

Real Insight from Code to Silicon

SourcePoint®  ScanWorks®

Debug Automation using the SourcePoint Command Language

Alan Sguigna
December 15, 2022

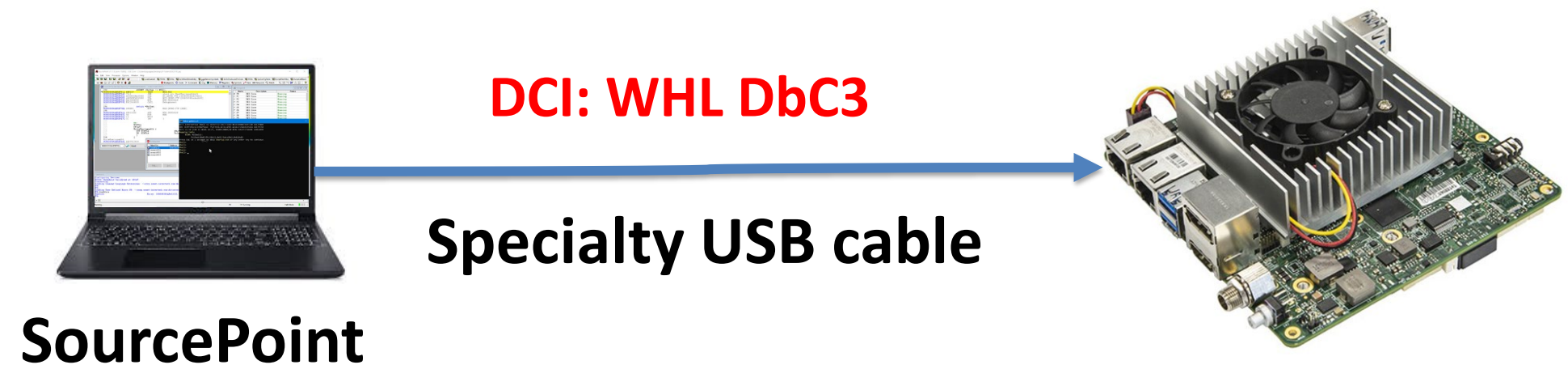
Agenda

- Introduction
- Reference Platform: AAEON Whiskey Lake w/DCI
- The Command Language
- Getting Help
- Macro buttons, User-Defined macros, Event macros
- The SourcePoint Assembler
- Built-in Macros: platform.mac, processor.mac, pch.mac, npk.mac
- Some helpful references
- Wrap-Up and Q&A

SourcePoint

- A very powerful JTAG-based debugger
- Optimized for low-level firmware debug
- Learning curve – **lots** of features

On AAEON Whiskey Lake (WHL) UP Xtreme board:



*WhiskeylakeOpenBoardPkg
DbC3 & FSP – “Choppy Seas”*

The Command Language

- A programming language very similar to 'C'
 - *Conditional logic, branching, loops, typed data, operators, functions, etc. etc.*
- Additional support for run-control, target access, and unique SourcePoint capabilities
 - *go*
 - *halt*
 - *dbgbreak*
 - *step*
 - *fprintf*
 - *strcmp*
 - *asm/endasm*
 - *etc. etc.*
- > 240 commands and control variables, ~ 400 pages of documentation

- ▼ SourcePoint Command Language
 - ▼ Overview
 - Introduction
 - Syntax Notation
 - Comments
 - Constants
 - Data Types
 - Expressions
 - Debug Variables
 - Debug Variable Arrays
 - Debug Procedures
 - Control Variables
 - Command Files
 - Filenames
 - Viewpoint Processor and Processor Overrides
 - Symbolic References
 - Qualified Symbol Names
 - ▼ Commands and Control Variables
 - aadump
 - abort
 - abs
 - acos
 - advanced
 - asin
 - asm
 - asmmode
 - atan
 - atan2

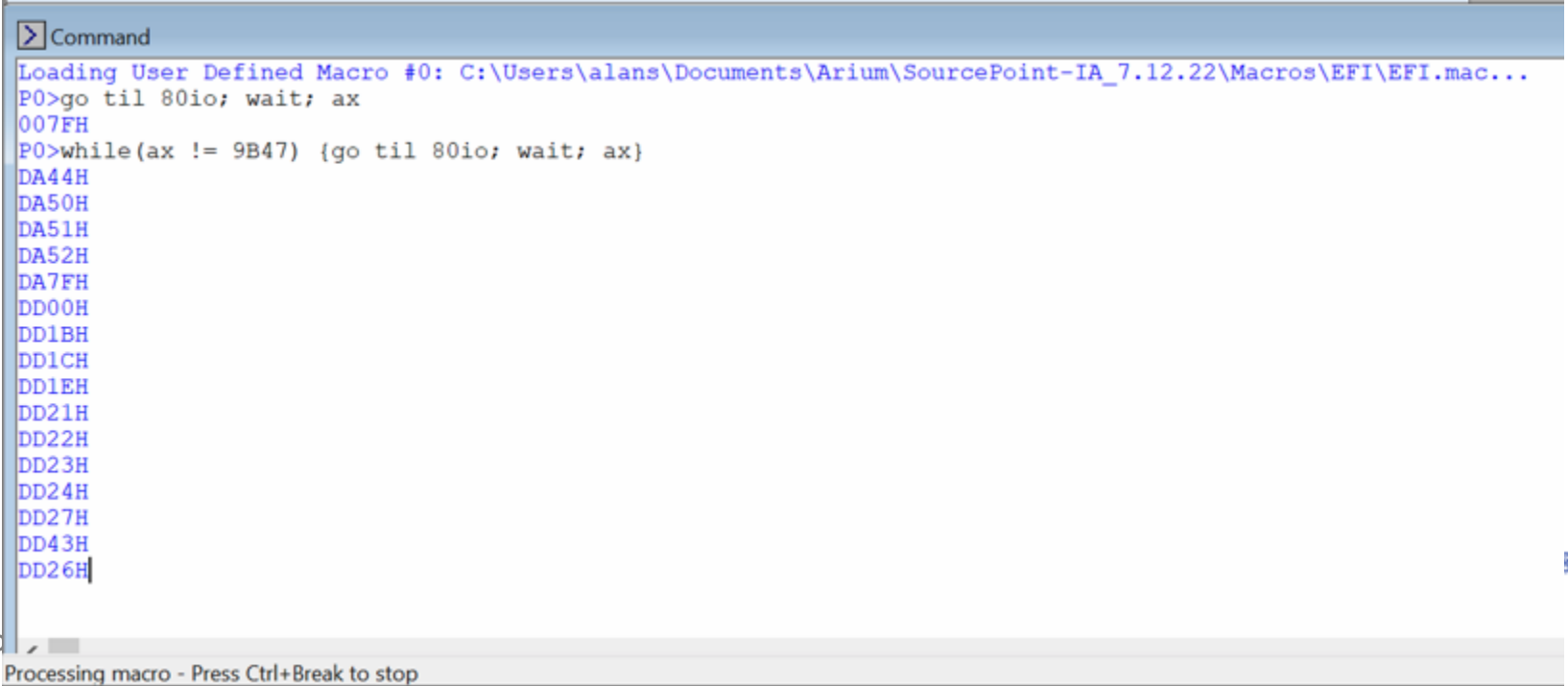
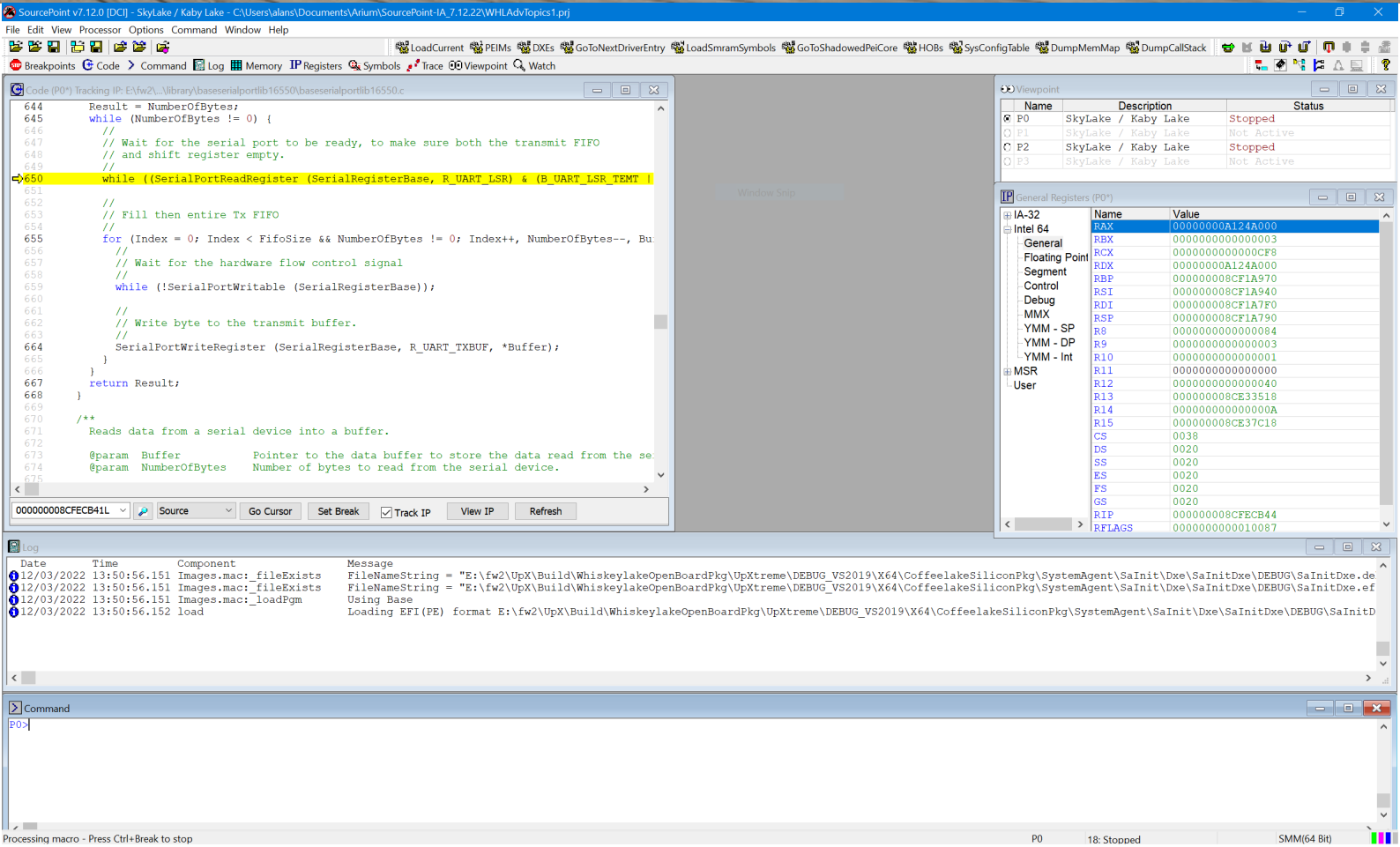
From the command line

Examples:

go til 80io

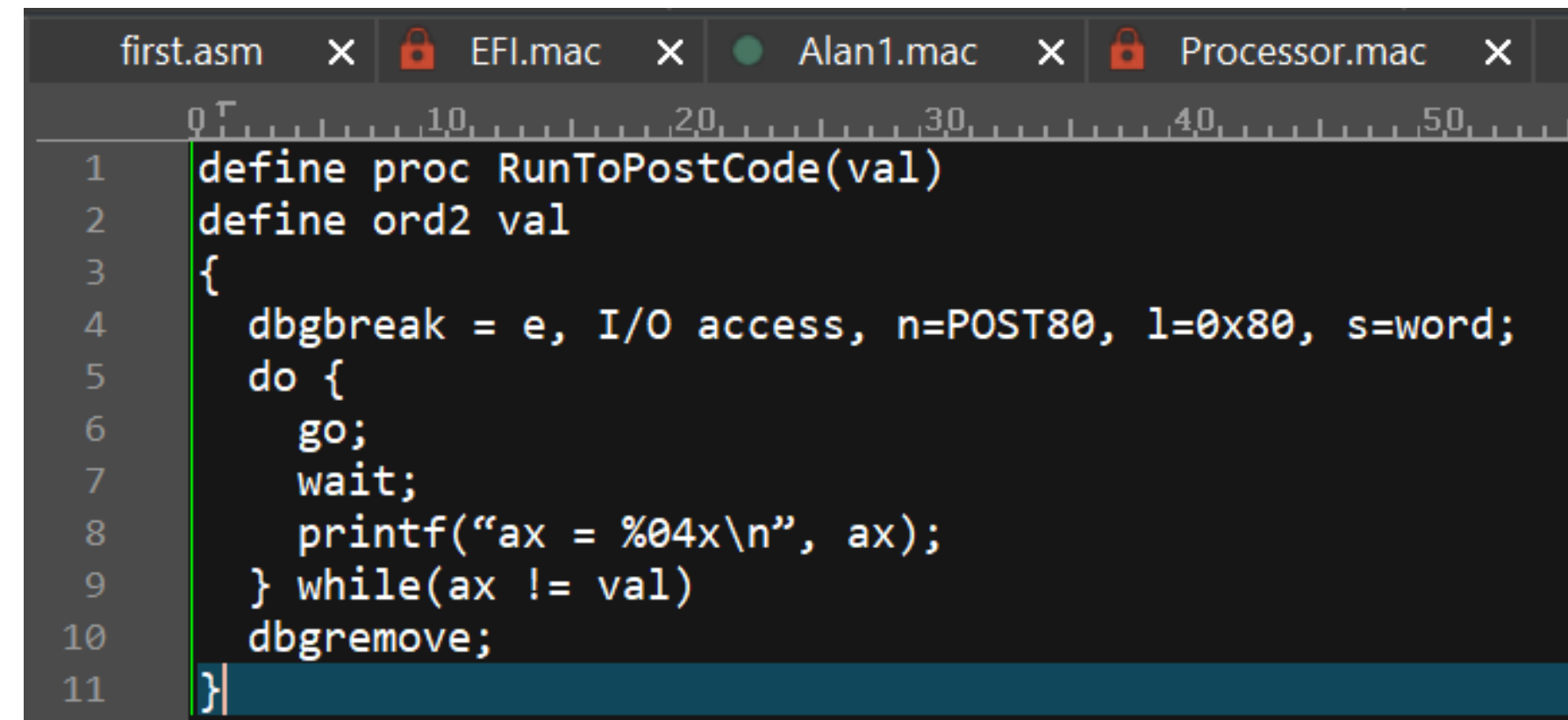
go til 80io; wait; ax

while(ax != 0x9B47) {go til 80io;
wait; ax}



In a macro

- Procedures are defined, and can be executed, within macro files
- Procedures have parameters (post code is “val”) as well as global and local variables
- do/while
- `dbgbreak` enables a breakpoint on an I/O access, named “POST80”, for port 80 and a word (16 bits)
- This routine is equivalent to:
 - `while(ax != val) {go til 80io; wait; ax}`

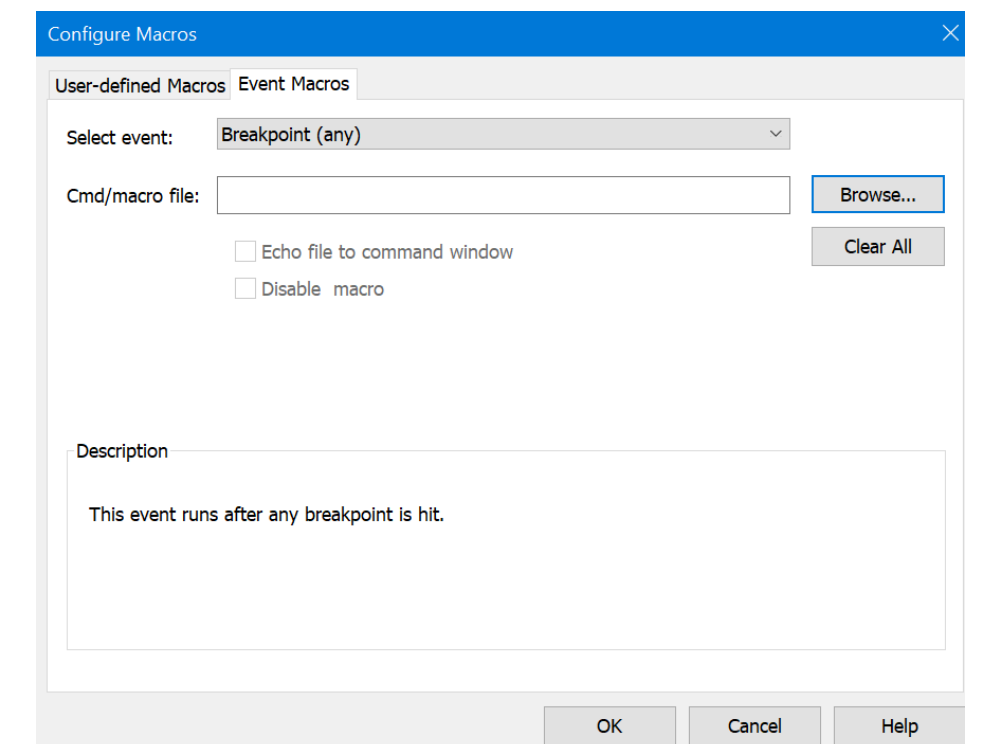
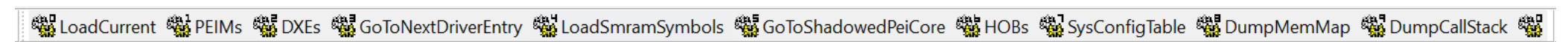
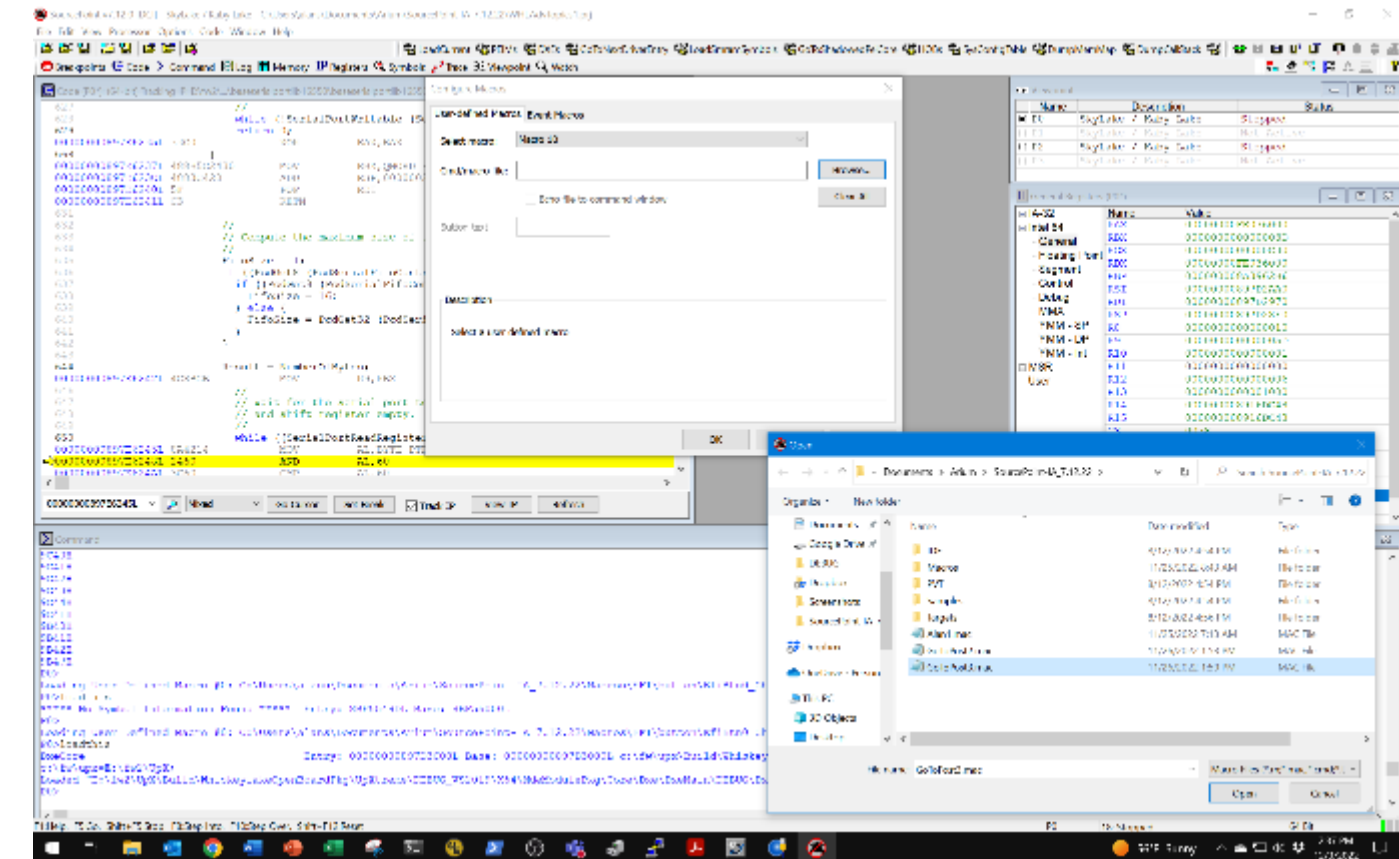


The screenshot shows a code editor with four tabs: first.asm, EFI.mac, Alan1.mac, and Processor.mac. The code in the editor is as follows:

```
1  define proc RunToPostCode(val)
2  define ord2 val
3  {
4      dbgbreak = e, I/O access, n=POST80, l=0x80, s=word;
5      do {
6          go;
7          wait;
8          printf("ax = %04x\n", ax);
9      } while(ax != val)
10     dbgremove;
11 }
```


Macros

- Can be assigned to buttons or events
- Up to 20 buttons of your own creation
- Event macros are very powerful, and can be triggered based upon:
 - Breakpoint (any)
 - Emulator connected
 - Go
 - Project Load/Unload
 - Reset (before, after)
 - Stop
 - Target configure
 - Target power on
 - etc.



The SourcePoint Assembler

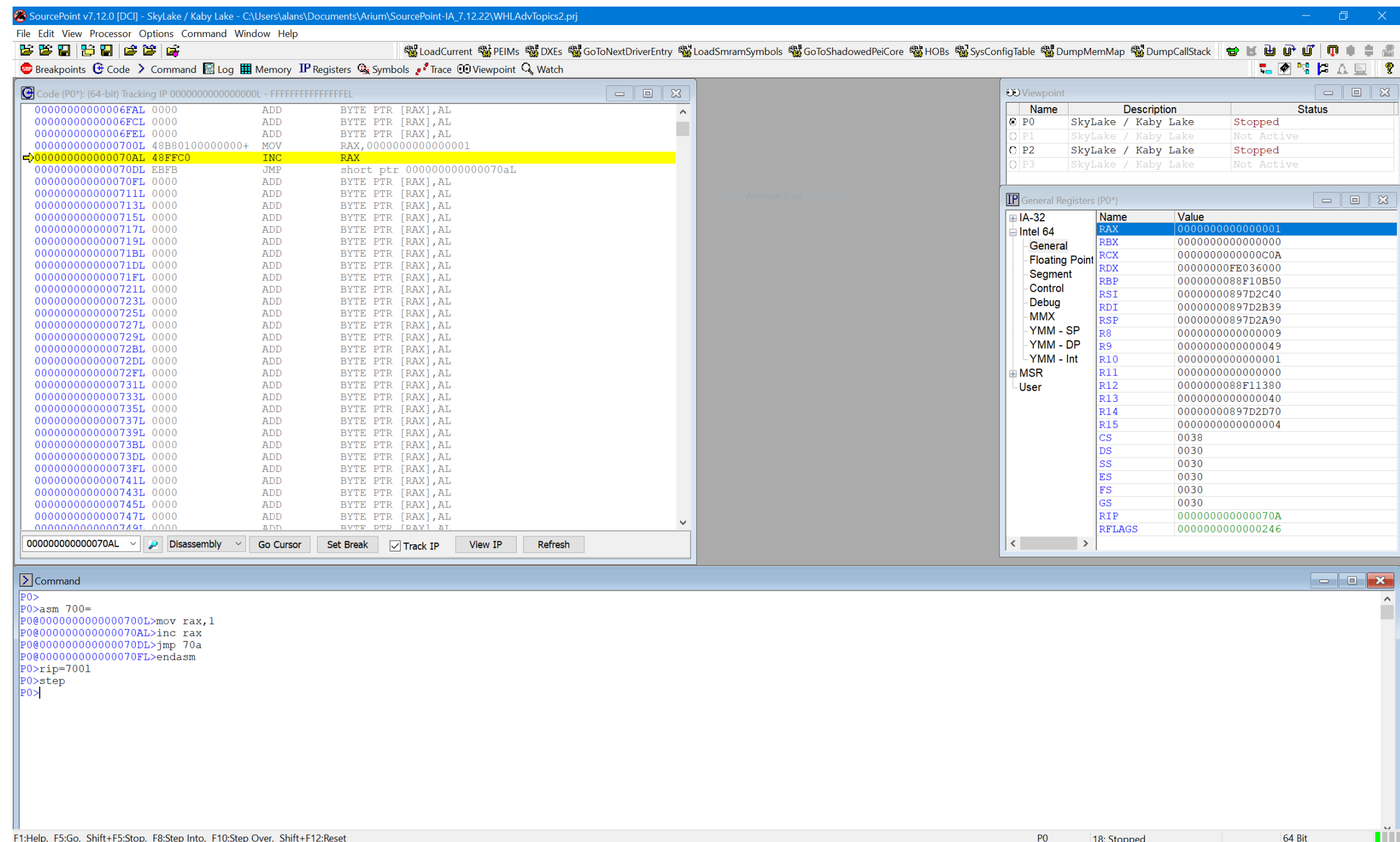
The command language provides access to the SourcePoint assembler

Can use the command line, or within macros

Single-pass assembler; cannot use labels not yet defined (but can use org directive for out-of-order assembly, and use command language for labels)

Example:

- asm 1000 =
- mov al, 01
- mov al, 02
- mov al, 03
- endasm



Useful macros within SourcePoint

- TraceHub.mac
- Platform.mac
- Processor.mac
- Example:
 - `define nstring strpath = defaultpath`
 - `defaultpath = strpath + \\Macros\\Intel`
 - `include platform.mac`
 - `pfmGetDetails()`
 - `show`

```
Command
P0>pfmGetDetails()
Macro Versions: Platform v1.04, Processor v1.03, PCH v1.03
Target Platform is Whiskey Lake
CPU: Whiskey Lake U-Processor Line (Mobile) Dual Core, Stepping V0, Processor ID: 0x806EC
IGD: GT1
PCH: CannonLake-LP Premium U, Stepping D0
P0>
```

F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset

Demo

Good Resources

- **SourcePoint Academy:**
<https://www.asset-intertech.com/resources/academy/sourcepoint-academy/>
“How To” Guides and all our technical documentation online
- **Blogs:**
<https://www.asset-intertech.com/resources/blog/category/arium-probes-sourcepoint/>
Tons of articles on timely debugging and JTAG topics
- **Videos:**
<https://www.asset-intertech.com/resources/videos/>
Our recorded webinars: DCI debug of UEFI and hypervisor technologies on the AAEON UP Whiskey Lake and Tiger Lake boards, JTAG-based debugging of AMD EPYC servers, etc.
- **Webinars:**
<https://www.youtube.com/c/UEFIForum/videos>
Beyond Printf – Real-Time Firmware Debugging
UEFI Debug with Intel Architectural Event Trace
JTAG-based Debug & Trace

Wrap-Up

Questions?

Reach me at alan.sguigna@asset-intertech.com,
DM @AlanSguigna

Real Insight from Code to Silicon

The logo for ASSET features the word "ASSET" in a bold, blue, italicized sans-serif font. A bright green swoosh underline starts under the 'A' and extends to the right, ending in three small green rectangular dashes. A small "TM" trademark symbol is positioned to the upper right of the 'T'.

ASSETTM