Application Report
SPRACJ9–April 2019

Fast Serial Interface (FSI) Skew Compensation

Nima Eskandari

ABSTRACT

This application report provides information on how to set up the Fast Serial Interface (FSI) on a C2000™ MCU. The integrated skew compensation block on the receiver is used to handle skew that may occur between the clock and data signals due to a variety of factors, including trace-length mismatch and skews induced by an isolation chip. Operating the FSI at maximum speed (50 MHz) at dual data rate (100 Mbps) may require the integrated skew compensation block to be configured according to the specific operating conditions on a case by case basis. This application report provides example software on how to configure and set up the integrated skew compensation block on the Fast Serial Interface.

Project collateral discussed in this document can be downloaded from the following URL: http://www.ti.com/lit/zip/spracj9.

Contents

List of Figures

List of Tables

Trademarks

C2000, Code Composer Studio are trademarks of Texas Instruments.

1 Introduction

The FSI module is a serial communication peripheral capable of reliable and robust high-speed communications. The FSI is designed to ensure data robustness across many system conditions such as chip-to-chip as well as board-to-board across an isolation barrier. Payload integrity checks such as CRC, start- and end-of-frame patterns, and user-defined tags, are encoded before transmit and then verified after receipt using without additional CPU interaction. Line breaks can be detected using periodic transmissions, all managed and monitored by hardware. The FSI is also tightly integrated with other control peripherals on the device. To ensure that the latest sensor data or control parameters are available, frames can be transmitted on every control loop period. With embedded data robustness checks, data-link integrity checks, skew compensation, and integration with control peripherals, the FSI can enable high-speed, robust communication in any system. One of the most important features of the FSI module, skew compensation, allows the FSI receivers to overcome the skew introduced to the data and clock signals in the high-speed communication. The software examples provided in this application report showcases how to setup the skew compensation block of the FSI receiver.

2 Software Example

The example software can be downloaded from http://www.ti.com/lit/zip/spracj9. The description for the content of the top-level folder is shown in Table 1.

Table 1. Top-Level Folder Description

Folder Name	Description
fsi_ex12_delay_tap_measurement	This example uses the HRCAP module to measure the FSI RX delay elements. The measure delays can be graphed using the FSI Skew Compensation Utility.
fsi_ex14_dual_line_delay_select_rx	Companion: fsi_ex14_dual_line_delay_select_tx In this example, the FSI module is configured to listen for a ping at dual data rate (using both RXD0 and RXD1). The software tests whether the ping sent from the TX device is correctly received against all combinations of delay elements activated. RXD0: 0-31 delay elements activated RXD1: 0-31 delay elements activated RXCLK: 0-31 delay elements activated The software stores the status of the ping received (fail/pass) for each of the 32x32x32 combinations of the delay line elements. This result can be graphed using the FSI Skew Compensation Utility.
fsi_ex14_dual_line_delay_select_tx	Companion: fsi_ex14_dual_line_delay_select_rx This example configures the FSI module to transmit pings at dual data rate (using both RXD0 and RXD1).
fsi_ex13_single_line_delay_select_rx	Companion: fsi_ex13_single_line_delay_select_tx In this example, the FSI module is configured to listen for a ping at single data rate (using RXD0). The software tests whether the ping sent from the TX device is correctly received against all combinations of delay elements activated. RXD0: 0-31 delay elements activated RXCLK: 0-31 delay elements activated The software stores the status of the ping received (fail/pass) for each of the 32x32 combinations of the delay line elements. This result can be graphed using the FSI Skew Compensation Utility.
fsi_ex13_single_line_delay_select_tx	Companion: fsi_ex13_single_line_delay_select_rx This example configures the FSI module to transmit pings at single data rate (using RXD0).
fsi_ex15_find_optimal_delay_device2	Companion: fsi_ex15_find_optimal_delay_device1 This example showcases how to find the optimal point for the number of delay elements activated on RXD0, RXD1 and RXCLK for optimal performance. The optimal number of elements selected for the FSI RX module can be calculated using both single and dual data rate. Then it will proceed to transmits pings using either single or dual data rate until the other device signals it to stop.
fsi_ex15_find_optimal_delay_device1	Companion: fsi_ex15_find_optimal_delay_device2 This example transmits pings using either single or dual data rate until the other device signals it to stop. Then it will switch to receive pings until it has calibrated it's skew compensation block.
FSI Skew Compensation Utility	This java-based utility can be used to visualize the results captured by the device. Graphs and csv files are generated.

3 FSI Delay Element Measurement

The FSI receiver module has a programmable delay line on each of the external signal inputs: RXCLK, RXD0, and RXD1. The delay elements introduce delays on the respective lines. This is to facilitate adjustment for signal delays introduced by system level components such as signal buffers, ferrite beads, isolators, and so on, or board delays such as uneven trace lengths, long cable length, and so on. The length of the delay is controlled by setting the RX_DLY_LINE_CTRL register values for each line. There are 32 delay elements available for each of the external signal input. These delay elements must be activated accordingly, in order to ensure that the FSI RX module will meet the requirements for the setup time and hold time. The amount of delay introduced by each delay element can be measure using the high-resolution capture (HRCAP) module. An example project is available with the name of fsi_ex12_delay_tap_measurement which measure the delay elements on RXD1 in nano-seconds.

3.1 Delay Element Measurement Example Software

The fsi_ex12_delay_tap_measurement example project measures the delay elements available on RXD1 using the HRCAP module. The measures are acquired by following the steps below:

1. Initialize the FSI module in loopback mode.
2. Initialize and configure the HRCAP6 module to operate in one-shot mode to capture falling and rising edges of the input signal.
3. Set the HRCAP6 module to calibrate periodically.
4. Set the number of delay elements activated.
5. Set the HRCAP6 module input to FSI RXD1.
6. Send FSI TX Flush Sequence.
7. Measure the delay introduced by the delay elements 10 times.
8. Repeat steps 4-7 for all 32 delay elements.

Follow the step by step instructions below to run the fsi_ex12_delay_tap_measurement example and view the results using the FSI Skew Compensation Utility. You must download and extract the example software zipped folder found http://www.ti.com/lit/zip/spracj9.

1. Launch Code Composer Studio™ (CCS) and import the fsi_ex12_delay_tap_measurement project into Code Composer Studio (CCS). This can be done by selecting Project → Import CCS Projects Browse to the example software folder and importing all the projects.

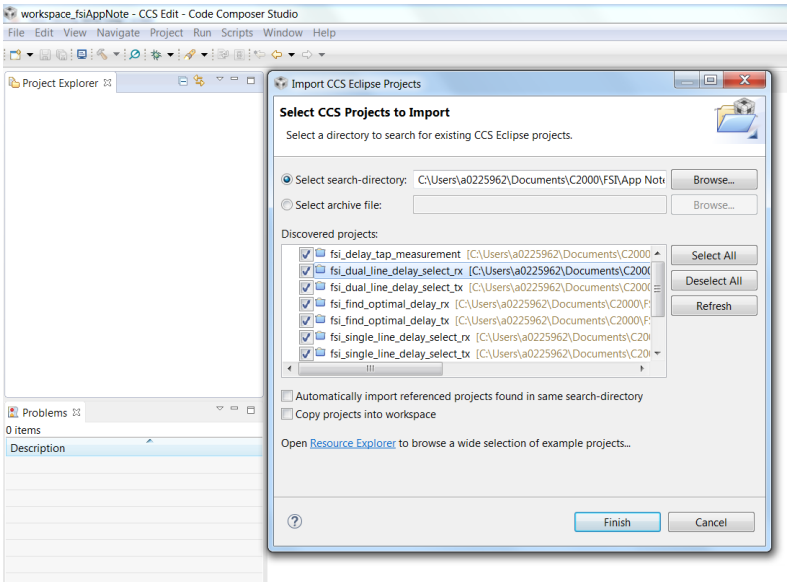

Figure 1. Import All Example Projects

2. After importing the projects, select the fsi_ex12_delay_tap_measurement project in the Project Explorer section, build and launch a debug session.

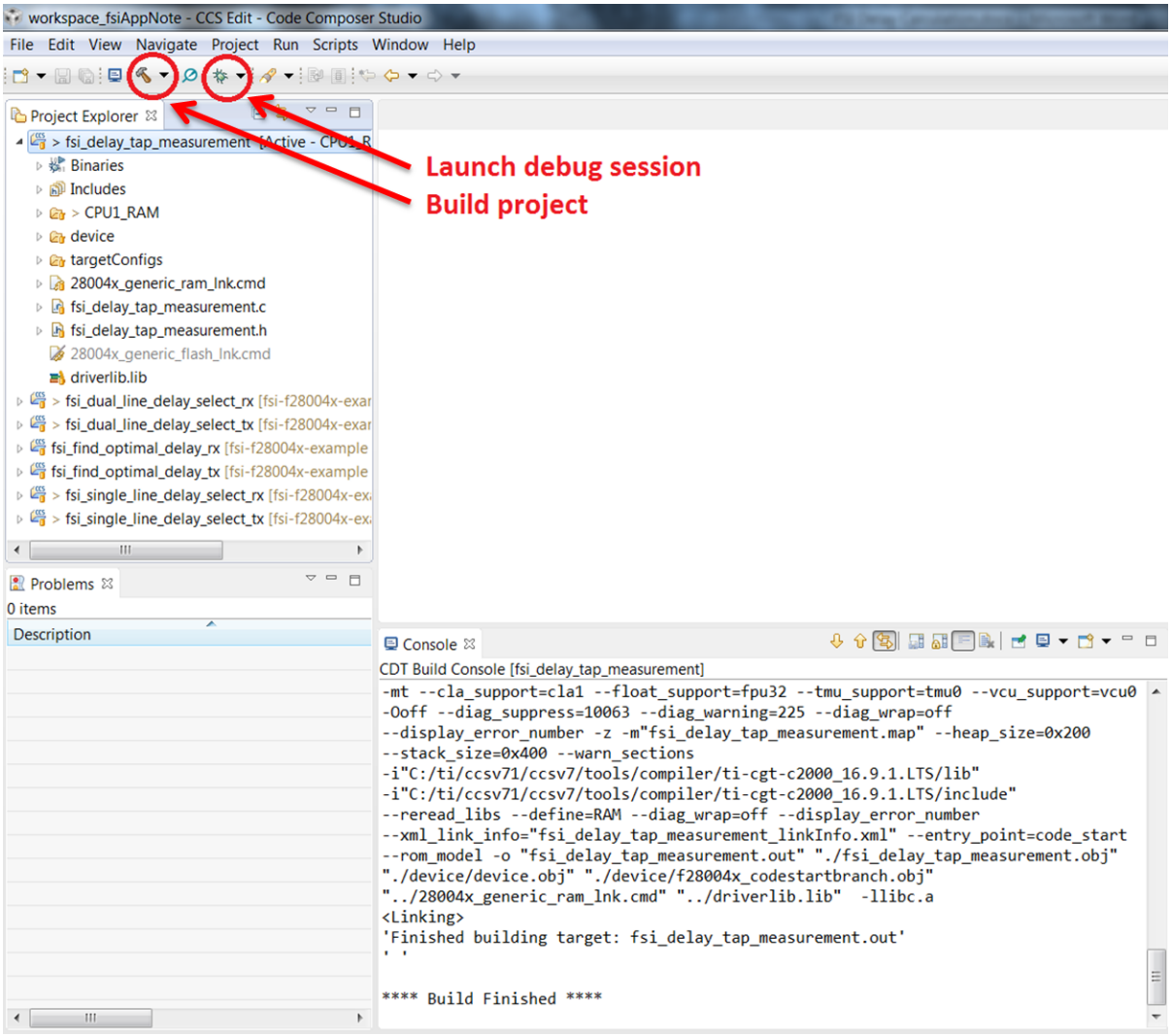

Figure 2. Build and Launch Debug Session

3. After the debug session has been launched, add the variable named “delays” to the Expressions window to view the measured delay introduced by all 32 delay elements (10 samples for each).

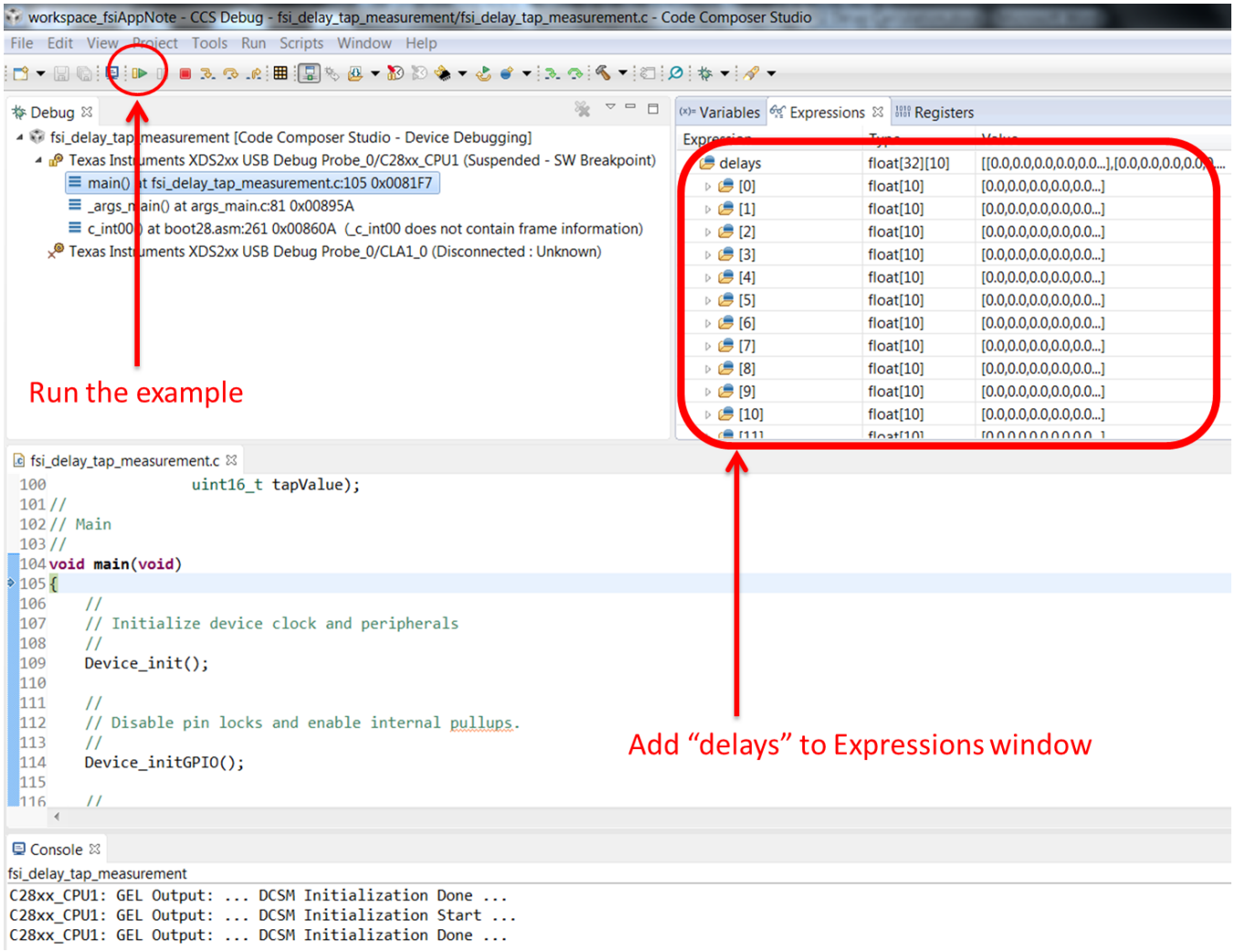

Figure 3. fsi_ex12_delay_tap_measurement in Debug Mode

4. Run the project and after all measurements are done, the device will halt and you can view all the measured delays in nano-seconds inside the watch window.

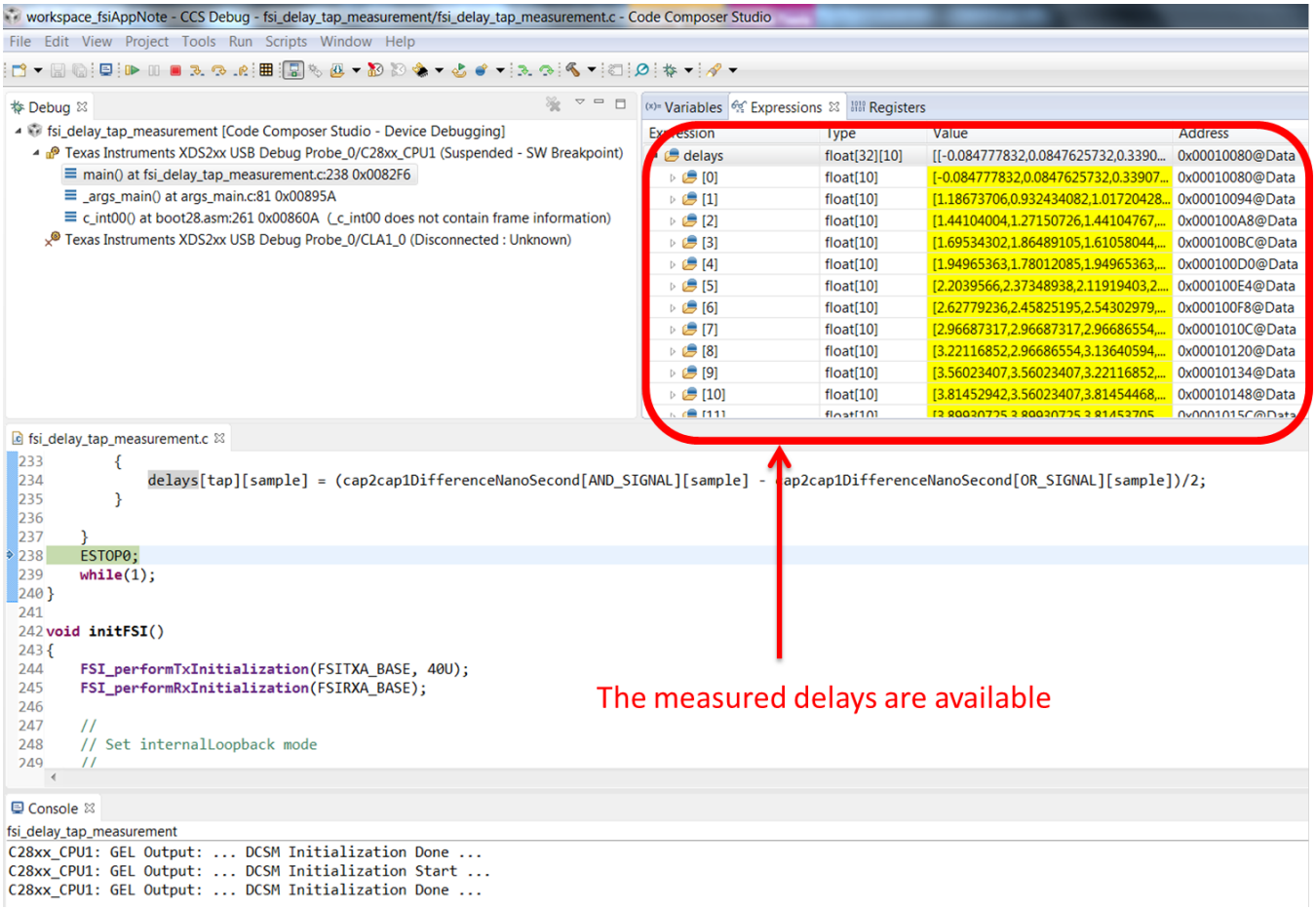

Figure 4. Measured Delay Elements in Expressions Window

5. To graphically view the delays introduced by each delay element, you must save the “delays” variable above using CCS, in a file that will be used as an input to the FSI Skew Compensation Utility. In order to save the “delays” variable, go to Tools → Save Memory.

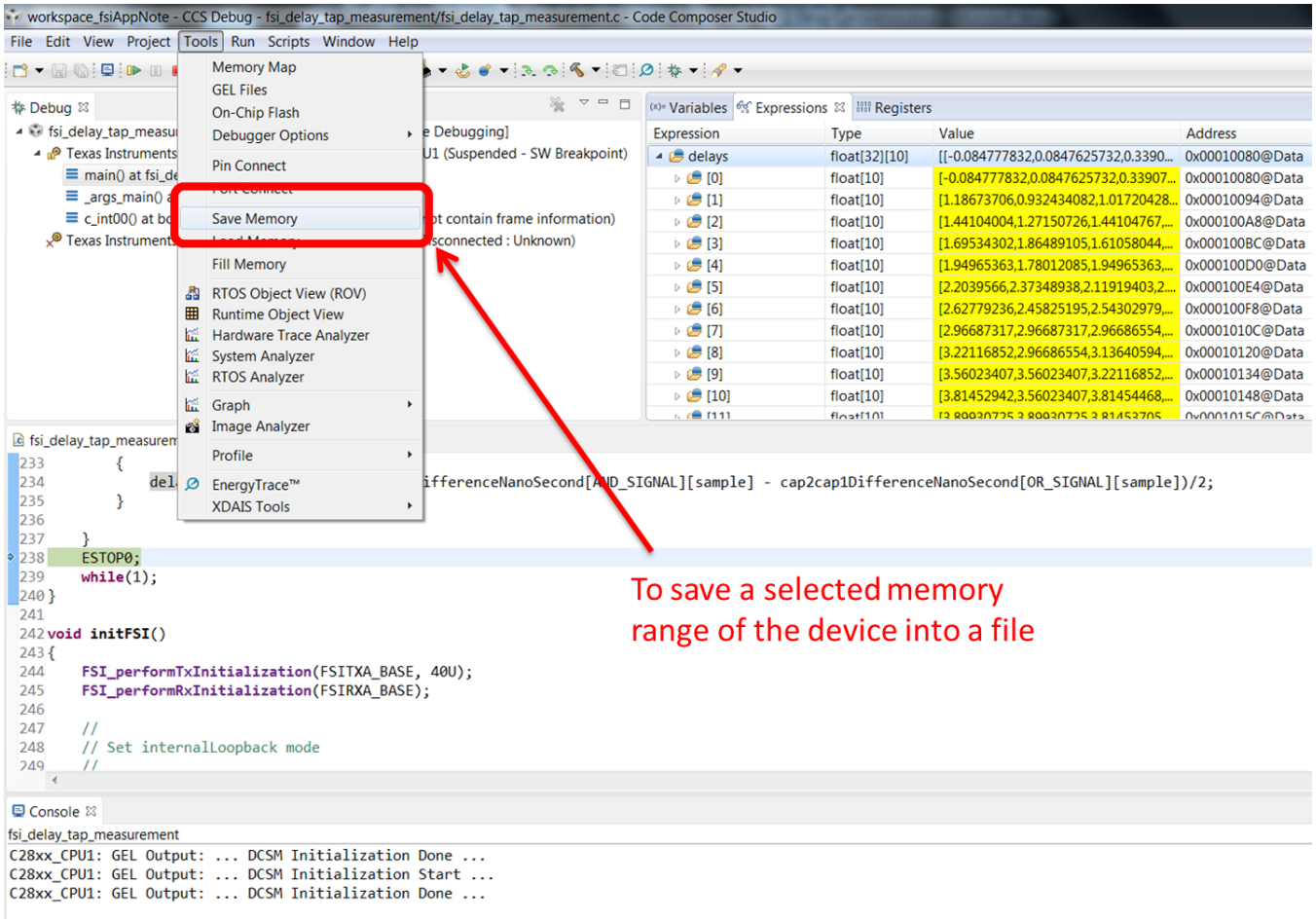

Figure 5. Code Composer Studio Save Memory Option

6. Enter the file location where the memory will be saved. Select “TI Data” in the File Type option and click “Next”. Populate the fields in the dialog as shown in Figure 6.

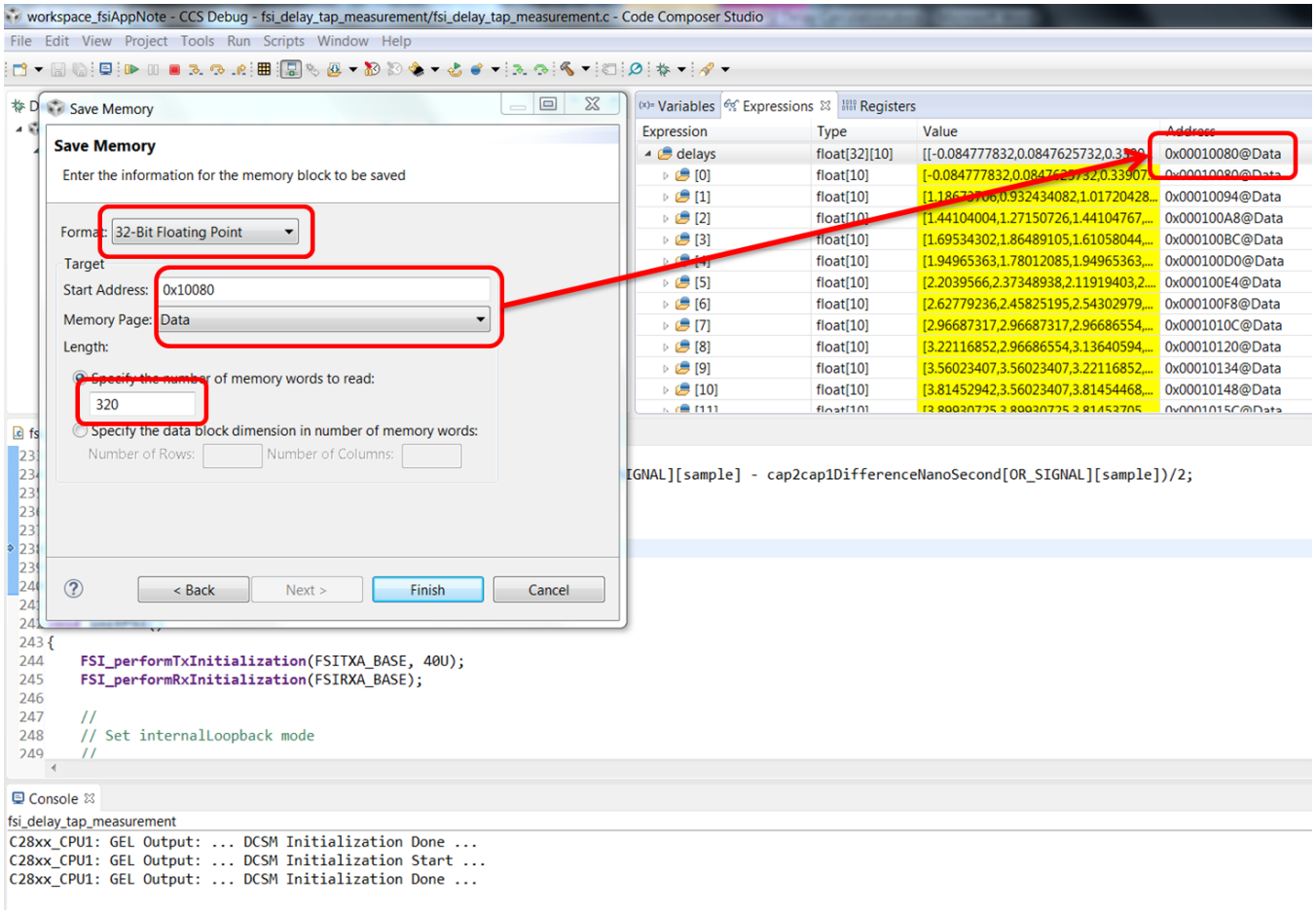

Figure 6. fsi_ex12_delay_tap_measurement Save Memory Options

7. It is possible that the “delays” variable is in another memory location than 0x10080. The location of the “delays” variable can be found in the “Expressions” window as shown in Figure 6. The saved file can be used by the FSI Skew Compensation Utility, to show the delays introduced by the delay elements in a scatter plot graph. Launch the FSI Skew Compensation Utility. In the main menu, select the “Delay Tap Measurement” option.

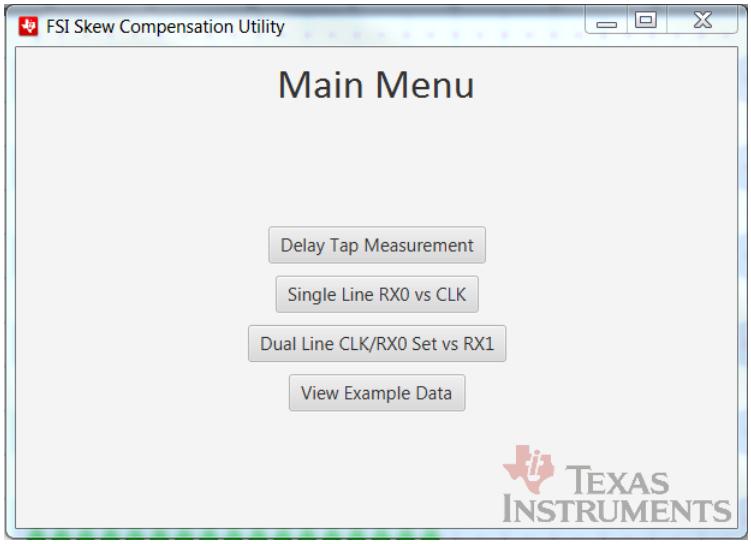

Figure 7. FSI Skew Compensation Utility - Main Menu

8. The utility will prompt the use for the memory file that was saved in the previous steps. Navigate and select the saved memory file. The utility should now graph the delays introduced by each delay element.

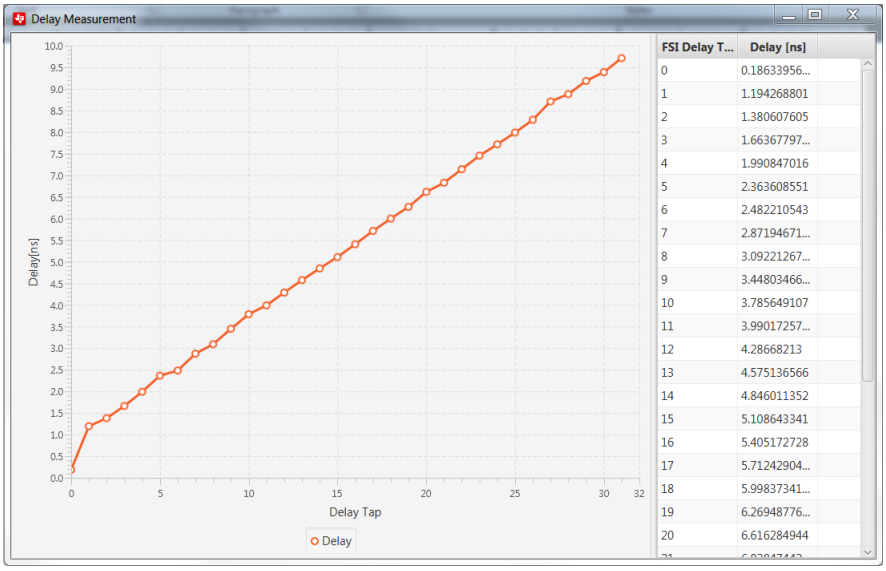

Figure 8. FSI Skew Compensation Utility - Delay Measurement

The delay introduced by each element is averaged over the 10 samples taken by the HRCAP. As seen in Figure 8, the delay elements can add up to approximately 10 ns of delay for each external signal (RXD0, RXD1, RXCLK). The delays are measured using RXD1. The delay elements on RXD0 and RXCLK are not measurable through software and are assumed to be the same.

4 Visualizing the FSI Single Data Line Skew Compensation

The FSI module can be operated using only one data line (RXD0). In this mode the RXCLK and RXD0 acts as clock and data while RXD1 is unused. To compensate for the skew introduced by board, isolators or any other factor, the skew compensation block must be utilized correctly.

For two FSI modules to communicate with one another, they must go through the initialization sequence described in the device-specific TRM. After the initialization sequence is completed, the two devices must establish the communication link. This is done through a sequence of ping frames being transmitted and received. An example of establishing the communication link is described in the device-specific TRM.

In order for the FSI skew compensation block to be calibrated correctly, the two devices will attempt to establish the communication link a number of times. Figure 9 shows the connection of the devices in this example. While there is no true concept of a master or a slave node in the FSI protocol, the example uses this nomenclature as a simple way to describe the data flow.

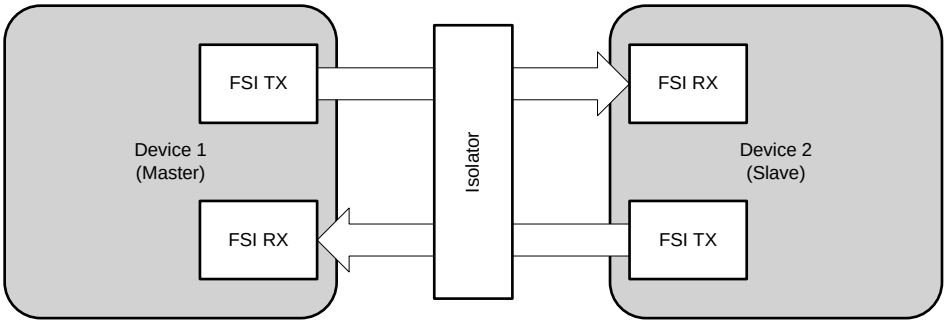

Figure 9. FSI Point to Point Connection

In order for Device 2 (slave) to calibrate its FSI RX module using the skew compensation block, it attempts to establish a communication link with Device 1 (master) a number of times. Each time, the FSI RX module will be programmed to operate at a different delay value set in RX_DLY_LINE_CTRL. The failure or success of the two devices in establishing the communication link will be logged inside the slave device.

An example is provided to show when the two devices succeed or fail at establishing the communication link. For single data line calibration, the example projects fsi_ex13_single_line_delay_select_rx and fsi_ex13_single_line_delay_select_tx are used. The fsi_ex13_single_line_delay_select_rx device attempts to establish a communication link at all 32 x 32 different possibilities for the RXD0 and RXCLK delay line settings. The results are logged and can be graphed using the FSI Skew Compensation Utility.

4.1 FSI Single Line Skew Compensation Example Software

The fsi_ex13_single_line_delay_select_rx and fsi_ex13_single_line_delay_select_tx projects can be used to visualize how the FSI delay line control can be used to compensate for different amounts of skews in the system. The example uses two GPIOs for FSI RX (RXD0 and RXCLK) and two GPIOs for FSI TX (TXD0 and TXCLK). The communication between the two devices are verified at all 32 x 32 (RXD0, RXCLK) different delay line configurations. The communication between the two devices is described in Figure 10 for each delay line configuration. When a device is waiting to receive data, a timeout is implemented to ensure that the never hangs.

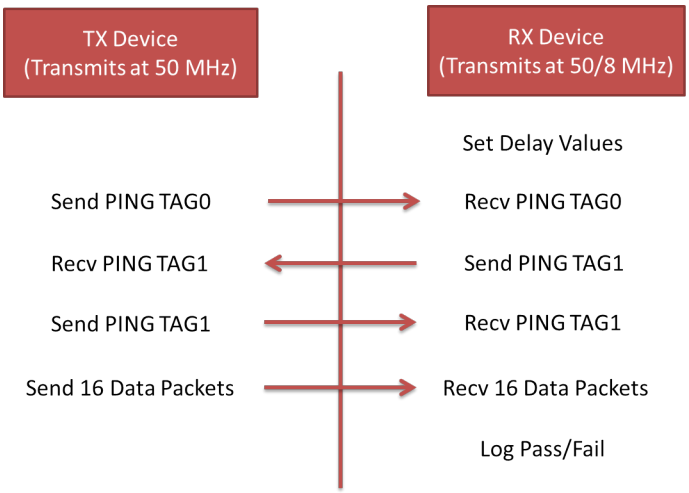

Figure 10. Delay Line Configuration Verification Flow

1. The two projects must be imported into CCS. It is recommended to have two different instances of CCS open and import each project into a different workspace. This way you can debug both projects at the same time.
   a. The fsi_single_line_delay_tx project must be started first.
   b. Aafter importing the project, build and run the example.
2. The fsi_single_line_delay_rx project must be started. It takes some time for this example to finish. The CPU will halt when the example is finished and the result is saved in a variable named “pingAndDataStatus”, which is a 32 element array. The index of the array 0-31 represents RXCLK delay from 0 to 31.The value of each pingAndDataStatus[x], where x is the RXCLK delay, represents the bit-packed status of ping and data transmission for RXD0 delay from 0-31. Therefore, pingAndDataStatus[12] = 0x00001FF0 means that data and ping transmission was successful for RXCLK delay = 12 and RXD0 delay = 4-12.

   Figure 11 shows two instances of CCS: one with fsi_single_line_delay_tx and the other with fsi_single_line_delay_rx, in debug perspective.

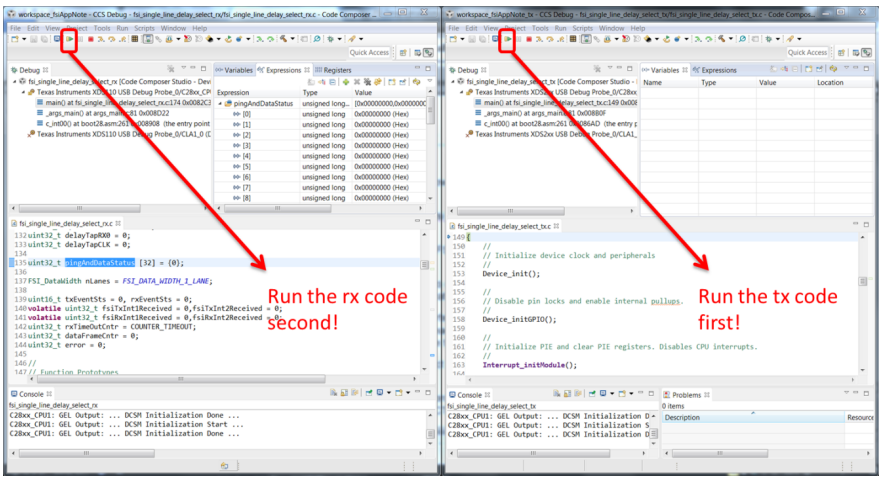

Figure 11. fsi_single_line_delay_tx And fsi_single_line_delay_rx Example Software in Debug Perspective

3. After both example projects are running, the fsi_single_line_delay_rx project will halt. It may take some time for the software to finish gathering the results. The fsi_single_line_delay_tx will not halt. It continues to send pings, which can be ignored for this example's purpose. The fsi_single_line_delay_rx project halts and the gathered result can be viewed in "pingAndDataStatus" as shown in the "Expressions" window.

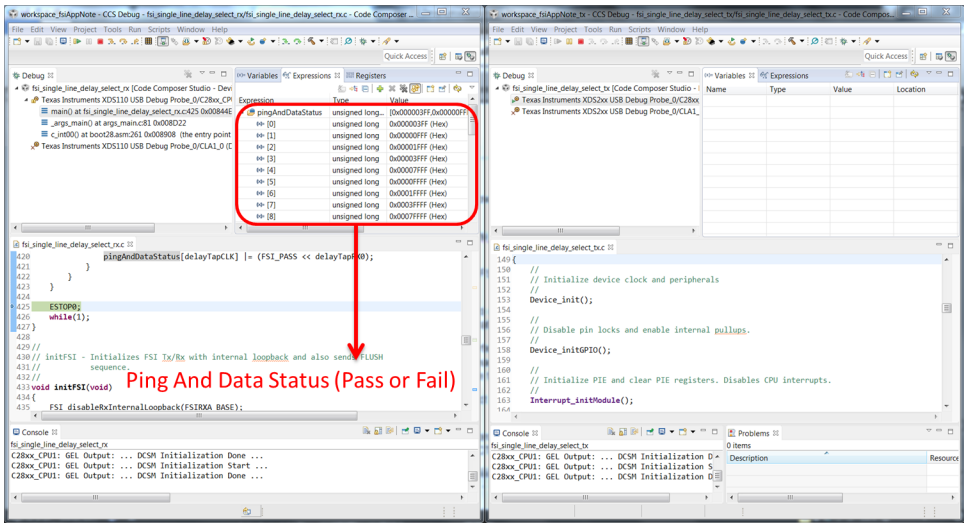

Figure 12. fsi_single_line_delay_rx Halted When Finished

4. The results captured in this example can be used by the FSI Skew Compensation Utility to generate graphs to visualize the data. In order to save the "pingAndDataStatus", go to Tools → Save Memory.

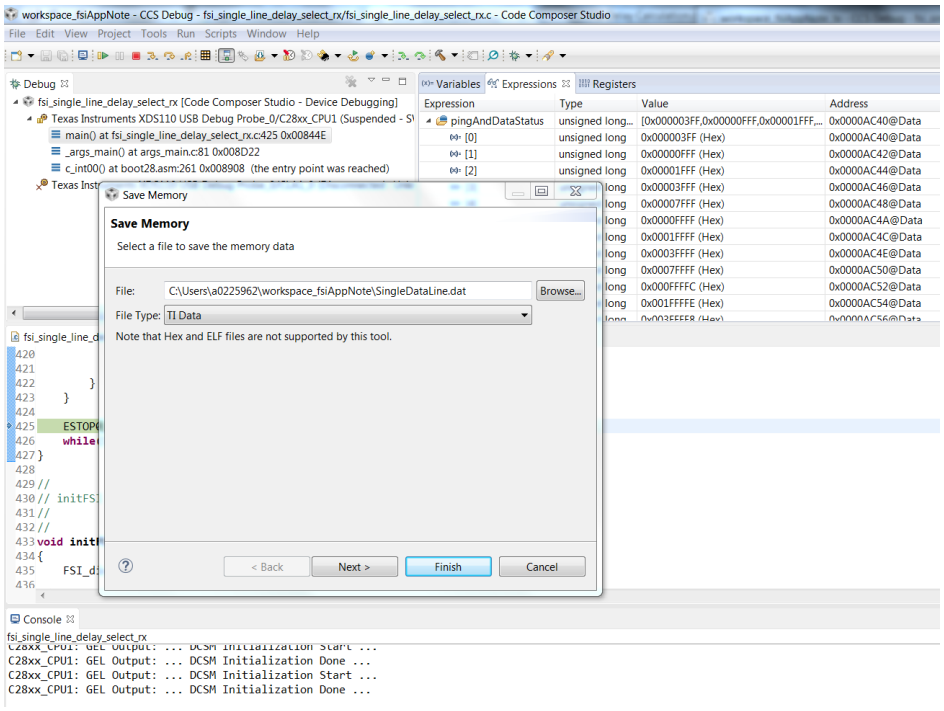

Figure 13. Save Single Data Line Communication Results – File Name

5. Click “Next” to enter the address, size, and format of the data to save.

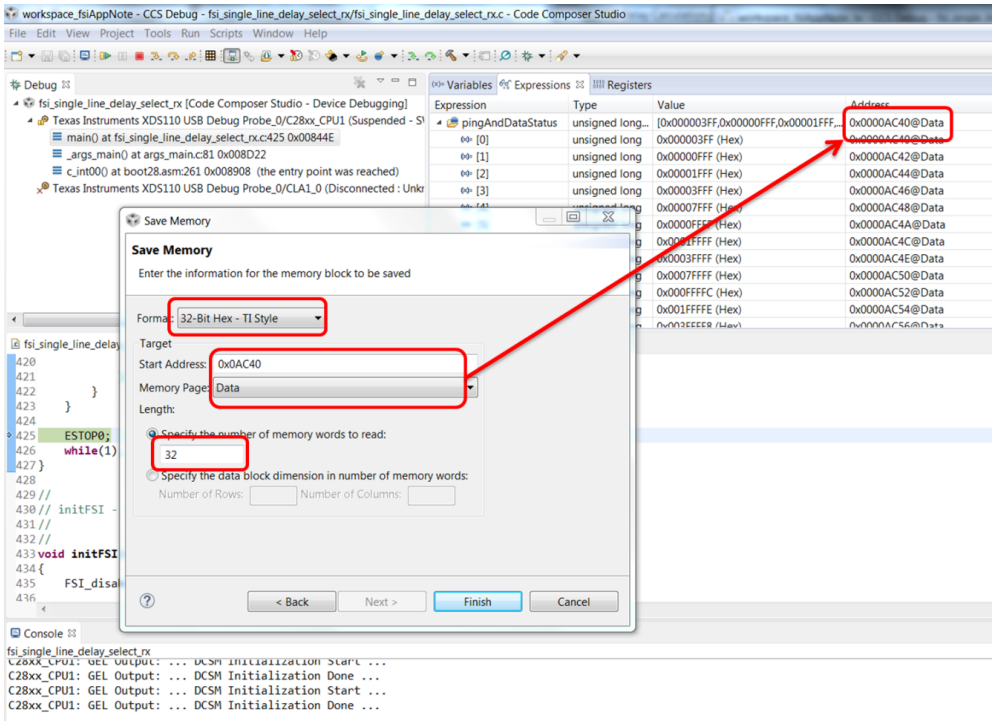

Figure 14. Save Single Data Line Communication Results – Address, Size And Format

6. The saved file can now be used as an input for the FSI Skew Compensation Utility. Launch the utility and select “Single Line RX0 vs. CLK”.

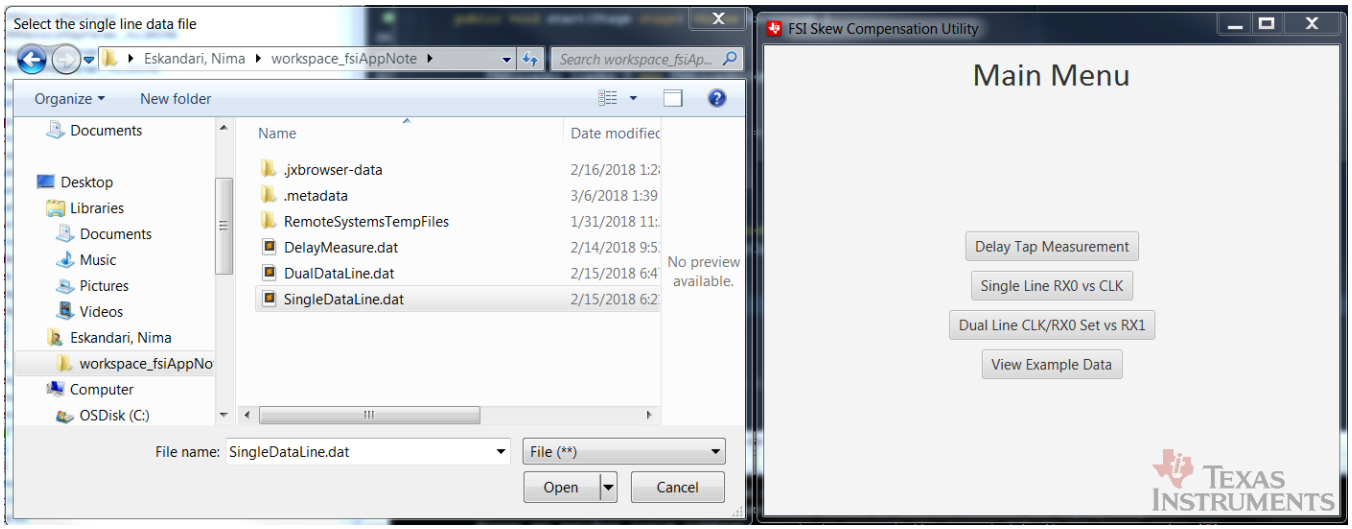

Figure 15. FSI Skew Compensation Utility - Single Line RX0 vs CLK

7. After the input file is selected, the utility graphs the result. This graph is shown in Figure 16. The green symbols show the RXD0 and RXCLK delay values at which the communication was successful. The red symbols show the delay values at which the communication fails. The blue symbols show the best delay values which will allow the most margins for data to pass.

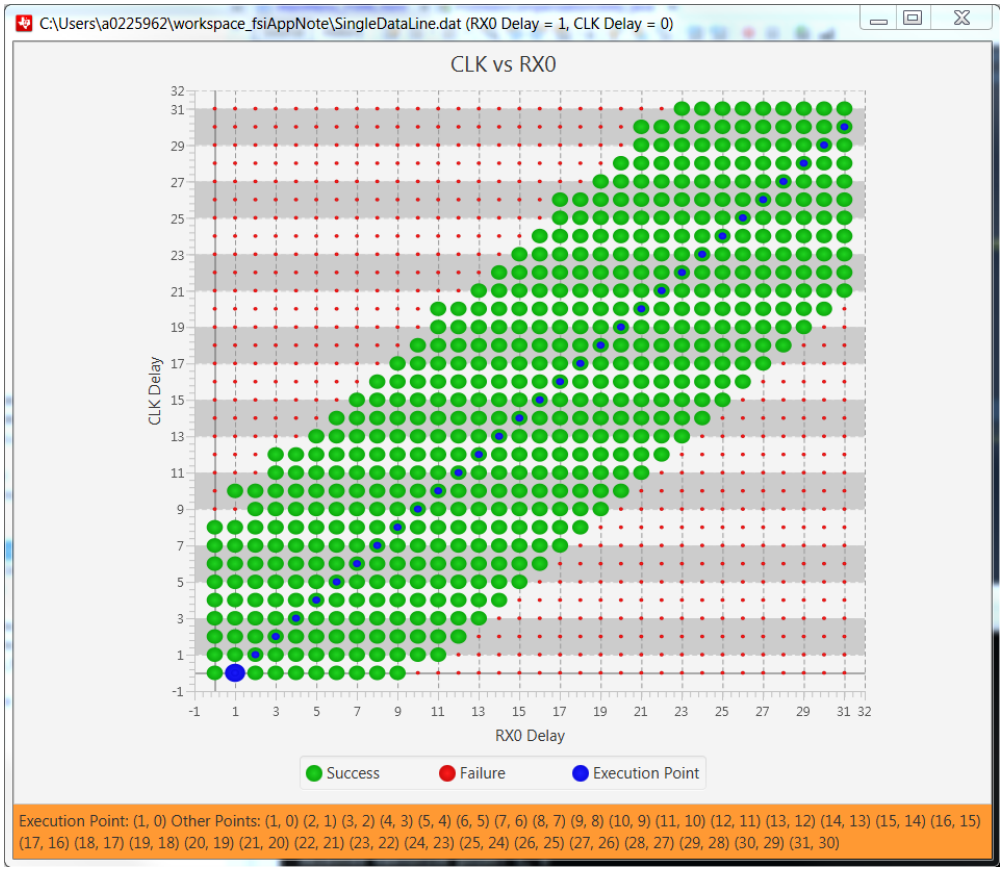

Figure 16. FSI Skew Compensation Utility - Single Line Data Result

8. The area where data passes can be shifted to the left or right in different operating conditions. All different scenarios can be view in Section 6.

5 Visualizing the FSI Two Data Line Skew Compensation

The FSI module can be operated using two data lines: RXD0 and RXD1. In this mode, the RXCLK, RXD0 and RXD1 act as clock and data lines. To compensate for the skew introduced by board, isolators or any other factor, the skew compensation block must be calibrated correctly. The skew compensation is more complicated for two data line communication.

The idea behind the skew compensation for two data line communication is the same as single data line communication, with the only difference being more iteration introduced by RXD1. There are 32 x 32 x 32 different combinations of delay line configurations. The example projects provided in this application report allow you to capture the delay line configurations at which ping and data transmission are successful. The FSI Skew Compensation utility can use the result of the example projects to visualize the captured data.

5.1 FSI Dual Line Skew Compensation Example Software

The fsi_ex14_dual_line_delay_select_rx and fsi_ex14_dual_line_delay_select_tx projects can be used to visualize how the FSI delay line control can be used to compensate for different amounts of skews in the system. The example uses three GPIOs for FSI RX (RXD0, RXD1 and RXCLK) and three GPIOs for FSI TX (TXD0, TXD1 and TXCLK). The communication between the two devices is the same as the single data line but instead of one lane, two lanes are used. The communication between the two devices is verified at all 32 x 32 x 32 (RXD0, RXD1, RXCLK) different delay line configurations.

1. The two projects must be imported into CCS. It is recommended to have two different instances of CCS open and import each project into a different workspace. This way the user can debug both projects at the same time. The fsi_dual_line_delay_tx project must be started first. So after importing the project, build and run the example.
2. The fsi_dual_line_delay_rx project must be started. It takes some time for this example to finish. The CPU halts when the example is finished and the result is saved in variable named "pingAndDataStatus", which is a 32 x 32 element array. The first index of the array 0-31 represents the RXCLK delay from 0 to 31. The second index of the array 0-31 represents the RXD0 delay from 0 to 31. The value of each pingAndDataStatus[x][y], where x is the RXCLK delay and y is the RXD0 delay, represents the bit-packed status of ping and data transmission for the RXD1 delay from 0-31. Therefore, pingAndDataStatus[12][10] = 0x00001FF0 means that data and ping transmission was successful for RXCLK delay = 12, RXD0 = 10, and RXD1 delay = 4-12. Figure 17 shows two instances of CCS, one with fsi_dual_line_delay_tx and the other with fsi_dual_line_delay_rx, in debug perspective.

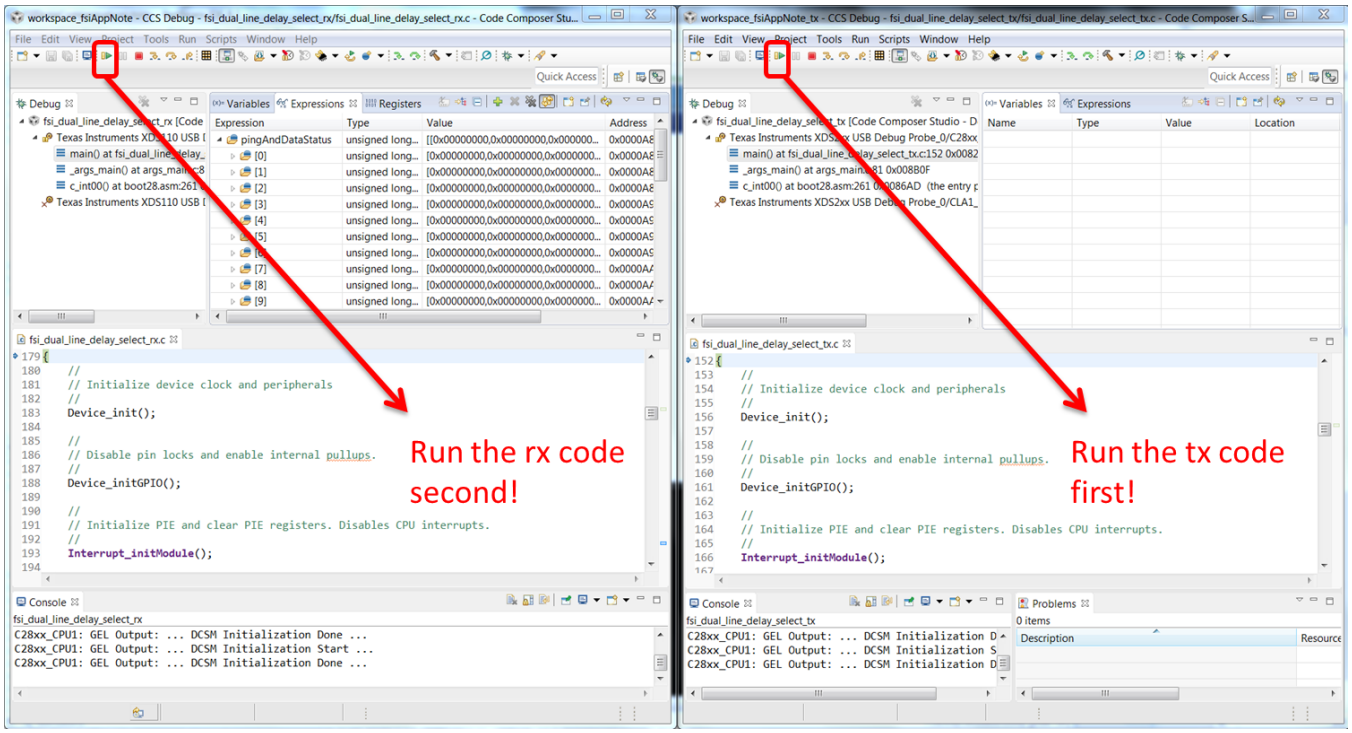

Figure 17. fsi_dual_line_delay_tx And fsi_dual_line_delay_rx Example Software in Debug Perspective

3. After both example projects are running, the fsi_dual_line_delay_rx project will halt. It will take some time for the software to finish gathering the results. Remember it is verifying all 32x32x32 delay line configurations. The fsi_dual_line_delay_tx will not halt. It will continue to send pings, which can be ignored for this example's purpose. The figure below shows the fsi_dual_line_delay_rx project halted and the gathered result in "pingAndDataStatus" is shown in the "Expressions" window.

   Now the results captured in this example can be used by FSI Skew Compensation Utility to generate graphs to visualize the data. In order to save the "pingAndDataStatus", go to Tools > Save Memory.

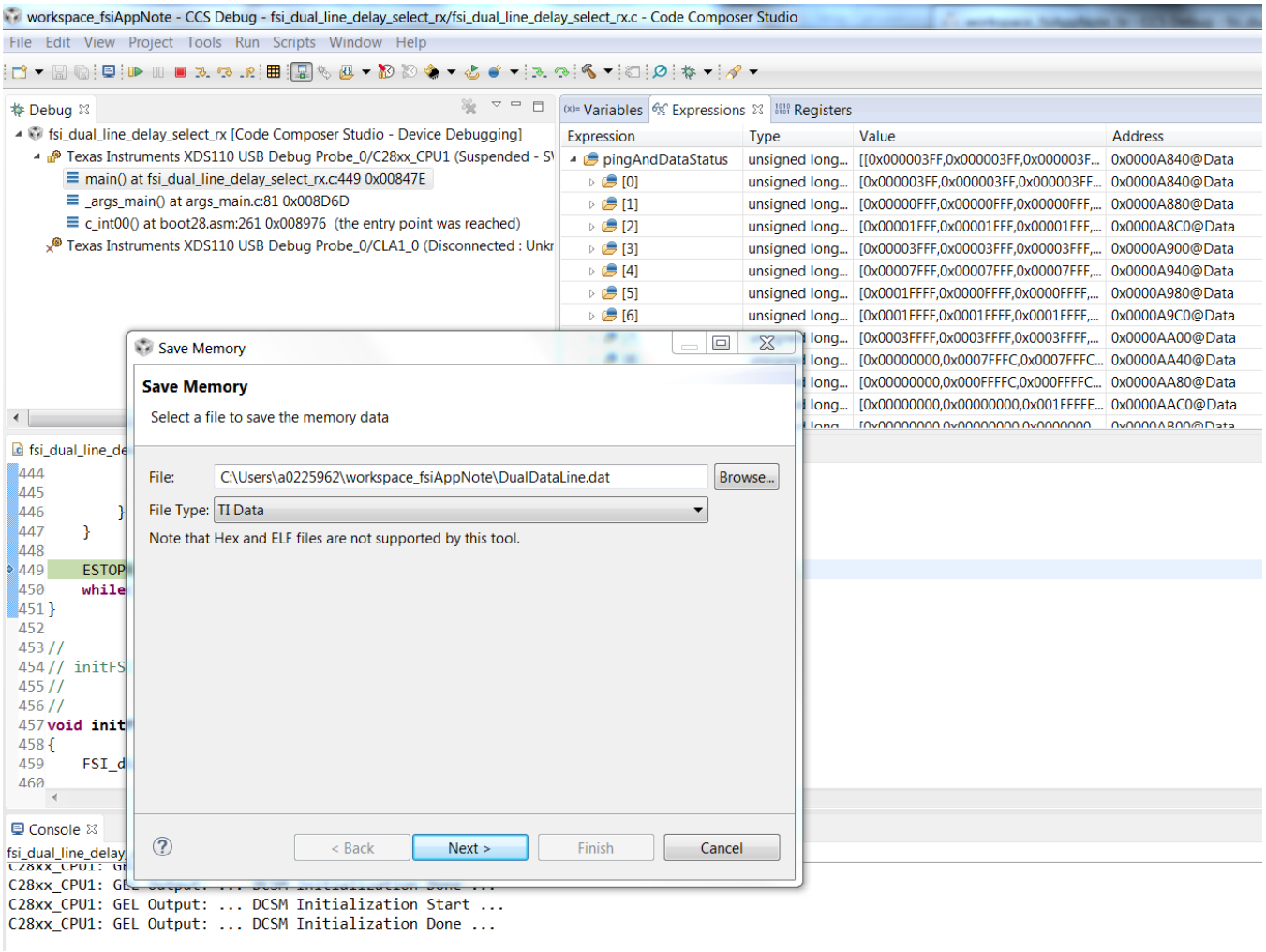

Figure 18. Save Dual Data Line Communication Results – File Name

4. Click “Next” to enter the address, size, and format of the data to save.

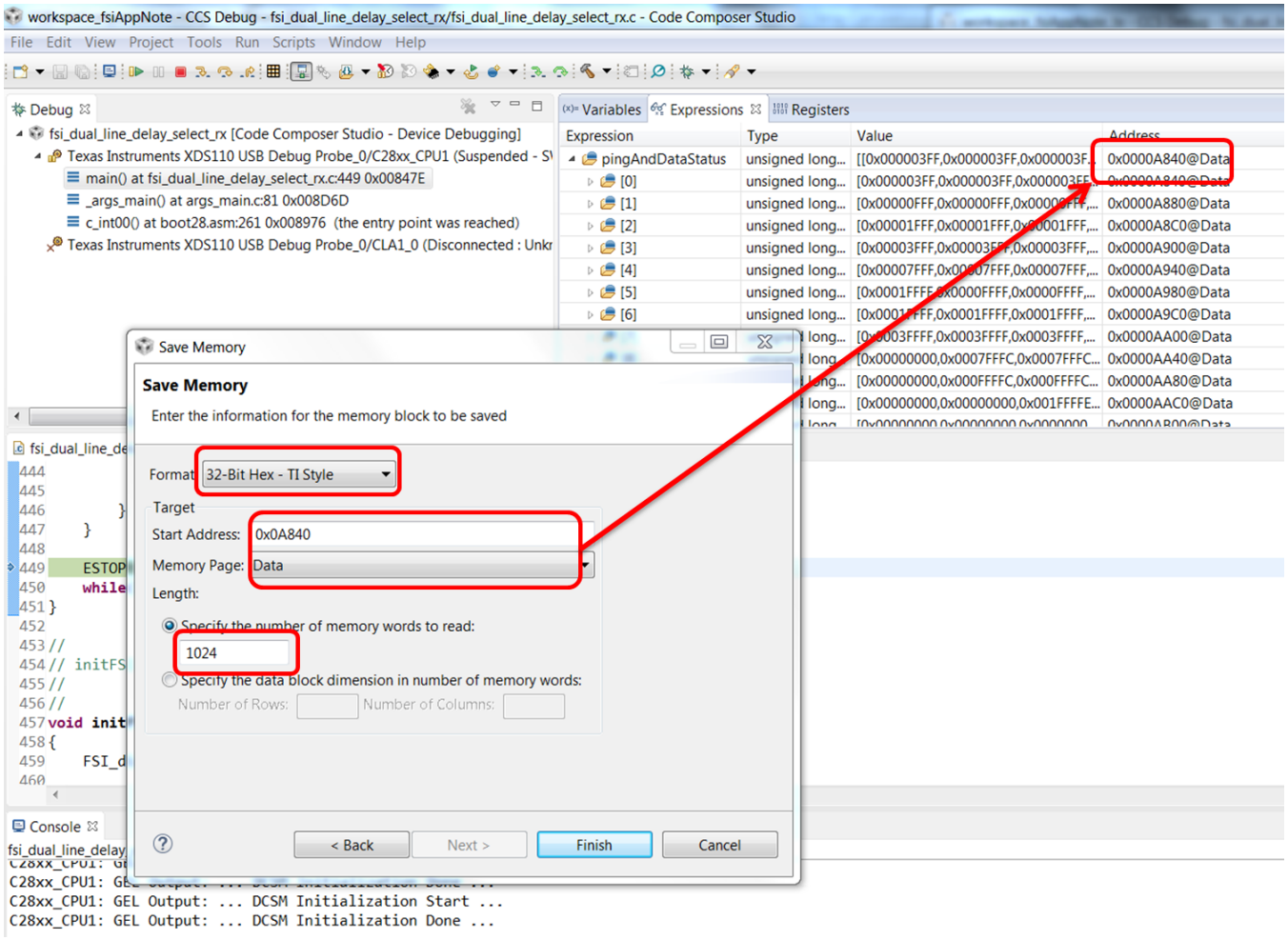

Figure 19. Save Dual Data Line Communication Results – Address, Size And Format

5. The saved file can now be used as an input for the FSI Skew Compensation Utility. Launch the utility and select “Dual Line RX0/CLK Set vs. RX1”. The utility will prompt you for both the single data line results and the dual data line results. If the user inputs both single data line and dual data line results, the utility will graph RXD1 vs RX0/CLK set at the optimal execution points found in single data line results. If only the dual data line result is inputted, the utility will graph RXD1 vs RXCLK at all RXD0 delays. Figure 20 shows FSI Skew Compesation Utility while selecting the dual data line results.

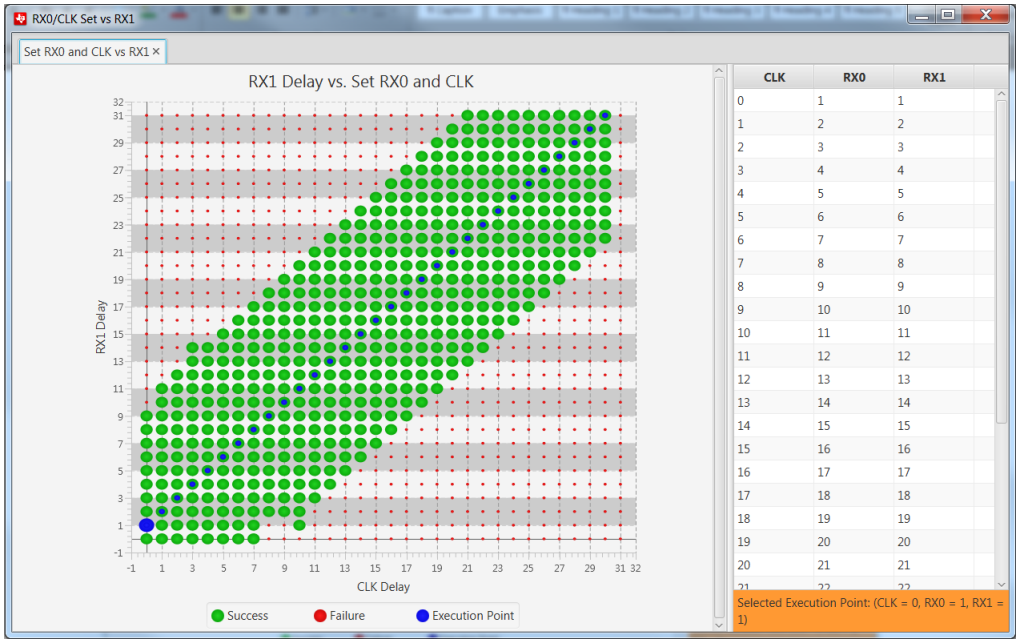

Figure 20. FSI Skew Compensation Utility - Dual Line CLK vs RX1 at Single Line Execution Points

Figure 21 shows an example that visualizes the results of the dual data line communication at all delay configurations.

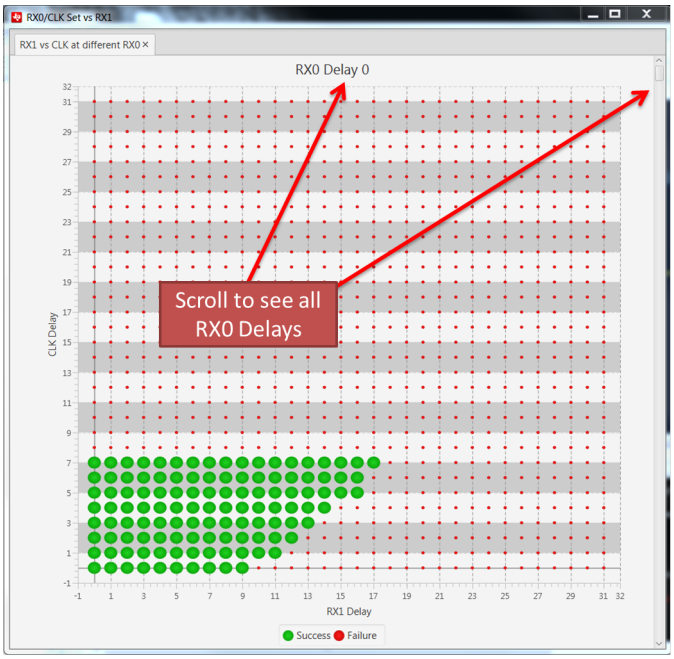

Figure 21. FSI Skew Compensation Utility - Single Line CLK vs RX1 at all RX0 Delay Values

6 Finding the Optimal FSI Execution Point

This section analyzes how the best delay line configuration for single and dual data line communication can be selected. In order to find the ideal FSI execution point, it is assumed that the delay elements on all three external signal inputs are identical. This means FSI RXD0, RXD1 and RXCLK are delayed by the same amount when the same value is written to RX_DLY_LINE_CTRL for each of the signals. There are eight total scenarios when comparing RXD0/RXD1 vs. RXCLK, assuming identical delay elements across all three external signals.

These eight scenarios are shown in Figure 22 through Figure 29. The green symbols represent a pass while the red symbols represent a failure in data transmission. The blue symbols represent the best execution points. The blue symbols always generate a line with a slope of one due to the assumption that the delay elements on all three external signals are identical.

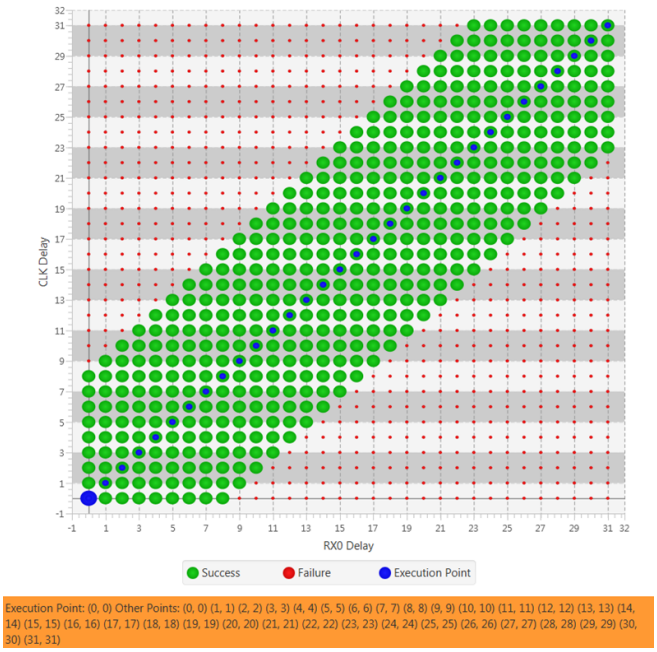

Figure 22. Scenario 1 FSI Execution Points

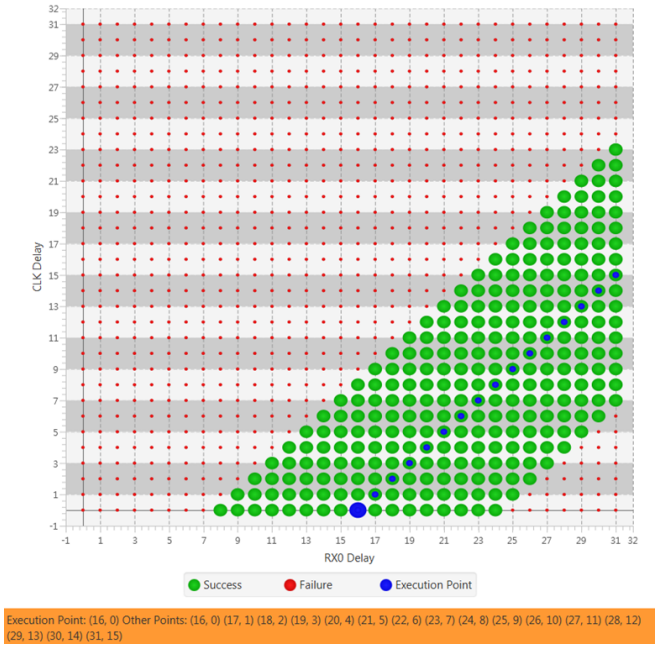

Figure 23. Scenario 2 FSI Execution Points

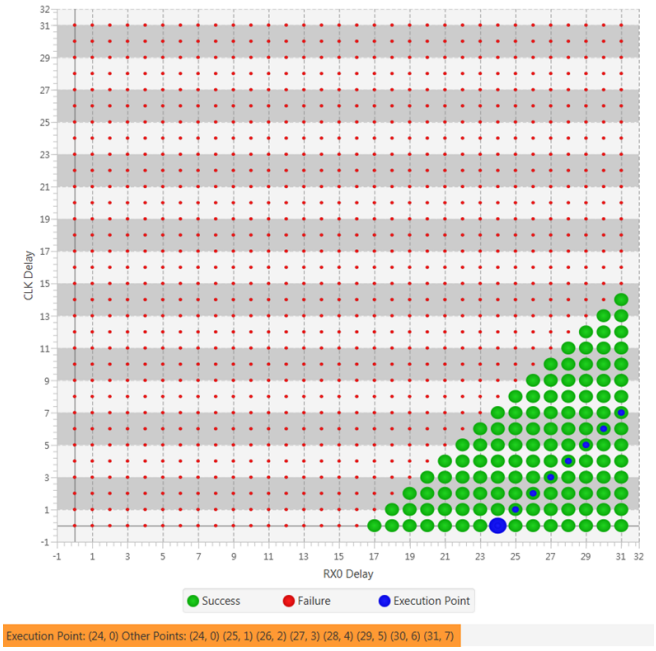

Figure 24. Scenario 3 FSI Execution Points

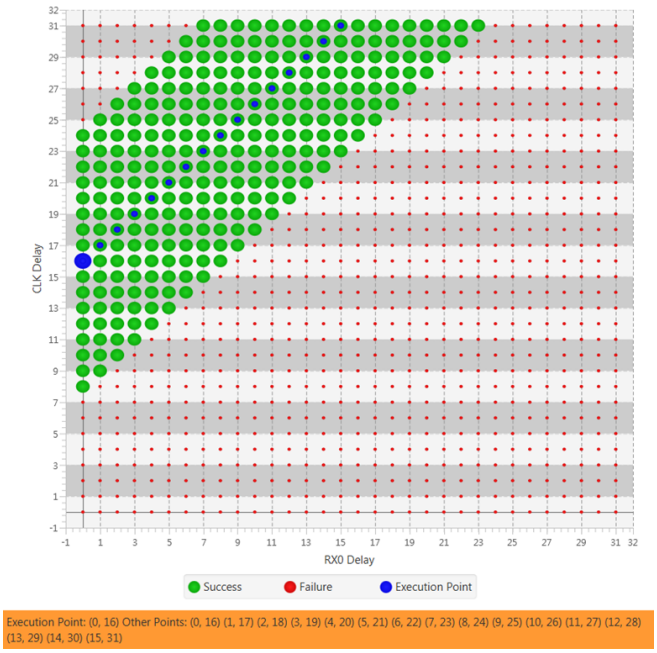

Figure 25. Scenario 4 FSI Execution Points

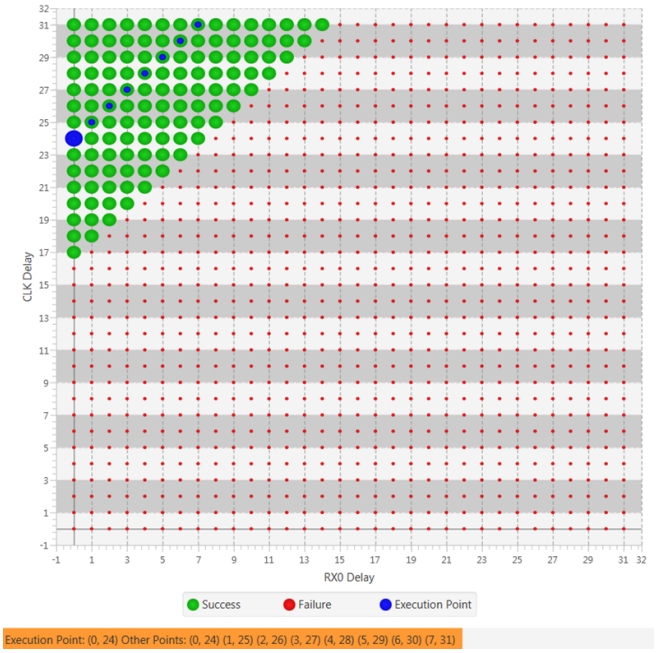

Figure 26. Scenario 5 FSI Execution Points

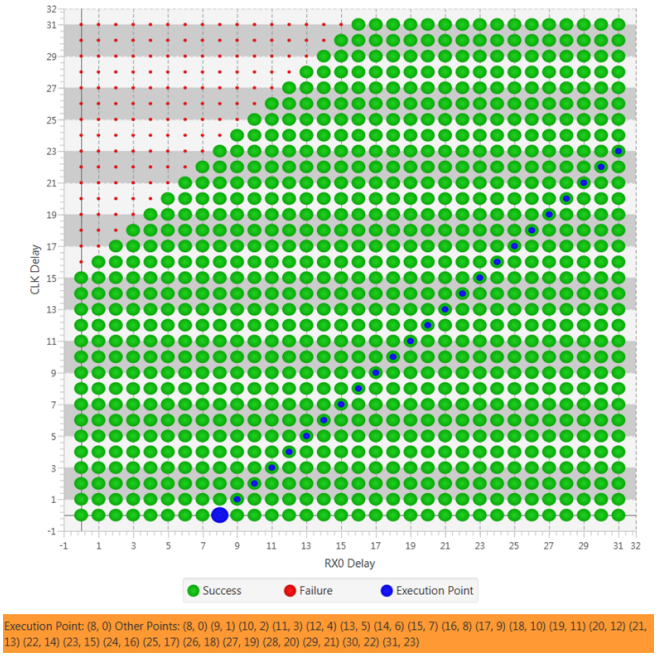

Figure 27. Scenario 6 FSI Execution Points

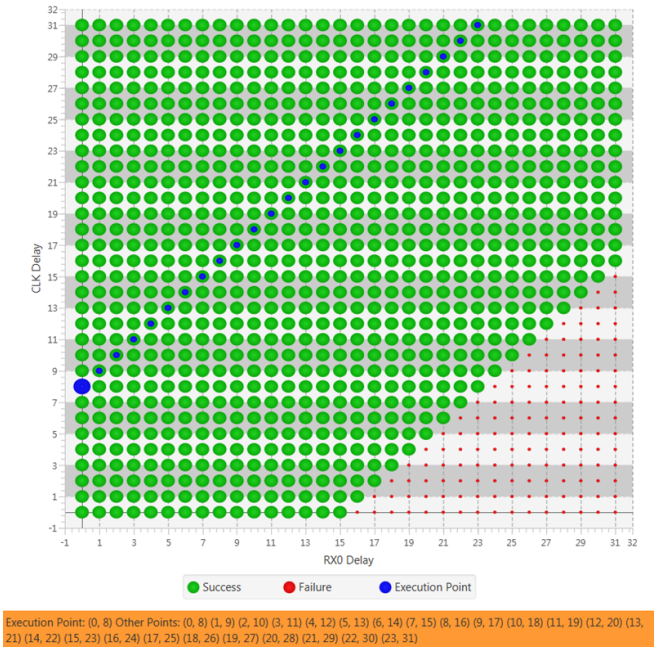

Figure 28. Scenario 7 FSI Execution Points

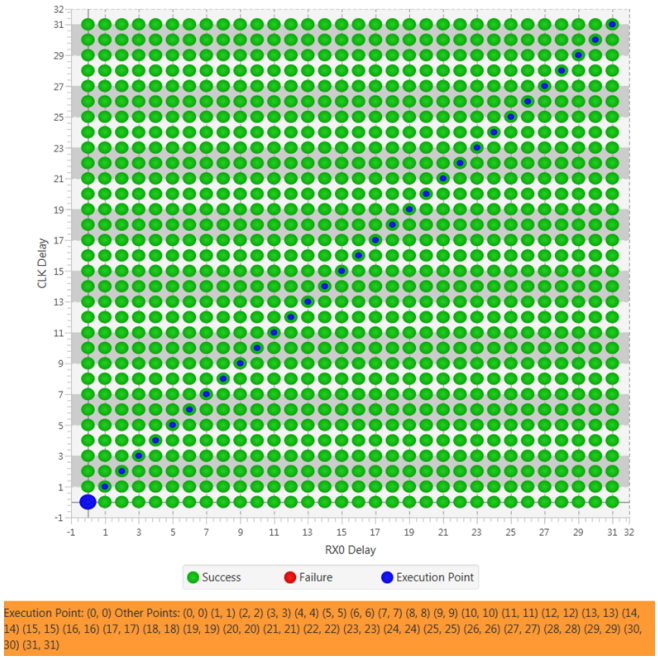

Figure 29. Scenario 8 FSI Execution Points

For each of the scenarios, the best execution point is selected to have equal delays margins for RXD0, RXD1 (if used), and RXCLK. After examining Figure 22 through Figure 29, it is clear that not all 32 x 32 configurations must be examined to find the optimal execution point. The only configurations that must be checked are the ones that lie on the x or y axis. Here is how each scenario can be detected:

1. Find the number of times the data switched between different statuses (failure to success or success to failure) on the x-axis
2. Find the number of times the data switched between different statuses (failure to success or success to failure) on the x-axis
3. Data passing at (0,0) will always count as an intercept on both x and y axis
4. The scenarios are then identified as described below:
   a. Scenario 1: two x-intercepts, two y-intercepts, and data passes at (0,0)
   b. Scenario 2: two x-intercepts, zero y-intercepts
   c. Scenario 3: one x-intercept, zero y-intercepts
   d. Scenario 4: zero x-intercepts, two y-intercepts
   e. Scenario 5: zero x-intercepts, one y-intercept
   f. Scenario 6: one x-intercept, two y-intercepts, and data passes at (0,0)
   g. Scenario 7: two x-intercepts, one y-intercept, and data passes at (0,0)
   h. Scenario 8: one x-intercept, one y-intercept, and data passes at (0,0)

The algorithm to find the optimal execution point is described below. The first step is to find the optimal execution point for single data line communication. In order to find this optimal point, the two devices attempt to establish a communication link. The device under calibration attempts to receive pings while iterating through the following two delay line configurations.

- Y-axis of RXD0 vs. RXCLK
  - RXD0 delay set to zero; RXCLK delay iterates from 0 to 32
- X-axis of RXD0 vs. RXCLK
  - RXCLK delay set to zero; RXD0 delay iterates from 0 to 32

The results of the 64 different configurations above are stored in two 32-bit variables. These two variables are used as x-axis and y-axis to identify which scenario best represents the results. After identifying the scenario, the optimal execution point for single data line transmission can be selected. If the skew compensation is to be done for only single data line transmission, the calibration process is terminated if dual data line calibration is required. The two devices will switch to dual data line mode and attempt to establish a communication link, knowing that the single data line execution point comes into effect when choosing which RXD1 delay and RXCLK delay configurations to test.

- X-axis of RXCLK vs. RXD1
  - RXD1 delay set to zero; RXCLK delay and RXD0 delay iterate from single line execution point to 32, incrementing both RXCLK delay and RXD0 delay by one every time
- Y-axis of RXCLK vs. RXD1
  - RXD0 and RXCLK delay set to single line execution point; RXD1 iterates from 0 to 32

After identifying the scenario, the optimal execution point for dual data line transmission can be selected. Example software is available to showcase how finding the FSI execution point is implemented. The software checks whether single data line or dual data line skew compensation is required. Then, the software follows the steps described in this section to validate whether a certain delay configuration will pass or fail. The steps shown in Figure 30 are executed on the two devices.

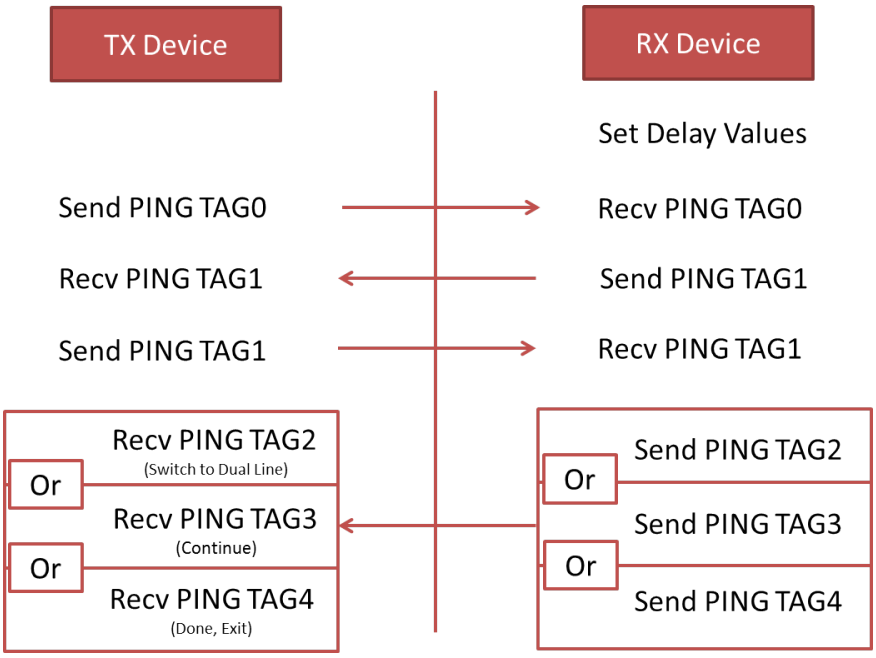

Figure 30. FSI Establishing Communication Link for Optimal Execution Point

The last ping sent from the calibrating device can have three different types of tags. If TAG2 is sent, the TX device (device that is not calibrating) switches to transmitting and receiving pings using dual data line communication. If TAG3 is sent, no change is needed. If TAG4 is sent, then the calibration process is done and both devices will stop transmitting and receiving pings.

6.1 Finding the Optimal FSI Execution Point Software Example

Example software is available to show how the FSI execution point can be selected by following the steps stated in this section. The fsi_ex15_find_optimal_delay_device2 and fsi_ex15_find_optimal_delay_device1 projects show how the FSI execution point is calculated. The example uses three GPIOs for FSI RX (RXD0, RXD1 and RXCLK) and three GPIOs for FSI TX (TXD0, TXD1 and TXCLK). The communication between the two devices are verified using 96 to 128 (RXD0, RXD1, RXCLK) different delay line configurations.

1. The two projects must be imported into CCS. It is recommended to have two different instances of CCS open and import each project into a different workspace. This way the user can debug both

projects at the same time. The fsi_ex15_find_optimal_delay_device1 project must be started first. So after importing the project, build and run the example. Finally, the fsi_ex15_find_optimal_delay_device2 project must be started. The CPU will halt when the example is finished and the RX_DLY_LINE_CTRL is set.

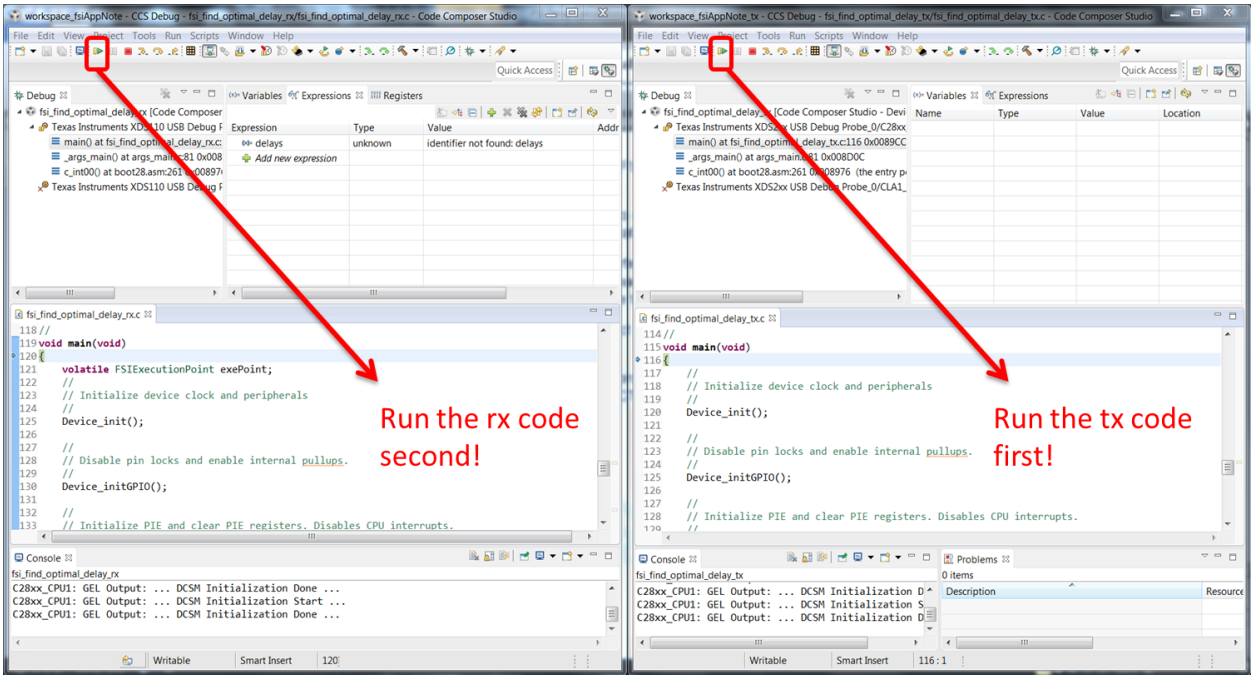

Figure 31. fsi_ex15_find_optimal_delay_device2 And fsi_ex15_find_optimal_delay_device1 Projects in Debug Perspective

2. The optimal execution point is a structure which contains a Boolean component named “Valid”. If “Valid” is false, then the software could not find an optimal execution point.

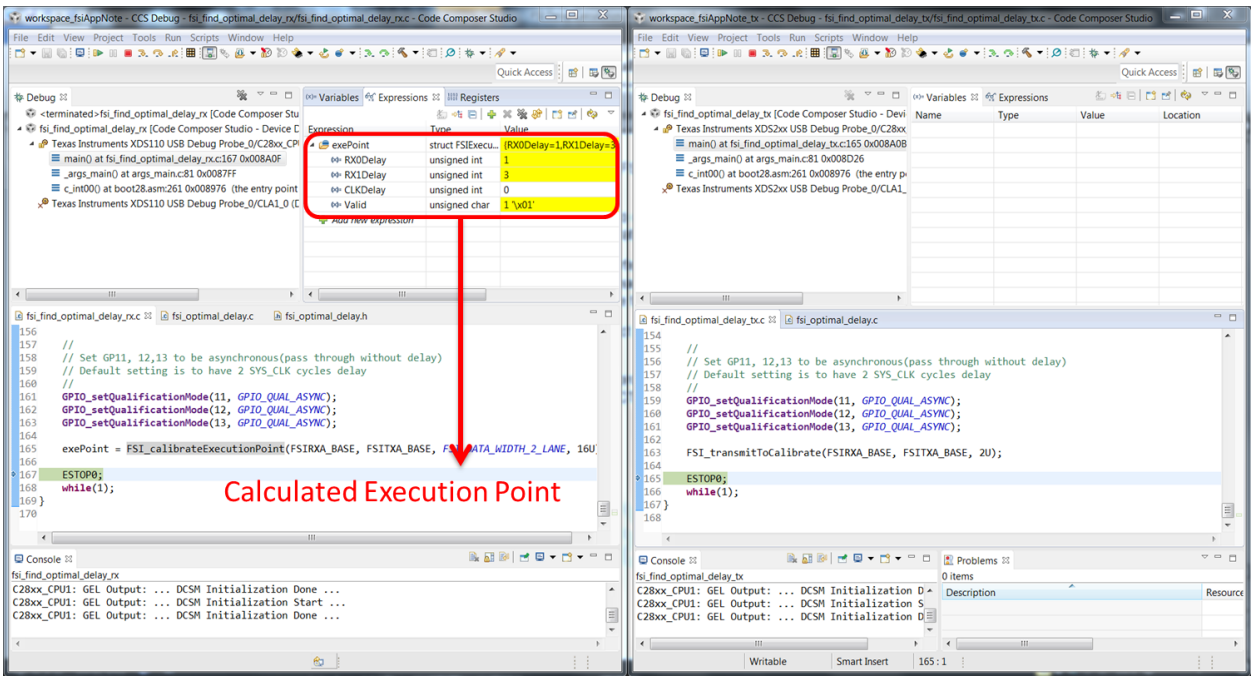

Figure 32. FSI Execution Point Found by fsi_ex15_find_optimal_delay_device2 Project

7 Summary

The Fast Serial Interface (FSI) module is a serial communication peripheral capable of reliable and robust high-speed communications. When using FSI, it may be required to use the integrated skew compensation block to overcome any introduced delays or skews, specific to your application. The algorithm and source code provided in this application report allows you to utilize the skew compensation block and choose the ideal setting for their application.

8 References

For more information on the FSI module on a specific C2000 devices, see the device-specific data sheet and technical reference manual (TRM).

This application report was written using the TMS320F28004x family of devices. The data sheet and TRM used for this application report are available below,

- Texas Instruments: *TMS320F28004x Piccolo™ Microcontrollers data sheet*
- Texas Instruments: *TMS320F28004x Piccolo Microcontrollers Technical Reference Manual*
- Additional support is provided by:
- TI E2E™ Community

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265