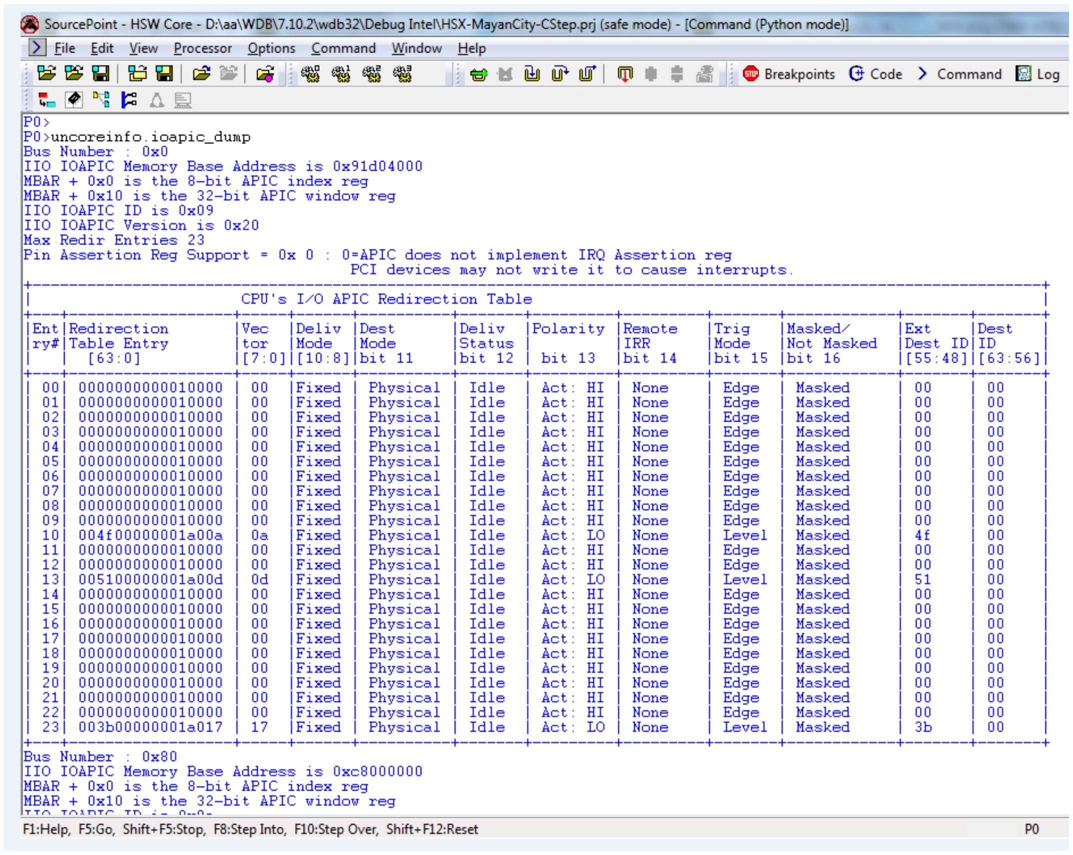
Platform Debug using Intel® Customer Scripts (CScripts)


SourcePoint - HSW Core - D:\aa\WDB\7.10.2\wdb32\Debug Intel\HSX-MayanCity-CStep.prj (safe mode) - [Command (Python mode)]
File Edit View Processor Options Command Window Help
Breakpoints Code Command Log
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset
P0

```
P0>
P0>uncoreinfo.ioapic_dump
Bus Number : 0x0
IIO IOAPIC Memory Base Address is 0x91d04000
MBAR + 0x0 is the 8-bit APIC index reg
MBAR + 0x10 is the 32-bit APIC window reg
IIO IOAPIC ID is 0x09
IIO IOAPIC Version is 0x20
Max Redir Entries 23
Pin Assertion Reg Support = 0x 0 : 0=APIC does not implement IRQ Assertion reg
                                  PCI devices may not write it to cause interrupts.
+----------------------------------------------------------------------------------------------------------------+
|                       CPU's I/O APIC Redirection Table                                                         |
+---+------------------+-----+------+----------+-------+---------+----------+-------+------------+-------+-------+
|Ent|Redirection       |Vec  |Deliv |Dest      |Deliv  |Polarity |Remote    |Trig   |Masked/     |Ext    |Dest   |
|ry#|Table Entry       |tor  |Mode  |Mode      |Status |         |IRR       |Mode   |Not Masked  |Dest ID|ID     |
|   |   [63:0]         |[7:0]|[10:8]|bit 11    |bit 12 | bit 13  |bit 14    |bit 15 |bit 16      |[55:48]|[63:56]|
+---+------------------+-----+------+----------+-------+---------+----------+-------+------------+-------+-------+
| 00| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 01| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 02| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 03| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 04| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 05| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 06| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 07| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 08| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 09| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 10| 004f00000001a00a | 0a  |Fixed | Physical | Idle  | Act: LO | None     | Level | Masked     | 4f    | 00    |
| 11| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 12| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 13| 005100000001a00d | 0d  |Fixed | Physical | Idle  | Act: LO | None     | Level | Masked     | 51    | 00    |
| 14| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 15| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 16| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 17| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 18| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 19| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 20| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 21| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 22| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI | None     | Edge  | Masked     | 00    | 00    |
| 23| 003b00000001a017 | 17  |Fixed | Physical | Idle  | Act: LO | None     | Level | Masked     | 3b    | 00    |
+---+------------------+-----+------+----------+-------+---------+----------+-------+------------+-------+-------+
Bus Number : 0x80
IIO IOAPIC Memory Base Address is 0xc8000000
MBAR + 0x0 is the 8-bit APIC index reg
MBAR + 0x10 is the 32-bit APIC window reg
```

eBook

by Larry Traylor

SourcePoint™
More visibility. Software Debug and Trace.

By Larry Traylor, Vice President Software Debug and Trace

Larry Traylor co-founded Arium Corporation in 1977. Larry served as president, CEO, and chairman of the board of Arium. He was instrumental in driving that company's vision for product creation of hardware-based program debug and code trace tools. In 2013, Larry joined ASSET InterTech when Arium was acquired by ASSET. He has a BSEE from Cal-Poly Pomona.

Table of Contents

Table of Figures

Introduction

Bringing up a new computer based on Intel processor(s) can be a daunting task. Intel® provides a group of scripts to its customers, that is intended to provide convenience and help when bringing up a new hardware design or debugging the firmware on a new hardware design, which uses Intel processors. The methods can range from basic state dump (register and memory dumps) to error injection/logging and sideband-enabled postmortem access.

These Intel Customer Scripts have classically been called 'CScripts' by Intel in order to differentiate them from the scripts Intel uses internally for silicon validation. This eBook refers to them as Intel Customer Scripts (ICS). This eBook is intended as an overview and introduction to these scripts and not an in-depth reference manual. There are several packages of ICSs, each with different features and functionality. However, there are many commonalities among them.

To understand the diversities in ICSs, it is necessary to take a brief look at their history. In addition, due to both scripting language and language revisions, it is important to know what the required runtime environment for each of these packages of scripts is.

While the functionality of each of these packages varies, each has fundamental groups of services. These groups will be described.

It also can be important to know what is coming from Intel and tool vendors in the future, so tool plans can be made accordingly. Intel is making great progress in providing a uniform environment to run these scripts and in enabling reasonable and available support for its customers.

What Intel Customer Scripts Do

ICSs are provided by Intel to their Original Equipment Manufacturers (OEMs), Original Design Manufacturers (ODMs) and others to ease the task of bringing products to market which are based on Intel Processors. ICSs are used at board bring up as well as at various times during debug or when an error occurs. They are also used to help find the cause of catastrophic failures such as 'three-strike-failures.' For many of the first use cases, the basic goal is to display the way a system is currently configured in a manner that can be easily compared to a known working version of a given platform type.

For example, the codename for the generic Intel topology of a two-package server system based on Haswell is code-named Grantley. Therefore, Intel distributes a package of ICSs for this platform. If an OEM has just powered up a new design that is based on Grantley and has run well into the PEIM-phase of a UEFI BIOS, then the OEM might run these Grantley (actually called "Haswell EP/…") ICSs and dump the state of many registers to a file that can be compared to one generated on the Intel reference design. If these files are identical or the differences can be logically explained, then much of PEIM module code that has run and the hardware it is accessing is likely mostly functional.

The teams at Intel refer to four major functional groups of features in ICSs as the BIG 4. These are:

a. sysError - extracts and decodes all error registers from each socket
b. sysInfo - displays decoded CPUID leaves, revision number of code and micro-code patches
c. sysTopo (formally sysStatus) – displays DDR, PCI, USB , SATA information
d. sysDump - dumps MCRs and other registers

In addition to the BIG 4 most sets of ICSs also contain:

a. Error injection, logging, and display
b. Other handy routines

The above groupings provide a very high level look at what is contained in ICS packages.

A Brief History of Intel Customer Scripts

ICSs have been around at Intel for many years. They were originally written in the C-like command language that has been in the Intel debuggers since the 1980s. Later, Intel began using a version of Python with an interface to the original ITP product.

One schema used to model the target (board and ICs in a system) was called Python for System Validation (PythonSV). Scripts in this language have been written and used inside Intel for several years.

More recently, Intel Customer Scripts have been published in Python. The ICSs based on the existing scripts have been written in PythonSV with an interface directly for the ITP II.

Additionally, newer ICS packages are sometimes written in the native Python of the ITP II called PythonCLI. This version uses a different target schema. Most client packages today are written in PythonCLI while server ICSs are all written in PythonSV. It is important to remember that both of these kinds of packages are written in Python. Just the domain naming schema is slightly different between the two.

A second change, moving from Python v2.6 to Python 2.7, occurred in 2014. Because some of the scripts within these packages are precompiled, it is mandatory that the scripts be run in a Python environment in the version that matches the scripts.

There are three main bodies of scripts. These are:

a. Client platforms ("Core") (Python-CLI)
b. Server Platforms ("XEON") (Python-SV)
c. Micro-server platforms (ATOM) (Python-CLI)

There are many versions of each of these ICS packages corresponding to different generations of the processors, such as Nehalem, Ivytown, Haswell, etc.

In all cases, the latest versions are produced in Python 2.7. When running older packages it is important to use a tool version that supports the correct Python version.

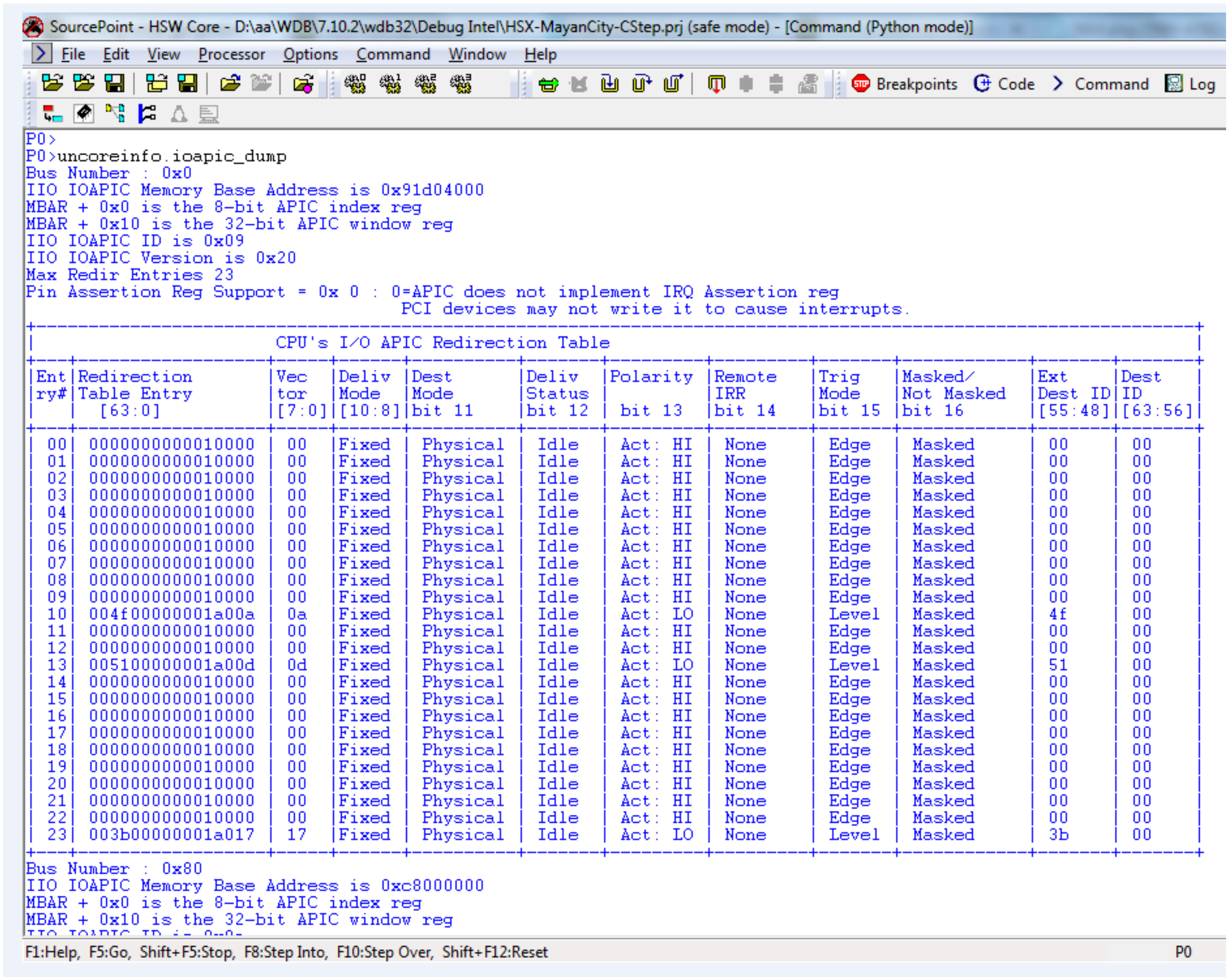
All later ICS packages are available on Intel's IBL website to those with authorized login credentials.

A More Detailed Look at Some Intel Customer Script Features

As mentioned earlier, there are four major classes of ICS functions plus several important miscellaneous ones. In order to look a little deeper into what exactly these scripts are, the following five classes will be examined:

a. sysInfo
b. sysTopo
c. sysError
d. sysDump
e. Error Injection

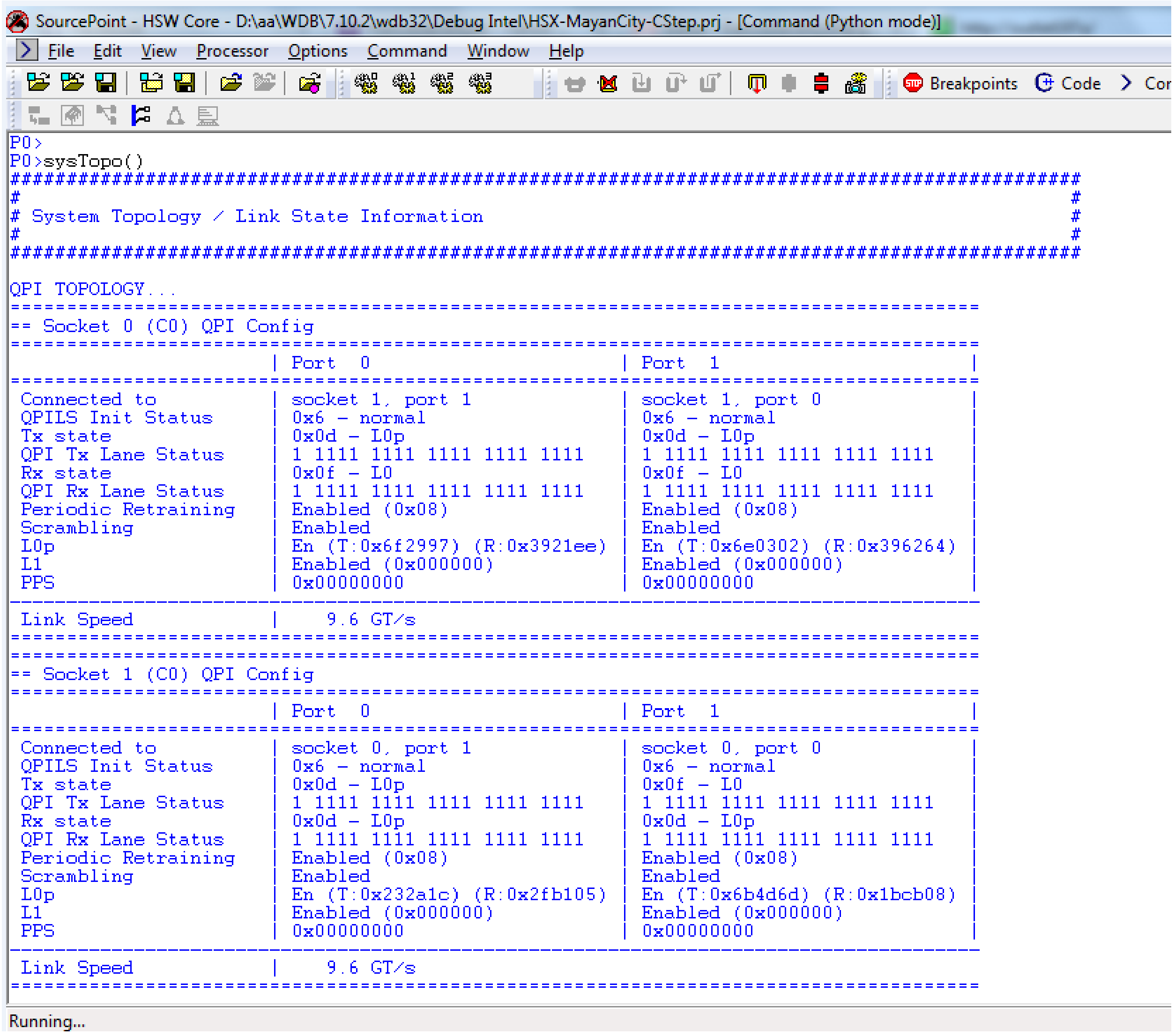
While there are functions outside of these classes, this set will provide a good understanding as well as cover the majority of the functions. Figure 1 shows an example output from running an ICS function. In this case the function was ioapic_dump.


SourcePoint - HSW Core - D:\aa\WDB\7.10.2\wdb32\Debug Intel\HSX-MayanCity-CStep.prj (safe mode) - [Command (Python mode)]
File Edit View Processor Options Command Window Help
Breakpoints Code Command Log
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset
P0

```
P0>
P0>uncoreinfo.ioapic_dump
Bus Number : 0x0
IIO IOAPIC Memory Base Address is 0x91d04000
MBAR + 0x0 is the 8-bit APIC index reg
MBAR + 0x10 is the 32-bit APIC window reg
IIO IOAPIC ID is 0x09
IIO IOAPIC Version is 0x20
Max Redir Entries 23
Pin Assertion Reg Support = 0x 0 : 0=APIC does not implement IRQ Assertion reg
                                   PCI devices may not write it to cause interrupts.
+-----------------------------------------------------------------------------------------------------------+
|                         CPU's I/O APIC Redirection Table                                                  |
+---+------------------+-----+------+----------+-------+----------+----------+-------+------------+-------+-------+
|Ent|Redirection       |Vec  |Deliv |Dest      |Deliv  |Polarity  |Remote    |Trig   |Masked/     |Ext    |Dest   |
|ry#|Table Entry       |tor  |Mode  |Mode      |Status |          |IRR       |Mode   |Not Masked  |Dest ID|ID     |
|   |  [63:0]          |[7:0]|[10:8]|bit 11    |bit 12 | bit 13   |bit 14    |bit 15 |bit 16      |[55:48]|[63:56]|
+---+------------------+-----+------+----------+-------+----------+----------+-------+------------+-------+-------+
| 00| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 01| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 02| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 03| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 04| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 05| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 06| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 07| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 08| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 09| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 10| 004f00000001a00a | 0a  |Fixed | Physical | Idle  | Act: LO  | None     | Level | Masked     | 4f    | 00    |
| 11| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 12| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 13| 005100000001a00d | 0d  |Fixed | Physical | Idle  | Act: LO  | None     | Level | Masked     | 51    | 00    |
| 14| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 15| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 16| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 17| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 18| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 19| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 20| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 21| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 22| 0000000000010000 | 00  |Fixed | Physical | Idle  | Act: HI  | None     | Edge  | Masked     | 00    | 00    |
| 23| 003b00000001a017 | 17  |Fixed | Physical | Idle  | Act: LO  | None     | Level | Masked     | 3b    | 00    |
+---+------------------+-----+------+----------+-------+----------+----------+-------+------------+-------+-------+
Bus Number : 0x80
IIO IOAPIC Memory Base Address is 0xc8000000
MBAR + 0x0 is the 8-bit APIC index reg
MBAR + 0x10 is the 32-bit APIC window reg
```

Figure 1: An example of an ICS output

SysTopo must be run in the emulator halted state. When run, it will print a tabular output containing information about what is in each socket on the board. Details include number of cores, SKU, frequency, along with many other details of what is physically in the target systems. This will also indicate what revisions of firmware and patches are contained in the system. This is a good place to start to see if the system is configured as expected, is up to date as expected and all the chips are accessible. Figure 2 shows a sysTopo() function output.


SourcePoint - HSW Core - D:\aa\WDB\7.10.2\wdb32\Debug Intel\HSX-MayanCity-CStep.prj - [Command (Python mode)]
File Edit View Processor Options Command Window Help
Breakpoints Code Cor
Running...

```
P0>
P0>sysTopo()
###############################################################################################
#                                                                                             #
# System Topology / Link State Information                                                    #
#                                                                                             #
###############################################################################################

QPI TOPOLOGY...
======================================================================================
== Socket 0 (C0) QPI Config
======================================================================================
                        | Port  0                      | Port  1                      |
======================================================================================
 Connected to           | socket 1, port 1             | socket 1, port 0             |
 QPILS Init Status      | 0x6 - normal                 | 0x6 - normal                 |
 Tx state               | 0x0d - L0p                   | 0x0d - L0p                   |
 QPI Tx Lane Status     | 1 1111 1111 1111 1111 1111   | 1 1111 1111 1111 1111 1111   |
 Rx state               | 0x0f - L0                    | 0x0f - L0                    |
 QPI Rx Lane Status     | 1 1111 1111 1111 1111 1111   | 1 1111 1111 1111 1111 1111   |
 Periodic Retraining    | Enabled (0x08)               | Enabled (0x08)               |
 Scrambling             | Enabled                      | Enabled                      |
 L0p                    | En (T:0x6f2997) (R:0x3921ee) | En (T:0x6e0302) (R:0x396264) |
 L1                     | Enabled (0x000000)           | Enabled (0x000000)           |
 PPS                    | 0x00000000                   | 0x00000000                   |
--------------------------------------------------------------------------------------
 Link Speed             |    9.6 GT/s
======================================================================================
======================================================================================
== Socket 1 (C0) QPI Config
======================================================================================
                        | Port  0                      | Port  1                      |
======================================================================================
 Connected to           | socket 0, port 1             | socket 0, port 0             |
 QPILS Init Status      | 0x6 - normal                 | 0x6 - normal                 |
 Tx state               | 0x0d - L0p                   | 0x0f - L0                    |
 QPI Tx Lane Status     | 1 1111 1111 1111 1111 1111   | 1 1111 1111 1111 1111 1111   |
 Rx state               | 0x0d - L0p                   | 0x0d - L0p                   |
 QPI Rx Lane Status     | 1 1111 1111 1111 1111 1111   | 1 1111 1111 1111 1111 1111   |
 Periodic Retraining    | Enabled (0x08)               | Enabled (0x08)               |
 Scrambling             | Enabled                      | Enabled                      |
 L0p                    | En (T:0x232a1c) (R:0x2fb105) | En (T:0x6b4d6d) (R:0x1bcb08) |
 L1                     | Enabled (0x000000)           | Enabled (0x000000)           |
 PPS                    | 0x00000000                   | 0x00000000                   |
--------------------------------------------------------------------------------------
 Link Speed             |    9.6 GT/s
======================================================================================
```

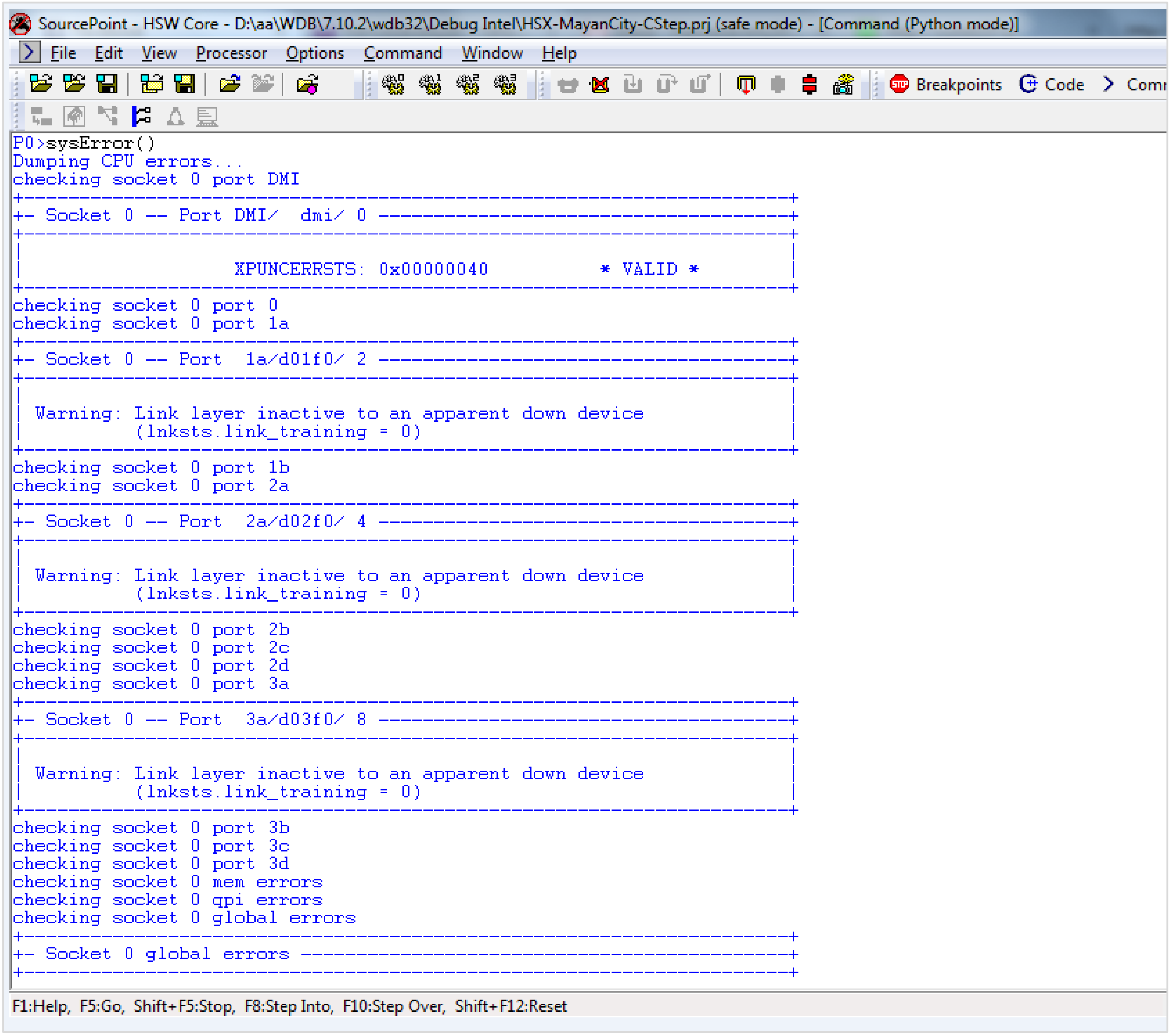
Figure 2: Output from a sysTopo() command

SysStatus provides more information about the dynamic configuration and state of the target. This will include a topological look at what peripherals are present as well as their operational status. This command is likely to indicate any areas that are not working in a target that is mostly working. Part of this display is a detailed display of what is on the PCI and PCIe connections, the status of the memory banks and many other things.

SysError and SysDump are used to find information left from the occurrence of an error. While SysError is usually used for non-fatal errors, SysDump performs a postmortem on a catastrophic error.

Error injections and logging provide a way to test and validate code that is intended to do error handling. A great example of this is injecting an error in a memory access in ECC-capable memory. Services provided by these functions provide both a facility to inject the error as well as logs to determine the occurrence of errors.

While only a few of the script functions have been discussed here, this should give a good overview of what the ICS packages contain and what they are used for. Figure 3 shows the first part of a sysError () execution.


SourcePoint - HSW Core - D:\aa\WDB\7.10.2\wdb32\Debug Intel\HSX-MayanCity-CStep.prj (safe mode) - [Command (Python mode)]
File Edit View Processor Options Command Window Help
Breakpoints Code Comm
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset

```
P0>sysError()
Dumping CPU errors...
checking socket 0 port DMI
+------------------------------------------------------------------------+
+- Socket 0 -- Port DMI/  dmi/ 0 ----------------------------------------+
+------------------------------------------------------------------------+
|                                                                        |
|                   XPUNCERRSTS: 0x00000040         * VALID *            |
+------------------------------------------------------------------------+
checking socket 0 port 0
checking socket 0 port 1a
+------------------------------------------------------------------------+
+- Socket 0 -- Port  1a/d01f0/ 2 ----------------------------------------+
+------------------------------------------------------------------------+
|                                                                        |
| Warning: Link layer inactive to an apparent down device                |
|          (lnksts.link_training = 0)                                    |
+------------------------------------------------------------------------+
checking socket 0 port 1b
checking socket 0 port 2a
+------------------------------------------------------------------------+
+- Socket 0 -- Port  2a/d02f0/ 4 ----------------------------------------+
+------------------------------------------------------------------------+
|                                                                        |
| Warning: Link layer inactive to an apparent down device                |
|          (lnksts.link_training = 0)                                    |
+------------------------------------------------------------------------+
checking socket 0 port 2b
checking socket 0 port 2c
checking socket 0 port 2d
checking socket 0 port 3a
+------------------------------------------------------------------------+
+- Socket 0 -- Port  3a/d03f0/ 8 ----------------------------------------+
+------------------------------------------------------------------------+
|                                                                        |
| Warning: Link layer inactive to an apparent down device                |
|          (lnksts.link_training = 0)                                    |
+------------------------------------------------------------------------+
checking socket 0 port 3b
checking socket 0 port 3c
checking socket 0 port 3d
checking socket 0 mem errors
checking socket 0 qpi errors
checking socket 0 global errors
+------------------------------------------------------------------------+
+- Socket 0 global errors -----------------------------------------------+
+------------------------------------------------------------------------+
```

Figure 3: Output of the SysError function

Future Plans at Intel for its Customer Scripts

These scripts are almost all based on scripts originally written for internal validation and troubleshooting inside Intel. Because of this, they were originally run on the internal versions of the tools that Intel uses. For many years, this was the ITP. Recently, this has been the newer version of the ITP (ITP II).

Many Intel customers (OEMs and ODMs) use tools from ASSET for BIOS debug. Because of this there has been a strong desire to run the ICSs on ASSET's SourcePoint (formerly Arium's SourcePoint). This is working today!

To ensure that this continues to work well, Intel is developing an environment for these scripts to operate on TPV tools. One example of these tools is ASSET InterTech's SourcePoint™.

The new interface spec for Python, called IPC by Intel, should accomplish Intel's goal of the scripts being available earlier in the board bring up process and with more robustness on third party tools.

ASSET InterTech Support for Intel CScripts

ASSET InterTech is taking a two-pronged path to ensure that SourcePoint users have a good experience when running Intel Customer Scripts on SourcePoint, ASSET's world class software debugger.

The first of these two solutions is already available in current SourcePoint products for Intel processors. This solution provides the ability to set the command line mode inside SourcePoint to Python and the support ICSs directly in this environment. This is available now for processors up through Haswell.

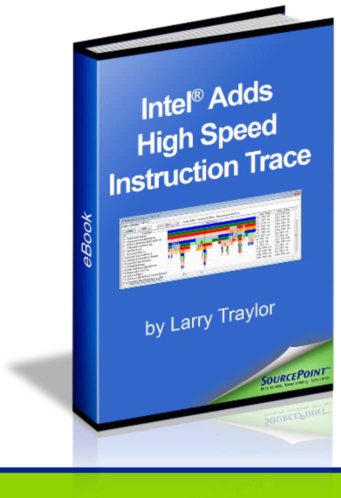
ASSET is also actively working with Intel to support the new IPC interface. With this feature ASSET's customers will have the choice of using the built-in SourcePoint Python shell or running a shell delivered by Intel to support ICSs. Support for the new IPC interface will be available beginning with the next micro-server chip introduction from Intel.

Conclusions

Intel Customer Scripts (CScripts) are very useful for board bring up, error injection and finding the root causes of failures. These scripts (ICSs) are delivered by Intel and tested for execution in several operating environments. One of these environments is ASSET InterTech's SourcePoint. For more information on SourcePoint, click here.

Learn More

You might also be interested in our "Intel Adds High Speed Instruction Trace" eBook. This eBook explains how you can take advantage of Intel's new trace capabilities to accelerate your debug processes.



Register Today!