

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

1	Introduction	3
2	Software Example	3
3	FSI Delay Element Measurement	4
4	Visualizing the FSI Single Data Line Skew Compensation	11
5	Visualizing the FSI Two Data Line Skew Compensation	15
6	Finding the Optimal FSI Execution Point	20
7	Summary	25
8	References	25

List of Figures

1	Import All Example Projects	4
2	Build and Launch Debug Session	5
3	fsi_ex12_delay_tap_measurement in Debug Mode	6
4	Measured Delay Elements in Expressions Window	7
5	Code Composer Studio Save Memory Option	8
6	fsi_ex12_delay_tap_measurement Save Memory Options	9
7	FSI Skew Compensation Utility - Main Menu	10
8	FSI Skew Compensation Utility - Delay Measurement	10
9	FSI Point to Point Connection	11
10	Delay Line Configuration Verification Flow	12
11	fsi_single_line_delay_tx And fsi_single_line_delay_rx Example Software in Debug Perspective	12
12	fsi_single_line_delay_rx Halted When Finished	13
13	Save Single Data Line Communication Results – File Name	13
14	Save Single Data Line Communication Results – Address, Size And Format	14
15	FSI Skew Compensation Utility - Single Line RX0 vs CLK	14
16	FSI Skew Compensation Utility - Single Line Data Result	15
17	fsi_dual_line_delay_tx And fsi_dual_line_delay_rx Example Software in Debug Perspective	16
18	Save Dual Data Line Communication Results – File Name	17
19	Save Dual Data Line Communication Results – Address, Size And Format	18
20	FSI Skew Compensation Utility - Dual Line CLK vs RX1 at Single Line Execution Points	19



21	FSI Skew Compensation Utility - Single Line CLK vs RX1 at all RX0 Delay Values	19					
22	Scenario 1 FSI Execution Points	20					
23	Scenario 2 FSI Execution Points	20					
24	Scenario 3 FSI Execution Points	20					
25	Scenario 4 FSI Execution Points	20					
26	Scenario 5 FSI Execution Points	21					
27	Scenario 6 FSI Execution Points	21					
28	Scenario 7 FSI Execution Points	21					
29	Scenario 8 FSI Execution Points	21					
30	FSI Establishing Communication Link for Optimal Execution Point	23					
31	fsi_ex15_find_optimal_delay_device2 And fsi_ex15_find_optimal_delay_device1 Projects in Debug Perspective	24					
32	FSI Execution Point Found by fsi_ex15_find_optimal_delay_device2 Project	24					
	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.

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.

Table 1. Top-Level Folder Description

Introduction



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.

 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

FSI Delay Element Measurement

www.ti.com

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).

😵 workspace_fsiAppNote - CCS Debug - fsi_delay_tap_measurement/fsi_delay_tap_measurement.c - 0	Code Composer Studio	No. of Concession, Name	- (100 K K)
File Edit View Project Tools Run Scripts Window Help			
☆ ▾ ::::::::::::::::::::::::::::::::::	Ø 🎋 🕶 🔗 🗸		
	(x)= Variables 😚 Expr	ressions 🖾 🔤 Register	'S
# Isi_delay_tap measurement [Code Composer Studio - Device Debugging]	Expression	Tune	Value
Texas Instruments XDS2xx USB Debug Probe_0/C28xx_CPU1 (Suspended - SW Breakpoint)	達 delays	float[32][10]	[[0.0,0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0,0.0,0]
main() at fsi_delay_tap_measurement.c:105 0x0081F7	Þ 🥏 [0]	float[10]	[0.0,0.0,0.0,0.0,0.0]
_args_nain() at args_main.c:81 0x00895A	Þ 🥏 [1]	float[10]	[0.0,0.0,0.0,0.0,0.0]
c_int00) at boot28.asm:261 0x00860A (_c_int00 does not contain frame information)	Þ 🥏 [2]	float[10]	[0.0,0.0,0.0,0.0,0.0]
🔎 Texas Inst <mark>u</mark> ments XDS2xx USB Debug Probe_0/CLA1_0 (Disconnected : Unknown)	Þ 🥏 [3]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [4]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [5]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [6]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [7]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [8]	float[10]	[0.0,0.0,0.0,0.0,0.0]
Run the example	Þ 🥏 [9]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	Þ 🥏 [10]	float[10]	[0.0,0.0,0.0,0.0,0.0]
	r111 🦛	float[10]	10000000001
<pre>101// 102// Main 103// 104 void main(void) 105 [106 // 107 // Initialize device clock and peripherals 108 // 109 Device_init(); 110 111 // 112 // Disable pin locks and enable internal pullups. 113 // 114 Device_initGPIO(); 115 116 //</pre>	d "delays" to	o Expressior	ns window
<			
Console 🛙			
fsi_delay_tap_measurement			
C28xx_CPU1: GEL Output: DCSM Initialization Done			

C28xx_CPU1: GEL Output: ... DCSM Initialization Start .. C28xx_CPU1: GEL Output: ... DCSM Initialization Done ...

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.

workspace_fsiAppNote - CCS Debug - fsi_delay_tap_measurement/fsi_delay_tap_measurement.c - C	ode Composer Studio			
File Edit View Project Tools Run Scripts Window Help				
□ ▼ 🔛 🕼 💷 🕨 💷 🗷 👁 🗤 🖩 🗐 % 😃 ▼ 🔊 % ♥ 🍲 🐇 ♥ (3. ৫) % ▼ (3)	Ø ☆ ▼ 🔗 ▼			
🏘 Debug 🛛 🍇 ▽ 🗖 🗖	(x)= Variables	ressions 🛛 🕮 Register	S	
# Si_delay_tap_measurement [Code Composer Studio - Device Debugging]	Ex lession	Туре	Value	Address
Texas Instruments XDS2xx USB Debug Probe_0/C28xx_CPU1 (Suspended - SW Breakpoint)	🔎 🥏 delays	float[32][10]	[[-0.084777832,0.0847625732,0.3390	0x00010080@Data
main() at fsi_delay_tap_measurement.c:238 0x0082F6	Þ 🥏 [0]	float[10]	[-0.084777832,0.0847625732,0.33907	0x00010080@Data
_args_main() at args_main.c:81 0x00895A	Þ 🥭 [1]	float[10]	[1.18673706,0.932434082,1.01720428	0x00010094@Data
c_int00() at boot28.asm:261 0x00860A (_c_int00 does not contain frame information)	Þ 🥭 [2]	float[10]	[1.44104004,1.27150726,1.44104767,	0x000100A8@Data
📌 Texas Instruments XDS2xx USB Debug Probe_0/CLA1_0 (Disconnected : Unknown)	Þ 🥭 [3]	float[10]	[1.69534302,1.86489105,1.61058044,	0x000100BC@Data
	Þ 🥭 [4]	float[10]	[1.94965363,1.78012085,1.94965363,	0x000100D0@Data
	Þ 🥭 [5]	float[10]	[2.2039566,2.37348938,2.11919403,2	0x000100E4@Data
	Þ 🥏 [6]	float[10]	[2.62779236,2.45825195,2.54302979,	0x000100F8@Data
	Þ 🥏 [7]	float[10]	[2.96687317,2.96687317,2.96686554,	0x0001010C@Data
	Þ 🥏 [8]	float[10]	[3.22116852,2.96686554,3.13640594,	0x00010120@Data
	Þ 🥏 [9]	float[10]	[3.56023407,3.56023407,3.22116852,	0x00010134@Data
	Þ 🥏 [10]	float[10]	[3.81452942,3.56023407,3.81454468,	0x00010148@Data
	. k 🛋 mm	float[10]	12 80020725 2 80020725 2 81452705	0v0001015C@Data
<pre>233 { delays[tap][sample] = (cap2cap1DifferenceNanoSecond[AND_S: 235 } 236 237 } 238 ESTOP0; 239 while(1); 240} 241 242void initFSI() 243 { 244 FSI_performTxInitialization(FSITXA_BASE, 40U); 245 FSI_performRxInitialization(FSIRXA_BASE); The 246 247 // 248 // Set internalLoopback mode 249 //</pre>	GNAL][sample] -	ap2cap1Difference	eNanoSecond[OR_SIGNAL][sample])/2;
E Console 🛛				
C20xx CDU1, CEL Output: DCSM Initialization Done				
C28xx_CPU1: GEL Output: DCSM Initialization Done C28xx CPU1: GEL Output: DCSM Initialization Done				

Figure 4. Measured Delay Elements in Expressions Window



FSI Delay Element Measurement

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



FSI Delay Element Measurement

www.ti.com

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.

V FSI Skew Compensation Utility	
Main Menu	
Delay Tap Measurement	
Single Line RX0 vs CLK	
Dual Line CLK/RX0 Set vs RX	1
View Example Data	
	INSTRUMENTS

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.



Visualizing the FSI Single Data Line Skew Compensation



Figure 10. Delay Line Configuration Verification Flow

- 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.

workspace_fsiAppNote - CCS Debug - fsi_single_line_delay_select_rx/fsi_single_line_delay.	select_rx.c - Code Composer 👄 💷	X	🖉 🐨 workspace_fsiAppNote_tx - CCS Debug - fsi_single_line_delay_select_tx/fsi_single_line_delay_select_tx.c - Code Compos 👄	D X
File Edit View Project Tools Run Scripts Window Help		_	File Edit View Project Tools Run Scripts Window Help	
🗂 🕶 📖 😳 🖝 🗉 🔳 🔍 👁 Le 🖩 😨 N 🕹 🕶 🖄 🖄 🔶 🖌 🗉 🔿 🔿	• < < 0 \$		I = 1 = 10 = 10 = 10 = 10 = 10 = 10 = 10	
	Quick Access	-	Quick Access	1 II
🕸 Debug 🛙 👘 🗸 🗢 🗖 👓 Variables 🛠 Expre	sions 12 III Registers		🛊 Debug 🛙 👘 🗸 👘 🗆 💀 Variables 🖬 🛠 Expressions 🔊 🗠 🖬 🖯 11 년 6	\$
# Isi_single_line_delay_select_rx [Code Composer Studio - Devi	🖉 🦛 🖻 🌩 🗶 🎇 🖬 🖬 👘	¢, ₹	A 😳 fsi_single_line_delay_select_tx (Code Composer Studio - Name Type Value Loca*	tion
Prevas Instruments XDS110 USB Debug Probe_0/C28xx_CPI Expression	Type Value	*	Texas Instruments XDS2xx USB Debug Probe_0/C28xx	
Imanity at fix jineje. Jime, delay, select, rxxx433 000487 Jime, Jim	us unsigned long (bx000003FF, bx0000 unsigned long bx00003FF (Hex) unsigned long bx00003FF (Hex) unsigned long bx00003FF (Hex) unsigned long bx00003FF (Hex) unsigned long 0x00007FF (Hex) unsigned long 0x00007FF (Hex) unsigned long 0x0003FFF (Hex)	IOFF	Teas Intureets XOS2x USB Debug Probe, 0/CIA1,	
<	unsigned long 0x0007FFFF (Hex)	~	· · · · · · · · · · · · · · · · · · ·	
R fsi single line delay select ry c 22				
420 pingAndDataStatus[delayTapCLK] = (FSI PASS <<	delayTap		I tsi_single_line_delay_select_bcc ==	
221) 223) 223) 224) 225 STOPNE 227 // 225 STOPNE 228 // 227 //	lso sendy FLUSH	8	<pre>109 // Initialize device clock and peripherals 101 // Initialize device clock and peripherals 102 // Perice_init(); 103 104 105 // Olsable pin locks and enable internal gullags. 105 105 // Olsable pin locks and enable internal gullags. 105 105 // Olsable pin locks and enable internal gullags. 105 105 // Olsable pin locks and enable internal gullags. 105 // Olsable pin locks and enable pin</pre>	Ш
432/// 432/// 432// 432// 432// 432// 432// 432// 432// 432// FSI disableRxInternalLoopback(FSIRXA BASE);	us (Pass or Fail)	-	<pre>1 160 // Initialize PIE and clear PIE registers. Disables CPU interrupts. 102 // Initialize PIE and clear PIE registers. Disables CPU interrupts. 103 Interrupt_initModule(); 104</pre>	
				P
Console II	in 🔐 📴 🖻 🖬 👻 🗸 🗸		Console 12 lik 🔤 🖙 📑 🖬 🕶 🐨 📑 🔮 Problems 12	0 0 E
fsigngle.lme.deMsy.gelect.x C28xx_CPU1: GEL Output: DCSM Initialization Done C28xx_CPU1: GEL Output: DCSM Initialization Start C28xx_CPU1: GEL Output: DCSM Initialization Done		^ 	Hs.imple.lms.delms.setex.tx 0 items C2Box_CFU1: 6EL Oxtput: DCSM Initialization D > Description ^ C2Box_CFU1: 6EL Oxtput: DCSM Initialization S ^ C2Box_CFU1: 6EL Oxtput: DCSM Initialization S ^	Resou
• m		•		

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.

workspace_fsiApp	pNote - CCS [Debug - fsi_si	ngle_line_delay_s	elect_rx/fsi_single_line_e	delay_select_n	.c - Code Com	poser Stu	dio	(milding)	Contract Shade	a h Clifford N. o
File Edit View P	Project Tools	Run Scrip	ots Window H	elp							
	Þ 00 🔳 🕉	s.e 🖩 🛛	2 % 🙆 🔻 🕅 1	8 🌸 🕶 🎸 📽 🖛 🔍 9	⊙ 🔨 ▼ 🖾	Ø 🎋 🕶 🛷	•				
‡ Debug ⊠				×	(x)= Variables	M Expression	is 23 1117 F	Registers	;		
4 🗘 fsi_single_line	e_delay_select	_rx [Code Co	mposer Studio -	Device Debugging]	Expression		Туре		Value		Address
🔺 🧬 Texas Instru	uments XDS1	10 USB Debu	ig Probe_0/C28x	CPU1 (Suspended - S)	🔺 🥔 pingAr	ndDataStatus	unsigne	d long	[0x000003FF	,0x00000FFF,0x00001FFF	0x0000AC40@Data
= main() a	at fsi_single_li	ne_delay_sel	ect_rx.c:425 0x00	844E	60° [0]		unsigne	d long	0x000003FF	(Hex)	0x0000AC40@Data
_args_m	nain() at args_	main.c:81 0x	008D22		60-[1]		unsigne	d long	0x00000FFF	(Hex)	0x0000AC42@Data
c_int00() at boot28.a	sm:261 0x008	8908 (the entry p	oint was reached)	(x): [2]		unsigne	d long	0x00001FFF	(Hex)	0x0000AC44@Data
📌 Texas Instr	Courses				- 1		22	long	0x00003FFF	(Hex)	0x0000AC46@Data
	Save me	nory					0.0	long	0x00007FFF	(Hex)	0x0000AC48@Data
	Save Mer	norv						long	0x0000FFFF	(Hex)	0x0000AC4A@Data
								long	0x0001FFFF	(Hex)	0x0000AC4C@Data
	Select a fi	le to save the	e memory data					long	0x0003FFFF	(Hex)	0x0000AC4E@Data
								long	0x0007FFFF	(Hex)	0x0000AC50@Data
	File:	C:\Users\a0	225962\workspace	e fsiAppNote\SingleDa	ataLine.dat	Bre	owse	long	0x000FFFFC	(Hex)	0x0000AC52@Data
								long	0x001FFFFE	(Hex)	0x0000AC54@Data
•	File Type:	TI Data				•		long	0~00355558	(Hev)	0v0000AC56@Data
fsi_single_line_d	Note that	Hex and ELF	files are not supp	oorted by this tool.							
420											
421											
422 }											
423 }											
425 ESTOP											
426 while											
427 }											
428											
429 //											
430// initFS.											
432//											
433 void init											
434 {											
435 FSI_d:	Ø		< Back	Next >	Finish	Cance	el				
436								<u> </u>			
U Console											
fsi_single_line_delay	_select_rx	DCSM .		m Start							
C28xx_CPU1: GEL	L Output:	DCSM 1	Initializatio	on Done							
C28xx_CPU1: GEL	L Output:	DCSM 1	Initializatio	on Start							
C28xx_CPU1: GEL	L Output:	DCSM 1	[nitializatio	on Done							

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.

e Edit View Proje	ct Tools Run Scripts Window Help				
- 🔛 🕼 🕒 🕩 🛛) = 3. 🕫 .e. 🖩 🖫 % 😃 + 🏷 🏷 📚 + 🕹 🖝 + 3. (० % ▼ <⊂ Ø क ▼ ⋪	? ▼		
Debug ⊠	×	(X)= Variables	ns 🖾 🛲 Registers		
fsi_single_line_del	ay_select_rx [Code Composer Studio - Device Debugging]	Expression	Type	Value	Address
Instrume	nts XDS110 USB Debug Probe_0/C28xx_CPU1 (Suspended - S)	pingAndDataStatus	unsigned long	[0x000003FF.0x00000FFF.0x00001FFF	0x0000AC40@Data
main() at fsi	_single_line_delay_select_rx.c:425 0x00844E	() ()	unsigned long	0x000003FF (Hex)	0+00004 C40@Data
_args_main() at args_main.c:81 0x008D22	69: [1]	unsigned long	0x00000FFF (Hex)	0x0000AC42@Data
c_int00() at	boot28.asm:261 0x008908 (the entry point was reached)	60: [2]	unsigned long	0x00001FFF (Hex)	0x0000AC44@Data
🔎 Texas Instrume	nts XDS110 USB Debug Probe_0/CLA1_0 (Disconnected : Unkr	60: [3]	unsigned long	0x00003FFF (Hex)	0x0000AC46@Data
(60- [A]	unsigned long	0x00007FFF (Hex)	0x0000AC48@Data
	Save Memory			0x0000FFFF (Hex)	0x0000AC4A@Data
	C		9	0x0031FFFF (Hex)	0x0000AC4C@Data
	Save Memory		9	0x0003FFFF (Hex)	0x0000AC4E@Data
	Enter the information for the memory block to be saved		9	0x0007FFFF (Hex)	0x0000AC50@Data
			9	0x000FFFFC (Hex)	0x0000AC52@Data
	Former DD Distley, TI Style		9	0x001FFFFE (Hex)	0x0000AC54@Data
	Forma: 32-Bit Hex - 11 Style		0	NVNN2EEER (Hev)	0v00000C56@Data
i_single_line_delay	Target	_ /			
0	Start Address: 0x0AC40				
1					
2 }	Memory Page: Data				
3 }	Length:				
5 ESTOPA	Specify the number of memory words to read:				
6 while(1)	22				
7}	32				
В	Specify the data block dimension in number of memory	y words:			
977	Number of Rows: Number of Columns:				
)// initFSI -					
void initFSI					
{					
5 FSI_disa					
6	? < Back Next >	Finish	ancel		
onsole 🛛					
ngle_line_delay_sel	ect_rx				

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".

Select the single line data file			FSI Skew Compensation Utility
Eskandari,	Nima 🕨 workspace_fsiAppNote 🕨	✓ ♣ Search workspace_fsiAp	Main Menu
Organize 🔹 New folder	·	= 0	
Documents	^ Name	Date modified	
Desktop	 jxbrowser-data .metadata RemoteSystemsTempFiles 	2/16/2018 1:2: 3/6/2018 1:39 1/31/2018 11:	
Documents Music S Pictures G Videos	 DelayMeasure.dat DualDataLine.dat SingleDataLine.dat 	2/14/2018 9:5: 2/15/2018 6:4 2/15/2018 6:2	Delay Tap Measurement Single Line RX0 vs CLK
 Eskandari, Nima workspace_fsiAppNc Computer 	y		Uual Line CLK/RX0 Set vs RX1 View Example Data
SDisk (C:)	 ✓ III e: SingleDataLine.dat 	File (**) Open ▼ Cancel	TEXAS INSTRUMENTS

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.



- 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.

workspace_fsiAppNote - CCS Debug - fsi_dual_line_delay_select_rx/fsi_dual_line_del	ay_select_rx.c - Code Compo	ser Studio	A anti-part totalla	A COMMANNA
File Edit View Project Tools Run Scripts Window Help				
🗂 🕶 🔛 🐚 🗳 🕪 💷 🔳 🌫 👁 📭 💷 🌆 🕼 💆 🖝 🍪 💣 🕶 🖎	∿ ∛ ▼ 🖾 🖉 🔅 ▼ 🦽	₽ ◄		
🏘 Debug 🛛 🍇 🗸 🗆 🗖	(x)= Variables	ns 🖾 🕮 Registers		
# Isi_dual_line_delay_select_rx [Code Composer Studio - Device Debugging]	Expression	Туре	Value	Address
A P Texas Instruments XDS110 USB Debug Probe_0/C28xx_CPU1 (Suspended - SV A) A)	🔺 🥭 pingAndDataStatus	unsigned long	[[0x000003FF,0x000003FF,0x000003F	0x0000A840@Data
main() at fsi_dual_line_delay_select_rx.c:449 0x00847E	Þ 🥏 [0]	unsigned long	[0x000003FF,0x000003FF,0x000003FF	0x0000A840@Data
_args_main() at args_main.c:81 0x008D6D	Þ 🥏 [1]	unsigned long	[0x00000FFF,0x00000FFF,0x00000FFF,	0x0000A880@Data
c_int00() at boot28.asm:261 0x008976 (the entry point was reached)	Þ 🥭 [2]	unsigned long	[0x00001FFF,0x00001FFF,0x00001FFF,	0x0000A8C0@Data
Yexas Instruments XDS110 USB Debug Probe_0/CLA1_0 (Disconnected : Unkr	Þ 🥭 [3]	unsigned long	[0x00003FFF,0x00003FFF,0x00003FFF,	0x0000A900@Data
	Þ 🥭 [4]	unsigned long	[0x00007FFF,0x00007FFF,0x00007FFF,	0x0000A940@Data
	Þ 🥭 [5]	unsigned long	[0x0001FFFF,0x0000FFFF,0x0000FFFF,	0x0000A980@Data
	Þ 🥏 [6]	unsigned long	[0x0001FFFF,0x0001FFFF,0x0001FFFF,	0x0000A9C0@Data
Save Memory		🔀 🛛 long	[0x0003FFFF,0x0003FFFF,0x0003FFFF,	0x0000AA00@Data
a save memory		I long	[0x0000000,0x0007FFFC,0x0007FFFC	. 0x0000AA40@Data
Save Memory		I long	[0x0000000,0x000FFFFC,0x000FFFFC	0x0000AA80@Data
Select a file to save the memory data		I long	[0x0000000,0x0000000,0x001FFFE	0x0000AAC0@Data
Select a file to save the memory data		Llong	000000000000000000000000000000000000000	0∨0000AR00@D≥t>
Isi_dual_line_de				
444 File: C:\Users\a0225962\workspace_fsiAppNote\DualData	aLine.dat Bro	owse		
445				
	•			
448 Note that Hex and ELF files are not supported by this tool.				
♦ 449 ESTOP				
450 while				
451 }				
452				
4557/ 454 // initES				
455 //				
456 //				
457 void init				
458 {				
459 FSI_d				
4				
Console S				
fsi_dual_line_delay cz8xx_CPU1: oi	Finish Cance	el		
C28xx_CPU1: GL				
C28xx_CPU1: GEL Output: DCSM Initialization Start				

Output: ... DCSM Initialization Done ...

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

Visualizing the FSI Two Data Line Skew Compensation

www.ti.com

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

👽 workspace fsiAppNote - CCS Debug - fsi dual line delay select rx/fsi dual line delay select rx.c - Code Composer Studio					
File Edit View Project Tools Run Scripts Window Help					
🗂 🕶 🔝 🖏 📮 🕪 🗉 🔳 🌫 🕫 📌 🗮 💷 % 😃 🕶 🗞 🇳 🛩 🍰 🗉	ಾ 🔦 ▼ 🖾 🖉 🎋 ▼ 🔗	•			
🎋 Debug ¤ 🐂 ⊂ 🗖	(x)= Variables	s 🖾 👭 Registers			
 Si_dual_line_delay_select_rx [Code Composer Studio - Device Debugging] Pexas Instruments XDS110 USB Debug Probe_0/C28xx_CPU1 (Suspended - S) main() at fsi_dual_line_delay_select_rx.c:449 0x00847E args_main() at args_main.c:81 0x008D6D c_int00() at boot28.asm:261 0x008976 (the entry point was reached) Texas Instruments XDS110 USB Debug Probe_0/CLA1_0 (Disconnected : Unkr 	Expression	Туре	Value	Address	
	🔺 🥏 pingAndDataStatus	unsigned long	[[0x000003FF,0x000003FF,0x000003F.	0x0000A840@Data	
	Þ 🥏 [0]	unsigned long	[0x000003FF,0x000003FF,0x000003EF,	0x000004840@Data	
	Þ 🥭 [1]	unsigned long	[0x00000FFF,0x00000FFF,0x00000FFF,	0x0000A880@Data	
	Þ 🥭 [2]	unsigned long	[0x00001FFF,0x00001FFF,0x00001FFF,	0x0000A8C0@Data	
	Þ 🌽 [3]	unsigned long	[0x00003FFF,0x000035FF,0x00003FFF,	0x0000A900@Data	
	Þ 🕭 [4]	unsigned long	[0x00007FFF,0x00007FFF,0x00007FFF,	0x0000A940@Data	
	Þ 🥭 [5]	unsigned long	[0x0001FFFF0x0000FFFF,0x0000FFFF,	0x0000A980@Data	
	Þ 🥭 [6]	unsigned long	[0x0001/FFF,0x0001FFFF,0x0001FFFF,	0x0000A9C0@Data	
Save Memory		Nong	[0v0003FFFF,0x0003FFFF,0x0003FFFF,	0x0000AA00@Data	
		llong	[0x00000000,0x0007FFFC,0x0007FFFC	0x0000AA40@Data	
Save Memory		Long	[0x0000000,0x000FFFFC,0x000FFFFC	0x0000AA80@Data	
Enter the information for the memory block to be saved		l long	[0x0000000,0x0000000,0x001FFFE	0x0000AAC0@Data	
là fsi_dual_line_de					
144 Format 32-Bit Hex - TI Style -					
445	1 Tarrat				
448 Start Address: UXUA840	Start Address: UXUA84U				
ESTOP Memory Page: Data					
while Length:					
451 }					
Specify the number of memory words to read:					
4557// 454// initES 1024	/ initES 1024				
455 // Specify the data block dimension in number of memory wo	Specify the data block dimension in number of memory words:				
456 // Number of Rows: Number of Columns:	Number of Rows: Number of Columns:				
457 void init					
458 {					
459 FSI_d					
400					
E Concolo M					
fi dual line datau ? < Back Nevt >	Finish				
	Curree	· · · · · · · · · · · · · · · · · · ·			
C28xx_CPU1: GL					
C28xx_CPU1: GEL Output: DCSM Initialization Start					

C28xx_CPU1: GEL Output: ... DCSM Initialization Done ...

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



11



0000000

19 21 23 25 27 29

Execution Point

Figure 24. Scenario 3 FSI Execution Points

RX0 Delay

•

Failure

. .

Point: (24, 0) Other Points: (24, 0) (25, 1) (26, 2) (27, 3) (28, 4) (29, 5) (30, 6) (31, 7)

9 11 13 15 17

Success

Finding the Optimal FSI Execution Point



Figure 25. Scenario 4 FSI Execution Points



Figure 26. Scenario 5 FSI Execution Points

Figure 27. Scenario 6 FSI Execution Points



Finding the Optimal FSI Execution Point





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.

🐝 workspace_fsiAppNote - CCS Debug - fsi_find_optimal_delay_rx/fsi_find_optimal_delay_rx.c - Code Composer Studio 🦳 📼 💥	😵 workspace_fsiAppNote_tx - CCS Debug - fsi_find_optimal_delay_tx/fsi_find_optimal_delay_tx.c - Code Composer Studio 🛛 🗕 🔍				
File Edit View Project Tools Run Scripts Window Help	File Edit View Project Tools Run Scripts Window Help				
🖻 🕆 🖩 🐚 🗳 🕨 🖩 🔍 🕫 🕫 🎆 🕼 🖓 🥹 🕶 🦄 🖄 🍲 🕹 🗉 🖉 🖓 🖓 🗸 🗸	[☆ •				
Quick Access 🛛 😰 🗎 🐯	Quick Access 😰 🛱				
💠 Debug 🕮 🐂 한 🗖 📴 🚾 Variables 🛠 Expressions 🕮 🕮 Registers 🖤 🗖	🏘 Debug 🛙 🐐 🛸 👘 🐨 Variables 🕮 🛠 Expressions 🦛 🖶 📑 😁 🌚 😤 🗉				
😨 <terminated>fsi_find_optimal_delay_rx [Code Composer Stu 🛛 🖄 🐗 🔁 📩 😤 😵 🛅 🔂 🌣</terminated>	▲ 😨 fsi_find_optimal_delay_tx [Code Composer Studio - Devi Name Type Value Location				
A 🕫 fsi_find_optimal_delay_rx [Code Composer Studio - Device C Expression Type Value	▲ P Texas Instruments XDS2xx USB Debug Probe_0/C28xx				
🔺 🖗 Texas Instruments XDS110 USB Debug Probe_0/C28xx_CPI 🦨 🅭 exePoint struct FSIExecu (RX0Delay=1,RX1Delay=3	main() at fsi_find_optimal_delay_tx.c:165 0x008A0B				
■ main() at fsi_find_optimal_delay_rxc:167 0x008A0F	■ _args_main() at args_main.c81 0x008D26				
= _args_main() at args_main.cstl 0x008/rF	= c_int00) at boot28.asm26.0 0x008976 (the entry p)				
Cintooy at bootzo.astrizo to woods for the entry point of the CLEDelay unsigned int of unsigned the operation of the entry point of the cleder of the c	X* Texas instruments AUS2XX USB Debug Probe_U/CLAL_				
x ⁺ Texas instruments Abstro Cos Debug Prote Orce X1 or C					
· Add new expression					
R fri find ontimal delay ty c 2 R fri ontimal delay c R fri ontimal delay h					
	I d fsi_find_optimal_delay_txc 🕸 d fsi_optimal_delay.c				
157 //	154				
158 // Set GP11, 12,13 to be asynchronous(pass through without delay)	156 // Set GP11, 12,13 to be asynchronous(pass through without delay) 157 // Default setting is to have 2 SYS_CLK cycles delay				
159 // Default setting is to have 2 SYS_CLK cycles delay					
160 // 161 GPTO setOuslificationMode(11 GPTO OUAL ASYMC):	158 //				
162 GPIO setQualificationMode(12, GPIO QUAL ASYNC);	<pre>159 GPID_setQualificationMode(11, GPID_QUAL_ASYNC); 160 GPID_cetQualificationMode(12, GPID_QUAL_ASYNC);</pre>				
163 GPI0_setQualificationMode(13, GPI0_QUAL_ASYNC);	160 GPI0_SetQualificationMode(12, GPI0_QUAL_ASYMC); 161 GPI0_setQualificationMode(13, GPI0_QUAL_ASYMC);				
164	162				
exePoint = FSI_calibrateExecutionPoint(FSIRXA_BASE, FSITXA_BASE, FSITX	<pre>163 FSI_transmitToCalibrate(FSIRXA_BASE, FSITXA_BASE, 2U);</pre>				
160 FSTOPA	164				
168 while(1); Calculated Execution Point	165 ESTUP0; 166 while(1):				
	167 }				
170	168				
۲	4 · · · · · · · · · · · · · · · · · · ·				
Se Consult as a co	Console & Consol				
C28xx CPU1: GEL Output: DCSM Initialization Done	C28xy CPUID: GEL Output: DCSM Initialization D Consisten				
C28xx_CPU1: GEL Output: DCSM Initialization Start	C28x_CPU1 GEL Output: DCSM Initialization S Description Resource				
C28xx_CPU1: GEL Output: DCSM Initialization Done	C28xx_CPU1: GEL Output: DCSM Initialization D				
	*				
4					
	Writable Smart Insert 165:1				

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 Copyright © 2019, Texas Instruments Incorporated