

SourcePoint

Getting Started Guide for the

AAEON UP Xtreme i11

Revision 2.4



©2024 ASSET InterTech, Inc.

Contents

REVISION HISTORY	4
WELCOME!	5
GETTING STARTED WITH SOURCEPOINT	6
Boards and Cables	6
BIOS Settings	8
DbCStatus.exe: Red is Bad, Green and Yellow are Good	12
Basic SourcePoint Debugging	14
Advanced Topics: Using Trace	26
Configuring the Intel Trace Hub	26
Using Architectural Event Trace (AET)	27
Intel Processor Trace	31
Basic SourcePoint Troubleshooting Tips	33
Trace buffer overflows	33
Intel Processor Trace can be slow	35
My board is not booting – what now?	
SOURCEPOINT DEBUGGING FOR WINDOWS	39
Configuring the target and setting up pre-requisites – Getting Started	42
How to Establish a SourcePoint WinDbg Session	44
Step 1: Connect SourcePoint to the target	44
Step 2: Start WinDbg via a SourcePoint macro	45
Step 3: Load symbols with the LoadCurrent macro	48
What happens if the symbols don't show up?	53
Getting SourcePoint to display module names as well as function names	55
Troubleshooting Tips and Errata	57
Windows crashes	57
WinDbg Classic is better than WinDbgX	57
Pause in Initial Symbol Load	58
LoadCurrent versus LoadAll	58
COM(32) Surrogate	58
Viewing the Stack	58
LoadCurrent intermittently fails in User code	60
Breaks are not process-aware	60



Mangled function names60
WinDbg FP register display is not working61
Problems with symbol loading from local cache61
SOURCEPOINT DEBUGGING FOR HYPER-V66
Getting Started
VMM breakpoints, and debugging the Secure Kernel68
VMCS Viewer/Editor
VM Exit breakpoints and Basic Exit Reasons72
Using Intel PT with Hyper-V74
Suggested Hyper-V Reading75
Troubleshooting Tips on Hyper-V/ VBS Enabled Targets77
VM Resume breakpoint with Intel PT crashes the target77
Hardware breakpoints don't work well in the Secure Kernel77
AET only partially functional77
Support for VM Exit Reasons > 63
Intel PT Call Chart does not work reliably79
CONCLUSION



REVISION HISTORY

Revision Number	Description	Date
1.0	Original document, describes v0000	November 28, 2021
	board support	
2.0	Added content for new support of v0001	May 30, 2022
	(with the Type-C connector removed)	
	AAEON UP Xtreme i11 board	
2.1	Updated for WinDbg support and other	December 3, 2023
	perfective changes	
2.2	Update for beta release SourcePoint	March 31, 2024
	7.12.52	
2.3	Production release documentation for	May 5, 2024
	7.12.53.	
2.4	Production release for 7.12.59.	July 13, 2024
	Consolidating SourcePoint UEFI and	-
	SourcePoint WinDbg documentation.	





WELCOME!

Thank you very much for your SourcePoint purchase! We appreciate you acquiring our best-in-class debugger, and hope you enjoy using it. We strive to deliver the most powerful, easy-to-use and polished product as possible. So, please feel free to share your feedback directly at our support site at <u>https://www.asset-intertech.com/support/</u>, or via your favorite social media outlet.

As with any new tool, mastering SourcePoint takes an investment in terms of time and effort. JTAG-based debug is a fairly specialized area, and low-level "on the metal" firmware development on x86 platforms is even more so. So, in your use of the tool, you may encounter behavior that seems non-intuitive or even wrong. You may be encountering a tool corner case, a limitation inherent in JTAG or DCI, or even a bug. If so, try a few different options as may be referenced in the <u>Troubleshooting</u> section of this Guide, and if it persists, give us a call. We are happy to support you.



GETTING STARTED WITH SOURCEPOINT

Read this section first! It provides an introduction to the configuration changes you need to make on the AAEON UP Xtreme i11 target, and provides a basic overview on running SourcePoint. Then, you can jump ahead to debugging <u>Windows</u> and <u>Hyper-V</u>.

Boards and Cables

The board covered in this document is the AAEON UP Xtreme i11 board, based upon the Intel Tiger Lake CPU.

The Tiger Lake boards come in four flavors: Celeron, i3, i5 and i7. All boards are suited for Windows debugging. The Celeron board is the least expensive, and supports all the latest Intel debug and trace features such as Intel Processor Trace (Intel PT), Intel Trace Hub, Architectural Event Trace (AET), and others.





WARNING:

Do <u>NOT</u> plug a regular USB cable into the target and attempt to use DCI. Specialty cables, with VBUS snipped, are required; using a regular USB cable may possibly fry your target, or worse.

The main source to purchase the specialty DCI cable needed for SourcePoint debugging is ASSET InterTech. This target has its Type-C port enabled for DCI. If you have a debug host with a Type-A port, you'll need to purchase the part # ITPDCIAMCM1MU (1.0 meter) cable. The longer 1.8-meter ITPDCIAMCM2MU cable will work as well. If your host has a Type-C port, purchase the ITPDCIC2CD2U1M. Using Type-A/C hubs have been seen to work, but are not warranted. Type A/C adapters have been seen <u>not</u> to work.

Contact your ASSET representative to obtain the cable needed for your configuration.



BIOS Settings

The AAEON UP Xtreme i11 boards come equipped with an AMI Aptio BIOS that is based upon typical Intel Customer Reference Board (CRB) BIOS.

Luckily, the platform comes with all the necessary hardware hooks and firmware straps to support Intel Direct Connect Interface (DCI) out of the box.

The USB Type C port on the board is the port of interest:



There are three BIOS settings to change for the board to work with SourcePoint and DCI:

There are three settings in the default AAEON BIOS that need to be changed to successfully JTAG debug the UP Xtreme i11 target:

- 1. Disable the WDT timer
- 2. Disable the TCO timer
- 3. Set the Platform Debug Consent to Enabled (USB2 DbC).

©2024 ASSET InterTech, Inc.



Go into the BIOS Boot menu (this is accomplished by rebooting the board, at the same time holding down the F7 key, and you'll be promoted for the password for the "CRB Advanced" BIOS menu. Note that CRB is Customer Reference Board, an Intel reference:

Enter	Password	

The password is upassw0rd.

And that is a number "zero" (0) in the password, not the letter "O".

Enter Setup, and use the arrow key to move over to the Boot menu. and set WDT Timer -> Disabled. If you don't, run-control will be successful, but the board will power-cycle every 30 seconds; putting a real crimp in your debugging!

Main Advanced Cl	Aptio Setup – AMI hipset Security Boot Save	8 Exit
Boot Configuration		▲ Enables or disables Quiet Boot option
Quiet Boot WDT Timer	[Enabled] [Disabled]	
FIXED BOOT ORDER P	riorities	
Boot Option #1	[USB Hard Disk]	
Boot Option #2	[USB CD/DVD]	
Boot Option #3	[USB Key]	
Boot Option #4	[USB Floppy]	++: Select Screen
Boot Option #5	[USB Lan]	↑↓: Select Item
Boot Option #6	[Hard Disk]	Enter: Select
Boot Option #7	[NVME:Windows Boot	+/-: Change Opt.
	Manager (KINGSTON	F1: General Help
	OM8SEP4256Q-A0)]	F2: Previous Values
Boot Option #8	[CD/DVD]	F3: Optimized Defaults
Boot Option #9	[Network]	 F4: Save & Exit ESC: Exit
V	ersion 2.21.1278 Copyright (C) 2021 AMI

There is another setting within the CRB Advanced menu to Disable the TCO Timer from being re-enabled by Windows; this is set to Disabled by default when shipped from AAEON, but if you run into issues with run-control stability, you might want to check this:

CRB Setup > CRB Chipset > PCH-IO Configuration > Enable TCO Timer **must be set to** Disabled.



Main	Aptio Setup – AMI	
Compatible Revision ID Legacy IO Low Latency PCH Cross Throttling PCH Energy Reporting LPM S0i2.0 LPM S0i2.1 LPM S0i2.2 LPM S0i3.0 LPM S0i3.1 LPM S0i3.2	[Disabled] [Disabled] [Enabled] [Enabled] [Enabled] [Enabled] [Enabled] [Enabled] [Enabled] [Enabled]	Enable/Disable TCO timer. When disabled, it disables PCH ACPI timer, stops TCO timer, and ACPI WDAT table will not be published.
LPM SOIS.3 LPM SOIS.4 C10 Dynamic threshold adjustment IEH Mode Enable TCO Timer Enable Timed GPIO0 Enable Timed GPIO1	[Enabled] [Enabled] [Disabled] [Disabled] [Enabled] [Enabled]	<pre>++: Select Screen f↓: Select Item Enter: Select +/-: Change Opt. F1: General Help F2: Previous Values F3: Optimized Defaults F4: Save & Exit ESC: Exit</pre>
Version	1 2.21.1278 Copyright (C) 2	021 AMI

Finally, go to CRB Setup > CRB Advanced > Debug Settings and set
Platform Debug Consent to Enabled (USB2 DbC):

Main	Aptio Setup – AMI	
Debug Settings Kernel Debug Serial Port Kernel Debug Patch Platform Debug Consent VT-d Debug Settings Advanced Debug Settings	[Legacy UART] [Disabled] [Enabled (USB2 DbC)]	To 'opt-in' for debug, please select 'Enabled' with the desired debug probe type. Enabling this BIOS option will override other debug-related BIOS options. ++: Select Screen 11 : Select Item Enter: Select +/-: Change Opt. F1: General Help F2: Previous Values F3: Optimized Defaults F4: Save & Exit ESC: Exit
Version	2.21.1278 Copyright (C) 2	021 AMI



Remember to do a "Save & Exit" with the F4 key after the changes! Otherwise, your changes won't be saved.

Power Tip: We have observed that, on very rare occasions, the saved settings that you've made will be undone by a "BIOS restore". If you begin to observe strange effects, like autonomous platform resets while in run-control mode, check to ensure that the changes you've made have not been undone.

You are now ready to test your connection, and then launch SourcePoint and begin debugging.



DbCStatus.exe: Red is Bad, Green and Yellow are Good

Luckily, there is a convenient application in the SourcePoint install directory that will tell you that the DCI driver is successfully installed on your computer, and it is possible to make a connection between SourcePoint and the target.

Navigate to C:\Program Files (x86)\Arium\SourcePoint 7.12.59 (where 59 is your current SourcePoint release), and launch the DbCStatus.exe. You should see the red ball, indicating that there is no connection:

DbC Connection Status		
Connection status:	No connection	
DCI driver version:	1.10.0.0	
		Close

Ensure that the Type-C cable is firmly connected to both the host and target, and power up the UP Xtreme i11. In a moment the ball should turn green:

DbC Connection Status		
Connection status:	USB 2.0	
DCI driver version:	1.10.0.0	
		Close

Let the platform boot to the UEFI shell. Congratulations! You have a working DCI connection. It's smooth sailing from here.

©2024 ASSET InterTech, Inc.



Power Tip: If you are using SourcePoint WinDbg for debugging this board, and have Windows already installed, there may be situations where you want to go to BIOS setup before booting all the way up to Windows. In this case, press F7 after powering on to stop at the BIOS setup.

Power Tip: We have found that some versions of the Celeron board will not power up successfully after a power cycle. Here's the workaround:

- 1. Apply power back to the target. You will see that it doesn't power up, and the blue light does not light up on the power switch.
- 2. Unplug the DCI cable from the Type-C plug.
- 3. The board will start to power up. Press F7 and you will stop at the UEFI menu password screen.
- 4. Hot Plug the Type-C cable back in. Your DCI connection will be restored, and you will see the DbCStatus ball turn green. You may continue your Windows debugging session.



Basic SourcePoint Debugging

When you launch SourcePoint for the first time, you will see the main screen, mostly gray:

SourcePoint							- 0	×
File Edit View Processor Options Window Help								
달 딸 김 김 날 김 수 알 날 수 없 없	ay ay ay ay	X . C . A . A	***	0.000	🗢 🕒 🔪 🔜 IP 🤅	. · · · · · · ·	図べばム県	# D.
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset					No power			

SourcePoint uses Projects (files with suffix .prj) as containers for your debugging session. You can create as many Projects as you want, with all your own preferences saved. Often, once you have the SourcePoint Project configured to your liking, you'll save it and use it repeatedly during your separate debugging sessions. Other users may wish to save a separate Project for each separate debugging session. That's really a matter of user preference and what you're debugging – it's your choice.

Now it's time to create the Project. Under File > Project... select New Project:





You'll be presented with the New Project Wizard (NPW). The emulator connection should be via DCI:





New Project Wizard: Welcome	×
Welcome to the New Project Wizard	
This wizard helps you:	
- Select or add an emulator connection.	
- Create a new project file.	
DCI, DCI · Add/Edit	
Select an emulator connection and click Next.	
< Back Next > Cancel	Help

After hitting Next, you'll be prompted for the Project Name, Location to store the Project, and the location of the Target Configuration file:



New Project Wiz	zard: Project File	×
Project file File name:	myproject	
Target conf	iguration file Browse Browse	
	Identify Target	
	< Back Next > Cancel Help	

Note that the Target Configuration (TC) files located in C: \My

Documents\Arium\Targets are used in conjunction with the jtag-devices.xml file to define the specific silicon and SourcePoint settings necessary to ensure a successful DCI connection.

For the UP Xtreme i11 boards, custom TC files have been created, so you shouldn't do an Identify Target to automatically select the TC file of interest. Rather, manually select the specific TC file that is customized for this target (TGL\UP-Xtremeill_DbC.tc) and hit Open:



🛞 Open			×						
\leftarrow \rightarrow \checkmark \uparrow 📜 $<$ Sour	$\leftarrow \rightarrow \checkmark \uparrow$ is sourcePoint-IA_7.12.20 \Rightarrow Targets \Rightarrow TGL \Rightarrow TGL-U \checkmark U \checkmark Search TGL-U								
Organize 🔹 New folder			• • •						
🦲 iCloud Drive 🖈 🔨	Name	Date modified	Туре						
Documents	TGL.tc	10/29/2021 4:15 AM	TC File						
📙 Intel	TGL_DbC.tc	10/29/2021 4:15 AM	TC File						
Screenshots	UP-Xtreme-i11_DbC.tc	11/19/2021 2:10 AM	TC File						
SourcePoint-IA_7									
😽 Dropbox									
OneDrive - Person									
🗢 This PC									
🧊 3D Objects									
📃 Desktop									
🖺 Documents 🗸 🗸	< Comparison of the second sec		>						
File nam	e: UP-Xtreme-i11_DbC.tc	✓ Target ConfOpen	iguration Files (*.tc) V						

Then, your screen should look something like this, after you've replaced the default Project file File name with your preferred name (in this instance, My Tiger Lake Project):



New Project Wi	zard: Project File	×
Project file		
File name:	My Tiger Lake Project	
Location:	C:\Users\alans\Documents\Arium\SourcePoint-IA_ Browse	
Target conf	figuration file	
C:\Users\a	alans\Documents\Arium\SourcePoint-IA_7.12.15\Ta Browse	
	Identify Target	
	< Back Next > Cancel	Help

Hit Next, then Finish, and SourcePoint should successfully connect to the target. You should see "JtagTest: Successful operation" followed by "Configuration state: Connected" in the Status bar at the bottom left:





Now the fun part begins. Click on the buttons at the top to set up the Viewpoint, Code, Command, Registers and other windows to your own preference. Move the windows around and resize them to take best advantage of your available screen real estate. You can right-click in the title bar of each window to change its type and, for example, to dock the window to the bottom, right side, etc.



A sample layout is below:

SourcePoint v7.12.0 [DCI] - TigerLake - C.\Users\alans\Documents\Arium\SourcePoint-IA_7.12.15\My Tiger Lake Project.prj (safe mode)				- 0	×
File Edit View Processor Options Code Window Help					
📴 🎬 🔛 😰 📽 🞼 🍘 💖 🖓 🖓 👹 🎆 🔛 🐨 👘 🖓 👘 🖓 👘 🔛 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘	Log 🎹 Memory	IP Register	s 🎕 Symbols 🧨 Trace 🤅	Viewpoint Q	Watch
					8
	● Viewpoint				23
The available - Processor not available	Name		Description	S	Status
	* P0	TigerLak	e	Running	
	• P1	TigerLak	e	Running	
	• P2	TigerLak	e	Running	
	° P3	TigerLak	e	Running	
	<				>
		Mamo	Value		
	Intel 64	RAX	22222222222222222222222222222222222222		_
	General	RBX	???????????????????????????????????????		
	Floating	FRCX	???????????????????????????????????????		
	Segment	RDX	3333333333333333333		
	Control	RBP	???????????????????????????????????????		
<u>`</u>	Debug	RSI	222222222222222222222222222222222222222		
Unknown V Source V Go Cursor Set Break Track IP View IP Refresh	MMX	RDI	2222222222222222222222222		_
	YMM - S	FR8	222222222222222222222222222222222222222		
🔤 Breakpoints	YMM - D	F R9	222222222222222222222222222222222222222		
Identifier Address Attributes	YMM - In	1 R10	222222222222222222222222222222222222222		
Numeri Address Address	MSR	R11	???????????????????????????????????????		
	User	R12	???????????????????????????????????????		
		R13	222222222222222222222222222222222222222		_
Edit Add Damous Damous All Enable All		R15	222222222222222222222222222222222222222		_
Luit Nethove Ali Linbule Disable Ali		CS	????		
	1	DS	????		
Det Blas demonstration		SS	????		
Date Time Component Message		ES	2555		
WINTERPORT FILTERPORT FILTERPORT	1	FS	????		
		BTD	222222222222222222222222222222222222222		_
	< >	RELAGS	222222222222222222222222222222222222222		
					_
Command					
Configuring Devices					^
Connecting					
Loading Command Language Extensions: C:\Users\alans\Documents\Arium\SourcePoint-IA_7.12.15\Macros\aa\aaextend.mac					
P0>					
					~
The In Fisher Fisher State Into Firsten Over Shift+Fi2Reset	P0 15- D	upping		Halt Mode	
rinely, 1330, anteriality, reality more more over, anterialitiese	IF: KI	unnind		Inter Mode	

Power Tip: the window layout can be saved as a separate file. That way, when you create a new Project, you can just Load the layout file separately, without having to create and move the windows around again. Choose File > Layout > Save Layout... to create a .lyt file, and then do a Load Layout... to save yourself time every time you create a new Project.

Exploring Options > Preferences... and some of the other menu items may give you some more labor-saving ideas. For instance, your workflow might suggest to disable both Load last project on startup and Save project on. This gives more control on entry and exit from the application:



Preference	s								×
General	Emulator	Breakpoints	Code	Memory	Program	IPC	Colors		
Project									
Lo	ad last pro	ject on startup)						
Pr	ompt befor	e automaticall	y saving	project					
Sa	ive project	on exit							
	ad target o	configuration fi	le when	project loa	ded				
Fi	le name:	C:\Users\alar	is\Docun	nents\Ariur	m\SourcePo	oint-IA_7	.12.1	Browse	
Llaan in	h (
User ir	iterrace								_
⊡ Sr	now advanc	ced configurati	on settin	igs	Preferre	ed editor:	notepad	d `	~
<mark>⊠ S</mark> ł	now tooltips	5							
Ti	med windo	w refresh							
int	erval: 10	second	S						
				Γ	OK		Cancel	Heli	p

Another suggestion is to check Save project on exit. It's a matter of preference.

This is a good point to do a Project > Save Project... That way, you don't have to start all over, if for some reason your project gets messed up.

At this point, you are ready to fully begin your debug session. Many of the operations can be accessed via the toolbar at the top of the screen. You can issue a Stop on the target, Step Into, Reset and halt at the reset vector, set some breakpoints, Go to the breakpoint, and so on. Use the buttons at the top, in the SourcePoint Icon toolbar section, to initiate these actions. Use the function keys (i.e. F8 for Step Into) as you gain experience.



SourcePoint v7.12.0 [DCI] - TigerLake - C.\Users\alans\Documents\Arium\SourcePoint-IA.,7.12.15\My Tiger Lake Project.prj					
File Edit View Processor Options Code Window Help					
👺 🎬 🔚 🖆 🔛 🎼 🚱 🗰 🥵 🐝 🐝 📾 📾 🖬 🖬 🖬 🖬 👘 🖬 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘	og 🎹 Memory	IP Registers	Symbols Irace	0 Viewpoint	Q Watch
	, ,	3	- / -		Ý 💡
GCode (P0*): (64-bit) Tracking IP 00000000000000 - FFFFFFFFFFFEL	COViewpoint				- 23
0000000640D200FL 498B80C0000000 MOV RAX,QWORD PTR [R8]+000000c0 A	Name	D	escription	a. 1	Status
0000000640D2016L BBCA MOV ECX, EDX	* P0	TigerLake		Stopped	
0000000640D201BL CIFI02 SAL ECA, 2 0000000640D201BL SB0001 MOV EX DWORD PTP [PCV1[Day]	• P1	TigerLake		Stopped	
	0 23	TigerLake		Stopped	
00000000640D2023L 3C02 CMP AL,02	- 15	TIGGIDAKO		bcopped	
0000000640D2025L 499B80C0000000 MOV RAX,QWORD PTR [R8]+000000c0	٢				>
00000006401202CL /50C JNE short ptr 0000000640d205aL	(100)]
000000066022031L OFB70401 MOVZX EAX.WORD PTR [RCA][RCA]	IP General Reg	gisters (PO*)			
0000000640D2035L E9C4000000 JMP 000000640d20feL	■IA-32	Name	Value		
0000000640D203AL 8BCA MOV ECX,EDX	Intel 64	RAX	000000000000000000000000000000000000000	0	
0000000640D203CL 8A0401 MOV AL, BYTE PTR [RCX] [RAX]	General	RBX	00000000606BF75	0	
0000000640D203FL 535A000000 JMP 000000640201eL	Floating F	PDY	000000000000000000000000000000000000000	8	
00000006401204BL 485C0 TEST RAX, RAX	Segment	RBP	0000000061DD699	8	_
00000000640D204EL 74A5 JE short ptr 0000000640d1ff5L	Control	RSI	0000000061CF214	4	
	Debug	RDI	0000000061DD699	8	
UUUUUUUUHUUZUIEL V V Disassembiy V Go Cursor Set Break V Track IP View IP Refresh	VMM - SI	RSP	00000000606BF69	0	
	YMM - D	R8	0000000061DD699	8	
🐨 Breakpoints 👘 🔛	YMM - In	R9	0000000061CF101	8	
Identifier Address Attributes	• MSR	P11	000000601D34203	0	
	User	R12	000000000000000000000000000000000000000	0	
		R13	00000000000003E	8	
		R14	0000000061CF214	4	
Edit Add Remove Remove All Enable Disable All		R15	0000000061DD699	8	
		CS	0038		
🔄 Log		DS	0030		
Date Time Component Message		55	0030		
1/17/2021 14:04:20.282 Exception Handling Exception: Translation error in 00000000640D201EL		FS	0030		_
		GS	0030		
		RIP	00000000640D201	E	
	< >	RFLAGS	000000000001020	6	
					_
Command					1
Scanning Uncore					^
Configuring Uncore					
Configuring Devices					- 10
Connecting					
Loading Command Language Extensions: C:\Users\alans\Documents\Arium\SourcePoint-IA_7.12.15\Macros\aa\aaextend.mac					
Po>					
502					~
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset	P0 18: St	topped	64 Bit	Halt Mode	

Hit the Refresh button in the Code window after the first Stop. This is only necessary once, to get out of Safe Mode; the Code window will automatically refresh with all run-control operations (stop, go, single-step, etc.) afterwards.

Refer to the <u>SourcePoint User Guide</u> in your install directory for detailed instructions on using all of the tool's features.

The board will halt automatically at the reset vector when you hit the SourcePoint Reset button (don't do this yet!):



🕒 Code (P0*): (1	6-bit) Tracking I	P 00000000L - FFFFFFFL		
FFFFFFEBL FFFFFFEDL FFFFFFEFL	0000 0000 00	ADD ADD DB	BYTE PTR [BX+SI],AL BYTE PTR [BX+SI],AL 00	^
FFFFFFF0L	90	NOP		
FFFFFFF1L	90	NOP		
FFFFFFF2L	E923C0	JMP	near16 ptr ffffc018L	
FFFFFFF5L	0000	ADD	BYTE PTR [BX+SI],AL	
FFFFFFF7L	OOFB	ADD	BL,BH	
FFFFFFF9L	0000	ADD	BYTE PTR [BX+SI], AL	
FFFFFFBL	0000	ADD	BYTE PTR [BX+SI],AL	
FFFFFFDL	OOFC	ADD	АН, ВН	
FFFFFFFL	FF	DB	ff	
FFFFFFOL	~ 🔎	Disassembly ~ Go	Cursor Set Break Track IP View IP Refresh	

This is NOT desirable, because some of the AAEON Tiger Lake boards have issues recovering from the reset vector. It is recommended that you disable the default halt at the reset vector by going to the Options > Emulator Configuration > Target Reset and click on "Run the target" in the "After SourcePoint reset" box:





Emulator Configuration	×
General JTAG Target Reset Switches	
Target reset signal is: O Bidirectional O Output only (driven by emulator)	
 Input only (sensed by emulator) 	
SourcePoint reset causes the emulator to: After SourcePoint reset: Action: Wait on manual (external) target reset Reset time: 60 60 ms After target reset, emulator will wait 5500 ms to ensure stability Description Target reset signal input only	
OK Cancel Defaults Help	

Congratulations, you have mastered SourcePoint's basic capabilities, and are using run-control. Many users are content to just use these basic operations, because run-control by itself is very powerful. However, if you wish to master the product and use some of its more advanced features, read on.



Advanced Topics: Using Trace

Trace is by far one of the most useful debugging utilities for triaging the most difficult, hard-to-reproduce bugs. Fortunately, the Tiger Lake CPU is equipped with all the latest-and-greatest Intel trace logic, and SourcePoint supports them all.

Let's look at a few of them, and how to configure their use in SourcePoint.

Configuring the Intel Trace Hub

Event tracing on the TGL platform is accomplished by the Intel Trace Hub. Fortunately, using DCI, events supported by the Intel Trace Hub can be streamed directly out of the system, well before Windows boots.

Boot to the UEFI shell. This is accomplished by powering on the target, and pressing the F7 key until you come to the password entry screen. Note the <u>Power Tip</u> above that references the newer Celeron boards, and the workaround necessary to get the target to power up.

Click on the Trace button in the toolbar at the top, to open the Trace window; then click on the Configure... button; then click on the Trace Hub tab. Set the settings as below:



	Trace Hub AET Intel PT Intel PT Memory
Masters to	trace
ONone	
() All	
List:	18
Trace rout	ing
Trace Hul	DbC V
Intel DT.	System Mamon
incer r 1.	System Herrory
AET:	Trace Hub
System me	emory trace buffer
O Use BIC)S settings
🖲 Use Sou	urcePoint settings
Base add	iress: 01000000P
Lonath	16k ~
Lengen	
Timestamp	
Timestamp	nt packets Frequency; CTC 16 ~
Timestamp	ent packets Frequency; CTC 16 ~
Timestamp Alignme Master / Cl	nt packets Frequency; CTC 16 ~

Once you've configured the Trace Hub this time, you don't have to set it up anymore. You can now take advantage of one of the main capabilities of the Trace Hub, Architectural Event Trace (AET).

Using Architectural Event Trace (AET)



Once the Trace Hub has been enabled for the features you need, click on the AET tab, select All as Processors to trace, and select RDMSR/WRMSR and Port In/Out as events to trace:

BR I	BTS	Trace Hub	AET	Intel PT	Intel PT Me	mory
Proce	essors	to trace				
ON	nne					
0	one					
Al	1					
OL	st:	PO				
0			- 1			
	1	e.g., P0, P4-	·/)			
Even	t sha	ring				
A	only a	- wents to all n	ncosse	ALC .		
O A		venus to all p	UCE550	<i>n</i> 5		
OA	oply e	events to:		(v)		
Even	ŧ			Enable	d IBR	
HW/9	SW I	nterrupt				
IRET				Ē	П	
Exce	epti	on				
RDMS	SR/W	RMSR		•		
Port	In	/Out		I		
Code	e br	eakpoint				
Data	a br	eakpoint				
BTM						
SMI	NMI	/RSM				
MONI	TOR	MWAIT				
	IVD					-
WBIN						~
SGX			1000	dunnand	Clear	all
SGX			F	Annencen		
SGX			F	AUVANCEU		
SGX			F	AUVBIICEU		
SGX			4	AUVAIICEU		(140)

Now, you can simply do a Go/Stop to capture the event trace data. Below shows using the Command window to simulate a break on any read/write of port x'CF8', the PCI CONFIG_ADDRESS. This is done by typing the following into the Command window:

go til cf8io

©2024 ASSET InterTech, Inc.



This will run the target until the next IN or OUT to CF8.

After issuing the command, you'll see something like this:

SourcePoint v7.12.0 [DCI] - TigerLake	- C:\Users\alans\D	locuments\/Arium\S	ourcePoint-IA_/	.12.15\My Liger	Lake Project.prj				- U ×
File Edit View Processor Options Trace	Window Help	-							
		Load UEFI Mac	ros 🖏 🖏 🖏		, 🖳 🔟 e 🛎 🕾	Breakpoin	ts 🕒 Code 💙 Command 📓	Log 🛄 Memory IP Registers 🔍 Symbols	Frace OViewpoint Q Watch
									8
Trace Hub - SW/FW Trace									
STATE ADDR	Event Trace								Status
No data available - Unable	STATE	Pn ADDR		INSTRUCTIO	ON			TIMESTAMP	Stopped (hit bre
		Event	: Port Out:	Port=0021,	Data=000000FF				Stopped (hit br
	-000000586	P0 000000	00067EE16D1	OUT	DX, AL			-72.813 us	Stopped (hit br
	000000550	Event	: Port Out:	Port=UUAI,	Data=000000FF			CC 40C	Stopped (hit bre
	-000000550	P0 000000	DUUG/EE16D/	Dout-1820	DX,AL			-66.406 US	
	-000000514	po ocooo	DODE7FF16DF	POPL=1850	FAX DX			-66 276 up	>
	000000314	Event	Port In:	Port=1830.	Data=80002033			001210 00	
	-000000478	P0 000000	00067EE16DE	IN IN	EAX. DX			-64.036 us	
		Event:	: Port Out:	Port=1830,	Data=80002030				
	-000000442	P0 000000	00067EE16E1	OUT	DX, EAX			-62.240 us	000008
		Event	: Port In:	Port=1830					000000
	-000000406	P0 000000	00067EE179E	8 IN	EAX, DX			-51.406 us	000000
		Event:	: Port In:	Port=1830,	Data=80002033				000CF8
	-000000370	P0 000000	00067EE179E	3 IN	EAX, DX			-48.281 us	0A3CC0
	000000000	Event	: Port Out:	Port=1830,	Data=80002033			45 404	0CD000
	-000000334	P0 000000	DOUG/EEI/9F	Dort-1920	DX, EAX			-46.484 US	B81C00
	-000000298	P0 000000	00067FF17A0	POIL-1030	FAX DY			=46 354 119	0A3C40
	000002.00	Event	: Port In:	Port=1830.	Data=80002033			40.004 48	E9D3B0
	-000000262	P0 000000	00067EE17A0) IN	EAX, DX			-44,115 us	00000E
		Event:	: Port Out:	Port=1830,	Data=80002033				000000
	-000000226	P0 000000	00067EE17A4	OUT	DX, EAX			-42.318 us	0A3AF0
		Event:	: Port Out:	Port=0021,	Data=000000FF				A23018
	-000000190	D0 00000	00067EE17DD) OUT	DX,AL			-24.922 us	000002
-000000375 Disassembly ~		Event:	: Port Out:	Port=00A1,	, Data=000000FF				000001
and see a	-000000154	P0 000000	00067EE17E7	OUT	DX, AL			-17.161 us	
Date Time (000000110	Event	: Port Out:	Port=00/0,	Data=000000B2			0.000	
@11/17/2021 16:35:39.976 r	-000000118	P0 000000	Post Out	Dort=0076	Data=00000005			=0.099 us	
11/17/2021 16:35:49.222	-000000082	P0 000000	DOD640CE1CE	OUT	DY AL			-494 791 nc	
-	00000002	Event	: Port Out:	Port=0CF8.	Data=80000008			434.732 115	
	-000000046	P0 000000	000640CE1E4	OUT	DX, EAX			+0 ns	
	-000000550	Disassembly	Configure	Dicolau	Filtor	Calibrata	Defrech		
		Siddaderritory 4	comgare	Dispidy	11001	Calibrate	Nerreall		
									10
Londing Poset (after), Cill	Icorc) al anc) l	Dogumontalari	um) Source D	oint-TA 7 1	2 15\Magrog\To	allant TCO	Timor Dicable mag		
ROappkEnableForce("cpcie=1"	"teact=1"	Documents (ALL	un (Sourcer)	offic-in_/.1	2.13 (PidC105 (11)	Ter (ADD_1CO	Timer_Disable.mac	· · · · · · · · · · · · · · · · · · ·	^
P0>go til cf8io	, couce-1	,							
P0>go til cf8io									
P0>go til cf8io									
P0>go til cf8io									
P0>									
F1:Help F5:Go Shift+F5:Stop F8:Step In	to F10-Step Over	r Shift+F12-Reset						PD 18: Stopped 64 Bit	Halt Mode

Scrolling up a little, you'll see a mix of Port In/Out and RDMSR/WRMSR, all timestamped.

Power tip: The Last Branch Record (LBR) stack associated with each event can be captured as well. This is a very powerful debugging utility, especially when troubleshooting code execution leading up to events before system memory is initialized and Intel Processor Trace is available. Note: If you try to collect too many events and their associated LBR instruction tracebacks, you will overflow the capacity of the DbC2 connection. Scale these back until the Trace Overflows are eliminated.

ce Configuration				×
BR BTS Trace H	lub AET	Intel PT I	ntel PT Memo	ry
Processors to trace				
List: p0				
	1 (70)			
(e.g., P0, P	+-P7)			
Event sharing				
Apply events to all	processor	_		
	processors	>		
○ Apply events to:		\sim		
				_
Event		Enabled	LBR	
HW/SW Interrupt				
IRET				
Exception				
RDMSR/WRMSR		Y	1 1	
Port In/Out		Y	1 1	
Code breakpoint				
Data breakpoint				
ВТМ				
SMI/NMI/RSM				
MONITOR/MWAIT				
WBINVD				
SGX				
1				
	A	dvanced	Clear all	
	ОК	Canc	el	Help
				•

©2024 ASSET InterTech, Inc.



Intel Processor Trace

Intel Processor Trace (Intel PT) is available only after system memory is initialized.

It's easy to set up. Click on the Trace button in the top toolbar, click the Configure... button, click on the Intel PT tab, put p0 in Processors to trace, and be sure that TSC and Cycle accurate under the Timestamp heading are enabled:

Trace Configuration	×							
LBR BTS Trace Hub AET Intel PT Intel PT Memory								
Processors to trace								
○ None								
● List: p0								
Share filter / timestamp settings Apply settings to all processors								
○ Apply settings to: P0								
Filters								
Range 1: Enter symbol or start-end								
Range 2: Enter symbol or start-end								
CPL: User V								
□ CR3:								
Timestamp								
⊡ TSC								
Trequency: CTC 6								
Cycle accurate Threshold: 0 (fine) ~								
OK Cancel Help								

That's all. Then use the go til cf8io trick to capture some instruction trace data:



🔛 🔁 🔛 🥔 🐨 🖓	ace window help	Load	UEFI Macros	4	5 45 6 15	00 0 U	🕸 🌲 🌋 💿 Br	eakpoints 🖸	Code	> Comman	l 🔝 Log 🎹 M	Aemory IP Re	gisters 🔍 Symb	ools 🧨 Tra	ace 🖲 Viewpoint 🤇
									_						
										-		in all a l			
TE ADDR	Event Trace													×	
ata available - Unab	1e STATE	Pn	ADDR	_	INSTR	JCTION						TIMES	TAMP	^	Stopped (h
	-000000658	PĤ	Event: P	or 67	Intel Processo	r Trace (PO*)									
			Event: P	or	STATE Pn	ADDR	INSTR	UCTION							TIMESTAMP
	-000000622	PO	00000000	67 -	00415 P0	000000005	F0B14C2 MOV	RJ	X, RE	BX					-267.970 us
			Event: P	or	PO	000000005	FOB14C5 MOV	[(00000	00005f0bb0	b8],RDI				
	-000000586	PO	00000000	67	PO	000000005	FOB14CC MOV	RI	3X, [F	RSP]+30					
			Event: P	or	PO	000000005	FOB14D1 ADD	R	SP,00	0000020					
	-000000550	PO	00000000	67	PO	000000005	FOB14D5 POP	RI	IC						
			Event: P	or	PO	000000005	FOB14D6 RETN								
	-000000514	PO	00000000	67 -	-00409 P0	00000006	40CD67D XOR	E	CX, EC	X:					-267.969 us
	101050200000000000000000000000000000000		Event: P	or	PO	00000006	40CD67F MOV	RI	BX, RA	X					
	-000000478	PO	00000000	67	PO	00000006	40CD682 CALL	00	00000	000640ce02	OL				
			Event: P	or	PO	00000006	40CE020 MOV	[]	RSP]+	08,RBX					
	-000000442	PO	00000000	67	P0	00000006	40CE025 MOV	[]	RSP]+	10,RSI					
			Event: P	or	PO	00000006	40CE02A PUSH	RI	DI						
	-000000406	P0	00000000	67	P0	00000006	40CE02B SUB	R	SP,00	0000020					
	and the second sec		Event: P	or	PO	00000006	40CE02F LEA	R	SI,[0	000000064	0cd000]				
	-000000370	P0	00000000	67	PO	00000006	40CE036 MOV	D	LL, CI	à					
			Event: P	or	P0	00000006	40CE039 TEST	CI	L,CL						
	-000000334	P0	00000000	67	PO	000000006	40CE03B JE	00	00000	000640ce12	dL				
			Event: P	or	PO	00000006	40CE12D MOV	RI	AX, [[000000064	0ce338]				
	-000000298	PO	00000000	67	PO	00000006	40CE134 XOR	EI	BX, EE	зх					
			Event: P	or	P0	000000006	40CE136 JMP	00	00000	000640ce14	8L				
	-000000262	PO	00000000	67	PO	000000006	40CE148 TEST	RJ	AX,R#	١X					
000375 Disassembly ~			Event: P	or	PO	00000006	40CE14B JNE	00		000640ce13	8L				
	-000000226	P0	00000000	67	P0	000000006	40CE14D TEST	D	LL,DI	L					
e Mime			Event: P	or	P0	000000006	40CE150 JNE	00		000640ce21	1L				
17/2021 16:35:49 222	-000000190	PO	00000000	61	PO	000000006	40CE156 DEC	10		0000640ce3	58 J				
17/2021 16:51:03 122	H		Event: P	or	PO	00000006	40CE15C MOV	Ež	AX, [[000000064	0ce3b8]				
1772021 10.51.05.122	-000000154	P 0	00000000	67	PO	00000006	40CE162 CMP	E	X,00	000000a	220				
	000000110	-	Event: P	OT	P0	000000006	40CE165 JNC	00	10000	000640ce21	11				
	-000000118	PU	00000000	64	PO	000000006	40CE16B MOV	EA	X, [[000000064	UCe3D81				
	-		Event: P	00	PU	00000000	AOCEI/I LEA	R/	w. 11	MATTRAA~2					
	-000000586	Disass	embly ~	Co	-00409	Disassembly	Configure	Display		Fliter	Calibrate	Refresh			
	-000000586	Disass	embly ~	Co	-00409	Disassembly	Configure	Display		Filter	Calibrate	Refresh			
til cf8io															
til cf8io															
til ofgio															
til cisio															
til afgie															
CIT CI010															

There's a lot more that you can see and do with these Trace utilities. SourcePoint can use Intel PT to display a Call Chart and Call Tree. You can open up Code Tracking windows that update dynamically as you walk through the code, showing you exactly where you are and the interaction between code and events. When you have source and symbols available, the firmware flow becomes much more intuitive and visual. Indulge your curiosity and imagination.

The video at <u>https://www.asset-intertech.com/wp-content/uploads/2021/11/UP-Xtreme-i11-Getting-Started.mp4</u> shows some of these capabilities. The <u>SourcePoint User Guide</u> also provides a very thorough, comprehensive review of the tool. And visit our <u>SourcePoint Academy</u> for helpful "How To" content.



Basic SourcePoint Troubleshooting Tips

At some point, you'll run into something strange. We're the first to admit that JTAGbased run-control and trace are not always deterministic. JTAG is a 30-year hardware protocol, and when something goes astray at a very low level, SourcePoint tries to (but sometimes doesn't) recover gracefully. There will be times that the board will power cycle on its own. Or the firmware thinks that a thread is running but gets out of sync with the SourcePoint software, which thinks it's halted. Or the DbCStatus.exe ball stays red instead of turning green, while you swear you have a good DbC connection. Sometimes you have no choice but to quit SourcePoint and power cycle the target. That usually clears up the one-of's. But if the issue is repeatable, we ask that you collect as much information as you can, and open a ticket with us at <u>https://www.assetintertech.com/support/</u>. We'll respond as soon as possible.

In the meantime, here are a few errata that we've noticed on the UP Xtreme i11, and the steps needed to mitigate.

Trace buffer overflows

DCI traffic processing has its limitations. When you try to collect too much trace data, the trace buffer overflows, causing aberrant behavior.

Although the trace data is highly compressed, some trace sources, specifically with AET, running through the Trace Hub can exceed the capacity of the USB 2.0 connection. In theory running at 480Mbps, in practicality SourcePoint can only process trace data at approximately 100Mbps. Beyond that, we collect ~ 20kB of trace data before a buffer in SourcePoint overflows, and we don't recovery gracefully.

You'll see these symptoms of this occurrence in the SW/FW Trace window:



Image: State Image: State<	
Image: Trace Hub - SW/FW Trace Image: Trace Hub - SW/FW Trace STATE ADDR INSTRUCTION TIMESTAMP Description STATE ADDR INSTRUCTION TIMESTAMP Description Description 00000067 -024348339 AET-18:03-000000007EF94B1 Lake Lake Lake 00000067 -024348320 AET-18:03-0000345000047ADE Lake Lake Lake 00000067 -024348321 AET-18:03-0000345000047ADE Lake Lake Lake 00000067 -024348322 AET-18:03-0000000007F99621 Lake Lake Lake 00000067 -024348755 AET-18:001-000000000001800 Description Lake Lake 00000067 -024348756 AET-18:001-00000000000000000000000000000000	
der@PD/LG STATE ADDR INSTRUCTION TIMESTAMP Description 00000067 -024348849 GLOBAL-0249700000 TIMESTAMP Description 00000067 -024348849 GLOBAL-0249700000 TIMESTAMP Description 00000067 -024348830 AET-18:001-0000180800EFD2C1 Lake Lake 00000067 -024348831 AET-18:001-0000000067EF94B1 Lake Lake 00000067 -024348812 AET-18:001-0000000067EF94B1 Lake Lake 00000067 -024348743 AET-18:001-0000000067EF94B1 Understamp Lake 00000067 -024348745 AET-18:001-00000000000000001808 Understamp Understamp 00000007 -024348776 AET-18:001-00000000000000000000000000000000	
JALE RUX LIAS INCLING Description 10000067 024448349 GLOBAL-0249900000 Description 00000067 024448389 AET-18:C01-000000007EFF4B1 Lake 10000067 024448380 AET-18:C01-0000140800EFF2C1 Lake 10000067 024448380 AET-18:C01-0000140800EFF2C1 Lake 10000067 024448380 AET-18:C01-0000140800EFF2C1 Lake 10000067 024448725 AET-18:C01-0000000007EFF4B1 Lake 10000067 024448725 AET-18:C01-0000000007EFF4B1 Lake 10000067 024448775 AET-18:C01-00000000000000000000000000000000000	
0000067 -02393830 AET-18:C01-0000000067EF94B1 Lake 0000067 -023948830 AET-18:C01-00001000067EF94B1 Lake 0000067 -023948821 AET-18:C01-000000007EF94B1 Lake 0000067 -023948821 AET-18:C01-00000000FFE94B1 Lake 0000067 -023948821 AET-18:C01-00000000FFE94B1 Lake 0000067 -024948735 AET-18:C01-00000000FFE94B1 Lake 0000067 -024948776 AET-18:C01-000000000FFE94B1 POT 0000067 -024948776 AET-18:C01-0000000000FFE94B1 POT 0000067 -024948776 AET-18:C01-0000000000FFE94B1 POT 0000067 -024948776 AET-18:C01-000000000FFE94B1 POT 0000067 -024948776 AET-18:C01-000000000FFE94B1 POT 0000076 -024948776 AET-18:C01-000000000000FFE94B1 POT 0000076 -024948776 AET-18:C01-00000000000000000000000000000000000	tion State
D000000000000000000000000000000000000	Stopped
0000000 -024948821 AET-18:C01-000345000047ADE Lake 000000 -024948812 AET-18:C01-00000000067999621 Lake 000000 -024948803 AET-18:C01-00000000067999621 Lake 000000 -024948735 AET-18:C01-00000000067199621 Lake 000000 -024948735 AET-18:C01-000000000001808 P01 000000 -024948735 AET-18:C01-00000000001808 P01 000000 -024948766 AET-18:C01-00000000007EF94B1 P01 000000 -024948767 AET-18:C01-00001000000FFE94B1 P01 000000 -024948769 AET-18:C01-0000100000FFE94B1 P01 000000 -024948769 AET-18:C01-0000000000FFE94B1 0000 000000 -024948749 AET-18:C01-000000000000FFE94B1 0000 000000 -024948740 AET-18:C01-00000000000FFE94B1 0000 -024948731 AET-18:C01-000000000000FFE94B1 0000 -024948731 AET-18:C01-0000000000000FFE94B1 00000 -024948731 AET-18:C01-00000000000001 +333.522 sec 0000 0000000 -024948709 GLOBAL-01 0000 000000000000000000000000000000000000	Stopped
100006 -024948912 AFT-18:C01-000000C0A7999C1 Lake 100006 -024948912 AFT-18:C01-000000C0A7999C1 Lake 100006 -024948913 AFT-18:C01-000000C0A79940C1 P000000000000000000000000000000000000	Stopped
0000007 -023948903 AET-18:C01-0000000067EF94811 0000007 -023948734 AET-18:C01-00000000000000000000000000000000000	Stopped
000006 024948794 AET-18:C01-000000000001008 000006 024948785 AET-18:C01-000000000478 000006 024948785 AET-18:C01-00000000478 000006 024948786 AET-18:C01-00000000078 000006 024948786 AET-18:C01-0000000078 000006 024948785 AET-18:C01-00001600007878 000006 024948789 AET-18:C01-0000000007878 000006 024948749 AET-18:C01-0000000007878 00000 000006 024948743 AET-18:C01-00000000007878 00000 000006 024948743 AET-18:C01-00000000007878 00000 000006 024948743 AET-18:C01-0000000000001 +333.522 sec 00000 000006 024948707 GLOBAL-00000000000001 +333.522 sec 00000 000006 024948707 GLOBAL-01 00000 00000 000006 024948707 GLOBAL-01 00000 00000 000006 024948707 GLOBAL-01 00000 00000 00000 000006 -024	beopped
-024948785 AFT-18:C01-000325000047ADE 0000067 -024948776 AET-18:C01-000000007930105 0000067 -024948767 AET-18:C01-000000007930105 0000067 -024948767 AET-18:C01-000000007930105 0000067 -024948769 AET-18:C01-00001000000000000000000000000000000	
000006 -024948776 ABT-18:C01-00000000787834015 PC* 000006 -024948776 ABT-18:C01-00000000787854015 B Value 000006 -024948776 ABT-18:C01-000010000787854015 B Value 000006 -024948778 ABT-18:C01-0000180000078785401 0000 0000 000006 -024948738 ABT-18:C01-0000000000787854315 00000 0000 000006 -024948731 ABT-18:C01-00000000000001 +333.522 sec 0000 000006 -024948731 GLOBAL-0000000000001 +333.522 sec 0000 000006 -024948709 GLOBAL-00000000000001 +333.522 sec 0000 000007 -024948707 GLOBAL-000000000000000000000000000000000000	
-022448767 AFT-18:C01-000000007EF94B1 POI 000006 -022448758 AFT-18:C01-00000000F7EF94B1 B Walue 00006 -024948758 AFT-18:C01-00001000EF1202C B 00000 00006 -024948749 AFT-18:C01-00000000F1E94B1 0000 00000 00006 -024948731 GLOBAL=00000000000001 +333.522 sec 0000 00006 -024948739 GLOBAL=000000000000001 +333.522 sec 0000 00006 -024948709 GLOBAL=0000000000000492 +333.522 sec 0000 00007 -024948707 GLOBAL=0000000000000492 +333.522 sec 0000 00006 -024948707 GLOBAL=000000000000000000000000000000000000	
-024948758 AFT-18:CO1-0000180800ET22C P Value 000006 -024948749 AFT-18:CO1-0004500004FADE 0000 000006 -024948740 AFT-18:CO1-0004500004FADE 0000 000006 -024948740 AFT-18:CO1-0004500004FADE 0000 000006 -024948731 AFT-18:CO1-00040000001FFF94B1 0000 000006 -024948721 GLOBAL-0000000000001 +333.522 sec 0000 000006 -024948720 GLOBAL-0100000000000000000000000000000000000	
000006 -024948749 AFT-18:C01-00034500004FADE 0000 000006 -024948740 AET-18:C01-0000000007933C49 0000 000067 -024948731 AET-18:C01-0000000000001 +333.522 sec 0000 000067 -024948720 GLOBAL=00000000000001 +333.522 sec 0000 000067 -024948709 GLOBAL=0100000000000000000000000000000000000	
000067 -024948740 ABT-18:C01-000000000A79A3C49 00000 000067 -024948731 ABT-18:C01-0000000000001 +333.522 sec 00000 000067 -024948721 GLOBAL-0000000000001 +333.522 sec 00000 000067 -024948721 GLOBAL-0000000000001 +333.522 sec 0000 000067 -024948707 GLOBAL-000000000000000000000000000000000000	000FED000FF
000000000000000000000000000000000000	0005F0A4C60
000067 -024948721 GLOBAL=0000000000000001 +333.522 sec 0000 000067 -024948707 GLOBAL=01 0000 0000 00067 -024948707 GLOBAL=01 0000 0000 00567E11 -024948707 GLOBAL=01 0000 0000 00567E11 -024948707 GLOBAL=000000000000000000000000000000000000	/0000000001
000067 **** Overflow - multiple masters *** 000067 -024948709 GLOBAL=01 0007671 -024948707 GLOBAL=000000000000492 +333.522 sec 0024948707 GLOBAL=000000000000492 +333.522 sec 0000 0024948696 AFT=18:C01=00000CA3115F 000000	000000000A1
0000761 -024948709 GLOBAL-01 00077617 -024948707 GLOBAL-000000000000492 +333.522 sec 0000 -024948766 AFT-18:C01-000025000047ADE +333.522 sec 00000	00063A335D0
-024948707 GLOBAL=0000000000492 +333.522 sec 0000 -024948596 AET=18:C01=000025000047ADE 000000000000000000000000000000000000	0005F0A4C68
006/EEII -024948696 AET-18:C01-00032500004FADE 00000	0005F0A4CA8
-024948687 AFT-18+C01=000000C0A7A11F1F	0005F0A4BE8
	00000000000
-024948678 AET-18:C01=000000067EF94B1 8000	0000000000E
-024948669 AET-18:C01=0000180800EFD31E 0000	000000000000
-024948660 AET-18:C01=000345000004FADE 00000	10005F0A4B80
-024948651 AET-18:C01=00000C0A7A12C09 0000	0000000000
-024948642 AET-18:C01=000000067EF94B1	00000000000
-024948633 AET-18:C01=00000000001808 0000	000633308F0
-024948624 AET-18:C01=000325000004FADE	
-024948615 AET-18:C01=00000C0A7A1D005	
-024948606 AET-18:C01=000000067EF94B1	
7/2021 024948597 AET-18:C01=0000180800EFD324	
7/2021 024948588 AET-18:C01=000345000004FADE	
-024948579 AET-18:C01=000000C0ATAIDC3D	
-024946570 RE1-161C01=000000067E94B1	
-024949259 Disassembly v Configure Display Filter Calibrate Refresh	
til cfRio	
til cf8io	
til cf8io	
g Reset (after): C:\Users\alans\Documents\Arium\SourcePoint-IA 7.12.15\Macros\Intel\ADL TCO Timer Disable.mac	
EnableForce("cpcie=1", "tsact=1")	
FSGo. Shift+E5:Stop. F8:Step Into. F10:Step Over. Shift+F12:Reset P0 18: Stopped 64.8	

A few of these overflows are no big deal. But, if you're tracing a huge amount of data, SourcePoint may spin, as it tries to process all that data, and deal with the mess. Sometimes, after maybe a few minutes, it recovers. Sometimes, you end up in limbo.

The only solution at this point is to quit SourcePoint, do an End task on the AssetDCI Background process, power cycle the target, and start over:



💐 Task Manager	_	•	×					
File Options View								
Processes Performance App history Startup Users Details Services								
^	31%	84%						
Name Status	CPU	Memory						
Adobe Collaboration Synchronizer 21.7 (32 bit)	0%	1.2 MB	^					
😂 Adobe Collaboration Synchronizer 21.7 (32 bit)	0%	1.1 MB						
🚔 Adobe Collaboration Synchronizer 21.7 (32 bit)	0%	2.0 MB						
🚔 Adobe Collaboration Synchronizer 21.7 (32 bit)	0%	2.2 MB						
> Adobe Genuine Software Integrity Service	0%	1.2 MB						
> III Adobe Genuine Software Service	0%	1.7 MB						
Adobe Installer	0%	0.9 MB						
🗐 Adobe IPC Broker (32 bit)	0%	2.8 MB						
> 💿 Adobe Update Service	0%	1.1 MB						
Application Frame Host	0%	8.3 MB						
🔳 ariumImd daemon	0%	1.4 MB						
🍇 AssetDCI (32 bit)	0%	51.5 MB						
CCLibraries	0%	0.1 MB						
CCXProcess	0%	0.1 MB	v					
<			>					
○ Fewer details								

Ultimately, we're working on improving the performing of the AssetDCI driver, and behaving more graciously when an overflow is encountered. But, in the interim, it is key to ensure that the trace data collected is relatively "sparse". Focus your debugging in the specific area of interest. Don't try to collect all Port IN/OUT from the reset vector all the way through to the UEFI shell. At 750K I/Os per second, you'll swamp the host debugger processing, and you can't deal with all that data anyway.

Note that this only happens with AET – that is, you can only overflow by collecting too much AET data. It does not apply to LBR, SW/FW Trace, SVEN, or Intel PT. It may take some trial and error to limit the scope of your event data collection.

Intel Processor Trace can be slow

When you configure Intel PT, you can specify the size of the buffer in system memory to collect trace data:



Trace Configuration		×						
LBR BTS Trace	Hub AET Intel PT	Intel PT Memory						
Trace buffer								
○ Use processor settings								
• Use SourcePoint settings:								
Base address:	10000000P	P						
Length per core:	16k ~							
Trace capture mode	32k 64k 128k							
 Overwrite 	256k 512k							
○ Append	1M 2M 4M							
	8M 16M							
	32M 64M 128M							
	256M 512M							
	1G 2G							
	20							
[OK Ca	ancel Help						

Note that Intel PT directs instruction trace data to system memory. Although highly compact and efficient, we are not streaming over DCI to the host in this case; we buffer the code execution data until the platform is halted, at which point SourcePoint uses JTAG over DCI to collect the trace data out of system memory, reconstruct and display

©2024 ASSET InterTech, Inc.


it. JTAG operates at fairly low speeds, and for large buffer sizes the transfer of all that data can be slow. It starts to become noticeable beyond a buffer size of 64kB.

SourcePoint compensates for that by only pulling in the trace data that's local to the code display you are in. Scrolling up in the Intel PT window will prompt SourcePoint to pull in more trace data, and display it. Be patient when this happens. If you try to collect 1GB of data from system memory over JTAG, it will take a long while for this all to be available. Displaying the Call Chart for this much data will take a while. You'll see this after initial trace data collection (that only takes a few seconds), but scrolling to the top of the Intel PT window with a 1MB trace buffer (which gives you ~ 300ms of execution trace data) takes ~30 seconds, and then doing a Call Chart takes much longer. Be patient.

Building tra	ice display
Status:	Analyzing Trace
Progress:	
	Processing trace This may take a few moments.
	Cancel

Analyzing t	race data	
Calls:	3048	
Progress:		4.0%
		Suspend



My board is not booting – what now?

Once in a while, especially during an intense debug session, we have found that the target goes into la-la land. You get to the UP splash screen, and then it just stops. Or the screen stays black. SourcePoint run-control continues to work, but it won't boot all the way up to the UEFI shell. Quitting SourcePoint, unplugging the DCI cable, killing the AssetDCI process, knocking out the AssetDCI process – all are good steps in this instance, when you need to recover.

But then, once in a while, you wait the needed 20 seconds; and it doesn't boot. The screen stays blank or frozen. We know that happens with the newest Celerons, and there's a <u>workaround</u> that involves unplugging and reconnecting the DCI cable, but this might not be the issue.

Don't panic! For reasons we're not sure of just yet, after about 60 seconds, the board "wakes up" and should boot all the way to the UEFI shell.

Clearing the CMOS on the target has also been known to help when it still won't boot up. There's only one thing: it has gone back to the factory settings, so you'll need to reset the WDT timer, and perform the other steps, as per <u>BIOS Settings</u> section in this manual.

Then, you'll be back in business.



SOURCEPOINT DEBUGGING FOR WINDOWS

Once you've configured the target and become familiar with the use of SourcePoint for basic UEFI debugging as described above, you can begin debugging Windows in earnest.

This section describes the use of SourcePoint WinDbg on a target with Hyper-V disabled. For debugging of Hyper-V, skip ahead to the section on <u>SourcePoint</u> <u>Debugging for Hyper-V</u>.

SourcePoint has implemented a connection to Microsoft's WinDbg application by using a public interface known as EXDI. A block diagram of how WinDbg is integrated with our SourcePoint debugger is as below:





The EXtended Debug Interface (EXDI) is used to connect a WinDbg debugging session to an existing SourcePoint JTAG-based connection to a target.

WinDbg is the controller in all transactions over EXDI, and SourcePoint is the worker. That is, the solution is most stable when run-control based operations (that is, Break, Go, single-step, etc.) are initiated via WinDbg. There are exceptions, particularly in the cases of using enhanced breakpoints for Hyper-V debug and Intel Trace features, that we will discuss later. But, in general, WinDbg issues debug primitive commands down to SourcePoint, which in turn uses JTAG-based run-control to perform operations on the target. Then, SourcePoint presents the results data back to WinDbg over the EXDI connection.



Power Tip: The UP Xtreme i11 boots to the UEFI shell when initially purchased. It is necessary to install Windows on the target. There are numerous references online on how to do this: it is recommended to go to the AAEON https://github.com/up-board/up-community/wiki/Windows-GSG site for helpful tips. In terms of driver installation, in most cases all that's needed is to install the Intel Graphics (igxpin.exe) – to improve the monitor resolution – and, optionally, the Intel LAN.

Power Tip: Be sure that your target has sufficient memory and storage to accommodate your Windows debugging needs. We typically recommend 16GB RAM, and a 256GB SSD.

Before we get started, the target needs to be configured to not interfere overmuch with JTAG-based run-control. Then, the steps needed to set up a debugging session will be covered.



Configuring the target and setting up pre-requisites – Getting Started

We first need to prevent Windows from changing power states from disrupting runcontrol prematurely, and VMX and VBS need to be disabled.

These steps are highly recommended (as of the time of writing) to have a successful initial debugging session, especially for newcomers to SourcePoint WinDbg. **Once** experience with SourcePoint is gained, Hyper-V can be turned back on again.

To adjust the power settings in Windows, open the Control Panel > Hardware and Sound > Power Options > Change plan settings > Change advanced power settings and set these per the below (use High performance dropdown). It also helps to set "Turn off the display" and "Put the computer to sleep" both to "Never":

🗃 Edit Plan Settings				-	- ×
\leftarrow \rightarrow \checkmark \uparrow 🦃 \diamond Control Panel	> Hardware and Sound > Powe	r Options > Edit Plan Settings	~	C Search Control Panel	Q
	Change settings for the p Choose the sleep and display set	olan: High performance tings that you want your computer to u	se.		
	🕑 Turn off the display:	Never	1	Power Options ?	×
	Put the computer to sleep:	Never \checkmark		Advanced settings Select the power plan that you want to customize,	and
	Change advanced power setting	s		 then choose settings that reflect how you want you computer to manage power. Change settings that are currently unavailable 	IT
	Restore default settings for this p	blan		High performance [Active]	_
			Save c	 Sieep Sleep after Setting: Never Allow hybrid sleep Setting: Off Hibemate after Setting: Never Allow wake timers Setting: Enable USR settings 	llts
				OK Cancel	Apply

For Windows VBS, go into Windows Security > Device security > Core isolation details, and ensure that Memory Integrity is off:





For VMX, boot the Tiger Lake board to BIOS settings menu (pressing the F7 key when restarting), enter the Advanced BIOS Setup (by entering the password upassw0rd) and follow the menu path CRB Setup > CRB Advanced > CPU Configuration and change "Intel (VMX) Virtualization Technology" to **Disabled**. Save and exit (pressing F4) and the target will reset.

Power Tip: Go to CRB Setup > CRB Advanced > Platform Settings > VTIO and make sure it is set to **Disabled**. This is the default in the AAEON Tiger Lake Debug BIOS, but it's worthwhile checking.

One last thing: To avoid the WinDbg error message "Unable to read debugger data block header" that indicates kernel debugging is not enabled, execute the command:

>bcdedit /debug on

on the target from an Administrator command prompt, then reboot the target. Note that this is not absolutely necessary if you're solely going to be using SourcePoint (with no WinDbg connection) for your debugging.

Now you're ready to set up a debugging session.



How to Establish a SourcePoint WinDbg Session

NOTE: With SourcePoint WinDbg, there is no need for the kdnet Ethernet connection, as all the traffic is over EXDI and the specialty USB cable.

Three steps are needed to begin a debugging session with SourcePoint WinDbg:

- 1. Connect SourcePoint to the target
- 2. Start WinDbg via a SourcePoint macro
- 3. Load symbols with the LoadCurrent macro.

Step 1: Connect SourcePoint to the target

Boot the target to Windows. Log into the Windows desktop.

Follow the steps as described in the <u>Getting Started with SourcePoint</u> section above.

Halt the target by hitting the Stop button in the SourcePoint Icon Toolbar at the top:



You will have to hit the Refresh button to see code displayed in the Code window. This transitions the Code view out of Safe Mode.

Your screen should then look something like this:



Now, you have a choice to set up your environment for Windows or UEFI debugging in this session.

Click on the Load WinDbg Macros button at the top of the screen. This will enable a number of new "Windows debugging friendly" macros available for later use. The screen should look like this:

	File Edit View Processor Options Code Window Help			
Outcome: Bit is a Lower Bit Newsy Farme Symper Line: Image: Symper Line: Image: Symper Line: Outcome: Bit is a Lower Line: Image: Symper Line: Image: Symper Line: Image: Symper Line: Outcome: Symper Line: Image: Symper Line: Image: Symper Line: Image: Symper Line: Image: Symper Line: Image: Symper Line: Image: Sympe Line: Image: Symper Line:	8 8 9 8 9 6 8 6 6	🏶 StartWinDbgC 🖷 StartWinDbgX 📽 LoadCurrent 📽 Load/	📲 🌇 LoadedModuleList 🖷 CachedModuleList 📽 Enabl	eTraceHub 🍪 🥵 📽 😁 🖻 🖬 🗊 🗊 🕸 🕸 🖉
	🐵 Breakpoints 🤁 Code > Command 📔 Log 📕 Memory IP Registers 🎕 Symbols 💉 Trace 👀 Viewpoint 🔍 Watch			🔋 📃 A 🛱 🐕 💁
Image: control of the control of th	GH Code (P0*): (64-bit) Tracking IP 00000000000000000000000000000000000	🙀 Symbols (P0") - Globals	D Uiewpoin	t 🗆 🖽 🖾
Comment Contrast Component Contrast Component Contrast Component Contrast Contrast	Construct <	Image: Symbolic (Phys.: Goldale Name Address C C Image: Symbolic (Locales) (Stack) Classes /	Control C	Control C
Component Message 05-03-2024 17:00:56.557 BuildBassalines Message Command Edit Command Component Constraint Party Component Constraint Party Component Constraint Party Command Constraint Party Component Constraint Party Constraints Constraint Party Constraints Constraints Constraints Constraints Constraints Constraints Constraints Constraints Constraints Constreations Constraints	had a second sec			
Consecting Londing Command Language Extensions: C:\Users\alans\Documents\Arius\SourcePoint-IA_7.12.5}\Macros\aa\aaextend mac P) v	Deter Time Component Message Description Translation error in FFFF70 Deter 2024 171.00.56.557 BuildDasamlines Description Component Command Scanning Moore Configuring Moore	902482 F44A2I	Edt	Add Ramove Ramove Al Enable
	Longering Longer	s∖aa\aaestend.nac		

You now have extra buttons showing up, including:

StartWinDbgC StartWinDbgX LoadCurrent LoadAll LoadedModuleList CachedModuleList This initiates a debug session with WinDbg Classic This initiates a debug session with WinDbgX This load symbols into SourcePoint at the current RIP Loads all current context symbols into SourcePoint Displays all loaded modules Shows the module list that is currently cached

Step 2: Start WinDbg via a SourcePoint macro

Next, it is time to run the SourcePoint macro that launches WinDbg and establishes the EXDI connection. For simplicity, click on either StartWinDbgC (Classic) or the StartWinDbgX macro button at the top of the screen. After about 30 seconds, WinDbg will open:



20 eUUI eddCLSUU(SYA1522-8847-4822-3.697-183.07-59725).kds.VerAddr-87050999/105,UatabreassExdr - WenDbg-100.22621.2425 AMU04 File Fide View Debug Window Heln		- 0 ×
Disastembly	Registers 🔤	Calls R 🔤
Offset (#Sscopeip Previous Next	Customize	Raw args Func info Source Addrs Headings Nonvolatile regs Frame nums
ffff802'75574/d9 CC int 3 ffff802'75574/d9 CC int 3	Reg Value ^	Source args
fffff802'755747db oc int 3	rax fffff8027557dcd0	More Less
fffff802 755747dd oc int 3	rcx ffffal8e5e6fdbb0	
Iffff802 755747de cc int 3 fffff802 755747de cn int 3	rds 0	Child-SP RetAddr Call Site
fffff802/755747e0 oc int 3	TDX U	fffff802'4ba74f88 fffff802'7557dcea 0xfffff802'755747f2 fffff802'4ba74f90_00000000'00000000
ffff802'75574'PL CC Int 3 ffff802'75574'PL CC int 3	rbp 0	
fffff802'755747e3 cc int 3 fffff802'755747e3 cc int 3	rsi ffffal8e5f011520	
fffff802'755747e5 cc int 3	rdi 0	
ffff802 7557476 bbb011840000000 ncp word pr [rax+rax]	18 U	
fffff802?755747f1 f4 hlt	r10 fffff80275571400	
fffff802 755747f3 cc int 3	r11 0	
ffff802'755747f4 cc int 3 ffff802'755747f5 cc int 3	r12 ffffffff	
fffff802'75574766 oc int 3	r13 ttttt80243a4c180	
ffff802'755747f oc int 3	r14 15148/81/	
fffff802'755747f9 0f1f800000000 nop dword ptr [rax] ffff800'7557400 4053	rip fffff802755747f2	
ffff802 75574802 8bc1 nov eax.ecx	ef1 50246	
IfIfI802/75574804 4C8DCa ROV 79.708 Iffff802/75574807 33db xor ebx.ebx	cs 10	
fffff802'75574809 33c9 xor ecx.ecx	es 1520	
fffff802 7557480d 0fa2 cpuid	fs 53	
IPTPTRI///SSZ2RIF 218901 BAO //BOWLDPY LPTF 244	gs 2b	
Command 2	ss 18 dr0 0	
EXDI: ExdiMotifyRunChange::Initialize() returned 0x00000000 EXDI: tueNerweitIng(:Initialize() returned 0x00000000000000000000000000000000000	dr1 0	
EXDI: Target initialization succeeded	dr2 0	
Connected to Windows 10 19041 x64 target at (Sat Apr 27 12:02:29.051 2024 (UTC - 5:00)), ptr64 TRUE	dr3 0	
Error: Change all symbol paths attempts to access 'C:\mysymbols' failed: 0x2 - The system cannot find the file specified.	dr6 11110110 dr7 400	
FARMANNER Path validation sunnary ************************************	fpow 0	
Nesponse Time (as) Location	fpsw 0	
Error C:\wysyabla Deferred CPVE::/www.blattte://aedl.nicrosoft.com/download/crahols	fptw 0	
OK C:\ProgramLata\dp\sym	st0 0.0000000000000000000000000000000000	
Symbol search path is: srv*;C:Nysymbols;SNV*c:Nsymbols*https://wsdl.wicrosoft.com/download/symbols;C:NProgramData/dbg/sym Executable search path is:	st2 0.000000000000e+000	
Loading symbols for ffff802'48c00000 ntkrnlnp.exe -> ntkrnlap.exe	st3 0.0000000000000e+000	
Windows 10 Kernel Version 19041 NF (8 procs) Free x64	st4 0.000000000000e+000	
Product: WinNt, suite: TerminalServer SingleUserIS Edition build lab: 19041 1 and64fre yb release 191206-1406	st6 0.000000000000000000	
Hachine Name:	st7 0.00000000000e+000	
Debug session time: Sat Apr 27 11:159 90 2024 (UTC - 5:00)	nm0 0:0:0:0	
System Uptime: 0 days 0:11:36.618 Laaded Usbelp extension DLL	na1 0:0:0:0 na2 0:0:0:0	
Loaded exts extension DLL	na3 0:0:0:0	
Loaded Kext extension DLL	na.4 0:0:0:0	
fffff802'755747f2 c3 ret	na5 0:0:0:0	
< >	nat5 0:0:0:0 na7 0:0:0:0	
0: kd>	< · · · · · · · · · · · · · · · · · · ·	Calls Locals Processes and Threads
	1	
		Ln 11, Col 43 Sys 0:eXDI KD Proc 000:0 Thrd 000:0 ASM OVR CAPS NUM

Power Tip: You have a choice of launching WinDbg Classic or WinDbgX via the macro buttons at the top. With this release of SourcePoint, *WinDbg Classic is more stable and higher performance*. See our <u>Troubleshooting</u> section. Alternatively, at the SourcePoint Command line, you can type in StartWinDbg(true) to start a WinDbg Classic session, or StartWinDbg(false) to start a WinDbgX session.

Power Tip: Ensure an environment variable _NT_WINDBG_VBS has been created, and set it to FALSE.

The target will be halted as part of this process, assuming the VBS override environment variable (_NT_WINDBG_VBS) has not been set:





Variable	Value	^
GPU_FORCE_64BIT_PTR	0	
GPU_MAX_ALLOC_PERCENT	100	
GPU_MAX_HEAP_SIZE	100	
GPU_SINGLE_ALLOC_PERCE	100	
GPU_USE_SYNC_OBJECTS	1	
OneDrive	C:\Users\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	1
	New Edit Delete	
stem variables	New Edit Delete	
stem variables Variable	Value	
stem variables Variable _NT_WINDBG_VBS	Value FALSE	
stem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE	Value FALSE EXDI	
stem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET	New Edit Delete Value FALSE EXDI C:\ScanWorks	
stem variables Variable <u>NT_WINDBG_VBS</u> _NT_WINDBG_WORKSPACE ASSET ComSpec	New Edit Delete Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe	
stem variables Variable <u>_NT_WINDBG_VBS</u> _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR	New Edit Delete Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\UIntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	
stem variables Variable <u>NT_WINDBG_VBS</u> _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData	New Edit Delete Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\WINDOWS\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData	
stem variables Variable <u>NT_WINDBG_VBS</u> _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK_DIR	New Edit Delete Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\UINDOWS\system32\cmd.exe C:\UIntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	

SourcePoint will then look for the KdVersionBlock structure, read the kernel memory and retrieve all the symbol information needed to match what WinDbg has (in terms of the Microsoft symbol server, or a local symbol cache). If you have the SourcePoint Log window open, you may see the symbol information being uploaded, but only for WinDbgX:

	1	
Tiae Component Message 00000606 69652855 6412535 2013255 52726557 *ianedBob.CcDireg* 0000050 69752844 2557259 2557257 *iateExternalCase 0000050 6975641 2557259 2557257 *iateExternalCase 0000050 657745 125725 255745 724525 *iateExternalCase 0000050 6976674 7904363		

If you don't have the Log window open, you will nonetheless see the SourcePoint "Dashboard Lights" at the bottom right lighting up as the JTAG-based memory reads are done:





When the symbol load is complete, you will see that WinDbg and SourcePoint break at the same place.

The SourcePoint Code and WinDbg Disassembly window show the same location. Both are typically (but not necessarily) halted on logical processor 0, at a RET instruction, as can be seen in the above image.

Step 3: Load symbols with the LoadCurrent macro

Symbols that are visible to WinDbg have to be made visible to SourcePoint as well, if we're going to get the most out of the joint solution.

SourcePoint has the ability to view Windows' symbols on its own, with no connection to WinDbg. To see what this looks like, just launch SourcePoint and load your Project, and follow the following steps:

Ensure that the target is in a Stopped state.

Click on the LoadCurrent macro button in the SourcePoint Icon toolbar at the top. After about 20 seconds, the SourcePoint Symbols window will display the module that the current instruction is in (skip to the next section, <u>What happens if the symbols don't</u> <u>show up?</u> if it does not):



🗣 Symbols (P0*) - Globals		
Name	Address	Va
<		>
Globals Locals A Stack A	Classes /	

Interestingly, SourcePoint will display the symbols associated with intelppm.pdb (sometimes). WinDbg does not generally display those symbols.

Expand the Labels within the Symbols window, and then you will see it populated with all functions that are in the current module, for example:



😪 Symbols (P0*) - Globals		×
Name	Address	^
	FFFFF8075F1A7F58L	
	FFFFF8075F0C54C0L	
	FFFFF8075F0C280CL	
	FFFFF8075F3BBE74L	
	FFFFF8075F3BBF94L	
	FFFFF8075F3B74C8L	
f PspSysAppIdClaim	FFFFF8075F4779A0L	
	FFFFF8075F3B8FFCL	
f PspSyscallProviderServiceDispatch	FFFFF8075EE38BD0L	
f. PspSyscallProviderServiceDispatchGeneric	FFFFF8075F3B91A8L	
	FFFFF8075F4771E8L	
	FFFFF8075F59E4E0L	
	FFFFF8075F477208L	
	FFFFF8075F477218L	
■ fx PspSystemRootSymlinkName	FFFFF8075F4779C0L	
	FFFFF8075F4771D8L	
—	FFFFF8075ED569A0L	
	FFFFF8075F3BA740L	
── f. PspTerminateAllProcessesInJobHierarchy	FFFFF8075F0A56F8L	
	FFFFF8075F1B3830L	
	FFFFF8075F3B9CF0L	
PenTarminataProcase	FFFFF8075F047624T	×
	>	
GIODAIS ALOCAIS AStack Aclasses /		

Power Tip: If WinDbg accesses symbols outside of intelppm.pdb (which it will during any typical debugging session), you'll need to run another LoadCurrent to additionally access these new symbols within SourcePoint.

Power Tip: If you want to load the symbols for an address outside of the current module or context, say at address 0xFFFF80643C46000, in the SourcePoint Command window type in LoadCurrentWinDbg(FFFF80643C46000).

Power tip: Right-click on a function name within the SourcePoint Symbols window, and you'll see a rich number of capabilities that can be applied to that function, such as setting breakpoints, opening the function's Code window, etc.

All the Windows kernel function name symbols are displayed in the SourcePoint symbols window, under the Globals tab. You can right-click in the window to see the function addresses as well as function names. Right-clicking on a function name gives you the context-sensitive options to work with these functions:





Now, it is possible to see the power of the two applications applied together. As an example, go into WinDbg and set a breakpoint on the entry point to the function MmCreateProcessAddressSpace:

bp nt!MmCreateProcessAddressSpace

Then hit Go within WinDbg.

Sometimes the breakpoint is hit right away. You might need to move the target's mouse around, or open a window on the target, before the breakpoint is hit.

You can then see the break in both applications. Do a LoadCurrent from within SourcePoint. You can see that the Code windows between the two applications are symmetrical:





😎 eXDI 'exdi:CLSID=(B5FA1822-6847-4A22-A369-216370F58F2B],Kd=VerAddr-8786269899768,DataBreaks=Exdi' - WinDbg:10.0.22621.2428 AMD64						– a ×
File Edit View Debug Window Help						
Disassembly				Calls		2 -
Offset @\$scopeip	Previous Next	Customia	£	Raw arc	gs Func info Source Addrs Headings Nonvolat	ile regs Frame nums
fif(f2) 47.54 fba 32.60 xor ext.ext fif(f2) 47.54 fba 32.60 xord ptr [rmp+6h] fif(f2) 47.54 fba 42.60 xord ptr [rmp+6h] fif(f2) 47.54 fba 42.60 xord ptr [rmp+6h] fif(f2) 47.54 fba add rmp ptr fba fif(f2) 47.54 fba add ptr fba fba fif(f2) 47.54 fba add ptr fba fba fif(f2) 47.54 fba add ptr fba fba fif(f2) 47.54 fba add add fba fba <td< td=""><td></td><td>Reg rax rcx rdx rbx rsp rbp rsi rdi rfi r10 r11 r12 r13 r14 r15 rip ef1 cs fs</td><td>- Value 1 (ff:1865%bbes) 0 0:59% 0 0:59% 1 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 1 (</td><td>Child Griff Child</td><td>Provide 2007 ADD Parage Notes 1-20 Provide 2007 ADD Parage 1-20 Provide 2007 ADD Parage 1-20</td><td>Call Bits Call Bits 1 Papel LocateProceeds 1 Papel LocateProceed 1 BitCreateDesrProce 0x00007ffe 37ede8f4</td></td<>		Reg rax rcx rdx rbx rsp rbp rsi rdi rfi r10 r11 r12 r13 r14 r15 rip ef1 cs fs	- Value 1 (ff:1865%bbes) 0 0:59% 0 0:59% 1 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 0 (ff:199417648) 1 (Child Griff Child	Provide 2007 ADD Parage Notes 1-20 Provide 2007 ADD Parage 1-20	Call Bits Call Bits 1 Papel LocateProceeds 1 Papel LocateProceed 1 BitCreateDesrProce 0x00007ffe 37ede8f4
Ifffff802'492a9206.498he8 and Phn n8		gs	2b			
Command	2. 1	ss dr0	18			
Connected to Windows 10 19041 x64 target at (Sat Apr 27 12:15:20.311 2024 (UTC - 5:00)), ptr64 TRUE Error: Change all symbol paths attempts to access 'C:\maysymbols' failed: 0x2 - The system cannot find the	file specified. ^	dr1	0			
Path validation summary ************		dr2	0			
Response Time (as) Location		dr6	11110110			
Error C. nysyabols		dr7	400			
Deterred SNV*C: %syabols*https://nsdi.microsoft.com/download/syabols OK C:\ProgramData/dbg\sya		fpcw	0			
Symbol search path is: srv*;C:\nysymbols;SRV*c:\symbols*https://wsdl.wicrosoft.com/download/symbols;C:\Pr Evenutable_search_path_is:	ogramData\dbg\syr	fpty	0			
Loading syabols for ffff802'48c00000 ntkrnlap.exe -> ntkrnlap.exe		st0	0.000000000000e+000			
NodLosd: ffff802 48c00000 ffff802 49c40000 ntkrninp.exe Windows 10 Kernel Version 19041 MP (8 procs) Free x64		st1	0.0000000000000 c+ 000			
Product: WinMt. suite: TerninalServer Single/SerTS Edition build lab: 19041 1 and/dfre vb release 191206-1406		st2	0.0000000000000000000000000000000000000			
Machine Name:		st4	0.0000000000000000000000000000000000000			
Debug session time: Sat Apr 27 12:15:16:200 2024 (UTC - 5:00)		st5	0.000000000000 c+ 000			
System Uptime: 0 days 1:04:52.822 Loaded dobable extension DLL		st6	0.0000000000000000000000000000000000000			
Loaded exts extension DIL		BB0	0:0:0:0			
Loaded Kext extension DLL		na1	0:0:0:0			
ffff802'755747f2 c3 ret 0: kd> bn ntlMcFreateProcessAddressSace		nn2	0:0:0:0			
		nx3	0:0:0:0			
ffff802 492a918 488bc nov rex.rsp		na5	0:0:0:0			
	×	na6	0:0:0:0	1		
6 · kd>		na7	0:0:0:0	1 Car	Louis Downwood Townsh	,
1		<u> </u>		- Cars	Locals Processes and Threads	
					Ln 11, Col 43 Sys 0:eXDI KD Proc 000:0 Thrd 00	6:0 ASM OVR CAPS NUM

SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\ala	ns\Documents\A	rium\SourcePoint-IA_7.12.53\Tiger_Lake_GSG_7-12-53.prj						(- a x
File Edit View Processor Options Window I	Help								
19 19 19 19 19 19 19 19 19 19 19 19 19 1				👹 StartWinDbg 👋 StartWinDbgX 👹 LoadCurrent 👹 Load	dAll 🖏 LoadedModuleList 🍕	🖥 CachedModuleList 🛛 📸 Ena	bleTraceHub 🍪 🍯 💕 🍪	🛛 🐿 🖬 🖬	∭ 0 0 0 0
Breakpoints (Code) Command 🔛 Log	Memory	IP Registers 🔍 Symbols 🥒 Trace 👀 Viewpoint 🔍 W	latch					. 4	1 LA D 7
G Code (P6*): (64-bit) Tracking IP 00000000000000000000000000000000000	FFFFFFFFFFFFFF	FFEL	- • ×	G Symbols (₽6") - Globals	- • •	DD Viewpoi	nt	1	
FFFFF802492A91D5L CC	int	3	^	Name	Address ^	Name	Description		Status ^
FFFFF802492A91D6L CC	int	3			FFFFF802494DFBA	C P0	TigerLake	Stopped	
MmCreateProcessAddressSpace:	X110			- f. NtSetInformationThread	FFFFF8024924375	C P1	TigerLake	Stopped	
FFFFF802492A91D81 488BC4	NOV	rax, rsp		 – f. NtSetInformationToken 	FFFFF802491E518	C P2	TigerLake	Stopped	
FFFFF802492A91DEL 48895808	807	gword ptr [rax+08].rbx		- f. NtSetInformationTransaction	FFFFF80248FD080	C P3	TigerLake	Stopped	*
FFFFF802492A91E3L 4C894018	3107	qword ptr [ras+18].r8		 – f. NtSetInformationTransactionManager 	FFFFF80248FD0CA	<			>
FFFFF802492A91E7L 48895010	2007	quord ptr [rax+10].rdx		- £ NtSetInformationVirtualMemory	FFFFF8024926112				
FFFFF802492A91ECL 57	push	rdi		 f. NtSetInformationWorkerFactory 	FFFFF80248EB5A8	IP General Registers (P6*)			
FFFFF802492A91EDL 4155	push	r13		- f. NtSetintervalProfile	FFFFF80249334F8	⊕ IA-32	Name Value		
FFFFF802492A91EFL 4155 FFFFF802492A91F1T 4157	push	r14 r15		- f. NtSetloCompletion	FFFFF802492FB3C	intel 64	RAX 0000000000	00001	
FFFFF802492A91F3L 4883EC40	sub	rsp,00000040		 – f. NtSetloCompletionEx 	FFFFF802492E14A	General	RBX 0000000001	06295	
FFFFF802492A91F7L 4883601000	and	gword ptr [rax+10].00000000		- f. NtSetIRTimer	FFFFF80248F6355	- Floating Point	PDY 0000000000	00000	
FFFFF802492A91FFL 488B359A406800	807	rsi.gvord ptr [PspMiniauaVorkingSet]		- f. NtSetLdtEntries	FFFFF80248FD012	- Segment	RBP FFFFF589441	76C40	
FFFFF802492A9206L 498BE8	31077	rbp.r8		- f. NtSetLowEventPair	FFFFF80249326B3	- Control	RSI 0000000000	00000	
FFFFF802492A92091 488HBC2498000000	307	rdi, qword ptr [rsp+00000098]		 – f. NtSetLowWaitHighEventPair 	FFFFF80249326B3	Debug	RDI 0000000000	00001	
FFFFF802492A9215L 4885C9	test	TON.TON		— f. NtSetQuotaInformationFile	FFFFF8024949920	- MMX	RSP FFFF589441	76448	
FFFFF802492A9218L 0F84500C1700	je	fffff80249419e6eL		- f. NtSetSecurityObject	FFFFF802492FC44	- YMM - SP	89 00000000000	00000	
FFFFF802492A92251 488B87D8090000	807	rbs. gword ptr [rax]		 	FFFFF8024955828	- YMM - DP	R10 7FFFFFFFFFF	FFFFC	
FFFFF802492A9228L 488BCB	3107	rcs.rbs		— f. NtSetSystemEnvironmentValueEx	FFFFF802495585A	- YMM - Int	R11 0000000000	0044E	
FFFFF802492A922BL E858ACC5FF FFFFF802492A9230T 85C0	call	MiMakePartitionActive		- f. NtSetSystemInformation	FFFFF8024928874	MSR	R12 FFFFA18E620	0D300	
FFFFF802492A9232L 0F84EC010000	je	MaCreateFrocessAddressSpace+24c		 f. NtSetSystemPowerState 	FFFFF8024959BF4	User	P14 00000000000	76DE0	
FFFFF802492A9238L 0FB713	ROWZX	edx, word ptr [rbx]		- f. NtSetSystemTime	FFFFF8024954EC5		R15 FFFFA18E62F	1D080	
FFFFF802492A923EL E835ACC5FF	call	MiSetProcessPartitionId		- f. NtSetThreadExecutionState	FFFFF802492F964		CS 0010		
FFFFF802492A9243L 4533C0	xor	r8d.r8d		- £ NtSetTimer	FFFFF80248F7914		DS 002B		
FFFFF802492492492491 4888008	nov les	edx dword ptr [r8+04]		- f. NtSetTimer2	FFFFF80248EAF65		55 0018		
FFFFF802492A924DL E80E8FB8FF	call	MiChargeConnit		- f. NtSetTimerEx	FFFFF80248EBF63		FS 0053		
FFFFF802492A9252L 85C0	test	eax.eax MnCreateProcessiddrensCrace+24c		- f. NtSetTimerResolution	FFFFF802493104C		GS 002B		
FFFFF802492A925AL 488D0DF7495A00	lea	rcx, qword ptr [MiState+1558]		- f. NtSetUuidSeed	FFFFF802493C773		RIP FFFFF802492	A91D8	
FFFFF802492A9261L 4C8DB780060000	lea	r14, qword ptr [rdi+00000680]	~	- f. NtSetValueKey	FFFFF8024923385		RFLAGS 0000000000	50202	
EEEEBaa40240108	Co Ouror	Cat Break Tradition View 10 Pafes	da .	- t NtSetVolumeinformationFile	FFFFF8024936D/E				
Disassembly V	Go Consor	Ser break 🖂 track the Alem the Kelle	H1	NtSetvynti-rocessivotincationEvent	FFFFF8024730A78				
				In Chutdown System	FFFFF002471D325				
				 International Applications and Applications 	FFFFF00240F5C76				
				 Intoignal/Antiviate or Single Object McSingle Disease Disease 	FFFFF002491002F				
				A MiStart Profile	FFFFF0024955DC6				
				Int StarDrofile	FFFFF0024955DC0				
				NtSubscribeWofStateChange	FFFFF802491F002				
				NtSuspandDocese	FFFFF8024950F6F Y				
				د	>				
				Globals (Locals) Stack Classes /					
									X
(P0)StartWinDbg(false);									0
P0>									
Loading User Defined Macro #2: C:NU: PhyLoadCurrentWinDbg():	sers\alans\l	Documents\Arium\SourcePoint-IA_7.12.53\Me	cros\WinDbg\but	ton/WinDbgBtnl_Chk.asc					
P6>									
									~
P									
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step C	Over, Shift+F12:R	eset				P6	18: Stopped	64 Bi	t 📕

Power Tip: between individual "Go" commands and breaks, the context of the code will often change (i.e. the value of CR3 changes). Even though the module name will still appear in the SourcePoint Symbols window, the needed symbols will no longer appear in its Code window. Hit the LoadCurrent button again to re-display the symbols.



Power Tip: Alternatively, take the following steps to ensure the Code window context is updated upon each break:

Under the File menu, select Macro > Configure Macros...

Click on the Event Macros tab.

Select Event: Breakpoint (any)

Then browse in the main folder, and select Events.mac.

This will slow down breakpoints ever so slightly, but it will ensure the code context is refreshed without manual intervention.

Power Tip: Once the PDB file is identified, SourcePoint will search for the symbol file in WinDbg's stored Symbol path, and then if not found, its Cache path. The symbol path in most WinDbg installations is something similar to:

srv*C:\Symbols*http://msdl.microsoft.com/download/symbols

and **SourcePoint has no knowledge of HTTP access**, so it will extrapolate only the C:\Symbols portion, and next go to, and include, the cache path.

What happens if the symbols don't show up?

In some instances, there are no symbols available for the module that SourcePoint is halted in, or the specified path to the symbols is invalid. You will see the following:



Check to ensure that your cached symbol path is correct by issuing the sympath command from the SourcePoint Command window. If it is correct, then the symbols are not local; fetch them from the Microsoft symbols server, presuming that they are available.



Another possibility is that SourcePoint's heuristic search for the PE32 header in system memory misses it, because the code is too small and it overshoots, or too large and exceeds the timeout period. There are two commands available from the SourcePoint Command line that can help:

timeout("nn") in seconds will change the timeout from default 30 seconds to 'nn' seconds



SearchPageSize(nnnnn) in hex will change the search decrement for detecting modules (base).

Command

```
P0>
P0>searchpagesize
00001000H
P0>searchpagesize(0x200)
P0>searchpagesize
00000200H
P0>
```



Getting SourcePoint to display module names as well as function names

WinDbg displays the fully qualified symbol name, including the module name, in its windows, as in nt!MmCreateProcessAddressSpace. SourcePoint truncates them by default to solely the function name, as in MmCreateProcessAddressSpace.

The module name prefix can be displayed by enabling SourcePoint's Qualified Symbol Name (QSN) format. In the Options menu, select Preferences, and click on "Use QSN in disassembler".

Preferences	×				
General Emulator Breakpoints C	ode Memory Program IPC Colors				
Source code	C++ symbol name demangler ☑ Demangled symbol names Compiler: GCC Standard ∨				
 ☐ Hide C++ internal symbols ✓ Smart symbol analysis ✓ Load from temporary copy of 	Use QSN in disassembler				
Share source file path map ar Internal globals are public	nong all programs Source Path				
Show individual inline functions Array expansion limit: 10000					
	OK Cancel Help				

The Code window display will now look something like this:



🕒 Code (P0*): (64-bit) Tracki	ng IP 00000000000000000	· FFFFFFFFFFFFFF	EL	- • ×
FFFFF80274682E07L	B948000000	MOV	ecx,00000048	
FFFFF80274682E0CL	0FB6D0	MOVZX	edx,al	1
FFFFF80274682E0FL	418895FA000000	MOV	byte ptr [r13+00000fa],dl	1
FFFFF80274682E16L	8BC2	mov	eax,edx	
FFFFF80274682E18L	48C1EA20	shr	rdx,20	
FFFFF80274682EICL	UF30	wrmsr		
FFFFF80274682EIEL	4180A5F8000000FE	and	Dyte ptr [rl3+0000018],Ie	
FFFFF80274682E26L	41BAU1000000	MOV	riud,uuuuuuui hata ata [aaa:50] a155	
FFFFF00274602E2CL	4430702430	cmp io	byte ptr [rsp+su],risb 	
FFFFF00274602E31L	/4/0	Je	and bute ptp [p12+00007e9a]	
FFFF80274682F3BI	4488702450	MOVZA	bute ptr [rep+50] r15b	
⇒FFFFF80274682E40L	84C0	test	al.al	
FFFFF80274682E42L	7465	ie	::ntkrnlmp.PpmIdleExecuteTransition+11b9	
FFFFF80274682E44L	65488B04252000+	mov	rax,qword ptr gs:[00000000000000000]	
FFFFF80274682E4DL	4C8D05ACD1D7FF	lea	r8,qword ptr [fffff80274400000]	
FFFFF80274682E54L	418BDA	MOV	ebx,r10d	
FFFFF80274682E57L	8B4824	MOV	ecx,dword ptr [rax+24]	
FFFFF80274682E5AL	4488B89A7E0000	MOV	byte ptr [rax+00007e9a],r15b	
FFFFF80274682E61L	418B9488D024D000	MOV	edx,dword ptr [r8][rcx*4+00d024d0]	
FFFFF80274682E69L	8BCA	MOV	ecx, edx	
FFFFF80274682E6BL	8BC2	mov	eax,edx	
FFFFF80274682E6DL	83E13F	and	ecx,000003f	
FFFFF80274682E70L	48D3E3	sal	rbx,cl	
FFFFF80274682E73L	48F7D3	not	rdx	
FFFFF80274682E1CL ~	🔎 Disassembly 🗸	Go Cursor	Set Break 🗹 Track IP View IP Refresh	

Power Tip: Note that SourcePoint's syntax is slightly different from WinDbg's:

WinDbg:	ntkrnlmp!PpmIdleExecuteTransition+11b9
SourcePoint:	::ntkrnlmp.PpmIdleExecuteTransition+11b9

Do a Project Save to save these settings into your Project, so they'll automatically load for your next session.

In an upcoming release, we'll make QSN the default in the disassembly for Windows debugging.



Troubleshooting Tips and Errata

Chances are, you'll run into something strange during your testing. We're the first to admit that JTAG-based run-control and trace are not always deterministic. JTAG is a 30-year hardware protocol, and when something goes astray at a very low level within the chip, SourcePoint tries to (but sometimes doesn't) recover gracefully. There will be times that the board will power cycle on its own. Or the firmware thinks that a thread is running but gets out of sync with the SourcePoint software, which thinks it's halted. Or the DbCStatus.exe ball stays red instead of turning green, while you swear you have a good DbC connection. Sometimes you have no choice but to quit SourcePoint and power cycle the target. That usually clears up the one-of's. But, of course, that means quitting out of WinDbg (preferably first), then quitting out of SourcePoint, power-cycling the target, and then re-establishing the connections from scratch. Tedious.

And, we all know that WinDbg has its quirks as well. And Windows sometimes objects to the presence of JTAG-assisted debuggers. Combine the three, and, well, you're bound to run into some bugs and misbehaviors.

Hopefully you don't run into this too many times. But, on the other hand, if you didn't, we'd have nothing to fix.

In the meantime, here are errata for the UP Xtreme i11, and the steps needed to mitigate where possible.

Windows crashes

If you work with SourcePoint WinDbg long enough, you'll likely crash Windows at some point. The vast majority of time, quitting out of SourcePoint and power cycling the target restores things to normal. But, in all of our testing, we've each had to reinstall Windows at least once. Really, it's no different from reinstalling Windows in a VM, only more onerous. But we've not ever been able to reproduce this type of failure.

Drop us a note on our <u>Support</u> line, or call us, if you can reproduce this.

WinDbg Classic is better than WinDbgX

WinDbgX, in intermittent circumstances, directs SourcePoint to do numerous memory reads at low memory. In which case, if you have the Log window open, will display messages like:

Page table is not present



Page table is not present. Linear address: 000000000001800L

Most of the time, these error messages are just informative. But they do occur much more frequently with WinDbgX than WinDbg Classic. In general, they can be ignored.

Pause in Initial Symbol Load

Intermittently, after issuing the first Break in WinDbgX, in the middle of the memory reads associated with the symbol loading, WinDbg stops sending commands to SourcePoint, and the transactions stop. The SourcePoint Dashboard Lights stop flashing, and a look at the Log window shows no traffic.

This issue seems to be very host and target specific. On some, it does not occur at all. In others, we see more frequent failures.

The only option at this point is to quit out of WinDbg and SourcePoint, power cycle the target, and start over. It is currently under investigation.

This issue only manifests itself with WinDbgX. **WinDbg Classic does not have this issue, so we recommend its use, instead of WinDbgX.**

LoadCurrent versus LoadAll

The LoadCurrent macro makes the symbols available within the module at the current instruction pointer visible to SourcePoint. LoadAll will retrieve all symbols for what's in the addressable context. It takes a long time.

COM(32) Surrogate

After a crash, when you restart SourcePoint, once in a blue moon it will misbehave. Run-control will not work properly.

Open Task Manager, and look for a COM(32) Surrogate task. If you see one, kill it.

Viewing the Stack

SourcePoint's Stack display in its Symbols window has been improved in the most recent release, but needs further enhancement. You will find that it gives different results from what will be seen in the WinDbg "k" command, and which may further differ from what you may see in an Intel Processor Trace traceback. Here is an example:







If you look carefully, you'll see that "guard_dispatch_icall" does not display in the Stack tab of the SourcePoint Symbols window. It has a special call frame/sequence that our Stack traceback does not display. But it does show up in Intel Processor Trace.



LoadCurrent intermittently fails in User code

When hitting a breakpoint set in user code, about 50% of the time a LoadCurrent will not successfully display the symbols within the SourcePoint Code window. WinDbg correctly displays the symbols. If you have a SourcePoint Log window open, you may see:

File doesn't exist -> \000000000000000000000000000000000

We are working on this.

Breaks are not process-aware

Setting breakpoints in WinDbg to break within a specific process, such as with:

bp /p <address> nt!NtReadFile explorer.exe

does not work properly. Instead of halting in the instance of nt!NtReadFile associated with explorer.exe, it will halt at the first instance of the shared code, likely in a different process. This is because EXDI does not provide process/thread information down to SourcePoint, unlike the standard WinDbg kdnet interface.

Mangled function names

SourcePoint has a built-in name demangler.

Some of the Windows kernel is written in C++ code, such as in win32kbase.sys, and the demangler comes in handy for this.

You may see this in action for example by looking at some of the win32kbase code here:



		(B)(B)	0		1 m 1 m		Concernant.	0.000
A Construction of the second s	International and a second sec	() ()	Construction C	Amp Previ Wash Hogod Amp Previ Wash Hogod mer Park Hose States Previ Amp and States Previ Amp And States Previ Amp And States Previ Amp And Amp And Previ Amp And Amp And Previ Amp And Amp And Previ Amp And	Image: Image: Image: Address: Image: Image: Address: Image: Image: Amage: Image: Image: TPTFC:	IP Consent Register BY IP Up As 22 To IP Consent Register BY IP IP Consent IP - Obset IP Consent - Obset IP IP - Obset IP Consent - Obset IP IP - WAR I	View Control 1 Type Lange 1 Type Lange <td>Control School S</td>	Control School S
Compared	Across and a second sec	In the second se	4. odd _cdecif 4. odd _cdecif 4. odd _cdecif 4. odd _cdecif 5. odd _cdecif	4028	106 c506 ≈ 00° 1 106 c506 ≈ 00° 1 106 c507 ≈	rear [resi] (Trivel devolut. rear [reset] (Trivel devolut. rear [reset] (Trivel devolut. rear [reset] (Trivel devolut. rear [reset] (Trivel devolut. rear [reset] (Trivel devolut. rear [reset] (Trivel devolut. (Trivel devolut. (Triv	s mayers 2015 Asset fil i miser an s Bahan	

Note that to see the demangled name in the SourcePoint Code window, you need to turn "Use QSN in disassembler" off under Disassembly in menu Options > Preferences > Program.

WinDbg FP register display is not working

WinDbg does not display the floating point registers contents when using EXDI. SourcePoint displays the register values correctly.

Problems with symbol loading from local cache

When symbols are loaded solely with SourcePoint (WinDbg is not launched), SourcePoint will refer to the WinDbg path local to your debugging PC. Type:

sympath

in the SourcePoint Command window, and you'll see where SourcePoint will look. For example:

c:\symbols;C:\ProgramData\dbg\sym

You can explicitly set the symbol path to be that of WinDbg Classic's, by typing in the Command line:

windbgc

```
©2024 ASSET InterTech, Inc.
```



Alternatively, typing:

windbgx

sets up the symbol path as defined within WinDbgX. Note that with WinDbgX, SourcePoint will automatically set its own sympath.

But, symbol and cache paths are a little trickier with WinDbg Classic. WinDbgX allows for an explicit description of the cache path:

Debugging paths	
Source path:	
Symbol path:	Browse SRV*c:\symbols*https://msdl.microsoft.com/download/symbols
	Browse
Default cache:	%PROGRAMDATA%\Dbg

Whereas WinDbg Classic does not; you have to embed it in the Symbol path, and that is squirrelled away in the registry:

Symbol path:		
SRV*c:\symbols*https://msdl.microsoft.com/download/symbols;C: \ProgramData\dbg\sym	^	OK Cancel
		Help
	\vee	Browse

It is important to be aware of Workspaces. SourcePoint will use the settings detected for default Workspaces (Default, AMD64 etc), but best practice is to create a separate Workspace; in this case, we name it EXDI:



Open Workspace	×
Workspaces:	
EXDI	
Workspace:	
EXDI	
OK Cancel Help	

Be sure to load the Workspace after you launch WinDbg, and use the SourcePoint windbgc and sympath commands to ensure the path is latched in. Once a Workspace is decided on, you can force the specific Workspace by setting an environment variable (_NT_WINDBG_WORKSPACE):





ironment variables		
ser variables for alans		
Variable	Value	^
GPU_FORCE_64BIT_PTR	0	
GPU_MAX_ALLOC_PERCENT	100	
GPU_MAX_HEAP_SIZE	100	
GPU_SINGLE_ALLOC_PERCE	100	
GPU_USE_SYNC_OBJECTS	1	
OneDrive	C:\Users\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	۷
	New Edit Delete	
	New Edit Delete	
ystem variables		
Variable	Value	^
_NT_WINDBG_VBS	FALSE	
	EVDI	
_NT_WINDBG_WORKSPACE	EXDI	
_NT_WINDBG_WORKSPACE ASSET	C:\ScanWorks	
ASSET ComSpec	C:\ScanWorks C:\WINDOWS\system32\cmd.exe	
ASSET ComSpec DALINSTALLDIR	C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	
ASSET ComSpec DALINSTALLDIR DriverData	C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData	
_NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK_DIR	C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	¥
NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK DIR	C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\ New Edit Delete	~

Note: WinDbg tends to store an extraneous "\sym" in the cache path that needs to be worked around. You'll see that SourcePoint handles that properly.



Power Tip: If you're using SourcePoint by itself, it may be helpful to store as many symbol files locally as possible. Use this following command (on the target PC) to download them, and then copy them over to your cache path on your host debug PC:

"C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\symchk.exe" /r c:\windows /s SRV*c:\symbols*http://msdl.microsoft.com/download/symbols

Power Tip: It's a good idea to disable Ethernet on the target for your debugging, to avoid Windows Update changing the modules and their GUIDs, requiring a reload to update the cached symbol files.



SOURCEPOINT DEBUGGING FOR HYPER-V

Debugging with Hyper-V and Virtualization-Based Security (VBS) is supported in the SourcePoint 7.12.59 and later releases. Visualization of the virtualization environment, and probe mode access to it, provide an unprecedented level of capabilities.

Getting Started

Let's get started. We'll walk through a subset of the capabilities. There are a multitude of areas to explore here.

Ensure that the target has Hyper-V and VBS enabled.

Then, change the Environment Variable _NT_WINDBG_VBS to TRUE, so that SourcePoint doesn't go scanning for the KdVersionBlock (it can't find it in a hypervisorenabled target anyways):





er variables for alans		
Variable	Value	^
GPU_FORCE_64BIT_PTR	0	
GPU_MAX_ALLOC_PERCENT	100	
GPU_MAX_HEAP_SIZE	100	
GPU_SINGLE_ALLOC_PERCE	100	
GPU_USE_SYNC_OBJECTS	1	
OneDrive	C:\Users\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	~
	New Edit Delete	
stem variables	Value	~
vstem variables Variable	Value	^
stem variables Variable _NT_WINDBG_VBS NT WINDBG WORKSPACE	Value TRUE EXDI	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET	Value TRUE EXDI C:\ScanWorks	
stem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec	Value TRUE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe	
vstem variables Variable <u>NT_WINDBG_VBS</u> _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR	Value TRUE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData	Value TRUE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\UINDOWS\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData	
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK_DIR	Value TRUE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK_DIR	Value TRUE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\ New Edit Delete	

It is so much easier to see VM transitions if only one processor is made active on the target:

Via the advanced BIOS settings (with the upassw0rd password – see the <u>SourcePoint</u> <u>UEFI Getting Started Guide</u>), go to CRB Setup > CRB Advanced > CPU Configuration, disable Hyper-Threading (if you're on a target that supports it – the Celeron board does not), and set Active Processor Cores to 1:



Disable the synthetic watchdog on the target to ensure that the target does not autonomously reset itself in probe mode: >bcdedit /set {default} loadoptions "systemwatchdogpolicy=disabled"

Power Tip: After crashing the target, which you will do periodically when you go "off the fairway" with VMM debug, you will need to power-cycle, and Windows will launch Automatic Repair to attempt to "restore" itself. Then you have to Restart again. This is not really necessary, and is inconvenient, so to save time, you can disable recovery boot from an Administrator CMD window using the two bcdedit commands: >bcdedit /set recoveryenabled No >NUL >bcdedit /set bootstatuspolicy ignoreallfailures >NUL

Boot the target to the UEFI shell. This is accomplished by power-cycling the target and holding down the F7 key until you see the BIOS login prompt:

Enter Password —	1
	I
	I

VMM breakpoints, and debugging the Secure Kernel



Launch SourcePoint, connect to the target, issue a Stop, Refresh the Code window, and set a VM Launch breakpoint:

Add Breakpoi	nt	×
Identifier:	VMLaunch	
Break on:	VM Launch 🗸	Advanced
Resource:	Processor ~	
Processor:	P0 ~	
Location:		<i>P</i> 1010
Translate:	~ ~	
Length:	\sim	
Data;		1010
External:		1010
Sequence;	~	
Cmd/macro:		Browse
	OK Cancel	Help

Hit Go in SourcePoint. Be sure to quit out of the target's UEFI shell password prompt (hit Esc and then accept "Yes") to start the Windows boot process. You will break at the first VM launch, and land in the hypervisor, in VM Guest mode. Click on the LoadCurrent macro button, and say "No" to the "Symbol not found" prompt to see we're in hvix64, for which there are no symbols:

Log				×
Date	Time	Component	Message	^
07/13/202	4 11:25:50.823	Images.mac:LoadImage	Performing Load Current @ 0xFFFFF86DEA43D314	
07/13/202	4 11:25:57.358	Images.mac:FindBase	MZ Header Found @ 0xFFFFF86DEA200000	
07/13/202	4 11:25:57.405	Images.mac:FindBase	PE Header Found @ 0xFFFF86DEA2000F8	
07/13/202	4 11:25:57.453	Images.mac:FindBase	PE32+ Header Found @ 0xFFFFF86DEA200110	
07/13/202	4 11:25:57.508	Images.mac:FindBase	Debug Directory Found @ 0xFFFF86DEA606210	
07/13/202	4 11:25:57.569	Images.mac:FindBase	Found .data Section @ 0xFFFF86DEA200250, Offset @ 0x600000, Size: 0x1EFAC8	
07/13/202	4 11:25:57.569	Images.mac:FindBase	Detected Module Base @ 0xFFFFF86DEA200000, Size: 0x1600000 [7 secs]	
07/13/202	4 11:25:57.633	Images.mac:LoadSymbols	Using Module Name @ 0xFFFFF86DEA60AF80	
07/13/202	4 11:25:57.698	Images.mac:GetSymEntry	File doesn't exist -> c:\symbols\hvix64.pdb\A1C1E0716D61D004FC934AA4D7CBA9A91\hvix64.pdb	
07/13/202	4 11:25:57.698	Images.mac:GetSymEntry	File doesn't exist -> C:\ProgramData\dbg\sym\hvix64.pdb\A1C1E0716D61D004FC934AA4D7CBA9A91\hvix64.pdb	۰ – د
07/13/202	4 11:27:41.504	Images.mac:LoadCurrent	File doesn't exist -> hvix64.pdb\&1C1E0716D61D004FC934&&4D7CB&9&91\hvix64.pdb	~
<				>



SourcePoint V12.0 (C) - TigLike - C(Viers) Man SourcePoint-A, 712.97 TU-Ceteron-742-55 pp O × E								
19 19 19 19 19 19 19 19 19 19 19 19 19 1	tWinDbgC 🚭 S	tartWinDbgX 📸 LoadCurrent 🧉	📓 LoadAll 🛛 📸 LoadedModuleList 👋	CachedModuleList 👋 EnableTraceHub 🗌 📽	សូសូ សុខភាព 🖞 🖞 🕯	a 🖬 🐔 🗣 🗾 🗈 🕸	<u>.</u>	8
🗢 Breakpoints 🕒 Code 🗲 Command 🔝 Log	Memory 1	IP Registers 🔍 Symbols 📌 Tr	ace 👀 Viewpoint 🔍 Watch					
G Code (P0*): (64-bit) Tracking IP 00000000000000000000000000000000000			2	😪 Symbols (P0*) - Globals			20 Viewpoint	
FFFF86DEA43D309L 48C7C010000000 FFFF86DEA43D310L 0F01C3 FFFF86DEA43D313L CC	NOV VAROSUNO int	rax.00000010	^	Name	Address		Name Description © P0 TigerLake	Stopped ^
	xor Vacall int dec jne rdtsc sal or sub shr add retp	ICK, ICK 3 Tox ffff86dem43d310L rdx.20 rox.rdx rox.rB rox.rB rox,rB rox,rB rox,rB rox,rB				IP General Registers (P0*) IP-1A-32 IP-1ntel 64	Ame Value 0 P3 TigerLake 0 P3 TigerLake 0 P3 TigerLake 0 P3 TigerLake 0 P3 TigerLake	Not Active Not Active
PTFFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF6424430342 C3 PTFF642443042 C3 PTFF642443042 C3 PTFF642443042 C3 PTFF642443042 C3 PTFF642443042 C3 PTFF642443074 C3 PTFF642443044 C3 PTFF64443044 C	retn int int int int int int int buy buy nov nov nov nov nov nov nov nov voresume int int int int int int int int int int	<pre> [rex][rex] [rex][rex] [rex][rex] [rex][rex] [rex][rex] [rex][rex] [rex][rex] [rex][rex][rex] [rex][rex][rex][rex][rex][rex][rex][rex]</pre>	* (6daa43d377] (daa43d36b))]	×	,	General Floating Point Segment General Gener	AC DFFFF0443958400 AC DFFFF0443958410 BFFFF76443958410 DEFFFF644398410 DEF FFFF7644398410 DEF FFFF7644398410 DEF FFFF7644398410 DEF FFFF7644398410 DEF FFFF7644398410 DEF FFFF76443994410 DE FFFF76402443994410 DE FFFF76402443994410 DE DFFF76402443000 DE OFF764000000010 DE OFF7640000000010 DE OFF764044303441001 DE OFF76404430314 DE OFF0764024430314 DE OFF0000500011902	
FFFFF86DEA43D314L V P Disassembly V	Go Cursor	Set Break 🗹 Track IP	View IP Retresh	Globals / Locals / Stack / Cla	sses /			
Date Time Component 017/11/2024 11:25:57 508 Inages.an 017/11/2024 11:25:57 508 Inages.an 017/11/2024 11:25:57 509 Inages.an 017/11/2024 11:25:57 509 Inages.an 017/11/2024 11:25:57 509 Inages.an 017/11/2024 11:25:57 630 Inages.an 017/11/2024 11:25:57 649 Inages.an 017/11/2024 11:25:57 649 Inages.an 017/11/2024 11:25:57 649 Inages.an	c:FindBase c:FindBase c:LoadSymbol c:GetSymEntr c:GetSymEntr c:LoadCurren	Message Debug Directory Fr Found data Secti Detected Module Ba Using Module Name y File doesn't exis y File doesn't exis t File doesn't exis	<pre>ound @ 0xFFFF86DEA2606210 on @ 0xFFFF86DEA200250. Off ase @ 0xFFFF85DEA20000. 5:</pre>	set @ 0x600000, Size: 0x1EFAC8 ze: 0x1600000 [7 secs] 1/C120716501004FC94AA407CBA9A91 hvist4 pdb/AIC1E0716061D004FC934A D004FC934AA4D7CBA9A91\hvist4.pdb	ivix64.pdb 4D7CEA9A91∖hvix64.pdb			v v
Command P) P) Confige Deer Defined Hacro #2: Conference Lass Documents Arius SourcePoint-1A.7.12: 59-Macros VinDephatton VinDephat_CAL sac								
RVSLoadCurrentWinDbg(); RV						Edit Add	Remove Remove All	Disable
Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F1:2:Reset PO 18: Stopped VM Guest 64 Bit							64 Bit	

Hit Go, and then LoadCurrent a second time. Do a LoadCurrent again. You're in hvloader (still no symbols), again in VM Guest mode.

Hit Go a third time, encountering the third VM Launch breakpoint, with the target again in Guest mode. Hit LoadCurrent again, and provided you have the securekernel.pdb file cached, you will see the symbols for the Secure Kernel:

Structure 1,123 (2001) Topic Astronomy (11) (2001) — 0 × How Reference Automa (Non-Mone-Mone-Astronomy Astronomy Astron							
🔋 🖉 🖓 🕼 🕼 🖓 🖓 🖓 🖓 🖓 🖓 👘 👘 🖓 🖓 🖓 👘 👘 🖓 👘 🖓 👘 🖓 👘 🖓 🎝 👷 👘 🖓 🎝 👷 👘 🖓 🎝 👷 👘 🖓 👘 🖉 👘 🖓 👘 🖉 👘 🖓 👘 🖉 👘 🖓 👘 🖉 👘 🖓 👘 🖉 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘							
📀 Breakpoints 🕒 Code 📏 Command 🔝 Log 🏢 Memory IP Registers 🎕 Symbols 🖋 Trace 👀 Viewpoint 🔍 Watch							
Code (P0*): (64-bit) Tracking IP 00000000000000000000000000000000000	Symbols (P0*) - Globals			DD Viewpoint			
FFFFF80339D02EF4L CC int 3	Name	Address ^		Name Description	Status ^		
FFFF80339D02EF6L 66660F1F840000+ nop [rax][rax+00000000]	HvisAnyHypervisonPresent	FFFFF80339CCAEUCL		O P1 TigerLake	Not Active		
Securekernel.Shvipttilentry: fFFFF80339D02F00L33C0 xor eax.eax	Hulshypervision/icrosoftCompatible HulSetupl.iveDumpBuffer	FFFFF80339D3D644L		O P2 TigerLake			
FFFFF80339D02F02L 4889442420 mov gvord ptr [rsp+20].rax	- £ input s	FFFFF80339CE5770L		O P3 TigerLake	Not Active		
FFFFF80339D02F0EL 650EF70C25A400+ movzx ecx.vord ptr gs:[00000000000044]	f invalid parameter	FFFFF80339D35324L		<	>		
FFFF80339D02F17L 85C9 test ecx.ecx	- f isdigit	FFFFF80339CE39DCL					
FFFFF80339D02F1BL 488D059E360800 lea rax.gword ptr [::securekernel.SkeDefaultNpxSt	- f. IsEmptyCollaborationId	FFFFF80339D388A8L	IP General Registers (P0*)				
FFFF80339D02F22L 0FAE00 ixsave [rox] FFFF80339D02F22L 0FAE00 ixsave [rox]	 f. IsTrustletCreateAttributeWellFormed 	FFFFF80339D2E8E8L	⊞-IA-32	Name Value			
FFFF80339D02F2AL 85C0 test eax	- f. lumApi_NtGENERIC	FFFFF80339CC8124L	⊜-Intel 64	RAX 000000000000000000000000000000000000			
FFFFF0339D02FZEL 90059C850600 nov dvord pt [::securekernel.SkPlaseIInitStatus]	f lumApi_NtTraceEvent	FFFFF80339CE2DICL	- General	RCX 00000000000011			
FFFF80339D02F34L E813240300 call ::securekernel.SkScrubLoaderBlock	A humbing GENERIC	FFFFF00239D3FCF0T	-Floating Point	RDX 0000000001A7260			
FFFFF80339D02F39L E80E jmp ::securekernel.ShvipVtilEntry+49	Inning PALPC MESSAGE ATRIBUTES	FFFFF00229CC01FCT	Segment	RBP 0000000001A73B0			
FFFFF00339D02F40L 65488E24253000+ mov rsp.qword ptr gs:[000000000000000]	A humbre DOBIECT ATTDIBUTES	FFFFF80339CC8814I	Control	PDI FFFFF60339D7F000			
FFFF80339D02F49L B80400000 nov eax,00000004	Impag PODECT_MINDOTES	FFFFF80339D2FE38L	Debug	RSP FFFFF80339C96ED0			
FFFFF0339D02F52L 65488324250800+ and qword ptr gs:[000000000000000000].00000000	4 JumAra PSID	FFFFF80339D2EF4CL	- MINA	R8 FFFFF803327A0028			
FFFFF03339D02F5CL E925040000 jmp ::securekernel.SkCallNornalMode+196	- 4 JumArg PSID AND ATTRIBUTES	FFFFF80339D2F080L	VIM DD	R9 00000000000FFF			
FFFFF80339D02F62L CC int 3	- 4 JumArg PUNICODE STRING	FFFFF80339D2F09CL	VMM lot	P11 000000000000000			
FFFFF0339D02F63LCC int 3	LumArg PWORKER FACTORY DEFERR.	FFFFF80339CC9138L	e.MSR	R12 000000000000002			
FFFF60339002F65L CC int 3	- f. lumArg PWSTR	FFFFF80339D2F080L	A VMCS	R13 0000000000000000			
FFFFF00309D02F63LCC int 3	- f. lumAssignMemoryToSocDomain	FFFFF80339D149C4L	User	R14 000000000000BB			
::securekernel ShviDenterVil:	- f. lumAwaitSmc	FFFFF80339D149C4L		CS 0010			
FFFF60339D02F70L 4881EC38010000 sub rsp.00000138	- f. lumCreateSecureDevice	FFFFF80339D2D958L		DS 0018			
FFFF80339D02F7FL 488D842400010000 lea rax, qword ptr [rsp+00000100]	- f. lumCreateSecureSection	FFFFF80339D1BC50L		SS 0000			
FFFFF00339D02F94L 0F297C2440 movaps maword ptr [rsp+40], max7	 – f. lumCreateSecureSectionSpecifyPages 	FFFFF80339D1BDA0L		ES 0019			
FFFFF00339D02F89L 440F2942450 movaps wnword ptr [rsp+50].xma8	- f, lumCrypto	FFFFF80339CC6AA4L		GS 0018			
FFFFF80339D02F95L 440F29542470 movaps maword ptr [rsp+70], mm10	- f. lumDebugAppendChar	FFFFF80339D2E6E0L		RIP FFFFF80339D02F00			
FFFFF0339D02F9BL 440F295880 movaps wnword ptr [rax=80] xmall	f lumDebugNumToString	FFFFF80339D2E708L		RFLAGS 000000000000000			
FFFFF0339D02FASL 440F2966A0 novaps snavord ptr [cax-6], snal3	- f lumDebugPrintContext	FFFFF80339D2C8E8L					
FFFFF80339D02FAL 440F2970B0 moveps snaword ptr [rag-50], sna14	- f. lumDebugPrintNt	FFFFF80339CB2B4CL					
C C C C C C C C C C C C C C C C C C C	 tumDebugenntivitList tumDmahlaphlamany 	FFFFF00229D1E1701 Y					
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT	<	>					
Internet of the set of	Globals (Locals) Stack) Classes /						
07/13/2024 11:34:30.215 Images.mac:FindBase Detected Module Base @ 0xFFFFF80339CA3000. S	ize: 0xFE000 [7 secs]				^		
07/13/2024 11:34:30.337 load using module wake @ UxFFFF80339D4E900 Loading PDB format c:\symbols\securekernel.p	db\3F38482DB080AF7428A6BB2D5374C1AD1\secureke	rnel.pdb					
Identifier Address Attributes							
13 VR Launch VR Launch (Frodessor P0)							
Command							
(P)LoadCurrentFinDbg(); (P)							
coding User Defined Macro #2: C:\Users\slams\Documents\kriux\SourcePoint-IA_7 12:59\Macron\VinUbgPtnl_Ckk.mac (9)LoadcurrentVinUbg(); (9)							
16Help, F5Gp, Shift+F53top, F8Step Into, F10Step Over, Shift+F12Reset							

All of the Secure Kernel functions are available for debug. Enjoy!



Power Tip: The above strategy works for having symbols display within SourcePoint while debugging in a Hyper-V environment. But what if we want the symbols to show up in WinDbg as well, while debugging via EXDI? There are several ways of doing this, but the easiest approach is to get the module base and size from within SourcePoint, and then use the .reload command within WinDbg to load the symbols there.

Example: while in the secure kernel with SourcePoint, click on LoadCurrent, and you'll see something like:

07/13/2024 13:57:30.211	<pre>Images.mac:IsKPCR</pre>	Invalid KPCR Pointer. Check MSR values.
07/13/2024 13:57:30.211	<pre>Images.mac:LoadImage</pre>	Performing Load Current @ 0xFFFFF8021FD48F00
07/13/2024 13:57:36.020	Images.mac:FindBase	MZ Header Found @ 0xFFFFF8021FCE9000
07/13/2024 13:57:36.066	Images.mac:FindBase	PE Header Found @ 0xFFFFF8021FCE9108
07/13/2024 13:57:36.066	Images.mac:FindBase	PE32+ Header Found @ 0xFFFFF8021FCE9120
07/13/2024 13:57:36.169	Images.mac:FindBase	Debug Directory Found @ 0xFFFFF8021FD8FED0
07/13/2024 13:57:36.272 Offset @ 0xB9000, Size:	Images.mac:FindBase 0x1A528	Found .data Section @ 0xFFFFF8021FCE9300,
07/13/2024 13:57:36.272 <mark>Size: 0xFE000</mark> [6 secs]	Images.mac:FindBase	Detected Module Base @ <mark>0xFFFFF8021FCE9000</mark> ,
07/13/2024 13:57:36.336	<pre>Images.mac:LoadSymbols</pre>	Using Module Name @ 0xFFFFF8021FD94900
07/13/2024 13:57:36.402 c:\symbols\securekernel.	load pdb\3F38482DB080AF7428A6B	Loading PDB format B2D5374C1AD1\securekernel.pdb

Note the highlighted sections above. Use these to load the symbols within WinDbg via .reload /f /i module=<address>,<size>:

kd>.reload /f /i securekernel.exe=0xFFFFF8021FCE9000,0xFE000

VMCS Viewer/Editor

But it gets more interesting. We are in the VTL 1 Secure Kernel, in Guest mode of course. Let's inspect both the Guest mode and Host mode VMCS (yes, it is possible to look at the state of the Host while you are in Guest mode – you can only do that with JTAG!).

In the General Registers window, click on VMCS and select Guest state. Open up a new Registers window, and select Host state, or any of VM-Entry, VM-Exit and VM-Control registers. Have fun.





VM Exit breakpoints and Basic Exit Reasons

For a more advanced topic, let's look at using VM Exit breakpoints to capture Guest to Host transitions.

Turn off the VM Launch breakpoint, and add a VM Exit breakpoint:


Add Breakpoir	nt	×
Identifier:	VMExit	
Break on:	VM Exit ~	Advanced
Resource:	Processor ~	
Processor:	P0 ~	
Location;		<i>P</i> 1010
Translate;	~	
Length:	\sim	
Data:	FFFFFFFFFFFFFF	1010
External:		1010
Sequence;	~	
Cmd/macro:		Browse
	OK Cancel	Help

Clicking on the 1010... to the right of Data, shows that we can trigger on any single or combination of VM Exit reasons, as detailed in the Intel <u>Software Developer's Manual</u>, Volume 3A, Appendix C, VMX Basic Exit Reasons. For now, let's just leave them as all F's.

Then hit Go. The VM Exit breakpoint will be hit, and this puts us back into hvix64, and this time we're in VM Host mode:



			ymbols (P0") - Globals		D Viewpoint
PFFF6DEA43E/FEL E%E600000 FFFF6DEA43E303L 0P01C2 FFFF6DEA43E305L F5DE00000 FFF6DEA43E305L F5DE00000 FFF6DEA43E305L 7442424000000 FFF6DEA43E31DL 40054C4320 FFFF6DEA43E31DL 400540 FFFF6DEA43E320L 400503 FFFF6DEA43E321L 400503 FFFF6DEA43E321L 400503 FFFF6DEA43E321L 400503 FFFF6DEA43E321L 400503 FFFF6DEA43E321L 400503 FFFF6DEA43E321 400503 FFFF6DEA43E320 FFFFF6DEA43E320 FFFFF6DEA43E320 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	jnp fffff85dma4382891 vnlaunch jnp fffff85dma4382891 00 cor dword ptr [rsp+30] 00 cor dword ptr [rsp+30] 00 cor dword ptr [rsp+31] 00 cor dword ptr [rsp] 00 cor dword	000000	te f. NoTerminateEnclave f. NoTerminateProcess f. NoTerminateProces f. NoTercontel f. Notercont	Address	Name Description Status P0 type:Lake Stopped P1 Type:Lake Stopped P2 Type:Lake Rot Act Not P2 VM Exit breakpoint (P0) - Reason: WRMSR e
A.2. ALL ALL <td>(a) (a) (b) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c</td> <td>All Carl (Jesc) Park All Carl (Jesc) Park (4023) Guest CR (Jesc) (4024) Guest CR (Jesc) (4024) Guest CR (Jesc) (4024) Guest EFF (4024) Guest EFF (4024) Guest EFF (4024) Guest EFF (4025) Guest EFF (4025) Guest CS Bale (4025) Guest C</td> <td>■ Brazer Strengther Strengthe</td> <td>VIEWIT LOS AND BARANDA VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT VIEWIT VIEWIT LOS LOS VIEWIT</td> <td>10 Objective taxes 6 10 Objective taxes 6 10 Objective taxes 1 10 Objective taxes 1</td>	(a) (a) (b) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c	All Carl (Jesc) Park All Carl (Jesc) Park (4023) Guest CR (Jesc) (4024) Guest CR (Jesc) (4024) Guest CR (Jesc) (4024) Guest EFF (4024) Guest EFF (4024) Guest EFF (4024) Guest EFF (4025) Guest EFF (4025) Guest CS Bale (4025) Guest C	■ Brazer Strengther Strengthe	VIEWIT LOS AND BARANDA VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT BITHAR VIEWIT LOS LEXT VIEWIT VIEWIT LOS LOS VIEWIT	10 Objective taxes 6 10 Objective taxes 6 10 Objective taxes 1 10 Objective taxes 1

Note that you can see above that the VM Exit reason is x'20 that is WRMSR, by typing in the cause command in the SourcePoint Command window, opening up the VM-Exit register in the Registers window and looking at VMEXIT_REASON, or hovering over the processor in the Viewpoint window and looking at the tooltip.

Using Intel PT with Hyper-V

Uncheck the VM Exit breakpoint, and use a VM Resume breakpoint to put the target back into the Secure Kernel.

Then open up a Trace window and activate Intel Processor Trace.

Disable the VM Resume breakpoint, and enable a breakpoint some point down into the Secure Kernel code.

Hit Go. You will collect Intel PT within the Secure Kernel! Here's an example:





Power Tip: Collecting Intel PT data from the Host while the target is in Guest mode will cause the target to crash. As one example, if you're in Host mode, set a VM Resume breakpoint, and hit Go while Intel PT is enabled, this will hose the target. As another example, while in Guest mode, if an interrupt comes along and takes the target from Guest to Host to Guest again, Intel PT will attempt to capture the Host mode instructions, and that will crash the target.

This will be addressed in a future SourcePoint release. Again, this capability will only be available via JTAG.

Suggested Hyper-V Reading

Suggested reading for this section is as follows, with some tips below.

Part 1: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI This is a basic introduction to enabling HV/SK, and the use of the VM Launch and VM Exit breakpoints.

Part 2: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI One thing to note is that the symbols for the securekernel are in fact in the public domain, on the Microsoft symbol server. You need to ensure that these are in your cache folder for SourcePoint to see them.

Part 3: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI This blog covers symbolic debug of the Secure Kernel, with Intel Processor Trace.

©2024 ASSET InterTech, Inc.



It highly recommends that the number of active processors is set to '1', in order to easily distinguish transitions with the hypervisor, secure kernel, and NT OS.

Part 4: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

Under the SourcePoint File menu, click on Macro > Load Macro... and mouse over to C:\Users\<my computer>\Documents\Arium\SourcePoint-IA_7.12.52\Macros\WinDbg and select vmcs.mac. This makes the dump, vmread, vmwrite, reason and ipt commands available. The ipt() function is crucial to ensure that Intel Processor Trace works properly between Host \Leftrightarrow Guest transitions. Note: the vmcs macro has been deprecated and replaced by the VMCS Viewer/Editor; but there's some interesting tidbits in this article regardless.

Part 5: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

This is a preamble article to using Intel AET to capture RDMSR and WRMSR events, and correlating them against the Windows MSR bitmap. For more advanced users only.

Part 6: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

An article about about Windows Secure Image Objects, an important structure used in sharing information between VTL 0 and VTL 1, the normal kernel and secure kernel.



Troubleshooting Tips on Hyper-V/VBS Enabled Targets

VM Resume breakpoint with Intel PT crashes the target

When transitioning from Host to Guest mode, and halting in Guest mode, with Intel PT active, the reads from Guest to Host memory do not succeed. This will crash the target. You will have to quit out of SourcePoint, power-cycle the target, and start over. We are working on this.

Note that if you have not disabled <u>Automatic Repair</u>, any system crash will often require two power cycles of the target. It is recommended to disable Automatic Repair with:

>bcdedit /set recoveryenabled No >NUL
>bcdedit /set bootstatuspolicy ignoreallfailures >NUL

Also, don't forget to turn off the synthetic watchdog:

>bcdedit /set {default} loadoptions "systemwatchdogpolicy=disabled"

Hardware breakpoints don't work well in the Secure Kernel

There are a few issues here, including:

- (1) BP indicators in the Code view come and go, which occurs when the current CR3 differs from CR3 when the BP was set.
- (2) BP set via WinDbg remains set in SourcePoint after the break.
- (3) The SourcePoint cause command (which displays why a breakpoint was hit) does not work. The DR6 bit is not getting set to indicate why the BP was hit.

These are all to be fixed in the upcoming release.

AET only partially functional

Intel's design for AET is only partially functional, with no knowledge of hypervisors and CR3 changes, unlike Intel PT. So, in some cases, when transitioning from Host mode to Guest mode, and where the events occur in Host mode, you don't see the actual disassembly in the Event trace window (remember, for now you can't read Host mode memory from Guest mode), but just the event itself:

©2024 ASSET InterTech, Inc.



🛃 Event Trace		83
STATE Pn ADDR INSTRUCTION	TIMESTAMP	^
-000001122 P0 Event: MSR Write: Addr=C0000087, Data=00000000000000000000000000000000000	-38.399 us	
-000000102 P0 Event: MSR Write: Addr=C0000103, Data=00000000000000	-36.770 us	
-000000057 P0 Event: MSR Read: Addr=C0000102, Data=00000000000000	+0 ns	
1		~
-000000192 Disassembly V Configure Display Filter Calibrate Refresh		

Use LBR where applicable to perhaps get some meaningful code insight, keeping in mind that LBR is an old instruction trace technology, and just uses MSRs to track to/from addresses, so it is not CR3-aware either):

🗾 Event Trace							23
STATE	\mathbf{Pn}	ADDR	INSTRUCTION		TIMEST	AMP	~
	P0	FFFFF8044EAF03A2	add	[rax],al			
	P0	FFFFF8044EAF03A4	add	[rax],al			
	P0	FFFFF8044EAF03A6	add	[rax],al			
	P0	FFFFF8044EAF03A8	add	[rax],al			
	P0	FFFFF8044EAF03AA	add	[rax],al			
	P0	FFFFF8044EAF03AC	add	[rax],al			
	P0	FFFFF8044EAF03AE	add	[rax],al			
	P0	FFFFF8044EAF03B0	add	[rax],al			
	P0	FFFFF8044EAF03B2	add	[rax],al			
	P0	FFFFF8044EAF03B4	add	[rax],al			
	P0	FFFFF8044EAF03B6	add	[rax],al			
-000000712	P0	FFFFF80447580028	MOV	rax, rcx			
	P0	FFFFF8044758002B	MOV	rcx,00000012			
	P0	FFFFF80447580032	vmcall				
-000000667	P0	Event: MSR Write	: Addr=000000	049, Data=0000000000000001	+0	ns	
-000000640	P0	FFFFF8044EAE97B0	add	[rax],al			×
-000000100		Disassembly ~ Configure	e Display.	Filter Calibrate Refresh			

This behavior of AET can actually be of benefit. In some cases, it is not possible for Intel PT to capture Host mode instruction execution while in Guest mode, without crashing the target; AET does not suffer from this limitation – you cannot see the instructions executed, but you can capture the occurrence of the event.

Support for VM Exit Reasons > 63

In the VM Exit breakpoint window, you can break on any single or multiple Basic Exit Reasons, from 0 to 63. As of the time of this writing, there are a total of 78 of them:

64 XRSTORS

65 PCONFIG

66 SPP-related event

67 UMWAIT

68 TPAUSE

69 LOADIWKEY





70 ENCLV

- 72 ENQCMD PASID translation failure
- 73 ENQCMDS PASID translation failure
- 74 Bus lock
- 75 Instruction timeout
- 76 SEAMCALL
- 77 TDCALL

It's a bit of a kludge to include the exit reasons beyond 63, but we're working on it. It will be in the next release.

Intel PT Call Chart does not work reliably

When using Intel PT for tracing code, for example, from Guest to Host transitions, you won't get the Call Chart with the pretty colors to appear; pressing the Analyze button just yields a blank display:







Although this feature works well with Hyper-V disabled, as SourcePoint is "aware" of function entries and exits, this is much more complex with VMM behavior, so we have not implemented it.



CONCLUSION

Thank you for getting this far! We hope that you have enjoyed the ride, and are using the power of SourcePoint successfully in your debugging and learning journeys. There are many new things to discover in UEFI and the Windows kernels enabled by this technology.

Feel free to browse the SourcePoint Academy at <u>https://www.asset-</u> <u>intertech.com/sourcepoint-academy/</u> for helpful reference guides, help material and "how to" videos.

If you ever have any questions, please call, email or open a Support Case here: <u>https://www.asset-intertech.com/support/</u>. We'll be glad to help!

