

SourcePoint WinDbg

Getting Started Guide for the

AAEON UP Xtreme i11

Revision 1.0



Contents

Revision History	3
Welcome!	4
Introduction	5
Configuring the target and setting up pre-requisites	8
How to Establish a SourcePoint WinDbg Session	10
Step 1: Connect SourcePoint to the target	10
Step 2: Run the StartWinDbgExdi macro	11
Step 3: Issue a Break from WinDbg	12
Step 4: Load symbols with the LoadCurrent macro	14
Getting SourcePoint to display module names as well as function names	19
Using Intel Processor Trace	21
Event Trace	26
First Step: Configuring the Intel Trace Hub	26
Second Step: Set up Architectural Event Trace	27
Troubleshooting Tips	32
Intel PT "Possible bad trace data"	32
Mangled function names	33
WinDbg doesn't like debugging over EXDI with VBS Enabled	33
WinDbg Register window slows things down badly	34
WinDbg FP register display is not working	35
Viewpoint window not refreshing initially	35
Pause in Initial Symbol Load	36
Intel PT and AET don't synchronize timestamps	36
.reload spurious error messages	36
LoadCurrent versus LoadAll	36
Windows crashes	37
Conclusion	



Revision History

Revision Number	Description	Date
1.0	Original document	December 6, 2023

Welcome!

Thank you very much for your use of our SourcePoint WinDbg product! We hope that you get great value using our tool for your debugging efforts.

If you do encounter issues or have questions on the use of SourcePoint WinDbg, please visit our support site at <u>https://www.asset-intertech.com/support/</u> to get in touch with our Support organization.

As with any new tool, mastering SourcePoint takes an investment in terms of time and effort. JTAG-based debug is a specialized area, and the JTAG, EXDI and Windows interactions sometimes behave non-deterministically – Windows in particular sometimes objects to a hardware-assisted debugger being present. We've done our best to mitigate these issues. Nonetheless, you may encounter behavior that seems non-intuitive or even wrong. If so, check out the <u>Troubleshooting</u> section of this document first. Secondly, view the Troubleshooting section of the <u>Getting Started Guide for the AAEON UP Xtreme i11</u>. Thirdly, refer to the Release Notes in the <u>SourcePoint</u> <u>Academy</u>. Finally, if you're still stuck, contact us at our Support page. We'll do our best to get you up and debugging again.

For those who are new to SourcePoint, it is highly recommended to review our <u>Getting</u> <u>Started Guide for the AAEON UP Xtreme i11</u> to get a jumpstart before using SourcePoint WinDbg. That, and the rest of the content within the <u>SourcePoint Academy</u>, are highly recommended background reading.





Introduction

It is recommended that all users have a working familiarity with SourcePoint installation, licensing, and basic usage. Installation and licensing are described fully in the <u>SourcePoint Installation and Licensing Guide</u> that is obtained from ASSET upon initial receipt of your shipment. For basic SourcePoint usage on the AAEON UP Xtreme i11, go to the <u>SourcePoint Academy</u> and read the <u>Getting Started Guide</u> for the AAEON UP Xtreme i11, <u>Xtreme i11</u> to learn the basics of SourcePoint run-control and trace.

The content that follows is based upon our using the AAEON UP Xtreme i11 Tiger Lake board. Of course, any Intel board that can support the Intel Direct Connect Interface (DCI) is suitable. Intel customers with the appropriate NDA will have access to a plethora of Customer Reference Boards (CRBs) that have DCI enabled out of the box. The AAEON UP Xtreme Whiskey Lake board (for which UEFI source code is available) is also a good platform. More information on the Whiskey Lake board is available here:

JTAG Debug using DCI on the AAEON UP Xtreme Whiskey Lake board

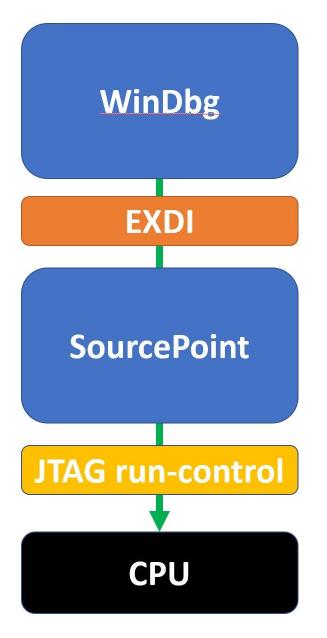
Hypervisor and OS Kernel Debug with DCI on the AAEON Whiskey Lake board

A key pre-requisite is that the platform must have debug consent enabled; that is, it must be in a debuggable state. If XDP access is available on the board, you can connect to it via the ASSET ECM-XDP3e hardware probe. Some small number of Commercial-Off-The-Shelf (COTS) boards support direct access via the Intel Direct Connect Interface (DCI). These include the AAEON UP Xtreme, and the AAEON UP Xtreme i11. Documenting the steps needed to enable JTAG-based debug on other boards is beyond the scope of this Guide; interested readers are referred to Satoshi Tanda's Debugging system with DCI and WinDbg.

The SourcePoint WinDbg application will work on Intel-based Windows platforms, on all CPUs that are supported by SourcePoint run-control. As of the time of this writing, all mainstream Intel CPUs are supported. AMD support will be in a future release.

A block diagram of how WinDbg is integrated with our SourcePoint debugger is as below:





The Microsoft 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 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 <u>https://github.com/up-board/up-community/wiki/Windows-GSG</u> site for helpful tips.

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

On the target, we'll need to prevent changing power states from disrupting run-control prematurely, and VMX and VBS need to be disabled.

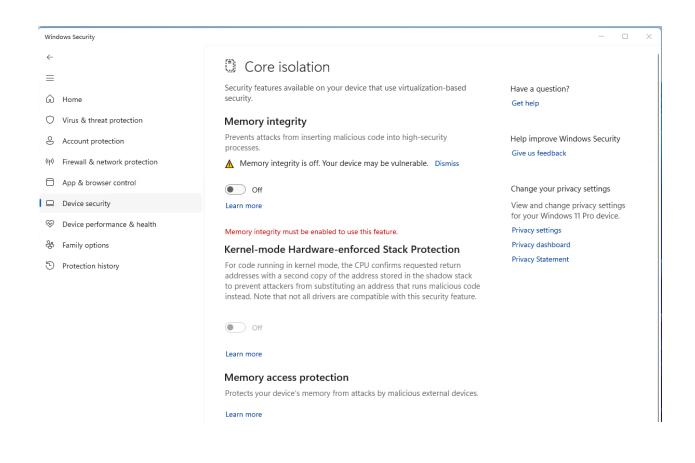
These steps are highly recommended (as of the time of writing) to have a successful debugging session.

To adjust the power settings in Windows, open the Control Panel > Hardware and Sounds > Power Options > Edit Plan Settings and set these per the below:

🗃 Edit Plan Settings				-		×
 È chit Plan Settings Change settings for the plan: High performance Choose the sleep and display settings that you want your computer to use. ^Q Turn off the display: ^{Never} ^Q Power Options ^P Advanced settings ^P Put the computer to sleep: ^{Never} ^{Never} ^Q Change advanced power settings Restore default settings for this plan ^{Savec} ^{Savec} ^{Savec} ^{Savec} ^{Savec} ^Q Restore plan defaults ^Q Restore	<i>م</i>					
	J J J	5 1	se.			
	Iurn off the display:	Never ~			?	×
	Put the computer to sleep:	Never ~		-	omize, and	
				 then choose settings that reflect how you we computer to manage power. Change settings that are currently unavailable 	int your	
			Save o	 Sleep after Setting: Never Allow hybrid sleep Setting: Off Hibernate after Setting: Never Allow wake timers Setting: Enable USR settings 	n defaults	ply
				Cancer	Арр	лу

For Windows VBS, go into Windows Security > Device Security > Core Isolation and turn Memory Integrity 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 and restart.

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.

NOTE: Debugging with VBS enabled will be covered in a future revision of this document.

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



How to Establish a SourcePoint WinDbg Session

Power Tip: If you're new to using WinDbg directly on a hardware target, getting acquainted with this beforehand will be very helpful. Setting up WinDbg/kdnet is described here:

https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-anetwork-debugging-connection-automatically

and getting started with kernel debug using WinDbg is here:

https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/getting-startedwith-windbg--kernel-mode-

NOTE: With SourcePoint WinDbg, there is no need for the Ethernet connection described in the above links, as all the traffic is over EXDI. Nonetheless, the target must be in a debuggable state prior to launching WinDbg.

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

- 1. Connect SourcePoint to the target
- 2. Run the StartWinDbgExdi macro
- 3. Issue a Break from WinDbg
- 4. 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 of the <u>Getting Started Guide for the AAEON UP Xtreme i11</u>. Your screen should look something like this:



dit View Processor Options Code Window Help						
※ 🔛 😫 😭 🚅 🎾 🥳 🦉 Load UEFI Macros 4월 4월 NPK 4월 E	Y	r			e 🛞 Viewpoint 🔍 Watc	
de P11: (64-b) Tacking # 0.0000000000000000000000000000000000	^	Rame Address	Ve Ve	D Viewpoint Name P1 TigerL P2 TigerL P3 TigerL P3 TigerL C P3 Tig	ske i ske i sko	Status Status Running Running Running Fot Active
TPT00007780012, 0F1F000000000 npp [rase00000 TPT00007780012, 0F0110000000 npp [rase000000000 TPT00007780012, 0F01100000000 npp npp npp TPT00007780012, 0F01100000000 npp npp npp npp TPT00007780012, 0F01100000000 npp	[Y ¹] 000 [Y ¹ +01] 000 [Y ¹⁺⁰⁰] 000 [Y ¹⁺⁰⁰] 000	€ ₹]3\Clobals/Locals}Stack/Classes/	Gene	ng Point Linx wat GBP ol SSI 9 R3P - SP R9 - JP R1 - Int R11 - Int R13 R14 CS SS SS SS SS SS SS SS SS SS	00000000000000000000000000000000000000	
Set Bank Set		or-aa-aaesterd, sac				

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



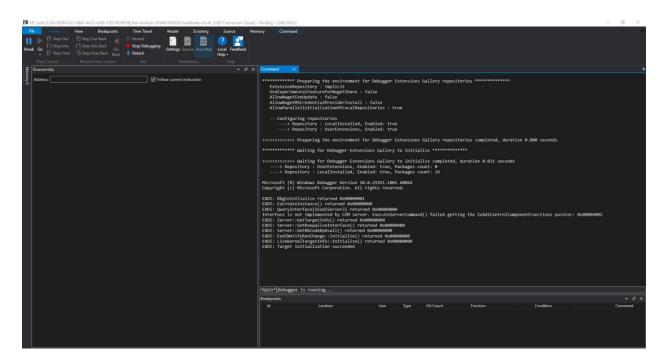
You may have to hit the Refresh button to see code displayed in the Code window.

Step 2: Run the StartWinDbgExdi macro

Next, it is time to run the StartWinDbgExdi.mac SourcePoint macro, that launches WinDbg and establishes the EXDI connection. Go to the File menu, select Macro > Load Macro... and select C:\Users\<your name>\Documents\Arium\SourcePoint-

IA_7.12.XX\Macros\WinDbg\StartWinDbgExdi.mac. After about 10 seconds, WinDbg will open:

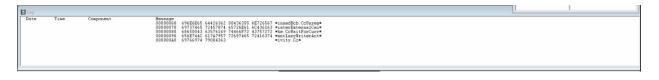




Power Tip: You can assign a button in the SourcePoint Icon Toolbar at the top to perform a one-click equivalent to the above operation, or to run any other macro. Refer to the <u>SourcePoint User Guide</u> or online help for how to do this.

Step 3: Issue a Break from WinDbg

Now, hit the Break key within WinDbg. It will take ~ 30-50 seconds for SourcePoint to 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 can see the symbol information being uploaded to WinDbg:



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 halted on logical processor 0, at a RET instruction:

E KE			6B47-4A22-A369-216370F58F2	B},Kd=VerAddr:-		ataBreaks=Exdi'											×
Fi			view Breakpoints	Time Travel	Model	Scripting	S	ource	Memory	Command							~
Brea	k Go •	{*} Step Into {} Step Over	T Step Into Back Go	Restart Stop Debuggin Detach End	ng Settings Sou	Assembly	Help										
			Reverse Flow Control		Prete			Help									
Mer	Disasse	mbly					×	Command	×								=
Memory 0		802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400 802`58f5400	2 4883.420 5 415e 5 415e 5 415e 5 415e 5 415e 5 415e 5 415e 5 456 5 4566 5 4566 5 4566 5 4566 5 4566 5 4566 5 4566 5 4566 5 4	add rsi pop riv ren int 3 int	rd ptr [rax+1	<pre>prax]] , eax], exx</pre>		EXDI: Co EXDI: Qu Interfac EXDI: Se EXDI: Se EXDI: Se EXDI: Se EXDI: EX EXDI: Ta Kernel D Connecte Symbol s Executab Loading ModLoad: Windows Product: Edition Kernel b Deburge S System U	CreateInst eryInterfa e is not i rver::Get rver::Get rver::Get diNotifyR veKernelT. rget init: ebugger co d to Wind weKernelT. earch path le search symbols fr ffff802 10 Kernel WinNt, s: build lab ase = 0xff	<pre>cc(1Exd(Sermy) mplemented argstInfo() (seepaliveInfo() blodeBpAvail mothange::Ini mothange::Ini mothange::Ini alization si mettodes: ws 10 22621 validation is: srv* path is: n fffff802 372 3f290000 fff Version 226 ite: Termin 22621.1928 ffff802 372 si ws 0.351.492</pre>	<pre>rned &x000000 wer3) returne yy COM server returned 0x0 rrface() retu (() returned itialize() re itialize() re itialize() re itialize() re itialize() re itialize() re itialize() re rfs802 402480 11 MP (4 proc ilserver Sing amd64fre.n1_ 0000 PsLoaded 1 31:9:20.96</pre>	d 0x00000000 .ExecuteSer 0000000 turned 0x00000 turned 0x0000 turned 0x000 t (Mon Sep 	verCommand 000 000000 000000 4 13:19:0: n -> ntkr p.exe prod3.2300 0xfffff8	3.197 2023 (U nnmp.exe 522-0951	тс - 5:0	3)), pt	
																(



SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\alans\E		ourcePoint-IA_7.12.33\TG	L_Demo_1.prj									- a x
File Edit View Processor Options Code Window		สป สปรีมาณ สปรีการป	eleTH 🎕 WinDbg symbol load 🕴	• 14 De	0.41 m + + +	Andreiter Contra	A Comment E	TR DATE	interes (D): Constantin			9 N II A E 9
	Load UEPI Macros	THE THE PARK THE ENDE	selfi 🦏 winDbg symbol load			Breakpoints G Code	Command	Log III Memory IF Keg	DO Viewpoint	* irace ee viewpoint 🔍 vi	aton 👘 🛌 (
Code (P0"): (64-bit) Tracking IP 00000000000000000000000000000000000				Ident		Attributes			Name	Description		Status
FFFFF00256F53FF1L CC FFFFF00256F53FF2L CC FFFFF00256F53FF2L CC FFFFF00256F53FF4L CC FFFFF00256F53FF4L CC FFFFF00256F53FF6L 66660F1F840000+ FFFFF00256F54000LFB	INT 3 INT 3 INT 3 INT 3 INT 3 INT 3	R&X] [R&X+0000000	-		undi vooriese	Autoures			C P1 T C P2 T	igerLake igerLake igerLake igerLake	Stopped Stopped Stopped Stopped	
FFFFF80258F54002L C3	RETN INT 3			Edi	t Add	Remove Remove All	Enable			General Registers (P0*)		
PTFFF015854084 CC PTFFF015854085 CC PTFFF015854085 CC PTFFF015854085 CC PTFF015854085 CC PTFF015854085 CC PTFF015854085 CC PTFF015854085 CC PTFF015854015 A155 PTFF015854015 A155 PTFF015854015 A155 PTFF015854015 A155 PTFF015854015 A155 PTFF015854025 A155 PTFF015854025 CC PTFF015854025	INT 3 INT 3 IN	PLX-00000000] EX AT RCT PX EX PX EX	EDX EDX	STATE	Ince (POY) Pa ADDE Source is not esabl Disseembly			- Giobale Value e (Locale), Stack), C	00 00 1255055	IA-32 Intel 64 General Floating Point Segment Control	Name NAX REX REX REX REX RES RES RES RES RES RES RES RES RES RES	Value 000000000000000000000000000000000000
Date Time Component		00000070 CDE82F 00000080 488B74 00000090 C3CCCC	174 114C8BCF 4D8BC448 RBD6 F77 FFFF488B 5C42048 BBC6 124 4048BFC 244488 5C420 CC CCCCCCC CCCCCCC CCCC 000 00000000	2438 *. 415E *H. 6666 *.	./H.\\$0H.1\$8* .t\$@H. \$HHA^*							
Command				_								×
Scanning Uncore												
Configuring Uncore Scanning Devices Configuring Devices Connecting Loading Command Language Extensions: C PO>	C:\Users\alans	\Documents\&rium\	SourcePoint-IA_7.12.33\Ha	cros/aa'	Naaextend.mac							
PO>												~

Step 4: 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. Follow the following steps:

Ensure that the target is in a Stopped state. Hit Break within WinDbg if necessary.

Load the symbols by going into SourcePoint, under the File menu, select Macro > Load Macro... and select C:\Users\<your

computer>\Documents\Arium\SourcePoint-

IA_7.12.XX\Macros\WinDbg\Load Current.mac. After about 10 seconds, the SourcePoint Symbols window will display the module that the current instruction is in:



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

Interestingly, SourcePoint will display the symbols associated with intelppm.pdb. WinDbg does not 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:



lame		Address	1
f .	PspStorageGetObject	FFFFF8075F1A7F58L	
f .	PspStorageInsertObject	FFFFF8075F0C54C0L	Τ
f .	PspStorageMakeSlotReadOnly	FFFFF8075F0C280CL	Τ
f .	PspStorageRemoveObject	FFFFF8075F3BBE74L	T
f .	PspStorageReplaceObject	FFFFF8075F3BBF94L	T
f .	PspSubtractAccountingValues	FFFFF8075F3B74C8L	T
f .	PspSysAppIdClaim	FFFFF8075F4779A0L	T
f .	PspSyscallProviderOptIn	FFFFF8075F3B8FFCL	T
f .	PspSyscallProviderServiceDispatch	FFFFF8075EE38BD0L	Ť
f ,	PspSyscallProviderServiceDispatchGeneric	FFFFF8075F3B91A8L	T
f .	PspSystem32String	FFFFF8075F4771E8L	T
	PspSystemCpuPartitionName	FFFFF8075F59E4E0L	Ť
	PspSystemDriveString	FFFFF8075F477208L	Ť
f .	PspSystemRootString	FFFFF8075F477218L	T
f .	PspSystemRootSymlinkName	FFFFF8075F4779C0L	Ť
	PspSystemRootTargetPrefix	FFFFF8075F4771D8L	T.
f .	PspSystemThreadStartup	FFFFF8075ED569A0L	
f .	PspTeardownPartition	FFFFF8075F3BA740L	T
f .	PspTerminateAllProcessesInJobHierarchy	FFFFF8075F0A56F8L	T
f .	PspTerminateAllThreads	FFFFF8075F1B3830L	T
f .	PspTerminatePicoProcess	FFFFF8075F3B9CF0L	Ť
2	PenTarminataProcaee	FFFFF8075F0&7624T	Ť,

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.mac" to additionally access these new symbols within SourcePoint.

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:





Edit	
Open Code Window	
Open Memory Window	
Set Breakpoint	F9
Go Until Cursor	F7
✓ Hexadecimal Display	
Show Names	
Show Types	
✓ Show Values	
Show Return Type	
Show Source Location	
Expand Children	
Collapse Children	
Viewpoint	•
Refresh	
Properties	

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:

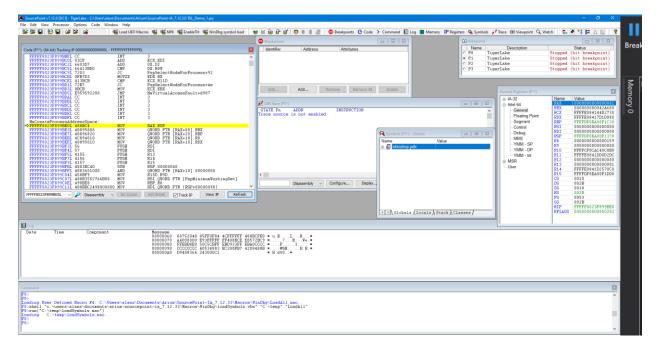
bp nt!MmCreateProcessAddressSpace

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. They are symmetrical:

📰 KD	'exdi:CLSID={B5FA1822-6B47-4A2	22-A369-216370F58F2B},K	d=VerAddr:-8786431403688,DataBreaks=Exdi			Quiet) - WinD	0bg 1.2308.2002.0						
Fil	Home View	Breakpoints Ti	me Travel Model Scripting	So	ource	Memory	Command						~
	Step Out 👘 Step		istart 📄 🚺 👘	2									
	The step Into Step	Into Back 📃 📕 Sto	op Debugging										
Break	Go Step Over	Go Over Back Back 📥 De	stach Settings Source Assembly	Local Help	Feedback								
l a l					Help								_
Memory	Disassembly			×	Command	×		* ****	10004100				-
26	Address: @\$scopeip		Follow current instruction		Response Deferred			Time (ms)	Location srv*				-
	11111002 21072004 4000/		IST' MANIN HEI FISH44011]			arch nath	h is: srv*		21.0+				
	fffff802`3f899bb9 4883c	:420 add	rsp, 20h		Executabl								
	fffff802`3f899bbd 5f	рор	rdi				or fffff802`3	f200000 ntl	krnlmp.exe -> n	tkrnlmp.exe			
	fffff802`3f899bbe c3								ntkrn1mp.exe				
	fffff802`3f899bbf cc	int						1 MP (4 procs)					
	fffff802`3f899bc0 03cf	add	ecx, edi					1Server Single					
	fffff802`3f899bc2 6603d		dx, di						lease svc prod3.2	30622-0951			
	fffff802`3f899bc5 66413		dx, r8w		Kernel ba	se = 0xff	ffff802`3f2000	000 PsLoadedMo	duleList = 0xffff	f802`3fe13460			
	fffff802`3f899bc9 7203	jb			Debug ses	sion time	e: Mon Sep 4	13:19:20.960	2023 (UTC - 5:00)				
	fffff802`3f899bcb 0fb7d		edx, bx		System Up	time: 0 d	days 0:35:43.0	964					
	fffff802`3f899bce 413bc		ecx, r11d		fffff802	58f54002		ret					
	fffff802`3f899bd1 72b7	jb	ntkrn1mp1PspSelectNodeForProce	25	0: kd> g								
	fffff802`3f899bd3 8bcb	mov	ecx, ebx		fffff802`			ret					
	fffff802 3f899bd5 e9595		ntkrnlmp!PspSelectNodeForProce 3	22					\Temp\WinDbgExdi.				
	fffff802`3f899bda cc fffff802`3f899bdb cc	int int	3				C:\Users\alan:	s\AppData\Loca	l\Temp\WinDbgExdi	.log'			
	fffff802`3f899bdc cc	int	3		0: kd> r								
	fffff802`3f899bdd cc	int	3		rip=fffff		802						
	fffff802 3f899bda cc	int	3		0: kd> 1m	11							
	fffff802 3f899bdf cc	int	3		start		end						
	nt!MmCreateProcessA						fffff802 4024	48000 <u>nt</u>	(pdb symbols) C:\Pr	ogramDa	ta\Dbg	g∖sy
	fffff802 3f899be0 488bc		rax, rsp		0: kd> .1								
	fffff802`3f899be3 48895		gword ptr [rax+8], rbx						\Local\Temp\WinDb	gExdi.log			
	fffff802`3f899be7 48896		gword ptr [rax+20h], rbp			nt!MmCre	eateProcessAde	dressSpace					
	fffff802`3f899beb 4c894		qword ptr [rax+18h], r8		0: kd> g								
	fffff802`3f899bef 48895		qword ptr [rax+10h], rdx				sAddressSpace						
	fffff802`3f899bf3 56	push	rsi		fffff802`	3f899be0	488bc4	mov ra:	x,rsp				-
	fffff802`3f899bf4 57	push	rdi										
	fffff802`3f899bf5 4155	push	r13	1									
	fffff802`3f899bf7 4156	push	r14		1: kd>								
	fffff802`3f899bf9 4157	push	r15		Breakpoints	×							-
	fffff802`3f899bfb 4883e	c40 sub	rsp, 40h										
	fffff802`3f899bff 48836	601000 and	gword ptr [rax+10h], 0				Location		Line Type	Hit Count	F	unction	
	fffff802`3f899c04 458bf	-9 mov	r15d, r9d		0 🔽	0xFFFFF	8023F899BE0		Software	1 nt!/	AmCreate	ProcessA	ddress
	fffff802`3f899c07 488b3	552766b00 mov	rsi, qword ptr [ntkrnlmp!PspMi										
	fffff802`3f899c0e 498be	8 mov	rbp, r8										
	fffff802`3f899c11 488bb		rdi, qword ptr [rsp+98h]										
	fffff802`3f899c19 48897		qword ptr [rax-38h], rsi	•									
	(1)				•								Þ
													~





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	Code 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 program 									
Share source file path map	among all programs Source Path								
 Internal globals are public 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) Tracking IP 0000000000000000	- FFFFFFFFFFFFFF		
FFFFF80274682E07L B948000000	MOV	ecx,0000048	
FFFFF80274682E0CL 0FB6D0	MOVZX	edx,al	
FFFFF80274682E0FL 418895FA000000	MOV	byte ptr [r13+000000fa],dl	
FFFFF80274682E16L 8BC2 FFFFF80274682E18L 48C1EA20	mov shr	eax, edx	
FFFFF80274682E1CL 0F30	snr wrmsr	rdx,20	
FFFFF80274682E1EL 4180A5F8000000FE		byte ptr [r13+000000f8],fe	
FFFFF80274682E26L 41840100000	mov	r10d.00000001	
FFFFF80274682E2CL 44387C2450	CMD	byte ptr [rsp+50],r15b	
FFFFF80274682E31L 7476	ie	::ntkrnlmp.PpmIdleExecuteTransition+11b9	
FFFFF80274682E33L 410FB6859A7E0000		eax,byte ptr [r13+00007e9a]	
FFFFF80274682E3BL 44887C2450	MOV	byte ptr [rsp+50],r15b	
⇒FFFFF80274682E40L 84C0	test	al,al	
FFFFF80274682E42L 7465	je	::ntkrnlmp.PpmIdleExecuteTransition+11b9	
FFFFF80274682E44L 65488B04252000+	MOA	rax,qword ptr gs:[000000000000000020]	
FFFFF80274682E4DL 4C8D05ACD1D7FF	lea	r8,qword ptr [fffff80274400000]	
FFFFF80274682E54L 418BDA	MOV	ebx,r10d	
FFFFF80274682E57L 8B4824 FFFFF80274682E5AL 4488B89A7E0000	MOV	ecx, dword ptr [rax+24]	
FFFFF80274682E5AL 4488889A7E0000 FFFFF80274682E61L 418B9488D024D000	MOV	byte ptr [rax+00007e9a],r15b edx,dword ptr [r8][rcx*4+00d024d0]	
FFFFF80274682E69L 8BCA	MOV	eax,awora ptr [roj[rcx*4+00a024a0] ecx.edx	
FFFFF80274682E6BL 8BC2	MOA	eax.edx	
FFFFF80274682E6DL 83E13F	and	ecx.0000003f	
FFFFF80274682E70L 48D3E3	sal	rbx,cl	
FFFFF80274682E73L 48F7D3	not	rbx	
	,		
FFFFF80274682E1CL V Disassembly V	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





Using Intel Processor Trace

Once using run-control is mastered, it is worthwhile testing out some of the SourcePoint advanced trace features, such as Intel PT.

First, ensure that the target is in a Stopped state. If not, issue a Break from within WinDbg.

Then, within SourcePoint, open up a Trace window, click on the Configure, and then click on the Intel PT tab at the top:



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

Click on "All" Processors to Trace", or select a processor from the list. Ensure both TSC and Cycle accurate are enabled.

Then click on the Intel PT Memory tab, and use a spare memory area to store the trace data:

©2023 ASSET InterTech, Inc.



Trace Configuration X
LBR BTS Trace Hub AET Intel PT Intel PT Memory
Trace buffer
O Use processor settings
Use SourcePoint settings:
Base address: 20000000P
Length per core: 16k \checkmark
Trace capture mode
Overwrite
○ Append
OK Cancel Help

NOTE: "Use processor settings" can be selected if the BIOS has been set up with this. For the UP Xtreme i11 board, this is not the default.

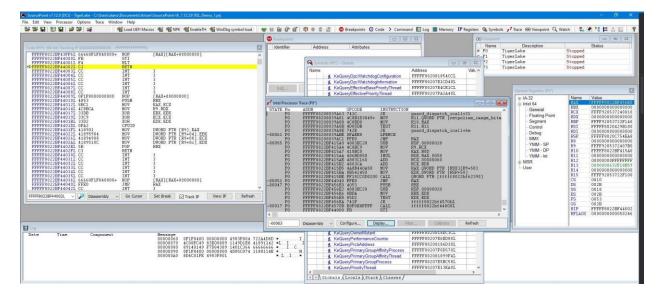
Click OK. The Intel Processor Trace window will, after a few seconds, refresh with some data. This first set of execution trace is not valid. It's a feature of SourcePoint that enables a JTAG "hotplug" dump of processor instructions if the emulator had been

©2023 ASSET InterTech, Inc. 23 SourcePoint WinDbg Getting Started Guide – UP Xtreme i11



unplugged and then plugged back in again. Only subsequent Go and Stop will yield valid trace data.

Then hit Go from within WinDbg, and then hit Break, and you will see something like the below in SourcePoint:

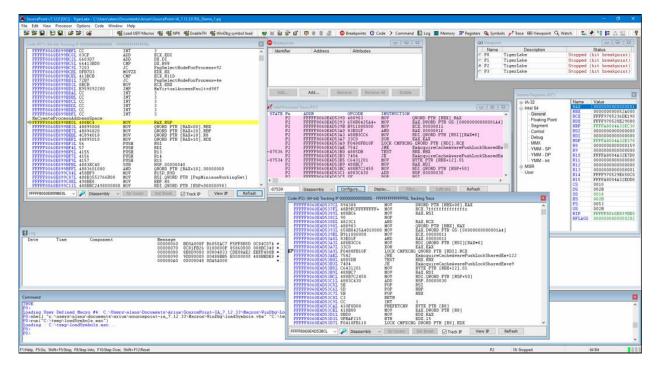


Feel free to resize the Intel Processor Trace window, and make it Floating, to see the trace data.

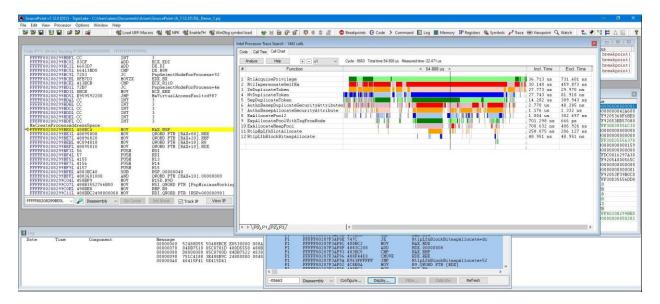
Click on the Display button within the Intel PT window, and be sure to click the appropriate buttons to ensure you see the symbols. These would include Object Code, Symbols, Pseudo-ops, Instruction Lines, Data Lines, and Labels Lines in the Disassembly section; and Source Lines in the Source code section.

You can click the cursor at any code line within the Intel Processor Trace window, and right-click to open up a Tracking Trace window that shows you the code and symbols (if available) for that line of code. You'll see the below when you open up the Tracking Trace window at an arbitrary line of the traceback:





To see a visual display of the trace data, right-click within the Intel Processor Trace window, click on Trace Search..., click on the Call Chart tab, and hit Analyze. You'll see something like this:



Move the time arrow by clicking on a section of code, or use your arrow keys. Expand the view of a particular area of code with the mouse wheel, or using the Expand (starts at x1) drop-down or +/- buttons at the top.

©2023 ASSET InterTech, Inc. 25 SourcePoint WinDbg Getting Started Guide – UP Xtreme i11



Event Trace

First Step: Configuring the Intel Trace Hub

Event tracing is accomplished with the Intel Trace Hub (ITH). Fortunately, using DCI, events supported by the ITH can be streamed directly out of system reset. The one limitation that exists is that some events (like Port IN/OUT tracing) happen so frequently at some points of the boot process that they overwhelm the capacity of the USB 2.0 (DbC2) connection and event processing, and thus cause trace buffer overflows – but these should be rare as long as the events collected are relatively close to the debug point of interest.

The first thing to do is to configure the ITH. Reset the target by clicking on the Reset button in the Icon Toolbar at the top of the screen, and it will halt at the reset vector, physical address FFFFFF0:

		\Documents\Arium\SourcePoint-IA_7.12.1	5\My Tiger Lake Proje	ct.prj (safe mode)					- t	5 ×
ile Edit View Processor Options Co	ode Window Help	📽 Load UEFI Macros 🏶 🎕 👹 🗧	K 🗗 🗗 🔍 🔟	🔹 🌲 🍙 💿 Brea	kpoints 🤁 Code 💙 Command 📓 L	.og 🎹 Memory	IP Register	s 🍇 Symbols 📌 Trace I	0€ Viewpoi	
										1
🕒 Code (P0*): (16-bit) Tracking IP 00	DOODOOL - FFFFFF					● Viewpoint				• *
FFFFFFEBL 0000	ADD	BYTE PTR [BX+SI],AL		^		Name		Description		Status
FFFFFFEDL 0000	ADD	BYTE PTR [BX+SI],AL					ligerLak		Stopped	
FFFFFFFFL 00	DB NOP									
FFFFFFF0L 90 FFFFFFF1L 90	NOP									
FFFFFFF2L E923C0	JMP	near16 ptr ffffc018L				• F2				
FFFFFFF5L 0000	ADD	BYTE PTR [BX+SI],AL				<				
FFFFFFF7L 00FB	ADD	BL, BH								
FFFFFF9L 0000	ADD	BYTE PTR [BX+SI], AL				IP General Reg				
FFFFFFBL 0000	ADD	BYTE PTR [BX+SI], AL				€IA-32	Name	Value	- II.	
FFFFFFFDL 00FC FFFFFFFFL FF	ADD DB	AH,BH ff				Intel 64	RAX	000000000000000000000000000000000000000	0	
	55					General	RBX	000000000000000000000000000000000000000		
						-Floating F	RCX	000000000000000000000000000000000000000	0	
						Segment	RDX	0000000000806C		
						Control	RBP	000000000000000000000000000000000000000		
				~		Debug	RSI	000000000000000000000000000000000000000		
FFFFFFOL V 🖉 Dis	assembly ~ G	o Cursor Set Break Track IP	View IP Re	fresh		MMX	RDI	0000000000000000		
		o track if				YMM - SP	RSP R8	000000000000000000000000000000000000000		
						-YMM - DI	RO	000000000000000000000000000000000000000		
		Breakpoints				YMM - In	R9	000000000000000000000000000000000000000		
		Identifier A	ddress A	ttributes		■MSR	R11	000000000000000000000000000000000000000		
						User	R12	000000000000000000000000000000000000000		
							R13	000000000000000000000000000000000000000	0	
							R14	000000000000000000000000000000000000000	0	
		Edit Add	Remove	Remove All E	Enable Disable All		R15	000000000000000000000000000000000000000	0	
							CS	F000		
Log							DS	0000		
Date Time	Component	Message					SS ES	0000		
11/17/2021 15:20:08.130			xpress Base				FS	0000		
-							GS	0000		
							RIP	0000000000000FFF	0	
						< >	RFLAGS	000000000001000		
										*
ommand										
canning Devices										^
onfiguring Devices										
onnecting										
oading Command Language	Extensions:	C:\Users\alans\Documents\Ar:	.um\SourcePoint	-IA_7.12.15\Mac	cros\aa\aaextend.mac					
·0>										
	:\Users\alans	\Documents\Arium\SourcePoint	-IA 7.12.15\Ma	cros\Intel\ADL	TCO Timer Disable.mac					
20>										
										v
rget Stopped: Target reset						P0 18: St	onned	Special	Halt Mode	

The next step is to run the ITH macros that enable the Trace Hub and hide it from the OS. Fortunately, SourcePoint comes equipped with a couple of macro buttons built in to make this process easier, At the top of the screen, you'll see two buttons labeled NPK



and EnableTH. Click on these, one at a time. Wait about five seconds after clicking on the second button.

You can then boot up to Windows (or not).

Second Step: Set up Architectural Event Trace

Now, it's time to tell the Trace Hub what you want to trace. 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 for the Tiger Lake platform:

BR B	TS	Trace Hub	AET	Intel PT	Intel PT Mem	ory
Master	rs to tr	ace				
O Nor	ne					
 List 	: 18	3				
Trace	routing	9				
Trace	e Hub:	DbC		~		
Intel	PT:	System N	1emory	~		
AET:		Trace Hu	b	~		
		n <mark>ory trace b</mark> settings	ouffer			
🔘 Use	e BIOS	settings cePoint set	tings)p		
🔘 Use	e BIOS e Sourc e addre	settings cePoint set)p		P





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:



R BTS	Trace Hub	AET	Intel PT	Intel PT M	emory
Processo	rs to trace				
ONone					
-					
All					
O List:	PO				
	(e.g., P0, P4-	P7)			
	(
Event sh	aring				
Apply	events to all p	rocesso	rs		
0.00			1000		
	events to:				
O Apply				ed LBR	^
Event HW/SW IRET Except RDMSR/ Port I Code b Data b BTM SMI/NM	Interrupt ion WRMSR n/Out reakpoint reakpoint I/RSM R/MWAIT			ed LBR	
Event HW/SW IRET Except RDMSR/ Port I Code b Data b BTM SMI/NM MONITO	Interrupt ion WRMSR n/Out reakpoint reakpoint I/RSM R/MWAIT				*
Event HW/SW IRET Except RDMSR/ Port I Code b Data b BTM SMI/NM MONITO WBINVD	Interrupt ion WRMSR n/Out reakpoint reakpoint I/RSM R/MWAIT				~
Event HW/SW IRET Except RDMSR/ Port I Code b Data b BTM SMI/NM MONITO WBINVD	Interrupt ion WRMSR n/Out reakpoint reakpoint I/RSM R/MWAIT	A			~

Now, you can simply do a Go/Stop to capture the event trace data. Below shows the use the Command window to simulate a break on any read/write of, say, port x'CF8', the PCI CONFIG_ADDRESS. This is conveniently done by issuing at the Command window P0> prompt:

go til cf8io

©2023 ASSET InterTech, Inc.



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

ace Hub - SW/FW Trace			
			Sta
TE ADDR data available - Unable STATE	Pn ADDR INSTRUCTION		ed (hit
ata avallable - Unable SiAIL	Event: Port Out: Port=0021, Data=000000FF		ed (hit
-0000005			
-0000003	Event: Port Out: Port=00A1, Data=000000FF	brobb	ed (hit
-0000005		-66,406 us	ed (hit
-0000005	P0 000000067EE16D7 OUT DX,AL Event: Port In: Port=1830	-00.400 us	
0000005		CC 07C	
-0000005		-66.276 us	
	Event: Port In: Port=1830, Data=80002033		
-0000004		-64.036 us	
	Event: Port Out: Port=1830, Data=80002030		
-0000004		-62.240 us 000008	
	Event: Port In: Port=1830	000000	
-0000004		-51.406 us 000000	
	Event: Port In: Port=1830, Data=80002033	000CF8	
-0000003		-48.281 us OA3CC0	
	Event: Port Out: Port=1830, Data=80002033	0CD000	
-0000003	P0 000000067EE179F OUT DX, EAX	-46.484 us B81C00	
	Event: Port In: Port=1830	0A3C40	
-0000002	P0 000000067EE17A0 IN EAX, DX	-46.354 us E9D3B0	
	Event: Port In: Port=1830, Data=80002033		
-0000002		-44.115 us 00000E	
	Event: Port Out: Port=1830, Data=80002033	000000	
-0000002		-42.318 us 0A3AF0	
	Event: Port Out: Port=0021, Data=000000FF	A23018	
-0000001		-24,922 us 000002	
0375 Disassembly ~	Event: Port Out: Port=00A1. Data=000000FF	000001	
-0000001		-17.161 us	
	Event: Port Out: Port=0070, Data=000000B2		
Time -0000001		-8.099 us	
/2021 16:35:39.976 r	Event: Port Out: Port=0076, Data=00000005	-0.035 us	
/2021 16:35:49.222 1 -0000000		-494.791 ns	
	Event: Port Out: Port=OCF8, Data=80000008	-494.791 115	
-0000000		+0 ns	
-0000000	PO UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU	+0 IIS	
-00000055	Disassembly Y Configure Display Filter Calibrate Refresh		
1			
	Documents\Arium\SourcePoint=IA 7.12.15\Macros\Intel\ADL TCO Timer Dis	able.mac	
g Reset (after): C:\Users\alar			
EnableForce("cpcie=1", "tsact=			
EnableForce("cpcie=1", "tsact= til cf8io			
EnableForce("cpcie=1", "tsact= til cf8io til cf8io			
EnableForce("cpcie=1", "tsact= til cf8io til cf8io til cf8io til cf8io			
ng Reset (after): C:\Users\alar &DableForce("cpcie=1", "tsact= til cf8io til cf8io til cf8io til cf8io			
EnableForce("cpcie=1", "tsact= til cf8io til cf8io til cf8io til cf8io			
EnableForce("cpcie=1", "tsact= til cf8io til cf8io til cf8io til cf8io			

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

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.

e Configuration			×
R BTS Trace Hub AET	Intel PT	Intel PT Memory	
Processors to trace			
○ None			
Onone			
List: p0			
(e.g., P0, P4-P7)			
 Apply events to all process Apply events to: 	ors		
Event	Enabled	LBR	
HW/SW Interrupt			
IRET			
Exception			
RDMSR/WRMSR	⊻		
Port In/Out	⊻		
Code breakpoint			
Data breakpoint BTM			
SMI/NMI/RSM			
MONITOR/MWAIT			
WBINVD			
SGX			
	Advanced	Clear all	



SourcePoint WinDbg Getting Started Guide – UP Xtreme i11



Troubleshooting Tips

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.

Intel PT "Possible bad trace data"

The format of Intel PT trace data changes from SKU to SKU, platform to platform. It's a lot for us to keep up with. More often than not, you'll see this error message(s) displayed below:



SourcePoint v7.12.0 (DCI) - TigerLake - C\\Users\alans\Documents\Ariu	um\SourcePoint-IA_7.12.33\TGL_Demo_1.prj						– Ø X
File Edit View Processor Options Trace Window Help					-		
📽 📽 🔛 😫 📽 😂 🖓 Load UEFI Ma	acros 🔹 📽 NPK 🥵 EnableTH 📽 WinDbg symbol load 🛛 4					* Trace 00 Viewpoint Q, W	
		Breakpoints			Name	Description	Status
Code (P0*): (64-bit) Tracking IP 000000000000000L - FFFFFFFFFFFFFF FFFFF8055B67006AL 7F04 JG		Identifier Address	a Attributes		@ P0 Tig	erLake	Stopped
FFFFF8055867006CL FR STI	IIIII8055b670070L					erLake	Stopped
-0FFFFF8055B67006EL EB01 JMP	fffff8055b670071L					erLake	Stopped
Improve Starbard Starbard FPE STI Improve Starbard Starbard CC Improve Starbard Improve Starbard Starbard CC Improve Starbard Improve Starbard CC Improve Starbard <	} } PRAT[EAX+00000000] 00000010380].00000000 DVGC0 PTF GS [000000000010380].0000000 A0000000 A1070 A1070 ffiffersber0005L4 A1070 ffiffersber0005L4 A1070 j A1070 </td <td>Building trace display Status: Analyzing Trace Progress: Possble bad trace d</td> <td>Emore Renor Al JECTRONICO De dissessably error *** ats. See bg for detail. not</td> <td></td> <td></td> <td>Serveral Register (PP) (1) Av 32 (2) Heat 44 (2) Serveral (2) Server</td> <td>Internet Value Internet Value Internet Value Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFF05581150 Internet PFFF055818100 Internet PFFF055811100 Interne PFFF055811100 <</td>	Building trace display Status: Analyzing Trace Progress: Possble bad trace d	Emore Renor Al JECTRONICO De dissessably error *** ats. See bg for detail. not			Serveral Register (PP) (1) Av 32 (2) Heat 44 (2) Serveral (2) Server	Internet Value Internet Value Internet Value Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFFF05581150 Internet PFFF05581150 Internet PFFF055818100 Internet PFFF055811100 Interne PFFF055811100 <
				Globals (Locals) Stack) C	lasses /		RIP FFFF8055E67006E RFLAGS 00000000010246
Log						1	
Date Time Component © 08/29/2023 13:12:36.359 stop	Message 00000000 00000000 00000000 00000000 0000	••					
Command Scanning Uncore							
Configuring Uncore Scanning Devices Configuring Devices Loading Command Language Extensions: C:\Users\al P0; P0;	lans\Documents\Arium\SourcePoint-IA_7.12.33\Ma	cros\aa\aaextend.mac					
Building trace display Press Ctrl-Break to abort					PO	1F: Running	

SourcePoint will say "Possible bad trace data. See log for details." It will then do its level-best to make sense of the trace data, and present it in a legible form. Most of the time, it succeeds.

Mangled function names

You may sometimes see a mangled function name, as in:

JNE ??_C@_0BH@CBDMLJDN@RtlCreateUnicodeString@

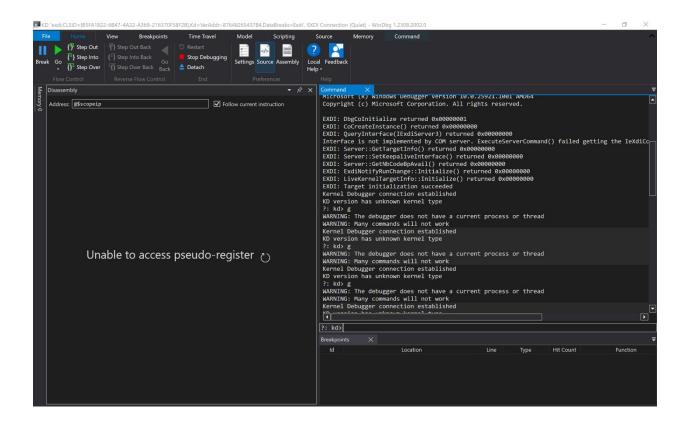
SourcePoint does not have a built-in C++ name demangler. It's on the to-do list.

WinDbg doesn't like debugging over EXDI with VBS Enabled

Virtualization-Based Security must be turned off on the target for the EXDI connection to work. If VBS is on, when you launch StartWinDbgEXDI.vbs, you'll see the below:







Interestingly, the SourcePoint connection to the target still works fine. We are working on this.

WinDbg Register window slows things down badly

Having the Registers window open within WinDbg slows things down quite a bit. Updating the WinDbg Registers view causes EXDI to transact hundreds of memory reads. This can cause problems. In particular, it has been seen to cause failures of the symbol load from WinDbg to SourcePoint. LoadCurrent.mac will fail quietly.

If it remains open, you may at some point see the below within WinDbg:

Registers		
Name	Value	
User	Unexpected	
Kernel	Unexpected	
SIMD	Unexpected	
VFP	Unexpected	
FloatingPoint	Unexpected	
CET	Unexpected	



Close it out (presuming that you had it open), and consider using the SourcePoint Registers window instead. You can see all the GPRs, Control Registers, Debug Registers, MSRs, VMX registers, and many more. And context-sensitive help (rightclick) provides the selective ability to find a particular MSR, open a Code or Memory window, and other features.

IA-32	Name	Value	Number	Description
Intel 64	IA32_ARCH_CAPABILITIES	000000000000006B	10AH	Enumeration of Architectural Features
General	IA32_APIC_BASE	00000000FEE00900	1BH	APIC Base
	MSR_BIOS_DEBUG		A7H	Indicates If WRMSR 79H Failed To Configure PRM Memory and
- Floating Point	MSR_BIOS_DONE		151H	BIOS Done
- Segment	MSR_BIOS_MCU_ERRORCODE		AOH	BIOS MCU ERRORCODE
- Control	IA32_BIOS_SIGN_ID	0000008A0000000	8BH	BIOS Update Signature Register
- Debug	IA32_BIOS_UPDT_TRIG	001C000000000001	79H	BIOS Update Trigger Register
MMX	IA32_BNDCFGS	000000000000000000000000000000000000000	D90H	Supervisor State of MPX Configuration
	IA32_CLOCK_MODULATION	000000000000000000000000000000000000000	19AH	ACPI Thermal Mo ConfigTDP Contr
-YMM - SP	MSR_CONFIG_TDP_CONTROL	000000000000000000000000000000000000000	64BH	
- YMM - DP	MSR_CONFIG_TDP_LEVEL1		649H	ConfigTDP Level
YMM - Int	MSR_CONFIG_TDP_LEVEL2	0000000000160078	64AH	ConfigTDP Level MSR Address: 151 v
MSR	MSR_CONFIG_TDP_NOMINAL	000000000000001E	648H	Nominal TDP Rat
General	IA32_COPY_LOCAL_TO_PLATFORM	**********	D91H	Copy Local Stat
MTRR	IA32_COPY_PLATFORM_TO_LOCAL	**********	D92H	Copy Platform S OK Cancel
	IA32_COPY_STATUS		990H	Status of Most
Machine Check	MSR_CORE_C1_RESIDENCY		660H	Core C1 Residency Counter
- Debug	MSR_CORE_C3_RESIDENCY	000000000000000000000000000000000000000	662H	Core C3 Residency Counter
X2APIC	MSR_CORE_C6_RESIDENCY		3FDH	Core C6 Residency Counter
UNCORE	MSR_CORE_C7_RESIDENCY		3FEH	Core C7 Residency Counter
VMX	IA32_CORE_CAPABILITIES	000000000000000000000000000000000000000	CFH	Core Capability
	MSR_CORE_GFXE_OVERLAP_C0	0000060AA84E7E62	65BH	Core and Graphics Engine Overlapped CO Residency
Jser	MSR_CORE_HDC_RESIDENCY	000000000000000000000000000000000000000	653H	Core HDC Idle Residency.
	MSR_CORE_UARCH_CTL	000000000000000000000000000000000000000	541H	Core Microarchitecture Control
	IA32_CPU_DCA_CAP	000000000000000000000000000000000000000	1F9H	CPU Direct Cache Access Capability
	MSR_CRASHLOG_CONTROL	000000000000000000000000000000000000000	1F1H	Write Data to a Crash Log Configuration

WinDbg FP register display is not working

WinDbg does not display the floating point registers. SourcePoint displays the registers correctly.

Viewpoint window not refreshing initially

When you first halt the target in Windows from SourcePoint, not all logical processors will display "Stopped" properly:

	Name	Description	Status	
۲	P1	TigerLake	Stopped	
0	P2	TigerLake	Stopped	
0	P3	TigerLake	Stopped	
0	P4	TigerLake	Not Active	



This is an outcome of an auto-scrolling usability enhancement feature developed recently to allow large-thread-count binned platforms to dynamically display the start of the stopped processors. This discrepancy can be safely ignored. Once all active processors have been discovered by run-control, the display will refresh correctly.

Pause in Initial Symbol Load

Intermittently, after issuing the first Break in WinDbg, 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.

Intel PT and AET don't synchronize timestamps

It is possible to run both Intel PT and AET concurrently, with synchronized timestamps. Thus, one could click on a particular event within the AET window, and the Intel PT window will automatically refresh and display the exact code that invoked the event. Then, you can trace back as far as you wish to view the code execution that might have led to, for example, a concurrency bug. Very powerful.

However, sometimes the Intel PT window doesn't refresh properly. This is to be fixed in the next release. As a workaround, for the moment, use AET in conjunction with LBR.

.reload spurious error messages

.reload issues from WinDbg runs very slowly. And, towards the end, it puts out spurious error messages. These can be ignored. This will be addressed in an upcoming release.

LoadCurrent versus LoadAll

The LoadCurrent macro makes the symbols available within the module at the current instruction pointer visible to SourcePoint. In an upcoming release, we will deliver a LoadAll macro that loads all symbols visible within WinDbg to SourcePoint.

Windows crashes

If you work with SourcePoint WinDbg long enough, you'll likely crash Windows at some point. Sometimes Automatic Repair will clean things up, sometimes it won't. In which case you will need to re-install Windows. Really, it's no different from reinstalling Windows in a VM, only more onerous.

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

Conclusion

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

