration Register**
  The control library computation requires a specific DSP configuration to run the control code most efficiently. If the DSP is used by any other application code module, which may use a different core configuration, this register needs to be saved, changed and restored.

Optional Features

This section can be used to add specific, generic features to the control code, which will be embedded in the assembly code for most efficient execution.

- **Add Core Configuration**
  This option may need to be selected if other application code modules are using the DSP with different configurations.

If it is sufficient to go with one DSP configuration for the entire application, it is recommended to configure the core in a separated initialization routine executed during startup rather than changing the contents of the core configuration in every control loop execution call.

- **Add Enable/Disable Switch**
  When enabled, this option will add a status bit to the `cNPNZ16b_t` data structure. This enable bit will be checked before every execution of the control update library code. When disabled, the control code will be bypassed and no data will be read nor written. When disabled, the histories will be frozen to their latest state, the last control output will remain as a constant, no ADC buffer reads and no output anti-windup (if selected) will be performed.
- **Add Input Normalization**
  dsPIC33FJ, dsPIC33EP, dsPIC33CH and dsPIC33CK ADC converters offer different data format options. Further data format may be different when multiple compensators are coupled in multi-loop systems. To deal with these differences in number scaling, the input data may of may not have to be normalized during the execution of the control loop code.
- **Add ADC Trigger Placement**
  Control loops, which depend on a precise ADC trigger point which needs to be synchronized to varying duty cycles or periods, will need the ADC trigger to be repositioned with the control output. This option should be selected to account for these requirements and execute the repositioning in the most efficient way
- **Create Local Copies**
  To allow other application code modules to track and monitor internal data, which will not be accessible from external code otherwise, local copies of specific data points can be copied to the data structure, through which they get published globally. These data points are
  - Most recent ADC sample
  - Most recent error
  - Most recent control output

Anti-Windup

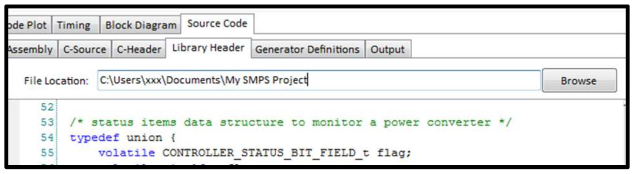
Digital controllers can clamp the control output to a user defined level. When using number clamping, the control history will be clamped at the defined maximum value without saturation effects known from analog control systems. When a digital controller with proper anti-windup clamping is used and the control loop hits output limits (minimum or maximum), it will be clamped there. When the system recovers the control loop will start to respond immediately.

- **Clamp Control Output Maximum**
  The control output will be monitored and clamped to a maximum value when exceeded
- **Generate Upper Saturation Status Flag Bit**
  When the control output gets overwritten by the defined maximum value, a status bit will be set within the status word of the controller to allow external application code modules to detect the saturation condition and respond to it accordingly.
- **Clamp Control Output Minimum**
  The control output will be monitored and clamped to a minimum value when underrun.
- **Generate Lower Saturation Status Flag Bit**
  When the control output gets overwritten by the defined minimum value, a status bit will be set within the status word of the controller to allow external application code modules to detect the saturation condition and respond to it accordingly.

7.0 CODE GENERATION

The code generator does not generate output files by default. This process must be deliberately executed by the user.

Figure 8 shows the file location entry on top of every generated code file (assembly source, C-source, C-header and library header). Use this entry text box to declare the path to the directory in which the generated code file should be located.



```
/* status items data structure to monitor a power converter */
typedef union {
    volatile CONTROLLER_STATUS_BIT_FIELD_t flag;
```

Figure 8: Source Code File Target Directory Declaration

Once all file locations have been declared, code can be generated by clicking on Tools → Export Generated Files → Export Files.

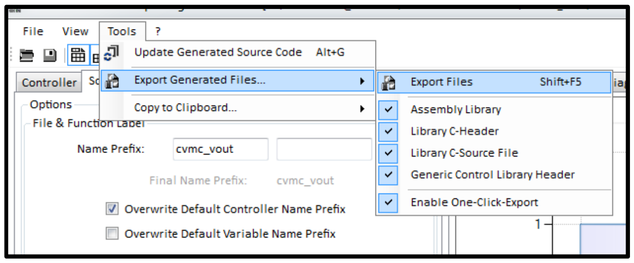


Figure 9: Code Generator Menu

If you'd like to restrict the generation of files to individual items, use the check box list within the same menu to determine which files should be generated. (see Figure 9)

PLEASE NOTE

The file location declarations also support relative file paths. The root path to which relative target paths are referred to is the physical file location of the recently opened DCLD configuration file.

8.0 USING DCLD WITH MPLAB® X

When installing DCLD on a Windows® computer, the setup program will associate the file type *.dcld with the Digital Control Loop Designer application executable DCLD.exe.

When you use this tool to create a control library for your dsPIC® project, the DCLD configuration file can be included in your MPLAB® X project files to ease access while working on application code.

Adding DCLD Configuration Files to MPLAB® X

The recommended procedure to add DCLD configuration files to your project is to place them in the *Important Files* folder, which is automatically created with the new project. This folder is also the home of the *Makefile* used by compiler and linker to build the project.

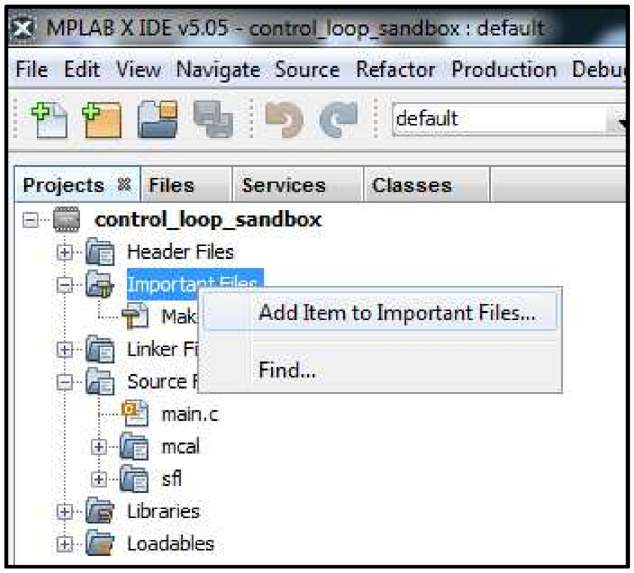
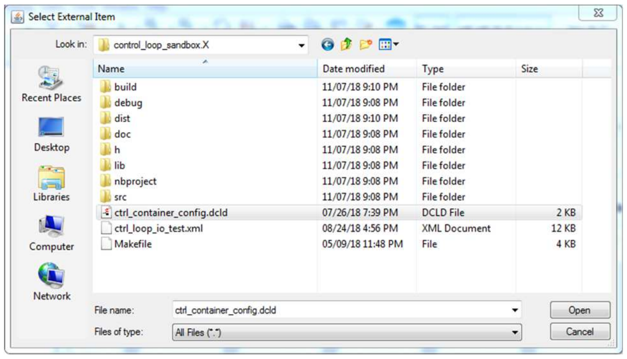

MPLAB X IDE v5.05 - control_loop_sandbox : default
File Edit View Navigate Source Refactor Production Debu
default
Projects Files Services Classes
control_loop_sandbox
Header Files
Important Files
Mak
Linker Fi
Source F
main.c
mcal
sfl
Libraries
Loadables
Add Item to Important Files...
Find...

Figure 10: Adding DCLD Configuration Files to a Project

Right-click on the **Important Files** folder in the project manager and select **Add Item to Important Files**.



Name	Date modified	Type	Size
build	11/07/18 9:10 PM	File folder	
debug	11/07/18 9:08 PM	File folder	
dist	11/07/18 9:10 PM	File folder	
doc	11/07/18 9:08 PM	File folder	
h	11/07/18 9:08 PM	File folder	
lib	11/07/18 9:08 PM	File folder	
nbproject	11/07/18 9:08 PM	File folder	
src	11/07/18 9:08 PM	File folder	
ctrl_container_config.dcld	07/26/18 7:39 PM	DCLD File	2 KB
ctrl_loop_io_test.xml	08/24/18 4:56 PM	XML Document	12 KB
Makefile	05/09/18 11:48 PM	File	4 KB

File name: ctrl_container_config.dcld
Files of type: All Files (*.*)

Figure 11: Select the DCLD Configuration File

From the File Browser dialog, select the DCLD configuration file which should be added to the project and click **Open**.

The selected DCLD configuration file will now be shown in the **Important Files** folder in the **Project Manager**. You can now open and access DCLD from the Project Manager view in MPLAB® X.

If you'd like to add multiple configurations for more than one control loop, repeat the described process until all control loop configurations for this project have been added.

PLEASE NOTE

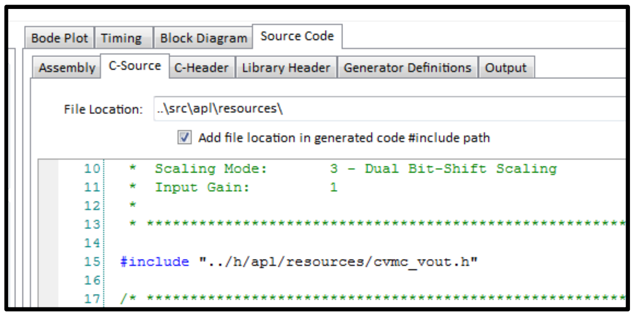
When creating multi-loop systems, each loop must have a unique name to prevent conflicts between files, variables and objects.

File Locations and Include Paths

Generated header files are associated by `#inlude` pre-compiler directives at the beginning of the C-source file and C-header file. In some projects it may be required to include the user-specified file location to this `#inlude` pre-compiler directive.

This is achieved by selecting the **Add file location in generated code #include path** option right below the **File Location** text box to be found on top of each code generator output tab. (see Figure 12)

To prevent confusion with file locations it's recommended to place the DCLD configuration file in the project folder of your MPLAB X project. In this case the compiler is referencing the relative path information correctly and as specified without need for manual edits after files have been generated.



File Location: ..\src\apl\resources\

```
 *  Scaling Mode:       3 - Dual Bit-Shift Scaling
 *  Input Gain:         1
 *
 * ***********************************************************

#include "../h/apl/resources/cvmc_vout.h"

/* ***********************************************************
```

Figure 12: Optimizing file locations

Opening DCLD from MPLAB® X Project Manager

When a control loop needs to be reconfigured during the development process, you can open the DCLD GUI directly from MPLAB® X Project Manager by following these steps (see Figure 13):

Right-click on the DCLD configuration file in the MPLAB® X Project Manager and click on **Open in System**. This will open your saved DCLD configuration in the DCLD GUI where you can modify your configuration.

When your edits to the settings are complete, click on Generate Files to update the control loop project files. MPLAB® X will immediately recognize the externally changed files and refresh them inside the editor window. The project can then be immediately built without further steps. The DCLD GUI can remain open to make further adjustments, if necessary.

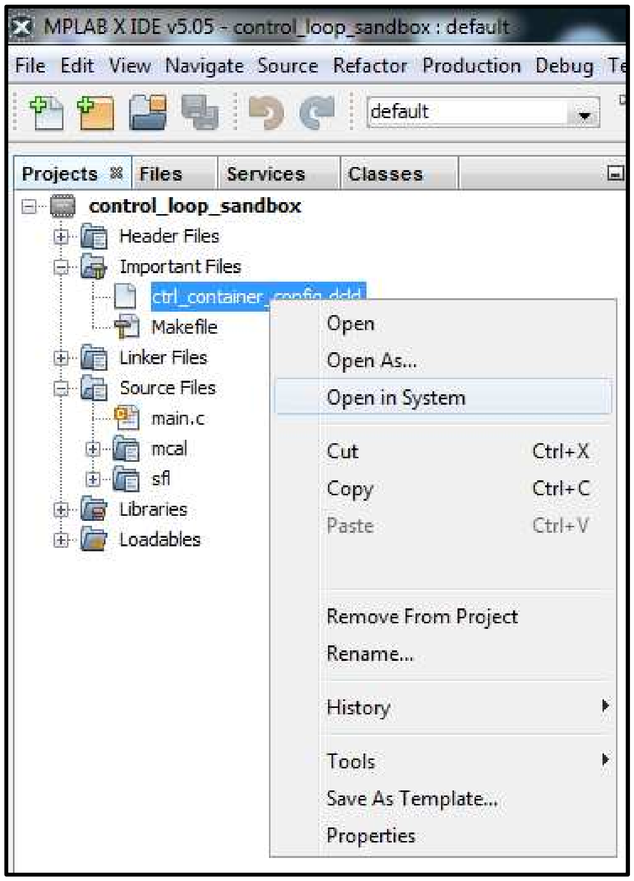


Figure 13: Opening DCLD from the MPLAB® X Project Manager

9.0 COMMON USE CASES

Depending on how complex the control system is you try to create, DCLD offers multiple options to only create content that's essentially required without wasting Flash memory of the target device.

The DCLD code generators is by default creating the following four essential library files described in 5.0 SOURCE CODE VIEW.

- **Assembly Library File**
- **Library C-Source File**
- **Library C-Header Files**
- **Generic Library Header File**

When creating a single-loop control system, all four files are required for a proper implementation of the control loop library code. In multi-loop systems, however, it might be beneficial to limit the number of generated files to save memory space in the target device.

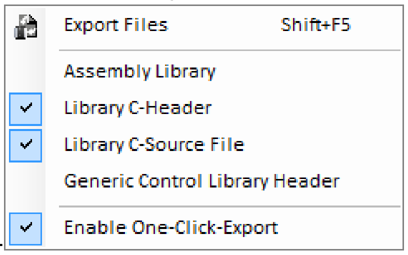
Multiple Controllers using the same Assembly Code

A most common use-case are average current controllers consisting of an outer voltage loop and an inner current loop. Usually both controllers are of Type II (2P2Z in digital) so that only one assembly library but multiple sets of coefficients as well as controller history arrays would be required.

In this case it's recommended to create two independent DCLD configuration files (one for the voltage loop and one for the current loop).

One of both configuration files needs to use the default setting, which will export all four library files. For every additional loop *based on the same compensator type, using the same number scaling method and code feature options*, it's only required to export the C-Source and C-Header file containing the coefficient and controller object declarations.

This can be configured individually by clicking on menu Tools → Export Generated Files. The File Export context menu offers options for every individual file to be exported or not



(see Figure 14

Figure 14)

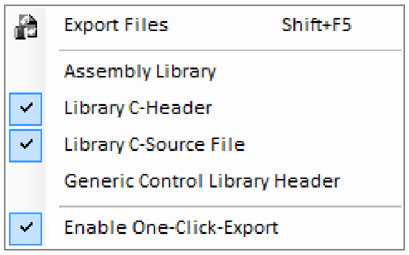


Figure 14: Selecting Files to be Exported

In this example, only the C-Header and C-Source File will be exported.

PLEASE NOTE

Function name labels of the initially generated assembly code will be determined by the DCLD configuration which was used to generate them.

When assembly files are used for multiple controllers, make sure the function calls placed in user code use the correct function name labels and hand over the correct pointer to the individual controller object.

For Example:

```
my_controller_Update(&controller_A);
(…)
my_controller_Update(&controller_B);
(…)
my_controller_Update(&controller_C);
```

Limitations

Using generated code for multiple loops has some limitation which need to be kept in mind to prevent address errors and other undesired conflicts.

Preventing these conflicts is the full responsibility of the user!

- **Main Filter Type Implementation**

  The selected compensator type (e.g. 2P2Z, 3P3Z, 4P4Z, etc.) will be used as template to determine how many filter iterations will be executed by the assembly code library block. To make this most efficient in terms of execution time, no dynamic adjustment to different filter types is made. Thus, the generated assembly library only supports the filter type selected.

- **Scaling Options**

  Different scaling options will equally result in incompatible code when used by different controller objects, which are not using the same number scaling format. Scaling factors, number normalization and resolution differ significantly depending on the selection made. Using controller objects configured for different scaling options therefore cannot use the same assembly library.

- **Code Features**

  Code feature selection will have an equally vital impact on the code integrity but does not necessarily exclude multiple controller object from using the same assembly library. Assuming multiple controller objects are built using the same controller/filter type and number-scaling method, but one controller needs anti-windup clamping while the other controller doesn't. In this case it is still possible to use the same assembly library, which, however, will *always* execute the anti-windup code block. Thus, the second controller needs to hold reasonable thresholds in its respective data structure spaces to not get cut off by clamping-to-zero.

- **Context Save/Restore**
  As long as all controller objects are based on the same compensator/filter type and using the same number scaling method, context save/restore options should be consistent. Nevertheless, if alternate working registers (ALTWREG) on dsPIC33EP, dsPIC33CH or dsPIC33CK are used, it is important to verify that all control library function calls like `xxx_Update(yyy)` are called on the same interrupt priority with a properly associated ALWREG set. These ALTWREG sets can be different but must be accessible and changes to the working registers must not result in conflicts with other tasks.

10.0 LEGAL TERMS FOR DEVELOPMENT BOARDS SOLD AND USED IN EUROPE REGARDING ZVEI REGULATIONS

IMPORTANT NOTICE TO CUSTOMERS

Boards marked as “NON-PUBLIC CONCEPT BOARD” are NOT part of Microchip’s usual development tool portfolio and no support is provided though Microchip’s common support channels. Changes may be applied without any further notice.

This specific hardware has been developed as proof-of-concept board or for training purposes for PROFESSIONAL USERS ONLY. You are not allowed to use this development board in any real application and others than professional lab environments. This board has not been certified or tested for any standards or any requirement such like EMC or safety.

LEGAL NOTICE

TRADEMARKS

CONTACT INFORMATION

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 ▪ Fax: 480-792-7277

Technical Support: http://www.microchip.com/support
Web Address: www.microchip.com

AMERICAS

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455
Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088
Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075
Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643
Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924
Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260
Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445
Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431
Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755
China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104
China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889
China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500
China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189
China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431
China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470
China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205
China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066
China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393
China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760
China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118
China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256
China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130
China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123
India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632
India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513
Japan - Osaka
Tel: 81-66-152-7160
Fax: 81-66-152-9310
Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122
Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302
Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934
Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859
Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068
Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069
Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850
Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955
Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-330-9305
Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102
Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393
Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829
France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79
Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44
Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781
Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340
Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91
UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

NOTES: