

# Interrupt Analysis with Trace

```
{{#switchcategory:MSP430=<McuHitboxHeader/>|C2000=<McuHitboxHeader/>|Stellaris=<McuHitboxHeader/>|TMS570=<McuHitboxHeader/>|MCU=<McuHitboxHeader/>|MAVRK=<MAVRKHitboxHeader/>|<HitboxHeader/>}}
```

## Contents

### Interrupt Analysis

- Overview
- Methodology
- Capturing Interrupt Analysis Data
- Decoder Prerequisites
- Command Line
- Sample Results Data
- Trace Script Download

### Trace Script Frequently Asked Questions

- Generic Scripting FAQs
- Interrupt Analysis Specific FAQs

## Interrupt Analysis

### Overview

Interrupt Analysis is a method of using XDS560 Trace to monitor the servicing of interrupts in a real-time application. In most cases, there are hard deadlines that need to be met. Interrupt analysis gives the user a picture of the servicing of the interrupts in their application. They can view the elapsed time between consecutive ISRs and determine the worst case scenarios. The Interrupt Analysis script generates a text output that can then be displayed in a graphical package as shown at right.

### Methodology

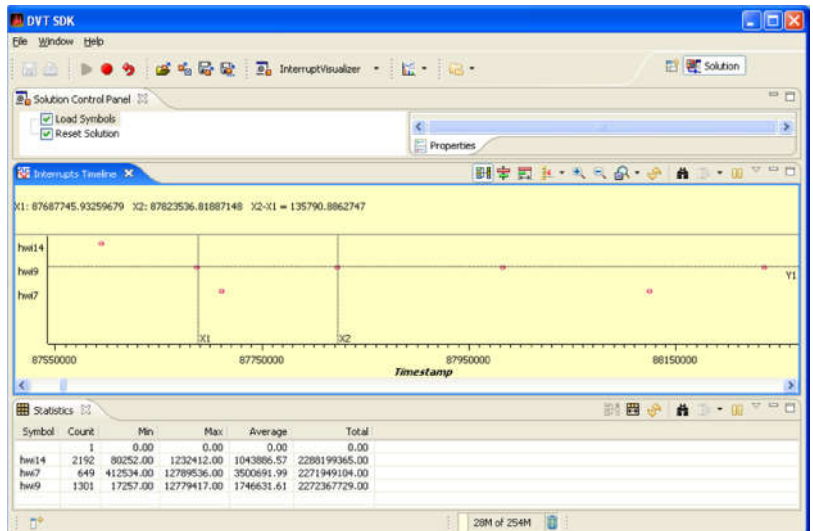
Interrupt Analysis is implemented by capturing only the execution of the interrupt vectors. This limits the amount of data captured by trace and focuses on when the interrupts are serviced. Essentially, we are capturing PC and Timing data whenever the Program Address is within a specified range (the location of the Interrupt Vector Table).

### Capturing Interrupt Analysis Data

Configure trace in the Trace Control menu as desired. Typically, "Stop on Buffer Full" mode is used with Interrupt Analysis, but it can also be used with "Circular Buffer" mode. If using an XDS560T, configure the desired trace buffer size. A larger buffer will capture much more data, but the data will take a longer time to process. A smaller buffer won't get nearly as much of the application, but will be post processed very quickly. The compression of data when capturing via this method will not yield nearly as much data as in some other cases, so the penalty for using a very large trace buffer will be much smaller.

The Unified Breakpoint Manager (UBM) Plugin in Code Composer Studio can be easily configured to capture Pipeline Stall Analysis data

- From the Breakpoint window, create a new Trace job.
- Edit the properties of the job, and select Trace Type->Standard, Actions->User Script, and Script Type->Event Analysis as shown in the image.
- Expand the Script Type Option
- Specify the Start of the interrupt vector table. This can be wither a symbolic or numerical value. Check the map file in order to determine the correct location.
- Click OK to save the configuration and ensure that the job is enabled in the breakpoint window.
- Run the application. You should see trace data being captured in the Trace Display Window.



### Decoder Prerequisites

In order for the Interrupt Analysis application to be able to process the data, the following fields must be in the data passed to the script

- Program Address
- Cycles
- Trace Status

### Command Line

One of the following commands can be used to process the trace data. Note that full or relative paths to each file supplied must be provided if all files are not in the current directory.

```
td.exe -bin XDS560_RecTraceData.bin -app <out file> <trace_decoder_options> | perl trace_interrupts.pl -n
```

or

```
td.exe -bin XDS560_RecTraceData.bin -app <out file> <trace_decoder_options>| trace_interrupts.exe -n
```

## Sample Results Data

The results data is output through stdout in comma separated value form. It's essentially a list of Interrupt Service Routine start addresses and the cycle at which each began to execute. The application below contains only a single interrupt A header line is prepended to the output, detailing the contents of each field. It can be suppressed by using the `--no_header` option with the script.

```
TimeStamp,IsrAddr
-----
1068130,0x00806DC0
2068148,0x00806DC0
3068126,0x00806DC0
4068123,0x00806DC0
5068127,0x00806DC0
6068127,0x00806DC0
7068129,0x00806DC0
8068131,0x00806DC0
9068150,0x00806DC0
10068135,0x00806DC0
```

This .csv file can then be consumed by another script or by a graphical package. It consists simply of a list of timestamps, along with the address of the ISR that started to be serviced at that time.

## Trace Script Download

The script package can be downloaded at the following location [https://www.a-ti.com/downloads/sds\\_support/applications\\_packages/trace\\_csv\\_scripts/index.htm](https://www.a-ti.com/downloads/sds_support/applications_packages/trace_csv_scripts/index.htm)

Note that in order to use the stand alone trace decoder, you must have version 3.0.0 of the scripting package or later.


# Trace Script Frequently Asked Questions

## Generic Scripting FAQs

- Q: Why do I sometimes see "UNKNOWN", "UNKNOWN" in the output for functions/filenames
  - A: The function/filename symbols are determined from the output of the ofd6x.exe (Object File Dump utility), which generates a list of functions and filenames from the .out file, along with their starting and ending addresses. If "UNKNOWN" values are showing in the output, it's because trace captured program execution that had a program address outside the ranges specified by the OFD utility. Common causes might be code that has been dynamically loaded/allocated which wouldn't have associated information in the .out file. You can usually determine the exact cause by looking in the file generated by ofd6x.exe and the .map file and determining why the offending program address is not defined.

## Interrupt Analysis Specific FAQs

Keystone=		C2000=For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article Interrupt Analysis with Trace here.		MSP430=For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article Interrupt Analysis with Trace here.		OMAPL1=For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article Interrupt Analysis with Trace here.		MAVRK=For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article Interrupt Analysis with Trace here.	
{		DaVinci=For technical support on DaVinciplease post your questions on The DaVinci Forum. Please post only comments about the article Interrupt Analysis with Trace here.		OMAP35x=For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article Interrupt Analysis with Trace here.		OMAPL1=For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article Interrupt Analysis with Trace here.		MAVRK=For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article Interrupt Analysis with Trace here.	
1. switchcategory:MultiCore=		For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum		For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum		For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum		For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum	
For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum		For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum		For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum		For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum		For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum	
Please post only comments related to the article Interrupt Analysis with Trace here.		Please post only comments related to the article Interrupt Analysis with Trace here.		Please post only comments related to the article Interrupt Analysis with Trace here.		Please post only comments related to the article Interrupt Analysis with Trace here.		Please post only comments related to the article Interrupt Analysis with Trace here.	
}		}		}		}		}	



Amplifiers & Linear Audio

DLP & MEMS High-Reliability Interface

Processors

- ARM Processors
- Digital Signal Processors (DSP)

Switches & Multiplexers Temperature Sensors & Control ICs Wireless Connectivity

<a href="#">Clocks &amp; Timers</a>	<a href="#">Logic</a>	▪ <a href="#">Microcontrollers (MCU)</a>
<a href="#">Data Converters</a>	<a href="#">Power Management</a>	▪ <a href="#">OMAP Applications Processors</a>

{{#switchcategory:MSP430=<McuhitboxFooter/>|C2000=<McuhitboxFooter/>|Stellaris=<McuhitboxFooter/>|TMS570=<McuhitboxFooter/>|MCU=<McuhitboxFooter/>|MAVRK=<MAVRKHitboxFooter/>|<HitboxFooter/>}}

Retrieved from "[https://processors.wiki.ti.com/index.php?title=Interrupt\\_Analysis\\_with\\_Trace&oldid=29391](https://processors.wiki.ti.com/index.php?title=Interrupt_Analysis_with_Trace&oldid=29391)"

**This page was last edited on 6 May 2010, at 16:47.**  
Content is available under [Creative Commons Attribution-ShareAlike](#) unless otherwise noted.