

C2000™ Software Frequency Response Analyzer (SFRA) Library and Compensation Designer in SDK Framework

The Software Frequency Response Analyzer (SFRA) is a software library that enables developers to quickly measure the frequency response of their digital power converter. The SFRA library contains software functions that inject a frequency into the control loop and measure the response of the system using the C2000 MCUs' on-chip analog to digital converter (ADC). This process provides the plant frequency response characteristics, the open loop gain frequency response and the closed loop frequency response of the closed loop system. This user's guide explains the software interface of the library and lists the steps needed to integrate the library into a project. The process to run the example project for the SFRA is also documented in this user's guide.

Contents

1	Introduction	2
2	Installing the SFRA Library	5
3	Module Summary	6
4	Compensation Designer	16
5	Case Study	20
6	Running Software Test Bench Example for SFRA	28
7	Using SFRA Without SFRA GUI Integration	33
8	FAQ	39

List of Figures

1	Digitally Controlled Power Converter.....	3
2	Control Design Using SFRA Library and Compensation Designer	4
3	SFRA Principle of Operation	6
4	Compiler Options for a Project Using SFRA Float Library	9
5	Adding Linker Options to the CCS Project to Include SFRA Library in Floating-Point With TMU Project.....	10
6	SFRA GUI Options.....	16
7	Standalone Compensation Designer When Launched From the GUI Folder Inside the SFRA Install Directory	18
8	Compensation Designer When Launched From Solution Adapter Page (both Modeled and SFRA data-based plant information can be used for compensation design)	19
9	Compensation Number and Style Selection.....	20
10	Software Diagram for Plant tf Extraction	21
11	Plant Frequency Response Plot.....	22
12	Comp Designer Being Used to for the Compensator Design for a Buck Power Stage, by Using SFRA Data From Open Loop Run	23
13	Closed Loop SFRA Software Diagram	24
14	SFRA GUI With OL Frequency Response Plot	25
15	Plant Frequency Response Modeled vs Measured on DPSWrkShpKit.....	26
16	Open Loop Frequency Response Modeled vs Measured on DPSWrkShpKit	27
17	Software Test Bench Example	28
18	SFRA Plot for the Plant, After STB Example 1 Run	29
19	SFRA Plot for Open Loop After STB Example 1 Run	30
20	Plant Frequency Response Measured vs Modeled in Software Test Bench Example 1	31

21	Open Loop Frequency Response Measured vs Modeled for Software Test Bench Example	32
22	Zoomed in Open Loop Frequency Response for STB Example 1, Measured vs Modeled Comparison	33
23	Memory Browser View of Stored SFRA Vectors	34
24	Save Memory Pop-Up Window	35
25	Save Memory Options	36
26	Selecting Data From .dat File to Put in the Excel.....	37
27	SFRA Data Copied in Excel File	38

List of Tables

1	Library Sub-Directory Structure	5
2	SFRA Cycle and Program Memory Usage.....	6
3	Floating Point Module Interface Definition.....	8
4	Different Compensation Style Supported by Compensation Designer.....	20

Trademarks

C2000, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

Texas Instruments' software frequency response analyzer (SFRA) library is designed to enable frequency response analysis on digitally controlled power converters using software only and without the need for an external frequency response analyzer. The optimized library can be used in high frequency power conversion applications to identify the plant, the closed loop and the open loop gain characteristics of a closed loop power converter, which can be used to get stability information such as gain margin, phase margin and open loop gain crossover frequency, to evaluate the control loop performance.

NOTE: This documentation pertains to the SFRA Library released in the [C2000WARE-DIGITALPOWER-SDK](#) framework. At the time of authoring of this document, only the single precision floating point format of the library is available inside the SDK framework, the fixed-point version is not yet ported to the new software framework. For the fixed-point version of the library, see [controlSUITE](#) and the [C2000™ Software Frequency Response Analyzer \(SFRA\) Library and Compensation Designer User's Guide](#).

Consider a digitally controlled closed loop power converter, as shown in [Figure 1](#), where:

- H is the transfer function of the plant that needs to be controlled
- G is the digital compensator
- GH is referred to as the open loop transfer function
- CL is referred to as the closed loop transfer function and is $GH/(1+GH)$
- r is the instantaneous set point or the reference of the converter
- Ref is the DC set point reference
- y the analog-to-digital converter (ADC) feedback
- e the instantaneous error
- d the sensor noise and disturbance
- u the PWM duty cycle

The key objectives of the compensator in a closed loop system can be summarized as:

- Ensure that the system is stable (for example, the system tracks the reference asymptotically)

$$e = \lim_{t \rightarrow \infty} e(t) = \lim_{t \rightarrow \infty} \frac{r}{(1 + GH)} \rightarrow 0 \quad (1)$$

- System provides disturbance rejection to ensure robust operation

$$S = \frac{y}{d} = \frac{1}{1+GH} \rightarrow 0 \quad (2)$$

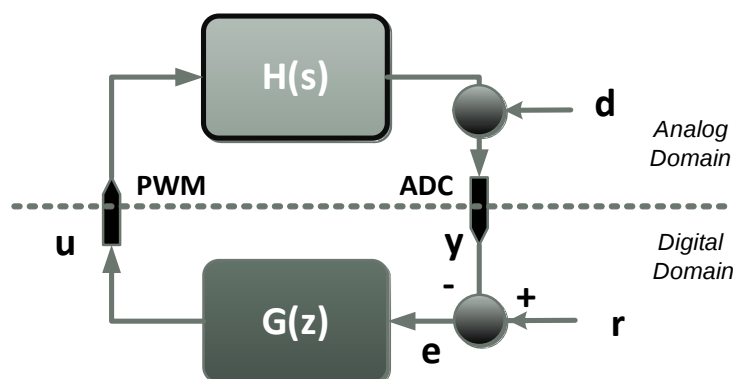


Figure 1. Digitally Controlled Power Converter

Whether or not the system meets the objectives can be determined by knowing the open loop transfer function (GH), as shown in [Equation 1](#) and [Equation 2](#).

A Bode plot of the open loop transfer function GH is frequently used for this purpose and quantities such as gain margin (GM), phase margin (PM) and open loop gain crossover frequency (Folg_cf) are often used to comment on the stability and robustness of a closed loop power converter.

The closed loop transfer function (GH/(1+GH)) provides an idea of the tracking that is how good the system is able to track to the reference commanded.

The SFRA library can enable measurements of the GH, GH/(1+GH) and H frequency response by software. This data can be used to:

- Verify the plant model (H) or extract the plant model (H)
- Design a compensator (G) for the closed loop plant
- Verify the close loop performance of the system by plotting the open loop (GH) or Closed Loop (GH/(1+GH)) Bode diagram

As the frequency response of GH and H carry information of the plant, the data can be used to comment on the health of the power stage by periodically measuring the frequency response.

The SFRA library is based on sinusoidal injection principle, where the assumption is that the injection amplitude causes very small deviation to the normal operating point of the converter. The SFRA library can be integrated into the control code of the power converter, this document details the steps to do so. All computations for the GH, H and CL calculations are done on the MCU and the entire arrays of the GH, H and CL magnitude and phase response are stored on the controller. For visualization purpose, a GUI is also provided with the tool. This GUI can be used to start a frequency response sweep, observe progress of the SFRA sweep and analyze the GH, H and CL plots generated after the FRA sweeps.

Once integrated into the code, the SFRA library can be used to design or fine tune the controller. For this, a typical flow of using SFRA library is:

1. Initiate a SFRA sweep in open loop through the SFRA GUI that populates the data in a comma separate value (CSV) file, stored at the location where SFRA GUI is invoked from. This information can then be used to identify the plant model for the steady state operating point at which the SFRA sweep has been conducted.

NOTE: Optionally, the SFRA GUI can output data to an excel file.

2. This SFRA CSV data can directly be used in the Compensation Designer to design a compensator that meets the system requirements. (Alternatively, if the Excel file option was used from the GUI, the MATLAB script provided with this package can be used to read that data into MATLAB and then curve fit the response to a transfer function. Sisotool can then be used to design the compensator).

3. New Compensator values can be copied from the compensation designer GUI or MATLAB into the Code Composer Studio™ project.
4. Compile and load the code with new coefficients into the microcontroller controlling the power stage. SFRA algorithm (Step 1) can be re-run to verify the closed loop system performance by measuring the open loop gain GH (also referred to as loop gain in many literatures).

In summary, TI's software frequency response analyzer along with the Compensation Designer provides a methodology to tune power supplies in a systematic way and enables quick and easy frequency response analysis for power converters without the need of external connections and equipment, as shown in Figure 2. Since no external connections are used, the SFRA can be run repeatedly to periodically assess the health of the power converter and get diagnostic information.

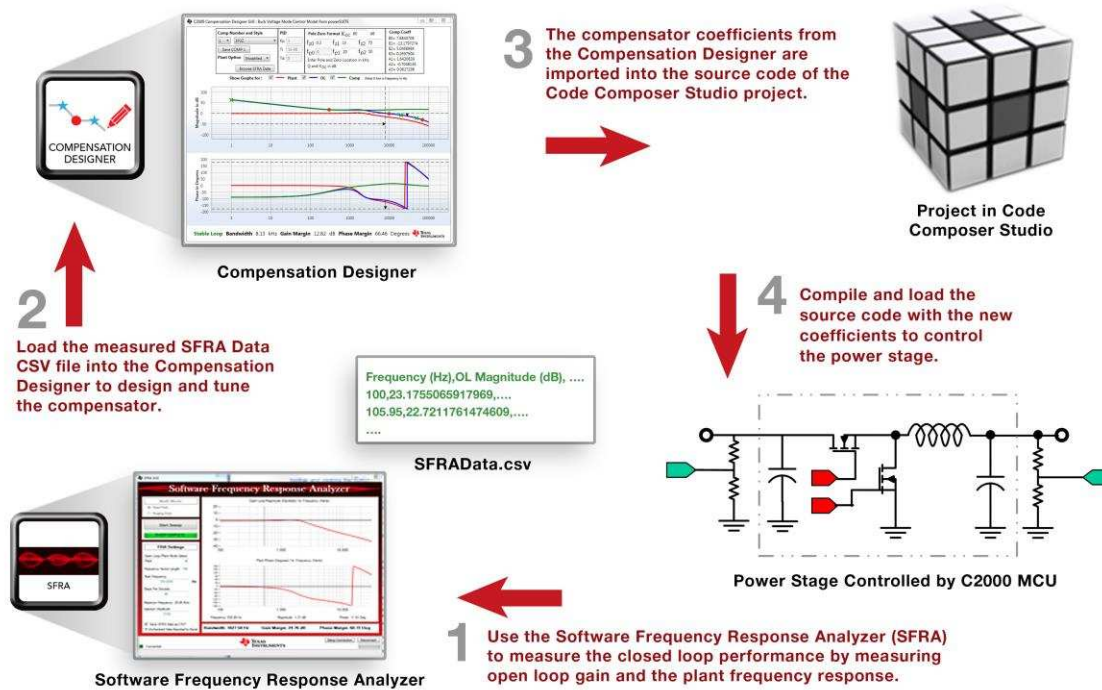


Figure 2. Control Design Using SFRA Library and Compensation Designer

2 Installing the SFRA Library

2.1 SFRA Library Package Contents

The TI SFRA library consists of the following components:

- Header files and software library for single precision floating-point computation
- [C2000 Software Frequency Response Analyzer \(SFRA\) Library User's Guide](#)
- SFRA Graphical User Interface to display the frequency response analysis results
- Scripts to import SFRA data into MATLAB
- Example software test bench project to quickly evaluate SFRA library

2.2 How to Install the SFRA Library

The SFRA library is distributed through the C2000WARE-DIGITALPOWER-SDK installer, which can be downloaded at <http://www.ti.com/tool/C2000WARE-DIGITALPOWER-SDK> , alternatively it may be available as part of other SDKs for C2000 MCUs that use SFRA library. By default, the installation places the library components in the following directory structure:

C:\ti\c2000\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra

The following sub-directory structure is shown in [Table 1](#) .

Table 1. Library Sub-Directory Structure

<base>\docs	Documentation
<base>\examples	Contains example projects for quickly evaluating SFRA (Software Test Bench). Currently one example included is example1_stb which is a software check for the SFRA library.
<base>\gui	Host side graphical user interface (SFRA_GUI.exe) that is used for displaying the frequency response and the GUI tool Compensation Designer (ComDesigner.exe) that can be used to design compensator based with the SFRA measured plant information. The SFRADData.csv file is also stored here.
<base>\include	Contains the header file that is needed to be included in the system project to use this library.
<base>\lib	The library file for float32 and float32+TMU processor options.
<base>\scripts	Scripts to import SFRA data for compensation design in MATLAB.
<base>	Software manifest with version number of the library.

3 Module Summary

3.1 SFRA Library Function Summary

The SFRA library consists of modules that enable the user to run SFRA on power converters. Table 2 lists the modules existing in the SFRA library and a summary of cycle counts on each instruction set variant of the SFRA library.

Table 2. SFRA Cycle and Program Memory Usage

	SFRA_INJECT	SFRA_COLLECT	SFRA_Background	Program Size	Data Size
Float	41	71	Typical ~ 50 Max 2350	1109 x 16-bit words	22 x 16-bit words (internal) + 31 x 16-bit words for SFRA object
Float TMU	37	67	Typical ~50 Max 407	1098 x 16-bit words	22 x 16-bit words (internal) + 31 x 16-bit words for SFRA object

The numbers reported above are for the data memory usage internal to the library and the memory used by the SFRA object instance. The memory used to store the frequency response data is not included as this is user specific, and depend on the number of frequency points the user wants to run the SFRA over. The number for inject and collect routine are reported for when active sweep is in progress and are the worst case. If no sweep in progress the routines take significantly less cycles.

NOTE:

- The SFRA library is non-re-entrant (only one instance of the SFRA library is supported in a particular project).
- The cycles reported, unless otherwise mentioned, are for maximum/worst case path in the operation. The cycles reported include the branch to the routine and return to the next instruction after the branch. Cycles taken to push parameters on the stack before the branch are not accounted for in this number as it can vary depending on adjoining code in the program.
- For the floating-point library, the memory for RTS library calls that execute trigonometric operations is not included.
- TMU provides additional acceleration that improves performance in the background loops.

3.2 Principle of Operation

The software frequency response analyzer is based on the principle of small signal sinusoidal injection. A small signal is injected on the reference of the controller, as shown in Figure 3, and the frequency response on feedback and controller outputs are calculated. This provides the plant frequency response characteristics and the open loop frequency response of the closed loop system.

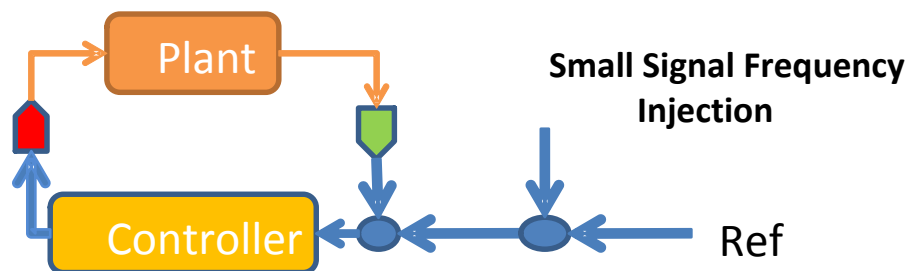


Figure 3. SFRA Principle of Operation

3.3 Per Unit Format

The SFRA library supports single precision floating-point-based math. Per unit, values are typically used for variables in control application and are used by the library. Per unit, value is found by dividing the current reading by the maximum value that can be read. For example, if the voltage sense max is 20 V and the instantaneous reading is 5 V, the per unit value is $5/20 = 0.25$.

3.4 Floating Point (Single Precision)

The following description applies to single precision floating point version of the SFRA library. This library can be run on any C2000 MCU that has single precision floating point unit. The following description applies to TMU version of the library as well, the only difference being the TMU provides cycle acceleration and more precision in the SFRA run.

3.4.1 Object Definition

The SFRA library defines the floating-point-based SFRA structure as discussed in the following sections:

```
typedef struct{
    float32_t *h_magVect;      //!< Plant Mag SFRA Vector
    float32_t *h_phaseVect;    //!< Plant Phase SFRA Vector
    float32_t *gh_magVect;     //!< Open Loop Mag SFRA Vector
    float32_t *gh_phaseVect;   //!< Open Loop Phase SFRA Vector
    float32_t *cl_magVect;     //!< Closed Loop Mag SFRA Vector
    float32_t *cl_phaseVect;   //!< Closed Loop Phase SFRA Vector
    float32_t *freqVect;       //!< Frequency Vector
    float32_t amplitude;       //!< Injection Amplitude
    float32_t isrFreq;         //!< SFRA ISR frequency
    float32_t freqStart;       //!< Start frequency of SFRA sweep
    float32_t freqStep;        //!< Log space between frequency points (optional)
    int16_t start;             //!< Command to start SFRA
    int16_t state;             //!< State of SFRA
    int16_t status;            //!< Status of SFRA
    int16_t vecLength;         //!< No. of Points in the SFRA
    int16_t freqIndex;         //!< Index of the frequency vector
    int16_t storeH;            //!< Flag to indicate if H vector is stored
    int16_t storeGH;           //!< Flag to indicate if GH vector is stored
    int16_t storeCL;           //!< Flag to indicate if CL vector is stored
    int16_t speed;             //!< variable to change the speed of the sweep
}SFRA_F32;
```

3.4.2 Module Interface Definition

Table 3. Floating Point Module Interface Definition

Module Element Name	Type	Description	Acceptable Range
h_magVect,gh_magVect, cl_magVect	Input	Pointer to the array that stores the magnitude of H, GH and CL measurements by SFRA . Pass NULL if you do not want SFRA to save that vector.	Pointer to 32 bit location, the location stores the value of the magnitude vectors in single precision (32-bit) floating point
h_phaseVect,gh_phaseVect, cl_phaseVect	Input	Pointer to the array that stores the phase of H, GH and CL measurements by SFRA . Pass NULL if you do not want SFRA to save that vector.	Pointer to 32 bit location, the location stores the value of the phase vectors in single precision (32-bit) floating point
freqVect	Input	Pointer to array of frequency values at which SFRA is performed.	Pointer to 32 bit location, the location stores the value of the frequency vectors in single precision (32-bit) floating point
amplitude	Input	Amplitude of small signal injection in pu.	Single precision (32-bit) floating point(-1,1)
isrFreq	Input	Frequency at which SFRA routine is called.	Single precision (32-bit) floating point
freqStart	Input	Frequency of the first frequency sweep data point.	Single precision (32-bit) floating point
freqStep	Input	$10^{(1/(\text{no of steps per decade}))}$.	Single precision (32-bit) floating point
start	Input	Command to start SFRA.	int16_t
state	Output	SFRA state. Non zero when SFRA injection is in progress, '0' if SFRA injection is not active/ in progress.	int16_t
status	Output	SFRA status. '1' is SFRA injection is in progress, '0' if SFRA injection is not active/ in progress.	int16_t
vecLength	Input	No of points for which SFRA is performed.	int16_t
freqIndex	Output	Frequency index number of freqVect at which SFRA is being performed.	int16_t (0-vecLength)
storeH	Output	Reflects the SFRA configuration, If one, H vector is stored If zero, H vector is not stored this happens when a NULL vector is passed for H mag or phase vector during SFRA configuration.	int16_t (0 or 1)
storeGH	Output	Reflects the SFRA configuration, If one, GH vector is stored If zero, GH vector is not stored this happens when a NULL vector is passed for GH mag or phase vector during SFRA configuration.	int16_t (0 or 1)
storeCL	Output	Reflects the SFRA configuration, If one, CL vector is stored If zero, CL vector is not stored this happens when a NULL vector is passed for CL mag or phase vector during SFRA configuration.	int16_t (0 or 1)
speed	Input	Used to change the speed of the sweep, need to be greater than 1. With 1 the STB example template sweep takes ~58 seconds. Actual speed in the system will depend on the frequency point being measured and the ISR rate used for calling the SFRA module. Higher the speed number the slower the sweep.	int16_t (Greater than 1)

3.4.3 Adding SFRA Library to the Project

1. Include the SFRA library in source code of your project where you want to run SFRA.

```
#include "sfra_f32.h"
```

Add the SFRA library path in the include paths under Project Properties → Build → C2000 Compiler → Include Options (see [Figure 4](#)).

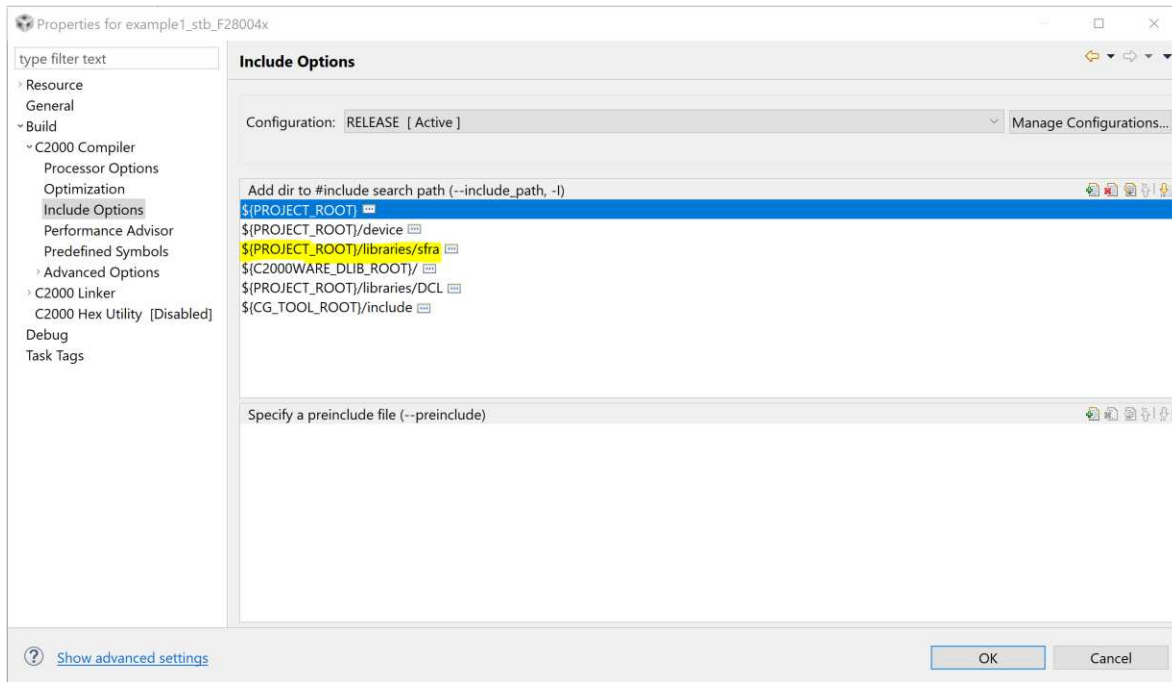


Figure 4. Compiler Options for a Project Using SFRA Float Library

NOTE: The exact locations may vary depending on where SFRA is installed and which other libraries the project is using.

Link SFRA library: (sfra_f32_coff/eabi.lib or sfra_f32_tmu_coff/eabi.lib) in the project depending on the device you have selected, the lib is located at:

```
<sfra_install_path>\lib
```

Figure 5 shows the changes to the linker options that are required to include the SFRA library.

NOTE: FastRTS library must be included, for steps on including fastRTS library. For more information, see the FastRTS library documentation found at:

<sdk_install_path>\C2000Ware\libraries\math\FPUfastRTS\c28

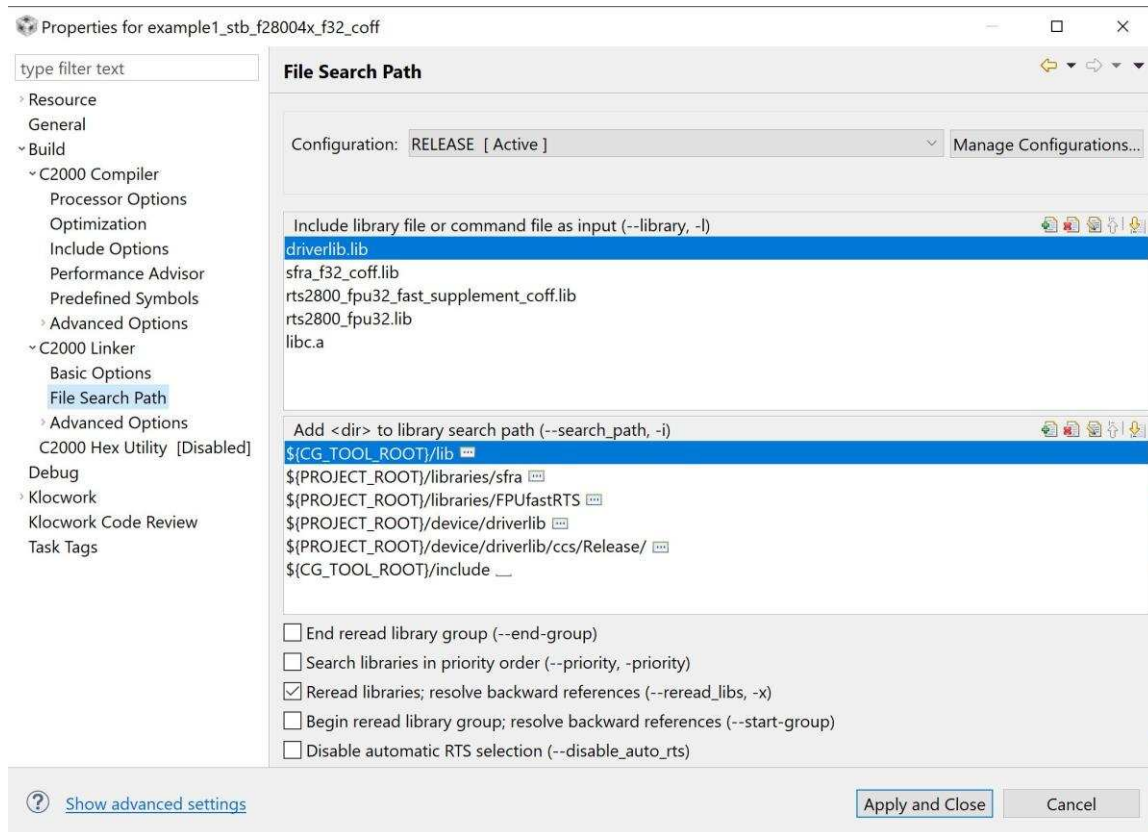


Figure 5. Adding Linker Options to the CCS Project to Include SFRA Library in Floating-Point With TMU Project

NOTE: The exact locations may vary depending on where controlSUITE is installed and which other libraries the project is using.

2. Add #defines for SFRA, such as the SFRA_ISR_FREQ, is the frequency of the ISR in which the SFRA routine will be called from. SFRA_FREQ_START is the start frequency of the SFRA analysis. SFRA_FREQ_LENGTH is the number of points in SFRA analysis. SFRA_FREQ_STEP_MULTIPLY specifies the the step size between individual points of SFRA sweep. The end frequency of the SFRA analysis is dependent on the start frequency, the length, and the step size.

```
#define SFRA_ISR_FREQ          CONTROL_ISR_FREQUENCY
#define SFRA_FREQ_START 2
//
// SFRA step Multiply = 10^(1/No of steps per decade(40))
//
#define SFRA_FREQ_STEP_MULTIPLY (float32_t)1.105
#define SFRA_AMPLITUDE (float32_t)0.005
#define SFRA_FREQ_LENGTH 100
...
```

3. Define SFRA_F32 object and arrays for the SFRA data storage. For the non TMU version of the library, add an extern to the the FPUsinTable to avoid compiler warnings.

```
SFRA_F32 sfra1;

float32_t plantMagVect[SFRA_FREQ_LENGTH];
float32_t plantPhaseVect[SFRA_FREQ_LENGTH];
float32_t olMagVect[SFRA_FREQ_LENGTH];
float32_t olPhaseVect[SFRA_FREQ_LENGTH];
float32_t clMagVect[SFRA_FREQ_LENGTH];
float32_t clPhaseVect[SFRA_FREQ_LENGTH];
float32_t freqVect[SFRA_FREQ_LENGTH];

//
//extern to access tables in ROM
//
extern long FPUsinTable[];
```

4. Steup of the SFRA module is carried out by the following routine. This routine is run before SFRA is called in the ISR. Pass a NULL for the arrays/vectors that are not stored.

```
//
//Resets the internal data of sfra module to zero
//
SFRA_F32_reset(&sfra1);

//
//Configures the SFRA module
//
SFRA_F32_config(&sfra1,
                SFRA_ISR_FREQ,
                SFRA_AMPLITUDE,
                SFRA_FREQ_LENGTH,
                SFRA_FREQ_START,
                SFRA_FREQ_STEP_MULTIPLY,
                plantMagVect,
                plantPhaseVect,
                olMagVect,
                olPhaseVect,
                clMagVect,
                clPhaseVect,
                freqVect);

//
//Resets the response arrays to all zeroes
//
SFRA_F32_resetFreqRespArray(&sfra1);

//
// Initializes the frequency response array ,
// The first element is SFRA_FREQ_START
// The subsequent elements are freqVect[n-1]*SFRA_FREQ_STEP_MULTIPLY
// This enables placing a fixed number of frequency points
// between a decade of frequency.
// The below routine can be substituted by a routine that sets
// the frequency points arbitrarily as needed.
//
SFRA_F32_initFreqArrayWithLogSteps(&sfra1,
SFRA_FREQ_START,
SFRA_FREQ_STEP_MULTIPLY);
```

NOTE: Even though in the above sequence the array is initialized with log step frequencies, the SFRA is in no way dependent on that. The frequency array initialization sequence can be replaced with any other as desired by the application.

5. Using the module, add the code to the ISR to call the SFRA as shown below around the compensator input and outputs:

- a. Using SFRA in a closed loop system to get loop gain(GH) and plant(H) frequency response:

```
interrupt void PWM_ISR(void)
{
    ...

    //
    // Read ADC and computer Fbk Value
    //
    Fdbk = (float32)Vout1R/(4096.0);

    //
    // Add FRA injection into the reference of the controller
    //
    Ref = SFRA_F32_inject(Vout1SetSlewed);

    //
    // Call the controller
    //
    Out=DCL_PI_C1(&dc1_obj,Ref,Fdbk);

    //
    // Update PWM value
    //
    EPwm1Regs.CMPA.half.CMPA = ((long)(BUCK_PWM_PERIOD))*Out;

    SFRA_F32_collect(&Out,&Fdbk);
    ...
}
```

- b. Using SFRA in open loop to get plant(H) frequency response:

```
interrupt void PWM_ISR(void)
{
    ...
    //
    // Read ADC and computer Fbk Value
    //
    Vout1_Read = (float32)Vout1R/(4096.0);

    //
    // Add SFRA injection into the duty cycle for the open loop converter
    //
    Duty_pu=SFRA_F32_inject(Duty_pu_DC);

    //
    // Update PWM value
    //
    EPwm1Regs.CMPA.half.CMPA=((long)(BUCK_PWM_PERIOD))* Duty_pu;

    SFRA_F32_collect(&Duty_pu,&Vout1_Read);
    ...
}
```

6. Background task: Call the background function in a slow task.

```
SFRA_F32_runBackgroundTask(&sfra1);
```

7. Linker command file changes (*.CMD): When running from FLASH for best performance, the inject and collect operations need to reside in RAM. The following are the changes and additions to CMD file for the SFRA library inclusion and best performance.

```
...
ramfuncs      : LOAD = FLASHD,
                RUN  = RAML0L1,
                LOAD_START(_RamfuncsLoadStart),
                LOAD_END(_RamfuncsLoadEnd),
                RUN_START(_RamfuncsRunStart),
                PAGE = 0

        {
            --library=sfra_f32_coff.lib<SFRA_F32_inject.obj>
            --library=sfra_f32_coff.lib<SFRA_F32_collect.obj>
        }

...

SFRA_F32_Data      : > dataRAM1, PAGE = 1
```

8. Also make sure the ROM location for the math tables is defined correctly. These locations are defined in the device command file, which is available through C2000Ware for devices that do not have TMU. Note that the following location is correct for F28069.

```
/* FPU Math Tables in Boot ROM */
FPUTABLES    : origin = 0x3FD860, length = 0x0006A0
...
/* Allocate FPU math areas: */
FPUmathTables : > FPUTABLES, PAGE = 0, TYPE = NOLOAD
```

9. For devices with TMU math tables that are not in ROM and must be assigned to a memory location to avoid linker warnings:

```
FPUmathTables : > FLASH1, PAGE =1
```

3.4.4 Adding Support for SFRA GUI

A few modifications to the Code Composer Studio [™] project are required to add support for the SFRA GUI.

1. Add "sfra_gui_scicomms_driverlib.c" to the project. This can be done in the project window, by right-click on your project and selecting "Add files to project", then browse to the file under <sfra-install-path>/gui/source.
2. Add "sfra_gui_scicomms_driverlib.h" to the include path for the project. The file is located at <sfra-install-path>/gui/include.
3. Call the SFRA_GUI_config function. The GUI can toggle an LED when connected, if this feature is used additional configuration parameters must be passed to the GUI config routine, otherwise pass zero in those parameters. Typical defines used on F28004x control card are shown below. Change them if your hardware requires any changes. Currently the SFRA GUI supports plotting of two vectors, passing a '1' for the plot options defaults the plant and open loop vectors. '2' plots the closed loop and the plant transfer function.

```
#define SFRA_GUI_SCI_BASE SCIA_BASE
#define SFRA_GUI_VBUS_CLK 50000000
#define SFRA_GUI_SCI_BAUDRATE 57600
#define SFRA_GUI_SCIRX_GPIO 28
#define SFRA_GUI_SCITX_GPIO 29
#define SFRA_GUI_SCIRX_GPIO_PIN_CONFIG GPIO_28_SCIRXDA
#define SFRA_GUI_SCITX_GPIO_PIN_CONFIG GPIO_29_SCITXDA

//
// if the following #define is set to 1 SFRA GUI indicates status on an LED
// otherwise LED code is ignored
//
#define SFRA_GUI_LED_INDICATOR 1
#define SFRA_GUI_LED_GPIO 31
#define SFRA_GUI_LED_GPIO_PIN_CONFIG GPIO_31_GPIO31
.....

//
// configures the SCI channel for communication with SFRA host GUI
// to change SCI channel change #define in the sfra_gui_scicomms_driverlib.c
// the GUI also changes a LED status, this can also be changed with #define
// in the file pointed to above
//
SFRA_GUI_config(SFRA_GUI_SCI_BASE,
                SCI_VBUS_CLK,
                SFRA_GUI_SCI_BAUDRATE,
                SFRA_GUI_SCIRX_GPIO,
                SFRA_GUI_SCIRX_GPIO_PIN_CONFIG,
                SFRA_GUI_SCITX_GPIO,
                SFRA_GUI_SCITX_GPIO_PIN_CONFIG,
                SFRA_GUI_LED_INDICATOR,
                SFRA_GUI_LED_GPIO, \
                SFRA_GUI_LED_GPIO_PIN_CONFIG,
                amp;sfra1,
                1);
```

4. SFRA_GUI_SCI_BASE : this is the base for the SCI that is used for SFRA communication.
5. SFRA_GUI_SCIRX/TX_GPIO: GPIO pin that are used for the SCI peripheral. These are defined currently for example on controlCARDS where the FTDI chip is connected for the serial link and it uses GPIO-28 and GPIO-29.
6. The GUI is setup such that it will blink an LED when GUI has established connection with the host. This LED can be specified in SFRA_GUI_LED_GPIO. Also make sure to change SFRA_GUI_LED_GPIO_PIN_CONFIG appropriately.
7. Also make sure that the SCI clock is enabled. Clock configuration is typically handled in setupDevice()->Device_init() functions. For powerSUITE solutions, these are in the board.c file.
8. In a slower task with a service rate of ~ 10 ms, call the following routine to keep the SFRA GUI connection alive. This can typically be kept close to the SFRA Background function.

```
SFRA_GUI_runSerialHostComms(&sfra1);
```
9. The project will now connect to the SFRA GUI. For more information on how to run the SFRA GUI, see [Section 3.6](#).

3.5 Script for Importing Frequency Response and Designing Compensation

The SFRA GUI, upon completing a frequency sweep, can populate a comma separated value (CSV) file or an excel sheet with the frequency response data. When CSV format is selected (which is the default), it can be saved as an excel file. The excel file data can be used to fit a pole zero format to extract the transfer function. This transfer function can then be used to design the compensation for the closed loop plant. The scripts for these operations that run on MATLAB are provided under:

SFRA/<version>/scripts

3.6 SFRA GUI Options and How to Run

The SFRA GUI can be used to start the frequency sweep and display the measured frequency response. [Figure 6](#) illustrates the different options that are presented.

- **MATH Mode Selection Radio Button:** This box must be selected appropriately according to the version of SFRA library being used on the MCU. For F28377 and F28004x as they are floating point devices, floating point check box must be checked.
- **Setup Connection Button:** This button is used to select the COM port and set the baud rate. Click on this button and a pop up window requests a Baud Rate selection, which is typically set to 57600. If the device is connected, click “Refresh Comports” and select the serial comport that the target is using.
 - If the comport that the target is connected to is known, please select it.
 - If not, and to find a valid comport, go to:
 - Control Panel → System → Device Manager → Ports (COM and LPT)
 - If using a serial port directly connected to a PC, look for a comport that shows up as “Communications Port” and select this it in the Setup Connection window. If using a USB to Serial adapter, look for the comport that shows “USB Serial Port”, then select this it in the Setup Connection window.

Uncheck “Boot on Connect” to use the GUI in conjunction with a project running from RAM in Code Composer Studio or when using a target that has been flashed with a working SFRA project.

- **Connect and Disconnect Button:** Once the Math mode is selected appropriately and COM port for the communication is selected, click this button to Connect to the MCU. This is the same button that is used for disconnection from the MCU.
- **Connection Status Indicator:** This indicates whether the connection was successfully established or if there were any problems.
- **Frequency Vector Length Label:** Once connected, this label shows the length of the FRA array that is programmed on the MCU.
- **Start Frequency Text Box:** This box shows the start frequency of the SFRA sweep. This can be changed by the user by entering a value in the text box and clicking enter. (Note the value on the controller side is only updated once the user starts a SFRA sweep).
- **Steps Per Decade Text Box:** Specifies the number of frequency points that FRA is performed per decade.
- **Maximum SFRA Frequency Label:** This is the maximum frequency until the SFRA will be performed. This is a function of the start frequency, the steps per decade and the frequency length.
- **Start Sweep Button:** Click this button to begin a frequency sweep on the controller.
- **SFRA Status Bar:** Indicates the progress of SFRA sweep.
- **Drop Down Menu to Select Open Loop or Plant Frequency:** Response to be displayed on the panel to the right.
- **Closed Loop Performance Parameters Panel:** If SFRA is performed on the closed loop, this panel displays the bandwidth, gain margin and the phase margin of the closed loop system.

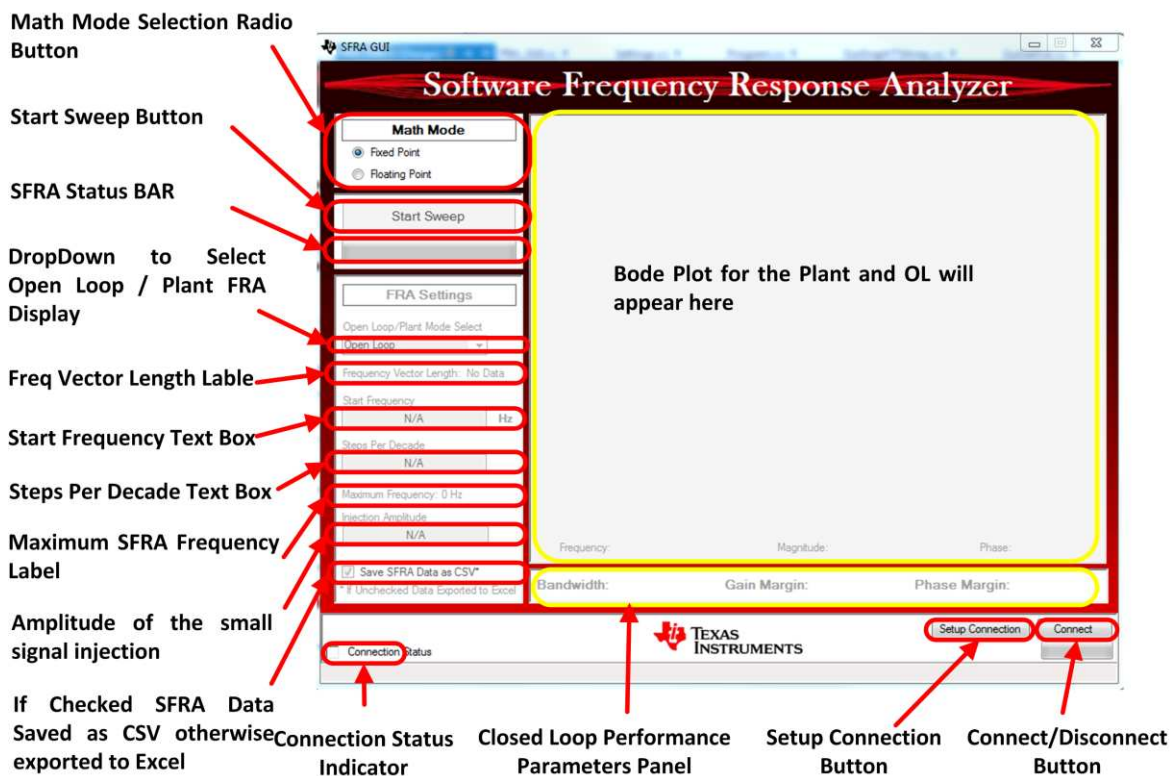


Figure 6. SFRA GUI Options

4 Compensation Designer

TI's Compensation Designer provides an easy GUI interface, to calculate the coefficients that are needed to be programmed onto the micro-controller to implement a compensator as part of a closed loop system. The GUI lets the user select different compensator styles like PID, Two Pole Two Zero, and so forth, and allows the user to save up to five compensator values.

The GUI displays the Plant, the Open Loop and the Compensator Frequency Response. The Pole and Zeroes of the compensator, are marked on the Open Loop Curve on the GUI using a cross and circle, respectively. Critical values such as open loop gain crossover frequency (Fol_{g_cf}), Phase Margin and Gain Margin are displayed on the GUI at the bottom, and a label showing "Stable Loop" and "UnStable Loop" is displayed if the system has healthy Bandwidth, Phase Margin and Gain Margin.

For Plant frequency response, depending upon where the Compensation Designer is invoked from, the user can select to use the mathematical model of the power stage or the experimental data gathered using SFRA, to design the compensator.

4.1 Launching Compensation Designer

The compensation design GUI can be launched using two methods.

4.1.1 Standalone From SFRA GUI Folder

Compensation designer can be launched standalone by double clicking on the “CompDesigner.exe” located inside <sfra_install_directory>/gui folder. The GUI can be used to design a compensator for desired control performance using plant information that has been gathered using the SFRA GUI in CSV format. The SFRA_GUI.exe puts the latest frequency sweep data inside

“<sfra_install_directory>/gui/SFRADData.csv” and also keeps all the frequency sweep data with time stamp inside the same folder. The compensation design GUI on launch defaults to this SFRADData.csv file. The user can select, during a session, to use another frequency sweep data file by clicking on “Browse SFRA csv Data”.

The GUI calculates the coefficients that can be programmed in the routines of Digital Controller Library (DCL) such as DCL_PI, DCL_DF22, and so forth. The user needs to enter the control frequency on the GUI, which is the rate at which the control loop is executed in kHz. This GUI can be used with solutions for which solution adapter GUI does not exist and SFRA has been gathered.

The GUI lets the user save different compensators, these compensators are saved in the Comp.xml file located inside the same folder from which the GUI is launched. To save a particular compensator setting, the user must hit the “Save Comp No” button, otherwise the compensator will not be saved.

NOTE: When using SFRA data the GUI displays only the frequency response data points for which the SFRA sweep has been performed. If the compensation zero or pole is not in that range it is not marked on the GUI.

Figure 7 shows the Compensation Designer GUI launched from the SFRA folder. Hovering over the drop down box for compensation style select, displays the exact equation used to generate the coefficients.

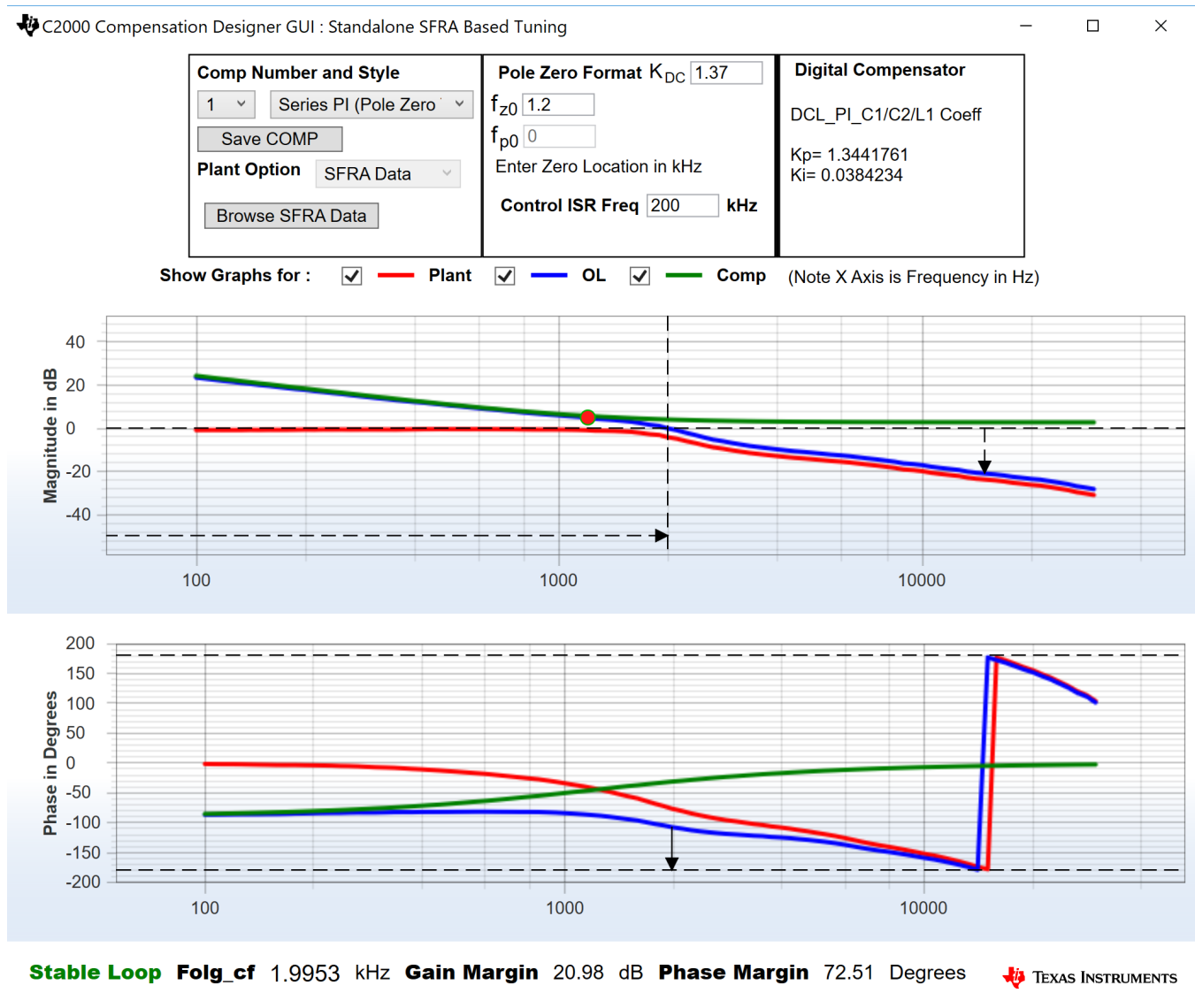


Figure 7. Standalone Compensation Designer When Launched From the GUI Folder Inside the SFRA Install Directory

4.1.2 From Solution Adapter Page

C2000 kits may support a solution adapter page, which lets the user launch the compensation designer directly from the solution adapter page. When launched from this page, the compensation designer models the power stage based on the parameters specified on the solution adapter page for inductance, capacitor, and so forth.

The control frequency that is used to calculate the compensator coefficients is also specified on the solution adapter page. The modeled plant can serve as a good starting point in the compensation design process before SFRA is run.

The user can also select SFRA data for the Plant Frequency Response and use the measured data to design the compensator just like when launched in standalone fashion.

NOTE: When Modelled Plant Option is selected the browsed SFRA Data csv file is not used. Before selecting SFRA Data, the user must browse to the appropriate SFRA csv file.

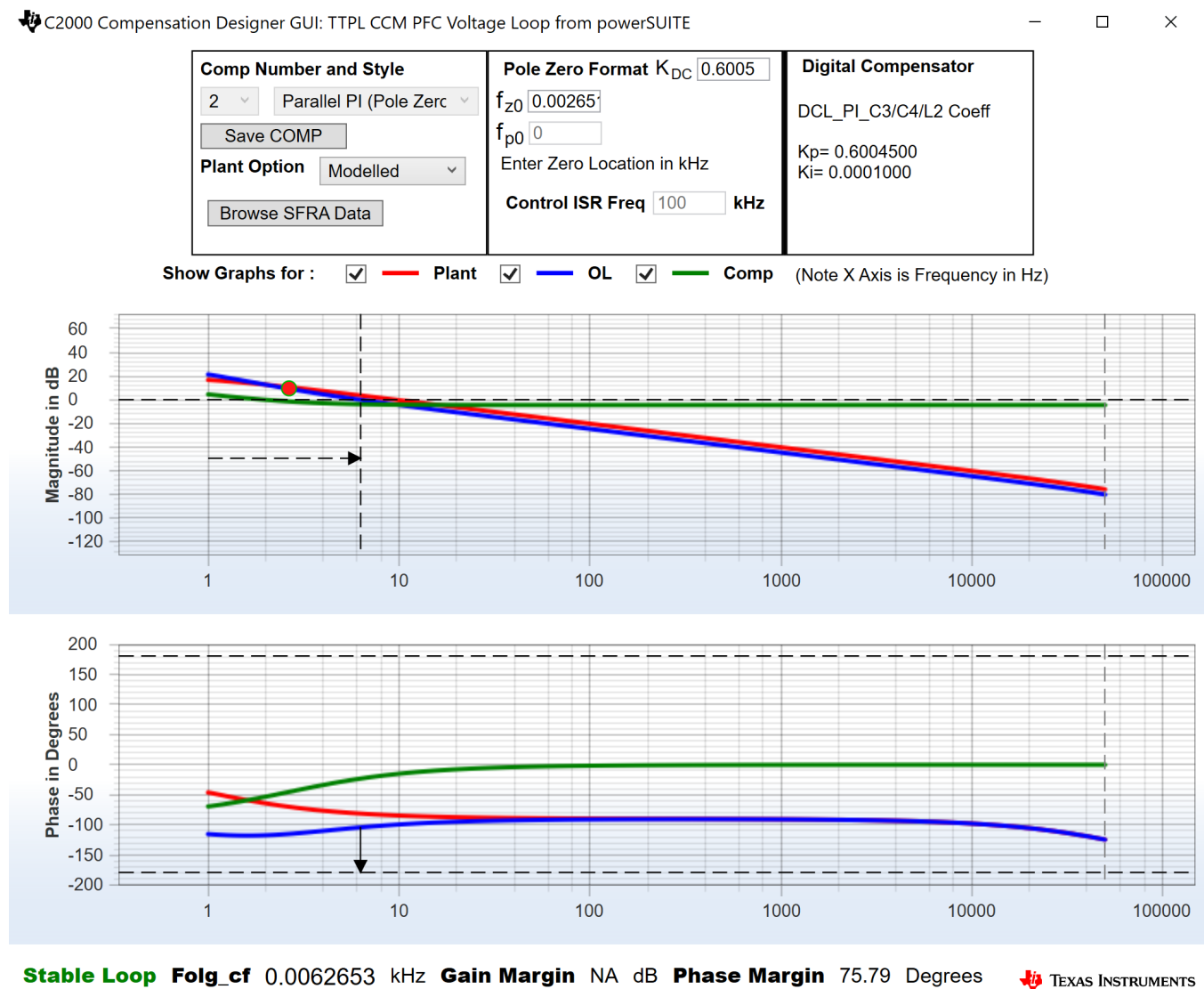


Figure 8. Compensation Designer When Launched From Solution Adapter Page
(both Modeled and SFRA data-based plant information can be used for compensation design)

4.2 Compensation Style and Number

Different styles of compensation are supported by the Compensation Designer and are listed in [Table 4](#).

Table 4. Different Compensation Style Supported by Compensation Designer

Style	Description	Implementation
Proportional Gain	Proportional gain controller	Simple multiply in control software
Series PI	Series proportional integrgral controller tuned with pole zero format	DCL_PI_C1/C2 or similar
Parallel PI	Parallel proportional integrgral controller tuned with pole zero format	DCL_PI_C3/C4 or similar
Two Pole Two Zero	Two pole two zero compensator	DCL_DF22 or similar
One Pole One Zero Lead Lag Compensator	Lead Lag compensator implemented using one pole one zero	DCL_DF11 or DCL_DF22 or similar

The standalone GUI when ran from the SFRA install folder enables designing up to ten compensators each of which can be tuned using any of the different compensation styles. The Compensation Style and Number can be selected using the drop down box.

Any changes to the compensation value are not saved until the button “Save COMP<number> is selected (see [Figure 9](#)). This prevents any inadvertent change to the compensation values.

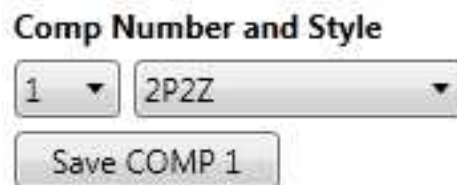


Figure 9. Compensation Number and Style Selection

Depending on which compensation style is selected, different panels on the GUI will become active. The coefficients for the DCL module based on the compensation style and the values entered in the tuning window are displayed on the GUI and can be copied into the user code.

NOTE: Assuming F_s is the control loop frequency or the rate at which the control loop is executed and $T=1/F_s$.

5 Case Study

To understand the use of SFRA, a case study of SFRA library as applied to buck converter is carried out in this section.

A typical workflow to use SFRA is as below:

1. Run the converter in open loop, perform a SFRA sweep.
2. Use this SFRA data to design a compensator by importing it into compensation designer.
3. Use the designed coefficients to run the closed loop response. Compare this with the designed and fine tune for performance.

5.1 Plant TF Extraction

If the plant model is not known the plant model can be identified by running the SFRA in open loop. For this, first a DC operating point is chosen. For example, in a buck converter a fixed duty cycle is provided to the converter. The SFRA_INJECT function is used to add small signal injection to the duty value. Which is then provided to the PWM driver. The SFRA_COLLECT function is used to analyze data from the excitation and calculate the plant transfer function, for example, (y/u) which, in case of a voltage controlled power supply, will be $V_{out}/Duty$. Figure 10 illustrates the software diagram for the SFRA inclusion.

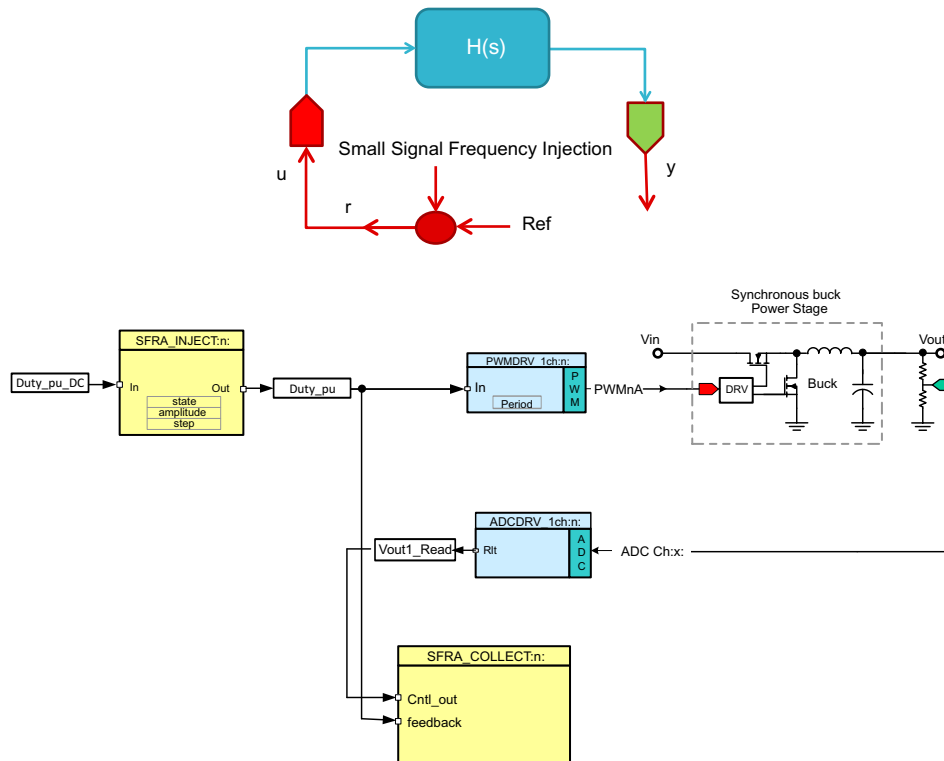


Figure 10. Software Diagram for Plant tf Extraction

Observe the PWM ISR code. Here, the SFRA_INJECT routine is called to add a sinusoidal injection on the DC operating point "Duty_pu_DC" to generate the duty cycle "Duty_pu" for the PWM module. Later in the code, the feedback is used to collect the response for the sinusoidal injection.

```
interrupt void PWM_ISR(void)
{
    ...
    //
    // Read ADC and computer Fbk Value
    //
    Vout1_Read = (float32)Vout1R/(4096.0);

    //
    // Add SFRA injection into the duty cycle for the open loop converter
    //
    Duty_pu=SFRA_F32_inject(Duty_pu_DC);

    //
    // Update PWM value
    //
    EPwm1Regs.CMPA.half.CMPA=((long)(BUCK_PWM_PERIOD))* Duty_pu;

    SFRA_F32_collect(&Duty_pu,&Vout1_Read);
    ...
}
```

Once the DC point is established, that is the converter is running and outputting the desired duty cycle. The SFRA GUI can be used to initiate a SFRA sweep. Open the SFRA GUI.exe located in the GUI folder of SFRA install.

For instruction on how to connect and what each panel on the SFRA GUI box means, see [Section 3.6](#). Click Setup Connection and set the baud rate to be 57600 on the pop up window. Uncheck boot on connect and select the appropriate COM port. For procedures to find out which COM port to select, see [Section 3.6](#). Click OK to close the pop-up window and return to the main screen.

On Main Window, click Connect. Once connected the GUI will parse the current settings for the FRA sweep from the controller, these include the Start Frequency of the sweep, the length of the frequency sweep array (this is fixed in the code and hence cannot be changed through the GUI), injection amplitude and steps per decade. Leave these as default for now.

Press the Start Sweep button. Wait for the status bar in the GUI to change to Sweep Complete. The results of the SFRA sweep are now displayed on the window. The open loop graph result has no meaning as the plant is not in closed loop operation. Therefore, select Plant in the drop down menu to the left. Once Plant is selected in the drop down menu, the GUI will look similar to [Figure 11](#).

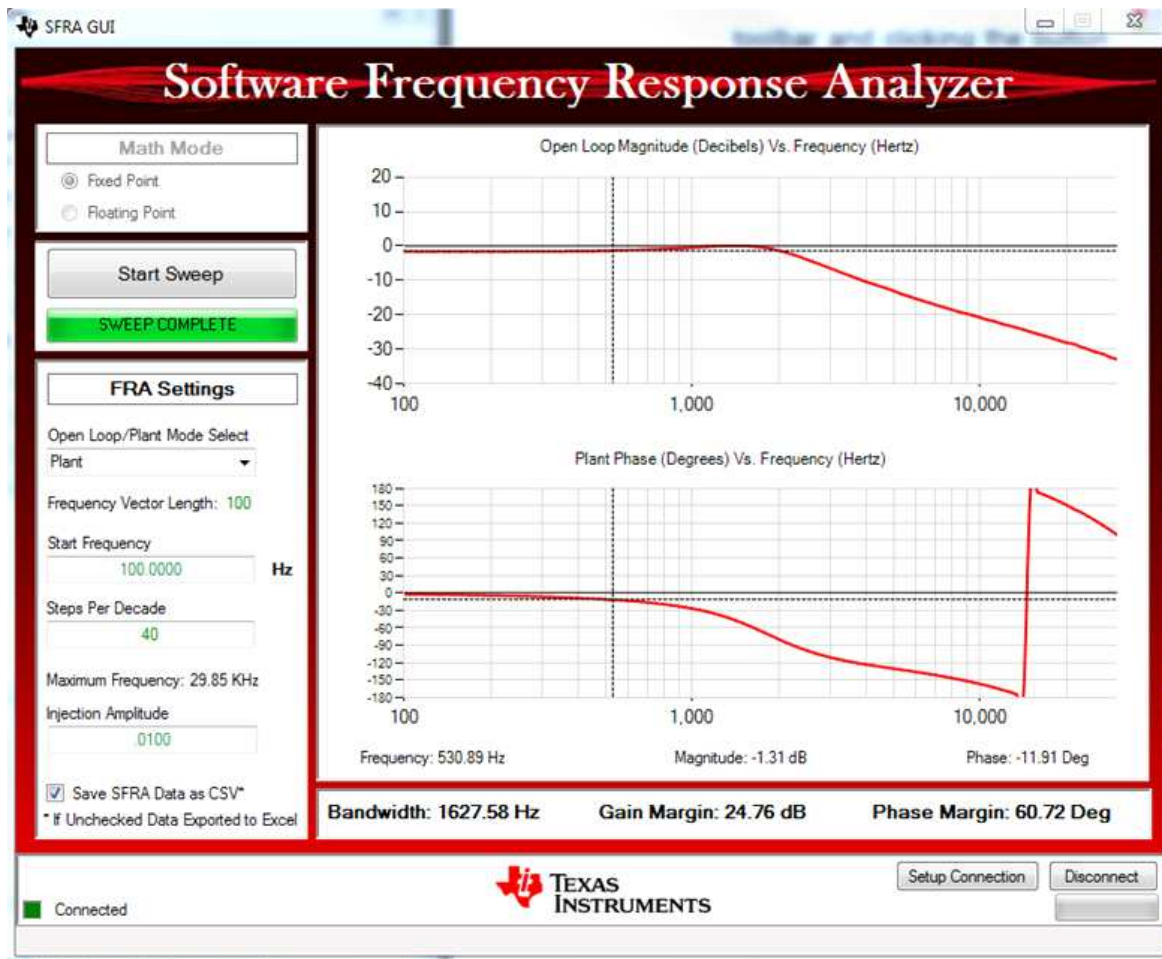


Figure 11. Plant Frequency Response Plot

Drag your mouse across either of the graphs to locate the values at specific frequencies used in the Frequency Response Analysis. A SFRADData.csv file inside the GUI folder is updated with the latest run of the SFRA library. All runs of the SFRA library and time stamps are saved under that folder. If the checkbox "Save SFRA Data as CSV" is unchecked, the excel sheet will pop up after each sweep with the data of the frequency response. Each subsequent sweep is added as a new page on the excel sheet.

NOTE: The change to the checkbox will only affect the save of the next run of the SFRA. The selection must be done before hitting “Start Sweep”.

The frequency response measured can then be used to design the compensation by importing the frequency data is MATLAB. The Scripts were described in [Section 3.5](#).

5.2 Designing Compensator Using Compensation Designer

The SFRA_GUI.exe puts the latest frequency sweep data inside the folder from which it is invoked, which typically is the GUI folder inside the install directory of the SFRA library. The compensation design GUI can be launched by clicking on the “CompDesigner.exe” located at the same folder. The Compensation Designer defaults to using the latest run of the SFRADData.csv file. A run of SFRA was completed in [Section 5.1](#). The CompDesigner will use that SFRA measurement to design the compensation.

Figure 12 shows the Compensation Designer with two pole two zero compensation style selected and used to design a stable closed loop system, based on plant data from SFRA run in the previous section. The compensator coefficients can be copied from the compensation designer GUI into the C code. This will be done in the next section.

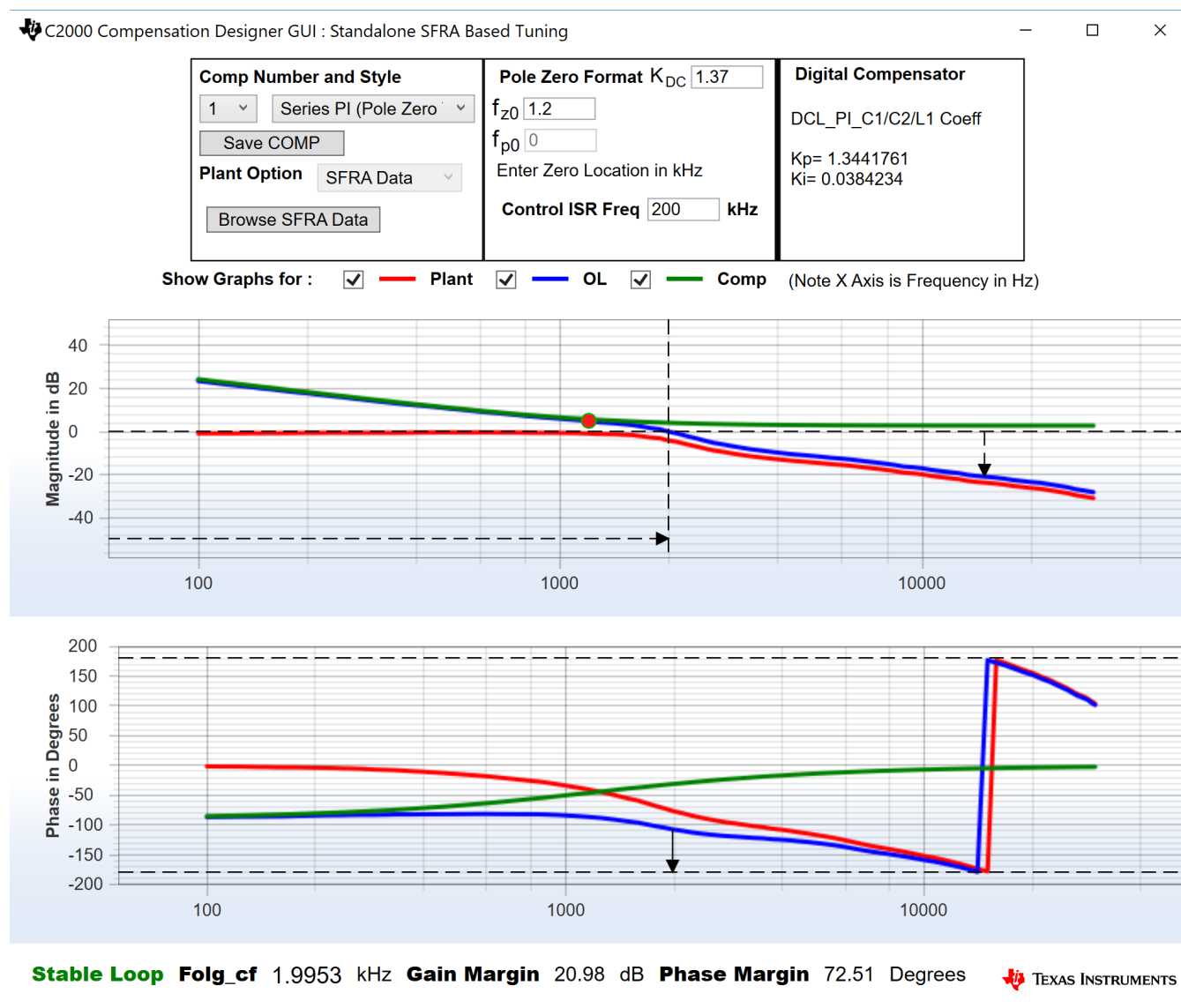


Figure 12. Comp Designer Being Used to for the Compensator Design for a Buck Power Stage, by Using SFRA Data From Open Loop Run

5.3 OL Measurement

The designed compensator can then be implemented on the controller and the SFRA library can be used to measure the open loop frequency response of the closed loop power converter. Figure 13 describes the software connections for a closed loop system using SFRA.

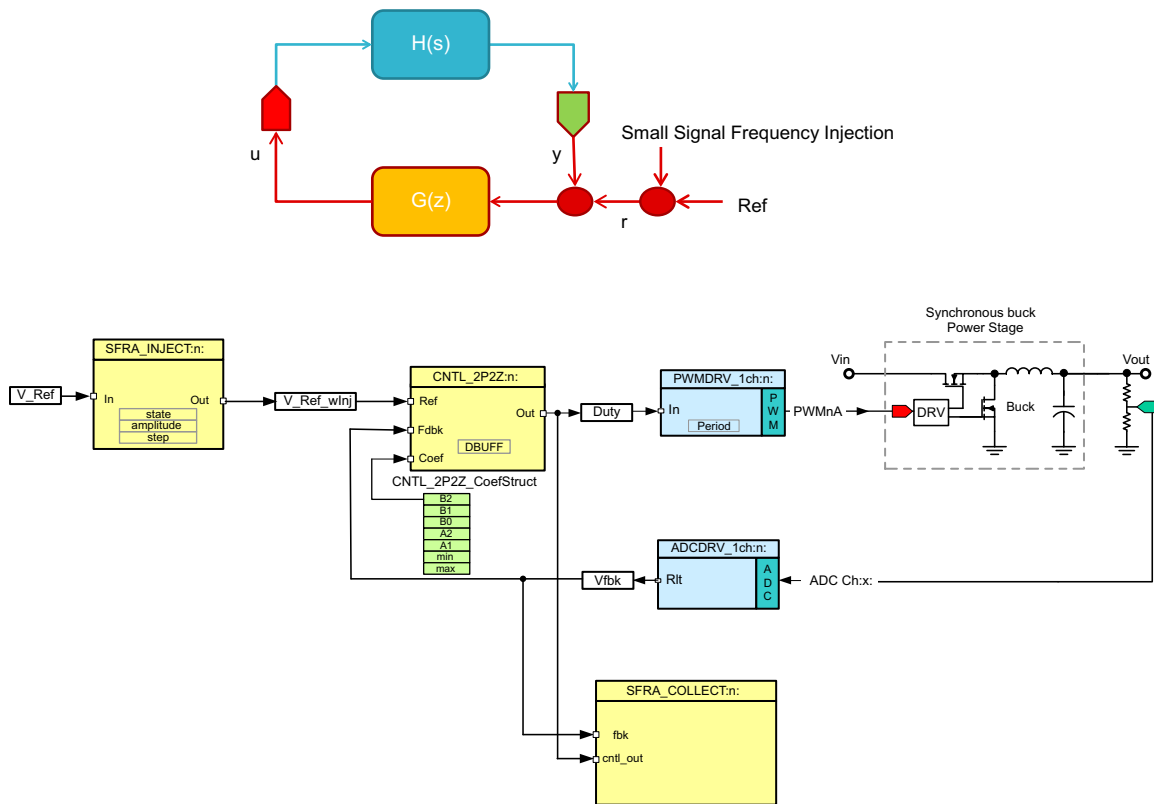


Figure 13. Closed Loop SFRA Software Diagram

The closed loop control ISR will look like the code shown below with the SFRA routines added.

```
interrupt void PWM_ISR(void)
{
    ...
    //
    // Read ADC and computer Fbk Value
    //
    Vout1_Read = (float32)Vout1R/(4096.0);

    //
    // Add SFRA injection into the reference of the controller
    //
    Vout1_Ref_wInj= SFRA_F32_inject(Vout1_Ref);

    //
    // Call the controller
    //
    gi_out=DCL_runPI_C1(&gi,Vout1_Ref_wInj,SFRA_F32_inject(Vout1_Read));

    //
    // Update PWM value
    //
    EPwm1Regs.CMPA.half.CMPA=((long)(BUCK_PWM_PERIOD))* gi_out;

    SFRA_F32_collect(&gi_out,&Vout1_Read);
    .....
}
```

1. Once the code is running, to run SFRA, open the SFRA_GUI.exe, located in the GUI folder under the SFRA install directory. Select "floating point" math for devices such as F28004x, F2837x.
2. Click Setup Connection and set the baud rate to be 57600 on the pop up window.
3. Uncheck boot on connect and select the appropriate COM port. For the procedure to find out which COM port to select, see [Section 3.6](#).
4. Click OK to close the pop-up window and return to the main screen.
5. On the Main Window, click Connect. Once connected, the GUI will parse the current settings for the FRA sweep from the controller. These include the Start Frequency of the sweep, the length of the frequency sweep array (this is fixed in the code and cannot be changed through the GUI), injection amplitude and steps per decade. Leave these as default for now.
6. Press the Start Sweep button.
7. Wait for the status bar in the GUI to change to Sweep Complete.
8. The results of the SFRA sweep are displayed on the large panel in the SFRA GUI. The control performance parameter like open loop gain cross over frequency gain margin and phase margin are also reported in a panel on the SFRA GUI. It is noted that the values reported match closely to what the system was designed for using the Plant Frequency Response Data from the open loop run.

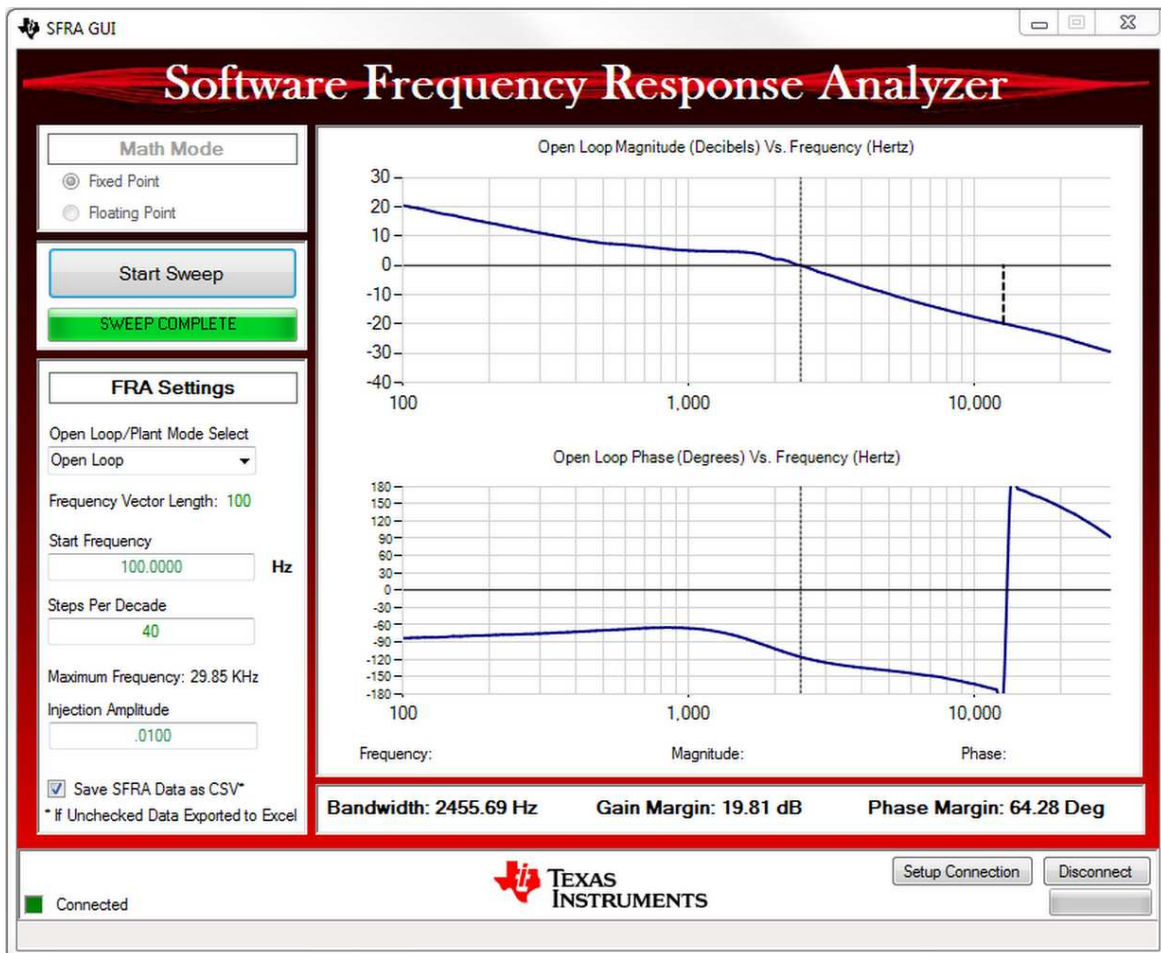


Figure 14. SFRA GUI With OL Frequency Response Plot

9. Drag your mouse across either of the graphs to locate the values at specific frequencies used in the Frequency Response Analysis.

10. A SFRAData.csv file inside the GUI folder is updated with the latest run of the SFRA library. All runs of the SFRA library and time stamps are saved under that folder. If the checkbox “Save SFRA Data as CSV” is unchecked, the excel sheet will pop up after each sweep with the data of the frequency response. Each subsequent sweep is added as a new page on the excel sheet.

NOTE: The change to the checkbox will only affect the save of the next run of the SFRA. The selection must be done before hitting “Start Sweep”.

11. Critical values like Bandwidth, gain margin and phase margin are brought out on the GUI. The values can be used to verify whether or not the designed compensation really met the goals.
12. Once finished, put SW1 in the OFF position, and the code can be halted and CCS debug session stopped.

5.4 Comparing SFRA Measured Frequency Response Versus Modeled

The SFRA library has been run successfully on a number of power topologies. The following section shows the comparison between modeled and measured frequency responses on the DPSWorkshop board, which is a buck converter. This clearly shows that the modeling was close enough to what was measured and vice versa. Some margin of error is expected due to parameter estimation error and non ideality of the model.

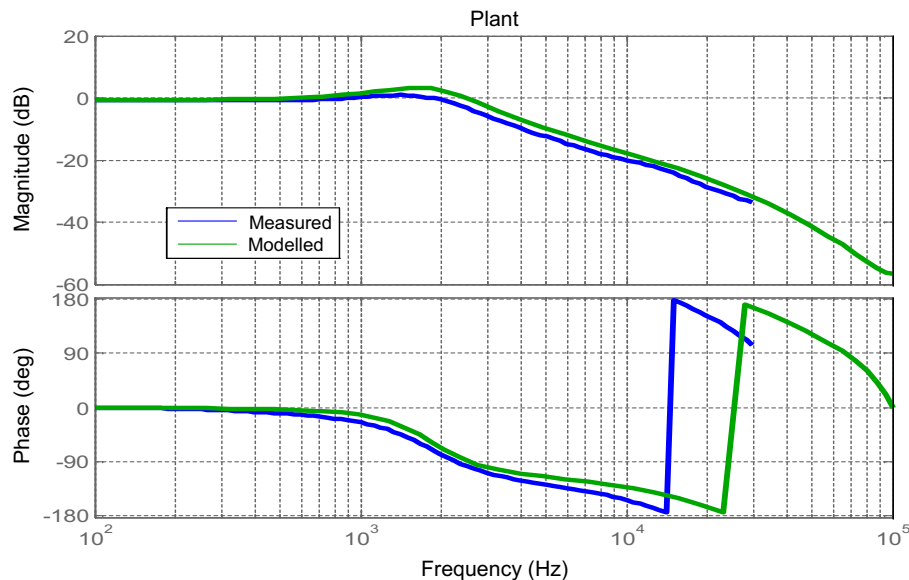


Figure 15. Plant Frequency Response Modeled vs Measured on DPSWrkShpKit

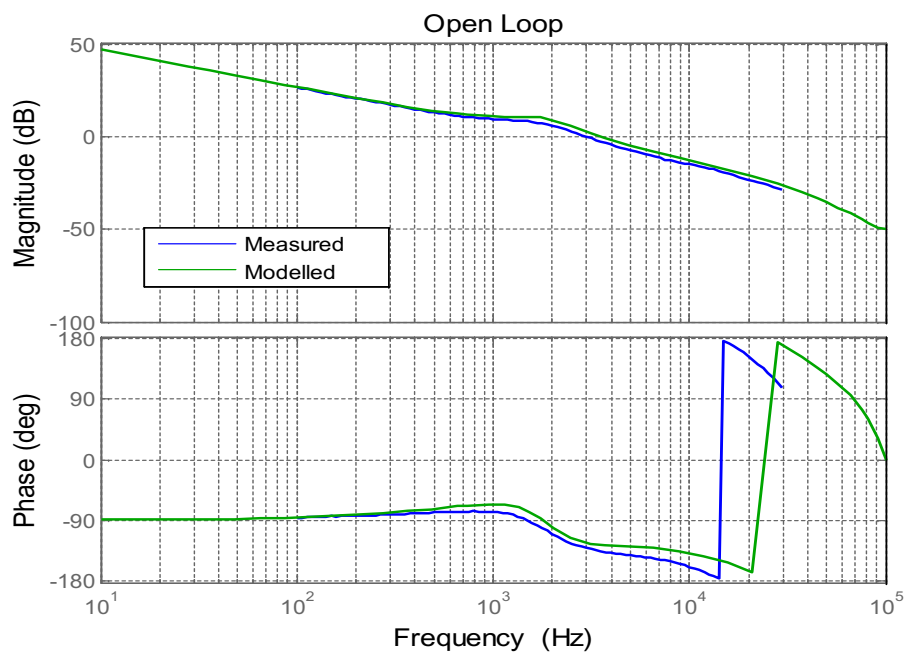


Figure 16. Open Loop Frequency Response Modelled vs Measured on DPSWrkShpKit

The SFRA GUI has been validated in the following TI Designs for different topologies.

- Buck ([BOOSTXL-BUCKCONV](#)): SFRA verified with model for voltage loop operation.
- Single Phase Bridged Boost PFC ([TMD5ILPFCKIT](#)): SFRA verified with model for current loop and voltage loop.
- Solar Micro Inverter Kit ([TMD5SOLARUINVKIT](#)) : Grid connected inverter current control loop verified with model and SFRA measurements. Flyback and the DC Bus regulation loop design based on SFRA plant measurement.
- LLC converter ([TIDM-1001](#), [TMD5HVRESLLCKIT](#)): Voltage loop designed using SFRA for LLC converter.
- Totem Pole PFC ([TIDM-1007](#), [TIDA-01604](#)): Voltage and current loop for PFC verified with SFRA and model comparison.
- Three Phase PFC using Vienna rectifier ([TIDM-1000](#)): Voltage loop and current loop verified from the model and compared with SFRA. DC Balance loop verified with SFRA measurement.
- Critical Mode PFC ([TIDA-0961](#))
- Motor Control with [FCL](#)
- Voltage source inverter ([TIDM-HV-1PH-DCAC](#)) Inner current loop and outer voltage loop with proportional resonant controllers verified with SFRA.
- Grid connected inverter ([TIDM-HV-1PH-DCAC](#)) Grid connected current loop verified with SFRA.
- In addition to the above, a software test bench is also provided as part of the SFRA package to self test the SFRA algorithm that is outlined in [Section 6](#).

6 Running Software Test Bench Example for SFRA

The SFRA library comes with a software test bench (STB) example that runs the SFRA around a fictitious loop that consists of a proportional integral controller, inherently due to the sampled nature of SFRA the digital update delay, as shown in [Figure 17](#).

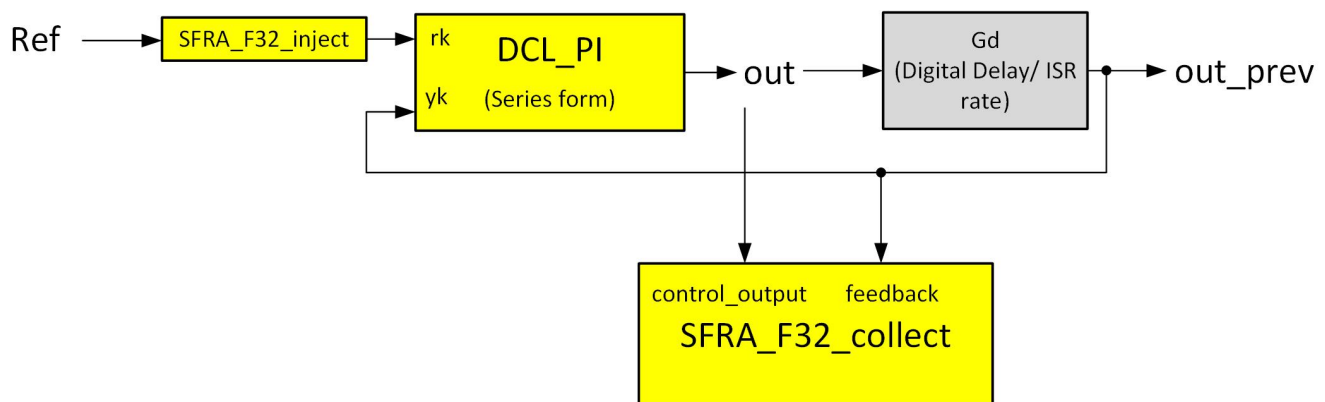


Figure 17. Software Test Bench Example

To import the STB project:

1. Open CCS and click on Project tab and click on Import CCS Project. In the window that pops up, browse to the SFRA install directory under examples <sfra_install>/examples/ccs. All of the available example projects will be displayed. Click "Finish" and the project will be copied into your workspace.

NOTE: The STB example can run the single precision floating point lib and the single precision floating point lib with tmu. A selection can be made while importing the project

2. Make sure the correct control card is connected to the host machine. That is, for the F28004x STB project, an F28004x control card must be connected to the machine. Compile and load the project, and run the project.

3. Open the SFRA GUI.exe as outlined in the section [Section 5.1](#). Once the SFRA GUI is connected , start a sweep. Once finished the SFRA will display under open loop curve the plot for a simple PI compensator as shown in [Figure 18](#) and the frequency response of a digital delay in the plant curve as shown in [Figure 19](#).

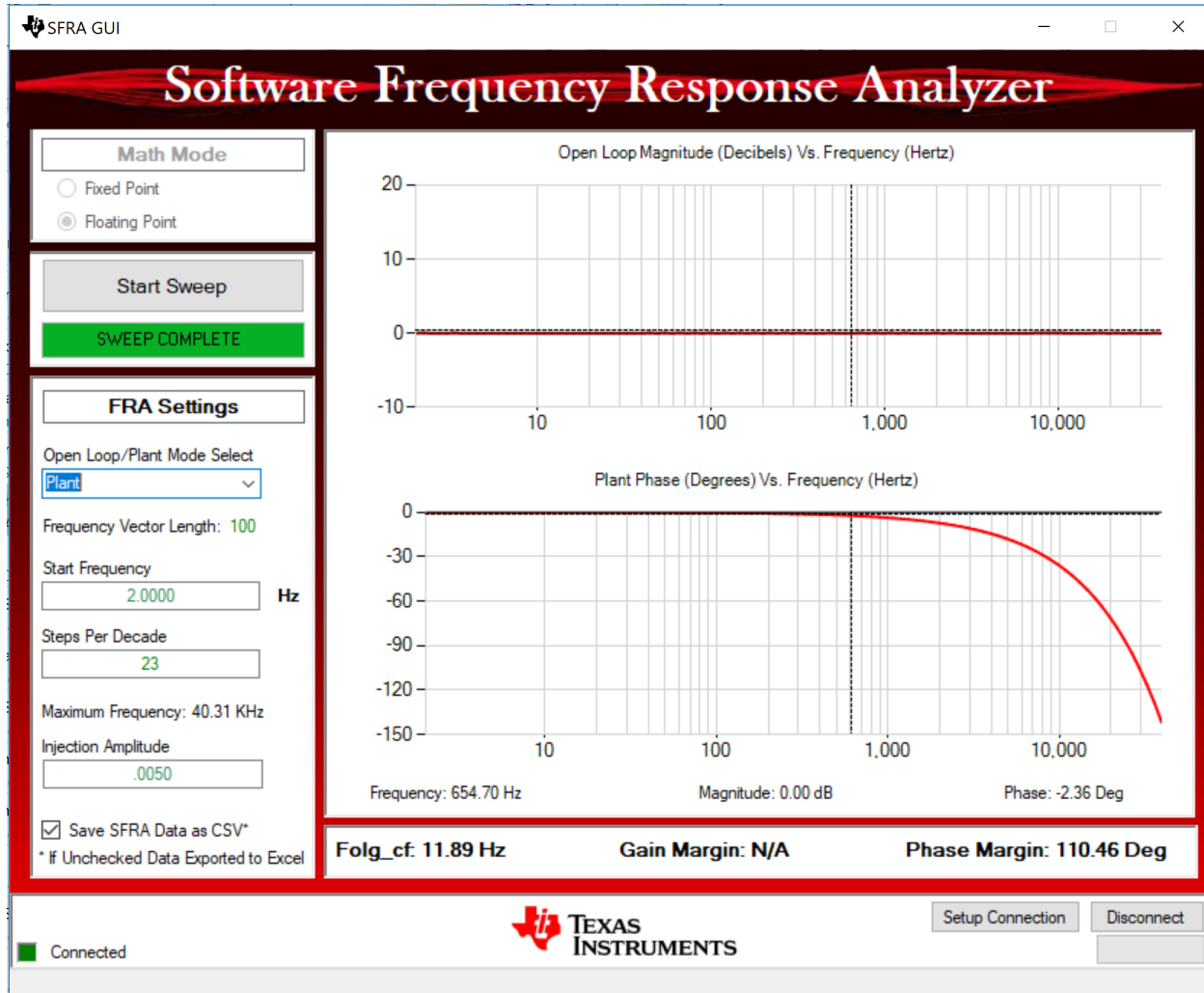


Figure 18. SFRA Plot for the Plant, After STB Example 1 Run

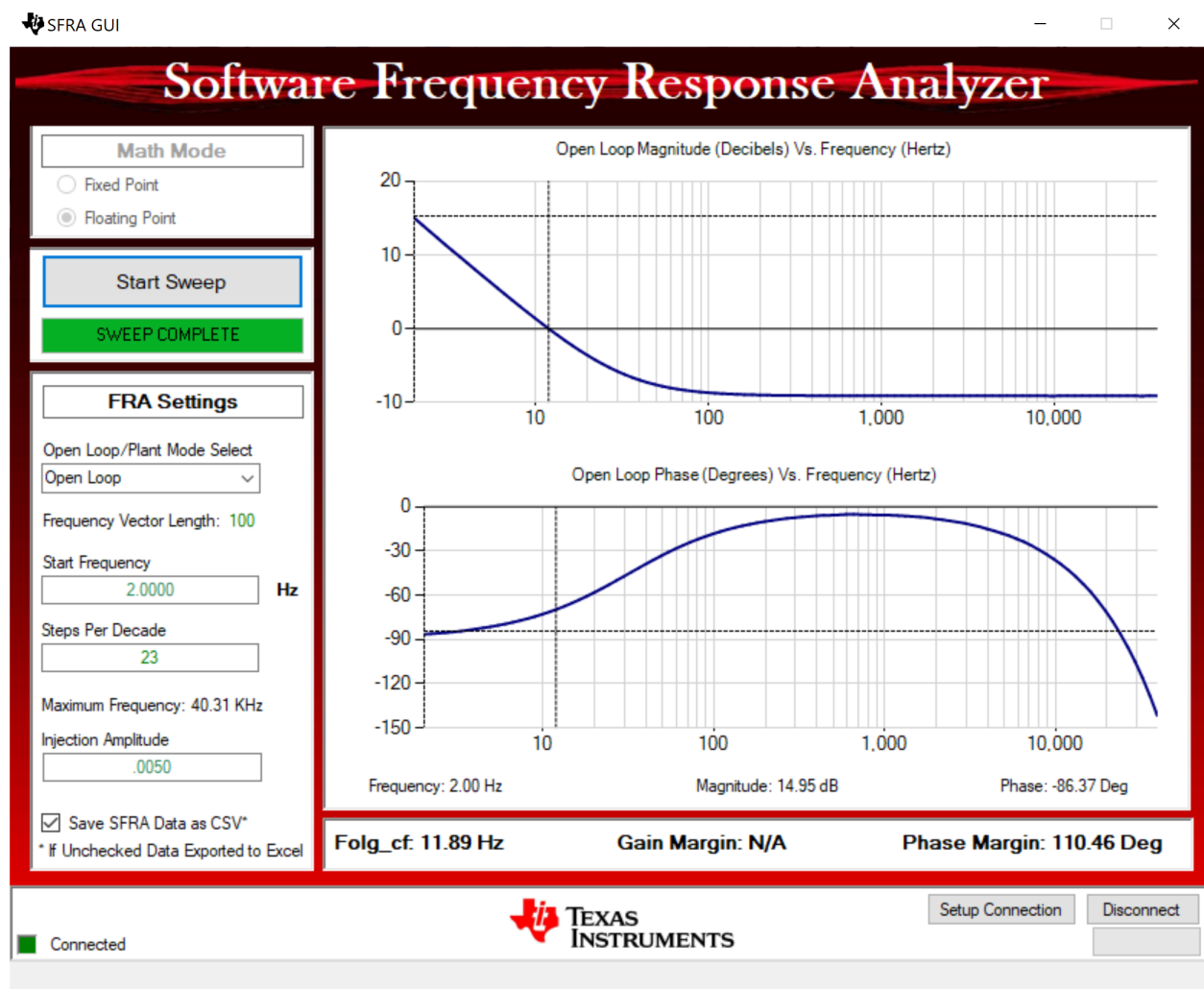


Figure 19. SFRA Plot for Open Loop After STB Example 1 Run

4. These are then compared with the frequency responses by simulating the same PI compensator in a simulation tool, such as MATLAB. This script is available along with the STB example 1 run data under the scripts folder in SFRA install directory. The file is called "STAB_Data_Compare_with_model.m". The comparison plots for plant is shown in [Figure 20](#), and comparison for open loop is shown in [Figure 21](#).

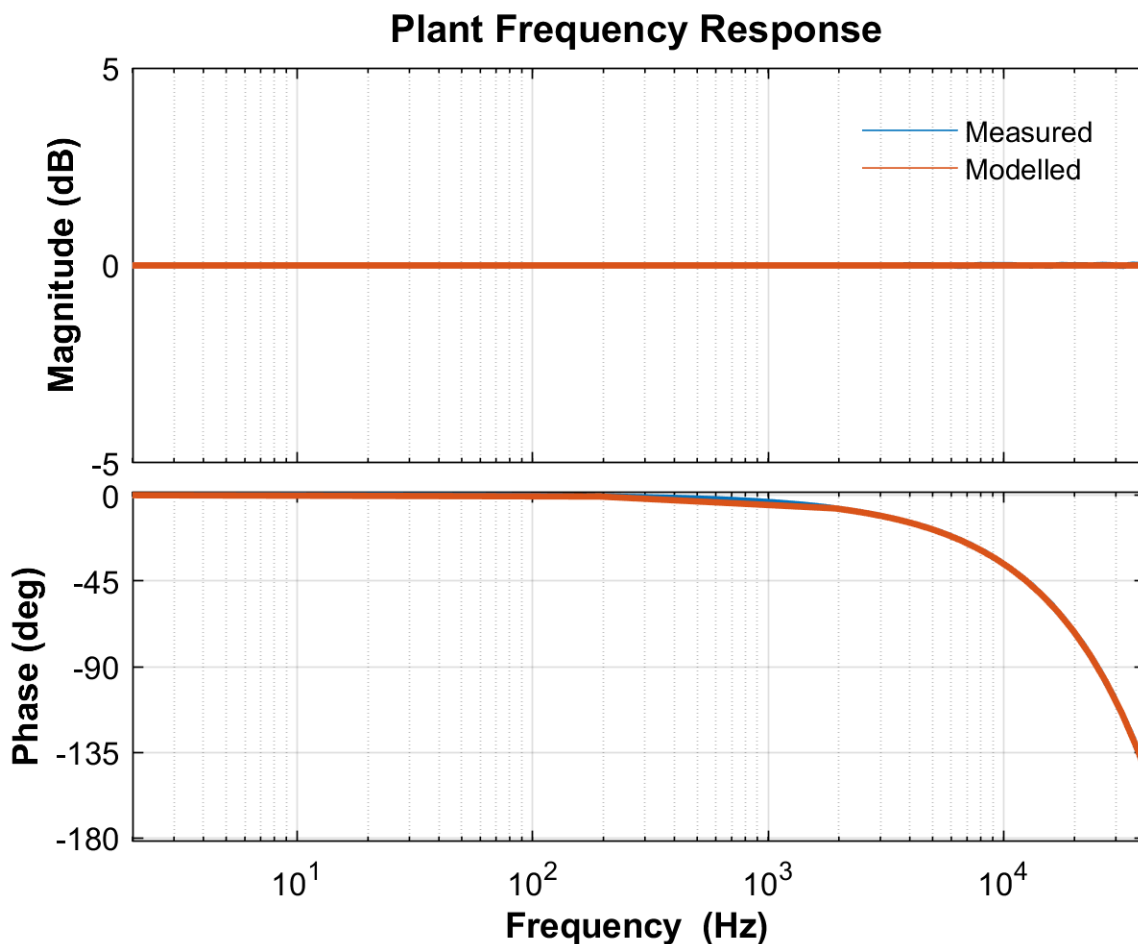


Figure 20. Plant Frequency Response Measured vs Modeled in Software Test Bench Example 1

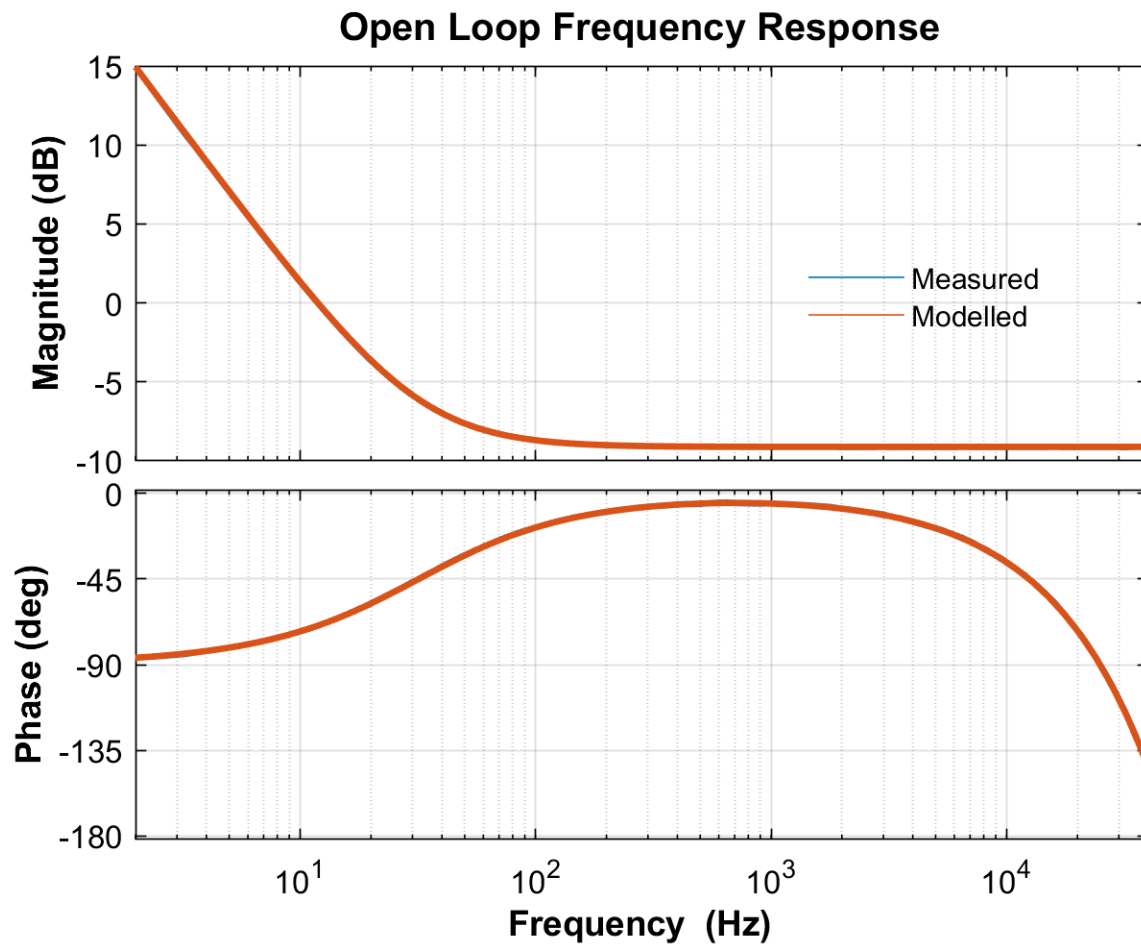


Figure 21. Open Loop Frequency Response Measured vs Modeled for Software Test Bench Example

5. The small errors in the measured and modeled frequency responses are not clear from the above plots. Hence, a zoomed view of the open loop plot for the STB example 1 is shown in [Figure 22](#).

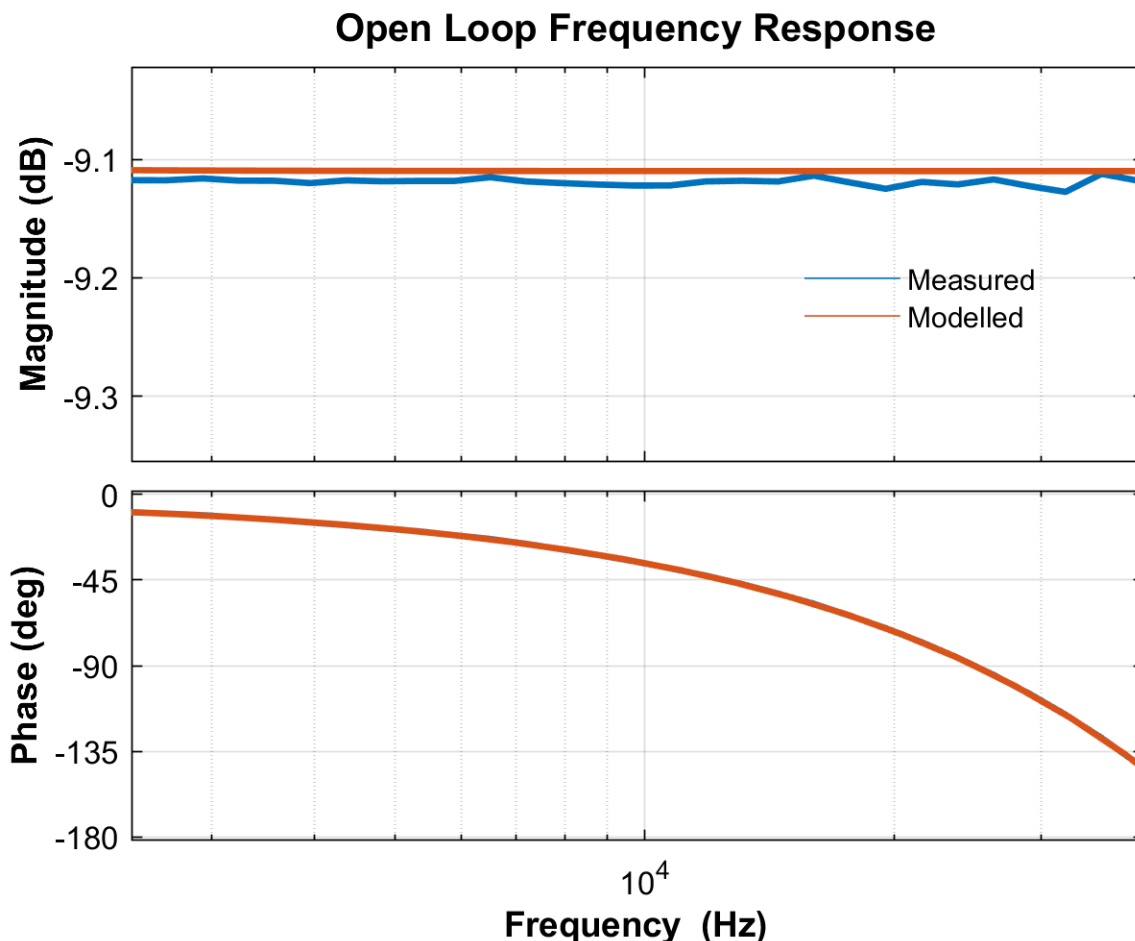


Figure 22. Zoomed in Open Loop Frequency Response for STB Example 1, Measured vs Modeled Comparison

There are some errors that are caused by floating-point operations and how they are handled on the MCU, and how matlab handles them. Overall, this verifies the SFRA algorithm function as the errors are within reasonable error limits. Closed loop transfer function plot can also be verified in a similar manner.

7 Using SFRA Without SFRA GUI Integration

Do these steps to integrate the SFRA in the project:

1. Because of resource limitations, some of the applications may not have an SCI port free that is needed by the SFRA GUI.
2. In that case, omit the steps for the SFRA GUI integration in [Section 3.4.4](#).
3. Integrate the Inject, Collect and background functions as described in [Section 3.4.3](#).
4. To start an SFRA sweep, put the SFRA object in the watch window.
5. Write SFRA_OBJ.start to 1, when you want the SFRA sweep to start. (This is the alternative to clicking start sweep on the SFRA GUI.)
6. Monitor the SFRA_OBJ.FreqIndex variable; it will gradually increment as SFRA sweep is performed.
7. Once the SFRA_OBJ.FreqIndex reaches Vec_Length, the SFRA sweep is complete.

8. As part of the SFRA initialization, as pointed out in [Section 3.4.3](#), the Open Loop and Plant Magnitude and phase are stored in arrays called.

```
//----- SFRA Related Variables -----
float plantMagVect[SFRA_FREQ_LENGTH];
float plantPhaseVect[SFRA_FREQ_LENGTH];
float olMagVect[SFRA_FREQ_LENGTH];
float olPhaseVect[SFRA_FREQ_LENGTH];
float freqVect[SFRA_FREQ_LENGTH];
```

9. Put these in the watch window to inspect and study the response.
10. Once the sweep is complete, click on View-> MemoryBrowser inside CCS.
11. Inside Memory Browser, enter &freqVect to see the frequency vector and select 32-bit floating point,

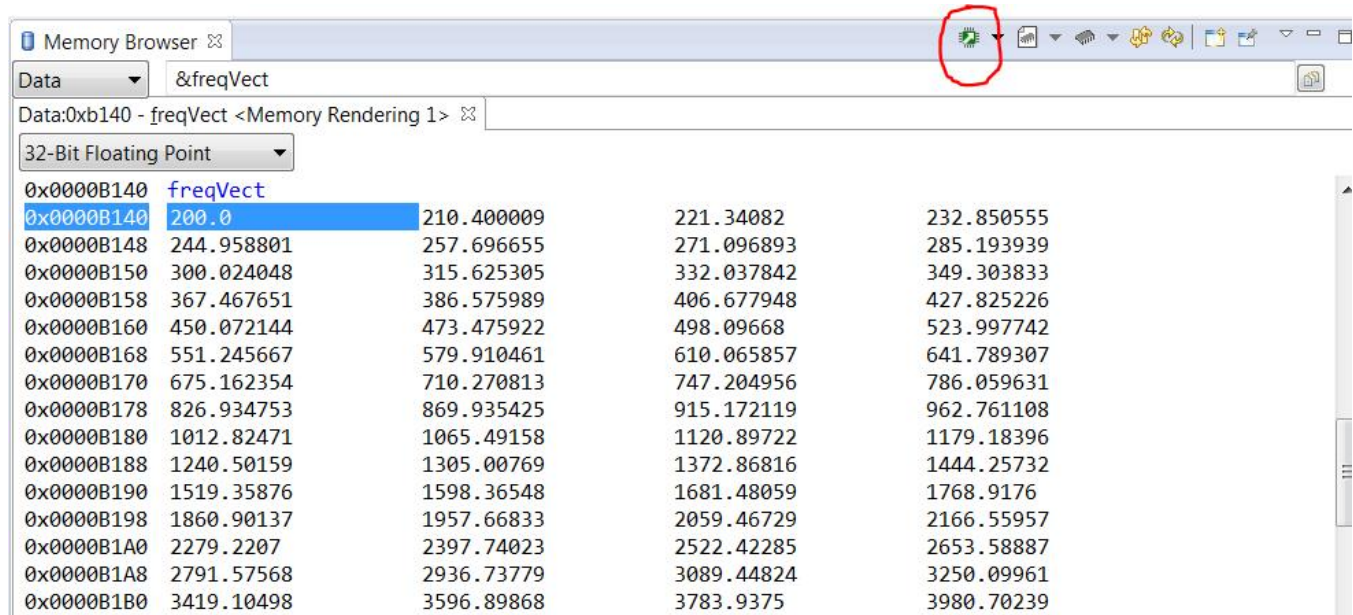


Figure 23. Memory Browser View of Stored SFRA Vectors

12. Click on save memory, shown encircled in [Figure 23](#).

13. A popup will appear. Select TI data and specify the file name *.dat in the location you prefer.

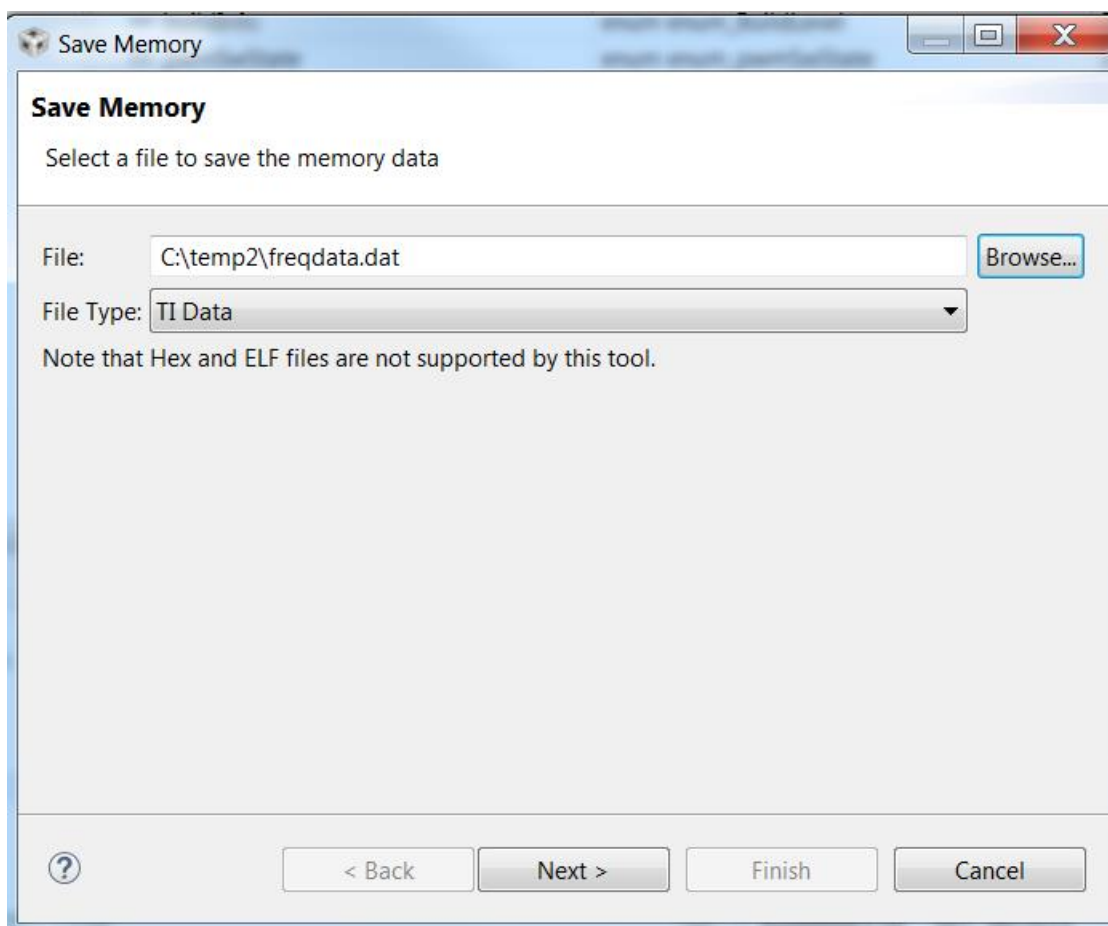
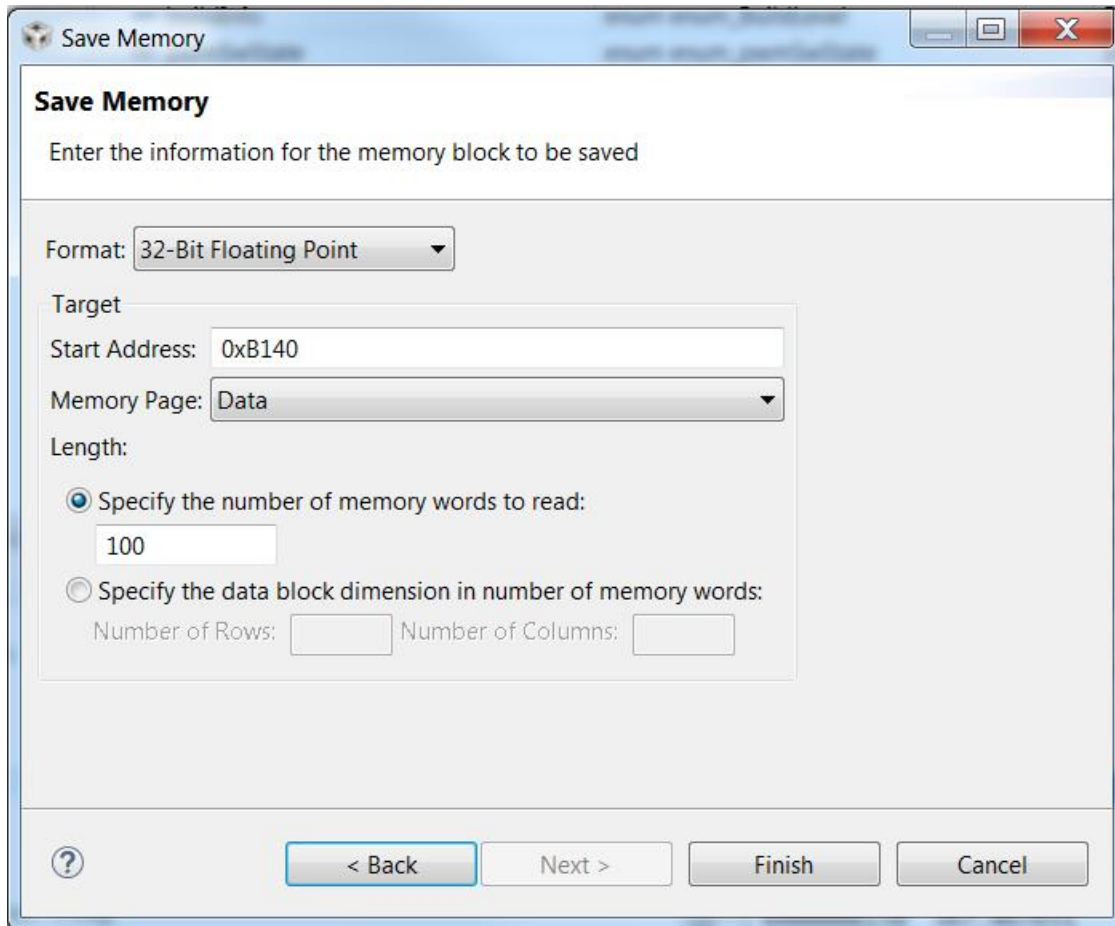


Figure 24. Save Memory Pop-Up Window

14. Click on Next and specify the address from the memory browser for the start of the array and then the length.

15. Make sure 32-bit floating point is selected. Click Finish.



Save Memory

Enter the information for the memory block to be saved

Format: 32-Bit Floating Point ▼

Target

Start Address: 0xB140

Memory Page: Data ▼

Length:

☒ Specify the number of memory words to read:

100

☐ Specify the data block dimension in number of memory words:

Number of Rows: Number of Columns:

? < Back Next > Finish Cancel

Figure 25. Save Memory Options

16. This will save the data in *.dat file.
17. Repeat this step for plantMagVect, plantPhaseVect, olMagVect, olPhaseVect, so you will have 5 *.dat files.
18. If you want to use this data in MATLAB or other tools, the data can be populated to an excel file.
19. Open the FRADData.xlsx file located at <install directory>\sfra\scripts in excel.
20. You can choose to re-name and save the file.
21. This excel sheet has four columns, in the first column is the frequency data.
22. Open the *.dat file that was saved.

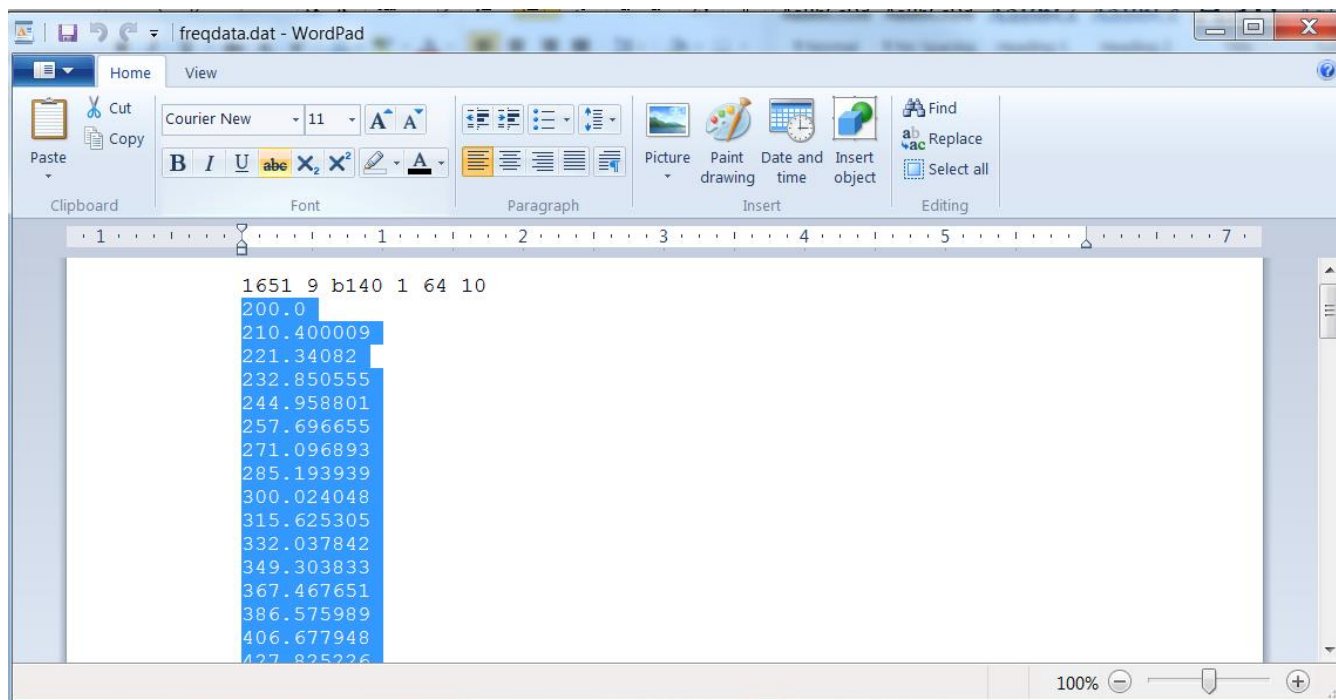
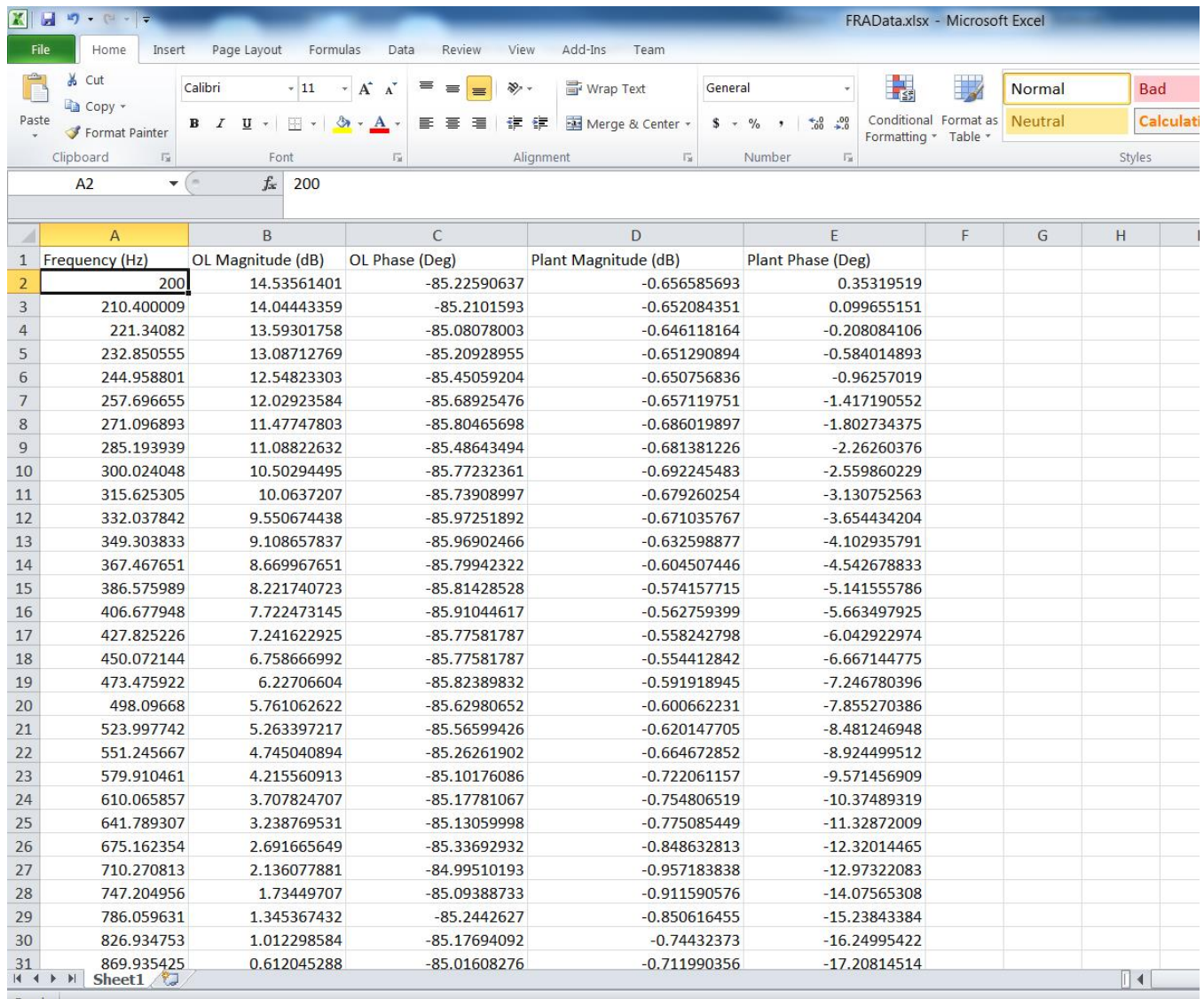


Figure 26. Selecting Data From .dat File to Put in the Excel

23. Select the data from the second line onwards to the end of the file and do Ctrl+C to copy the data.
24. Open the Excel File, go to the first element under the corresponding vector and do Ctrl+V to copy the array.



	A	B	C	D	E	F	G	H	I
1	Frequency (Hz)	OL Magnitude (dB)	OL Phase (Deg)	Plant Magnitude (dB)	Plant Phase (Deg)				
2	200	14.53561401	-85.22590637	-0.656585693	0.35319519				
3	210.400009	14.04443359	-85.2101593	-0.652084351	0.099655151				
4	221.34082	13.59301758	-85.08078003	-0.646118164	-0.208084106				
5	232.850555	13.08712769	-85.20928955	-0.651290894	-0.584014893				
6	244.958801	12.54823303	-85.45059204	-0.650756836	-0.96257019				
7	257.696655	12.02923584	-85.68925476	-0.657119751	-1.417190552				
8	271.096893	11.47747803	-85.80465698	-0.686019897	-1.802734375				
9	285.193939	11.08822632	-85.48643494	-0.681381226	-2.26260376				
10	300.024048	10.50294495	-85.77232361	-0.692245483	-2.559860229				
11	315.625305	10.0637207	-85.73908997	-0.679260254	-3.130752563				
12	332.037842	9.550674438	-85.97251892	-0.671035767	-3.654434204				
13	349.303833	9.108657837	-85.96902466	-0.632598877	-4.102935791				
14	367.467651	8.669967651	-85.79942322	-0.604507446	-4.542678833				
15	386.575989	8.221740723	-85.81428528	-0.574157715	-5.141555786				
16	406.677948	7.722473145	-85.91044617	-0.562759399	-5.663497925				
17	427.825226	7.241622925	-85.77581787	-0.558242798	-6.042922974				
18	450.072144	6.758666992	-85.77581787	-0.554412842	-6.667144775				
19	473.475922	6.22706604	-85.82389832	-0.591918945	-7.246780396				
20	498.09668	5.761062622	-85.62980652	-0.600662231	-7.855270386				
21	523.997742	5.263397217	-85.56599426	-0.620147705	-8.481246948				
22	551.245667	4.745040894	-85.26261902	-0.664672852	-8.924499512				
23	579.910461	4.215560913	-85.10176086	-0.722061157	-9.571456909				
24	610.065857	3.707824707	-85.17781067	-0.754806519	-10.37489319				
25	641.789307	3.238769531	-85.13059998	-0.775085449	-11.32872009				
26	675.162354	2.691665649	-85.33692932	-0.848632813	-12.32014465				
27	710.270813	2.136077881	-84.99510193	-0.957183838	-12.97322083				
28	747.204956	1.73449707	-85.09388733	-0.911590576	-14.07565308				
29	786.059631	1.345367432	-85.2442627	-0.850616455	-15.23843384				
30	826.934753	1.012298584	-85.17694092	-0.74432373	-16.24995422				
31	869.935425	0.612045288	-85.01608276	-0.711990356	-17.20814514				

Figure 27. SFRA Data Copied in Excel File

25. Repeat the steps for each column.
26. Once the excel file is updated for all five columns, use the matlab script to import the FRA data. Then, use it inside sisotool to design compensator and carry out stability analysis.
27. The file can also be downloaded as a *.csv file and used with the compensation designer tool.

8 FAQ

- **GUI will not connect**

Make sure the SCI pins are configured to be used as SCI and the SCI module clock rate value provided to the SCI init routine is correct.

- **GUI will connect but start button does not become active**

If your native language is not English (USA) and the above behavior is observed, try selecting the English (USA). To do this, go to "Region and Language" using search in the Start Button → select under Formats → English (USA) → Apply.

- **For measurable response, what is the maximum frequency SFRA?**

Maximum frequency measurable is determined by the SFRA_ISR_FREQ and is limited by the nyquist criterion (half of the switching frequency or the frequency at which the SFRA routine is called).

- **What is the lowest amplitude signal measurable by SFRA?**

SFRA uses the on-chip ADC whose noise floor for a 12-bit ADC is at approximately 60dB-70dB. Therefore, readings of the FRA below 60dB should be interpreted with caution as there may be significant noise elements.

- **Why does the SFRA sweep take so long?**

The time taken by the SFRA depends on two factors:

- The ISR frequency in which the SFRA routine is called.
- The rate at which the background task is called, therefore, it is expected that SFRA will run much quicker when used to measure the response of a 200 kHz power converter compared to a one running at 10 kHz-20 kHz.

- **How to make sure SFRA does not run in production code.**

Unless the SFRA is started by writing a '1' to the 'start' variable in the SFRA object (the SFRA GUI also writes to this variable to initiate a sweep when the start sweep button is clicked), the SFRA injection routine will not do anything to the reference value (inject/collect).

If it is desired to completely remove the SFRA routines, the call to run the SFRA routines can be put under a #define and for release code that #define can be defined as something else or none. As shown below in a typical code, the SFRA may be needed on the voltage or the current and typically the software has a define for this. Hence, to remove the SFRA routines, it can be defined as SFRA_NONE, which will make sure SFRA routines are not compiled into the object.

Also note that although the parameters to the collect routine are passed by reference, the parameters are not modified by the routine.

```
//SFRA Options
#define SFRA_NONE 0
#define SFRA_CURRENT 1
#define SFRA_VOLTAGE 2
#define SFRA_TYPE          SFRA_NONE
.....
#if SFRA_TYPE == SFRA_VOLTAGE
    gv_out=GV_RUN(&gv, SFRA_F32_inject(vBusRefSlewed), vBus_sensed);
#else
    gv_out=GV_RUN(&gv, vBusRefSlewed,vBus_sensed);
#endif

#if SFRA_TYPE == SFRA_VOLTAGE
    SFRA_F32_collect(&gv_out,&vBus_sensedFiltered_notch2);
#endif
```

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (September 2018) to A Revision	Page
• Updates were made in Section 1	2
• Update was made to Table 1	5
• Update was made in Section 3.1	6
• Updates were made in Section 3.4.1	7
• Updates were made in Section 3.4.2	8
• Updates were made in Section 3.4.3	9
• Figure 6 was updated in Section 3.6	15
• Updates were made in Section 5.1	21
• Updates were made in Section 5.3	24
• Updates were made in Section 6	28

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](#) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2022, Texas Instruments Incorporated