CCS Debug and Trace

Debug Tools Value Chain

Board Bring-Up & Drivers Development → **Kernel & Application Development** → **Application & System Optimization** → **Deployment**

Board Bring-Up & Drivers Development	Kernel & Application Development	Application & System Optimization	Deployment
JTAG Debug for Cortex, DSP, & Accelerators (IVA)	Kernel & MMU Aware Debug	Trace Based Function Profiling & Code Coverage	Embedded Debug & Analysis APIs
Boot Mode Debug Support	Symmetrical Multi Processing (SMP) Debug	Cache & Cycles Profiling	Standalone Trace Import & Analysis
Debug Across Core & System Resets	GDB based OS App Debug Linux & TI RTOS Aware	Time-Correlated Multicore Instrumentation	
Debug Across Power Transitions	Cross Triggering for Debug & Real-Time Trace	Non-Intrusive Throughput, Bus Bandwidth, & Traffic Profiling	
Global Execution Control	Program Execution & Data Trace	Non-Intrusive Power & Clock Management Optimization	

Quality

Low Cost

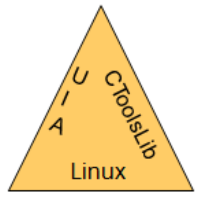
Time to Market

Debug Probes & Trace Receivers

Debuggers & Development Environments

IAR SYSTEMS
LAUTERBACH DEVELOPMENT TOOLS
Aarium
GDB The GNU Project Debugger
Green Hills SOFTWARE
ARM
CODESOURCERY now part of Mentor Embedded
Mentor Graphics
ENEA software

Embedded & Hosted APIs



Tools Capabilities

Stop Mode Debug

•JTAG (1149.1) support
•C66x, Cortex A15, Cortex M4, EVE(ARP32), IVA(ARM9), and PRU debug
•C66x, CortexA15, Cortex M4, ARM9, ARP32 supports HW breakpoints, Watchpoints, and performance counters
•Debug code from reset vector

•CPU and system reset debug
•Symmetrical multi processing (SMP) debug
•Global run and halt across processors
•Debug across low power transitions
•Linux/Android kernel aware debug
• Hypervisor (VM and Guest OS) Debug (2Q 2014)

Run Mode Debug

•GDB based
•Linux and Android support
•Process level execution control
•Process level breakpoints

•Linux eclipse plug-in add-on to CCS for simultaneous kernel and app debug

Processor Trace and Profiling

•DSP Trace – PC, cycles, data, & events (PG 1.0 not working)
•Cortex A15 – PC and cycles
•Embedded trace buffer (ETB)
•Off-chip trace via Pro Trace (Aug 2013)

•DSP cache and pipeline stalls profiling (PG 1.0 not working))
•PC based function level profiling
•CToolsLib embedded trace instrumentation
•Trace based code coverage (CCS v5.5 - 4Q 2013)
•CToolsProf: GNU style CLI for trace and profiling (1H 2014)
•Trace over USB3 (2Q 2014)

SoC Instrumentation and Analysis

•Chip level system SW instrumentation view
•System Trace Linux drivers for HW assisted printf
•Embedded Trace Buffer (ETB) & off-chip collection
•Cache performance measurement for IPU, and EVE (CTSET)
•Bus bandwidth and latency profiling (Stats Collect.)

•Bus interconnect traffic profiling (OCPWP)
•Power and clock management profiling
•IVAHD accelerators execution profiling
•CToolsLib for embedded debug and trace
•CToolsProf: GNU style CLI for trace and profiling (1H 2014)

JTAG Probes and Trace Receivers

XDS100v2/v3

XDS200

XDS510

XDS560v2 STM

XDS560v2 Pro Trace

XDS JTAG and Trace Receivers

<u>XDS100 v2</u>
- Entry level JTAG for hobbyist and universities
- CCS Cortex A download speed ~30 KB/sec
- USB 2.0
- TI 14 and CTI 20 native
- Open HW reference design

<u>XDS200</u>
- Performance JTAG at low cost for serious users
- CCS Cortex A download speed ~300 KB/sec
- ARM SWD and SWO support
- USB 2.0 and optional ENET
- Bi-direction GPIOs for instrumentation
- TI, MIPI, and ARM connector option
- 3P EPK licensed

<u>XDS510</u>
- Performance JTAG for serious users
- CCS Cortex A download speed ~250 KB/sec
- USB 2.0
- TI 14 and CTI 20 native
- 3P EPK licensed

<u>XDS560 v2 STM</u>
- High performance JTAG & cJTAG for professional users
- CCS Cortex A download speed ~600 KB/sec
- Low bandwidth trace receiver (STM & Cortex M)
- USB 2.0 and ENET
- 4 pin @ 100 MHz with auto skew & jitter calibration
- 128 MB storage
- MIPI 60 native with add-on adapters
- 3P EPK licensed

<u>XDS Pro Trace Receiver</u>
- High performance JTAG & cJTAG for professional users
- CCS Cortex A download speed ~600 KB/sec
- High bandwidth dual-channel trace receiver (DSP, Cortex, & STM)
- USB 2.0 and ENET
- 32 pin @ 250 MHz DDR with auto skew & jitter calibration
- 2 GB trace storage buffer
- MIPI 60 native with add-on adapters
- 3P EPK licensed

Embedded Debug and Analysis APIs

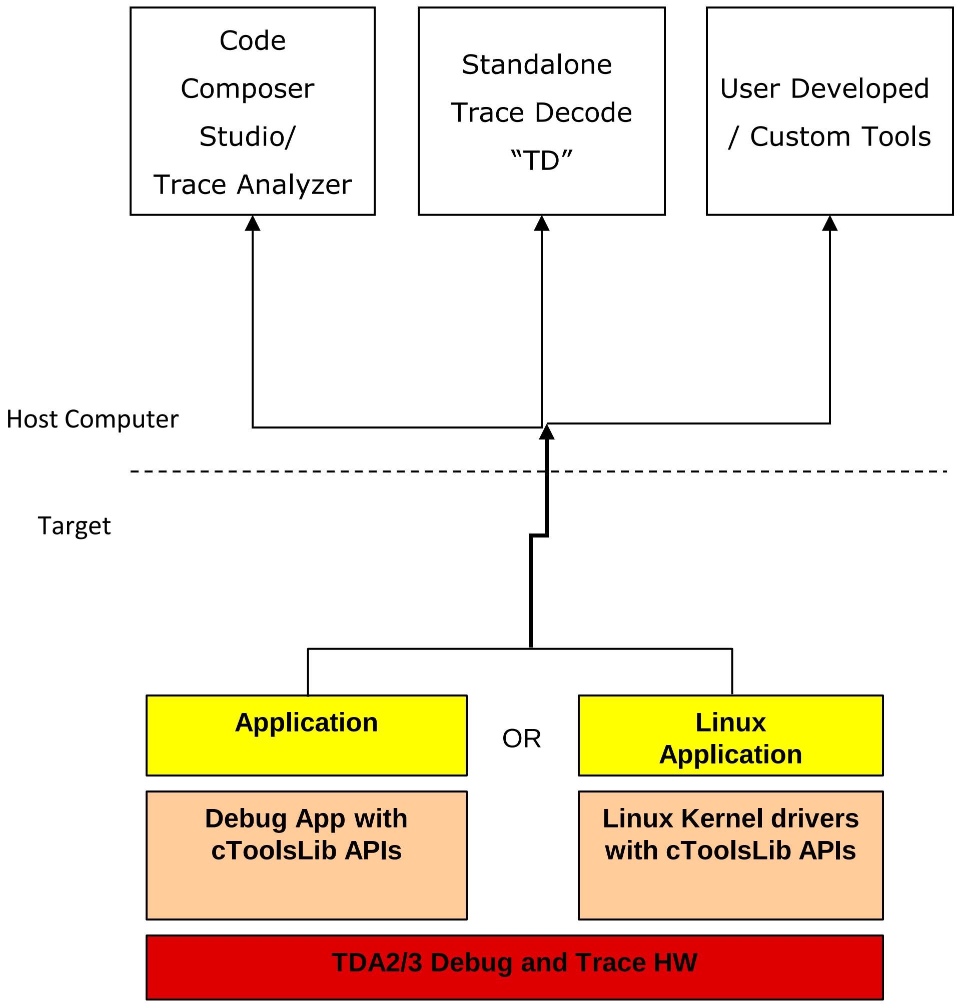
Field Deployed Debug and Trace

http://processors.wiki.ti.com/index.php/CToolsLib

- CtoolsLib – Enabling embedded debug, trace setup, and analysis usage
- Easy access to debug capabilities via simple C APIs
- Very low latency and small footprint (order of few KBs)
- Easy OS integration
- Easy import and data visualization via CCS

API	Jacinto6
ETBLib - Programmatic access to program and read trace data from an Embedded Trace Buffer	√
ETMLib - Programmatic access to setup Cortex A15 trace export	√
STMLib – Printf like instrumentation APIs with HW acceleration	√
STM Linux driver – Enables C printf output redirection to STM port (HW acceleration with SoC Timestamping)	√
SCLib - Programmatic access to setup Statistics Collectors for SoC visibility	√
PMCMLib - Programmatic access to setup power and clock management profiling visibility	√

Embedded Debug and Analysis APIs – Usage Flow


Code Composer Studio/ Trace Analyzer
Standalone Trace Decode "TD"
User Developed / Custom Tools
Host Computer
Target
Application
OR
Linux Application
Debug App with cToolsLib APIs
Linux Kernel drivers with cToolsLib APIs
TDA2/3 Debug and Trace HW

Data Import/Analysis:

Code Composer Studio

Standalone Trace decoder (TD)

User developed custom tools

Data Transport:

Ethernet, JTAG, or user defined

Triggering/Collection:

CToolsLib APIs

Texas Instruments

Technology Overview

TDAx Debug and Trace – Key Goals

Efficiency

- Reuse ARM IP with TI differentiation additions

Consistency

- Across all TDAx family devices

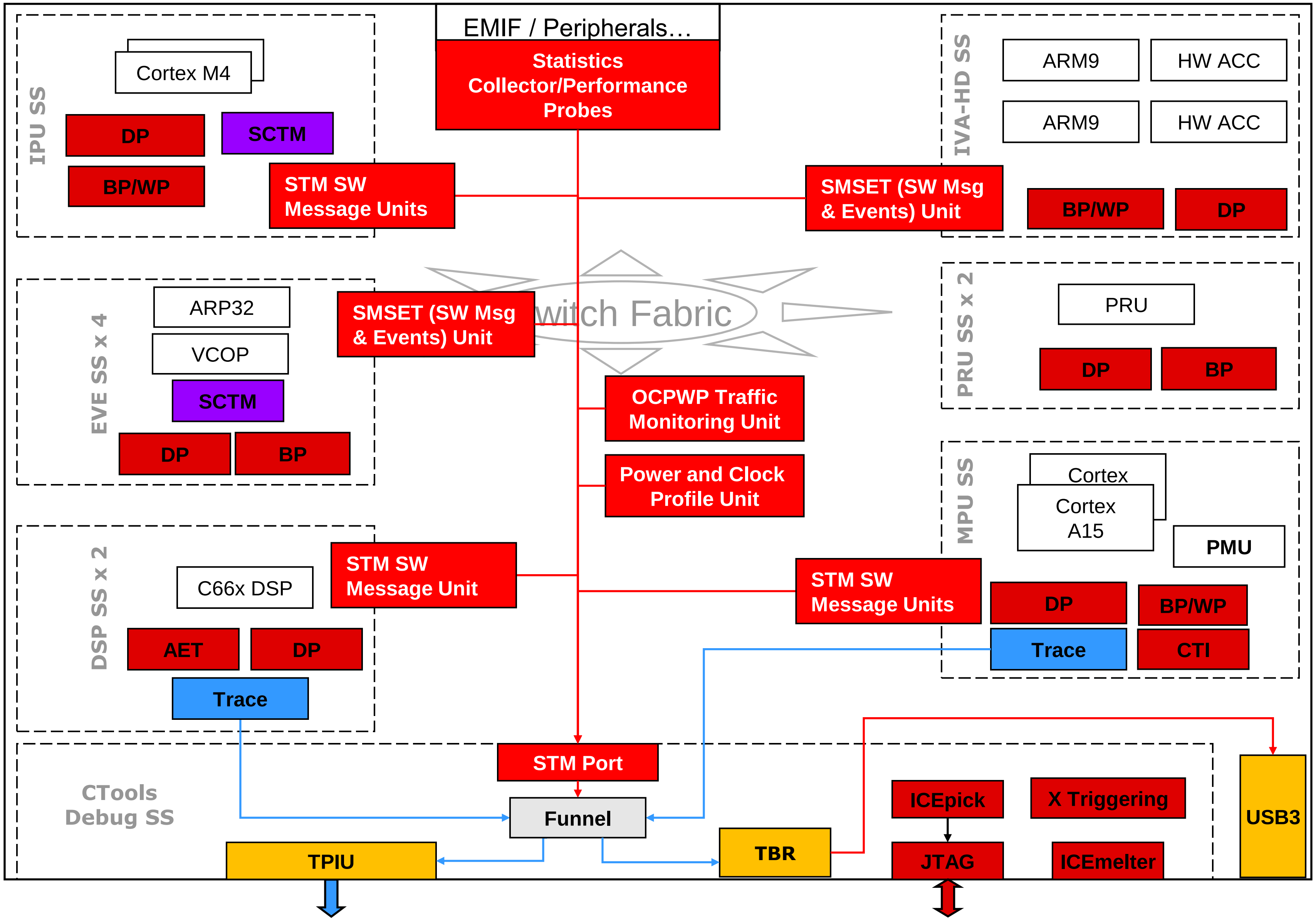
Product Life Cycle

- Development and deployment

Eco System Enablement

- 3P framework and low cost tools

TDA2x Debug and Trace View



Texas Instruments

TDA2x Debug and Trace – Capabilities (i)

Debug

- **Debug Interface**
  - JTAG 1149.1 : 5 pin JATG debug interface for stop mode debug
  - cJTAG 1149.7: 2 pin compact JATG (cJTAG) debug interface for stop mode debug.
  - IcePick: Dynamic scan chain, power, clock, and reset management
  - ICEMelter: For controlling power and wakeup of DebugSS
- **Debug Ports (DP), Breakpoints, and Counters**
  - Cortex-A15: ARM CoreSight on-chip debug breakpoint, watchpoint, and performance measurement units (PMU), Debug access port (DAP)
  - C66x DSP: TI ICEMaker on-chip debug, Advanced event triggering (AET) for SW/HW breakpoints, watchpoints, and profile counters
  - IVAHD (ARM9): ARM debug with TI ICECrusher extensions for HW breakpoints and watchpoints
  - EVE: TI debug IP with SW/HW breakpoints on ARP32; macro level stepping for VCOP
  - PRU: TI debug IP with SW breakpoints
- **Cross Triggering**
  - MPU SS: Arm CoreSight Cross triggering
  - Debug SS level : TI X triggering to propagate debug (trigger) events from one processor subsystem/module to another.
- **Sub-system Counter and Timers (SCTM)**
  - Cache performance measurement for the IPU, and EVE sub-system

TDA2x Debug and Trace – Capabilities (ii)

Trace

- **Real-time Processor Trace**
  - Cortex-A15 processor trace consisting PC and timing
  - C66x DSP processor trace with PC, timing, data, and events
- **Real-time System Trace (STM)**
  - SW Messages - Hardware accelerated multi-core software instrumentation
  - Statistics collectors– Non-intrusive SoC bus bandwidth and latency profiling events
  - IVAHD - Software Message & System Events (SMSET)
  - Power and Clock Management (PM/CM) profiling
  - OCP Watchpoint (OCP_WP) – bus traffic monitoring
  - Chip level timing correlation
- **Embedded Trace Buffer Router (TBR)**
  - On-chip trace capture buffer for real-time trace
  - Routing trace to USB3 interface
- **Trace Port Export Unit (TPIU)**
  - Off-chip Cortex, DSP, and STM trace export
  - Requires external trace receiver (like XDS560v2 STM or XDS560v2 Pro Trace)

CCS Debug Setup

Setup and Installation Overview

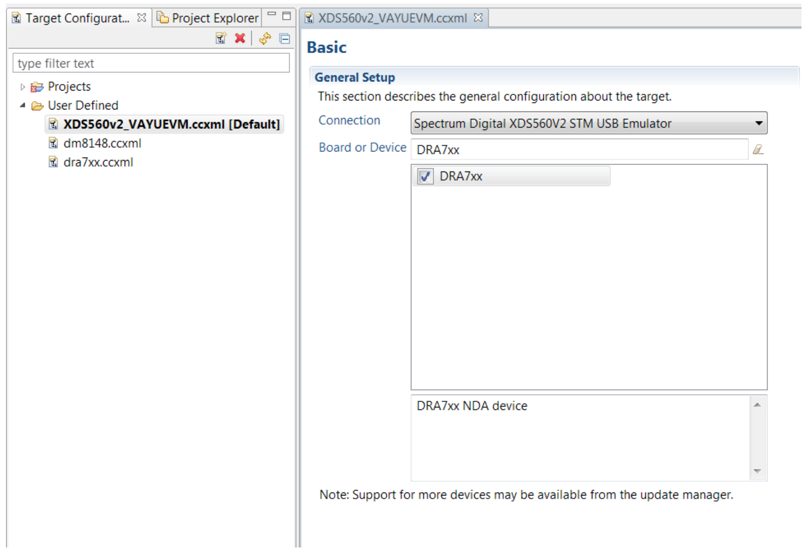
- **Hardware Setup**
  - XDS560v2 STM Trace
  - TDA2x/3x EVM

- **Software Installation**
  - Code Composer Studio
    - CCS v5.5

      http://processors.wiki.ti.com/index.php/Download_CCS
  - Latest TI Emulator Update via Update Manager in CCS
  - TDA2x/3x Chip Support Package (CSP)

TDA2x EVM Debug Setup

- In CCS, setup TDA2x target configuration with Spectrum Digital XDS560v2 STM USB Emulator connection
  - Go to File → New → Target Configuration File
  - Type file name as XDS560v2_VAYUEVM and click Finish
  - Now select Connection as "Spectrum Digital XDS560V2 STM USB Emulator"
  - Type ADAS-S28 in the Device field; device names will be filtered; select/check ADAS-S28
  - Click on Target Configuration from Advanced Setup (RHS)
  - Check Bypass for the unused CPUs to save time when launching.
  - Select View->Target Configurations to see a list of all configuration files.
  - Select the one you just created (as XDS560v2_VAYUEVM.ccxml) under User Defined.
  - Launch the debug session by selecting the Launch Selected Configuration in the context menu.
- Connect Cortex A15 core



Basic

General Setup

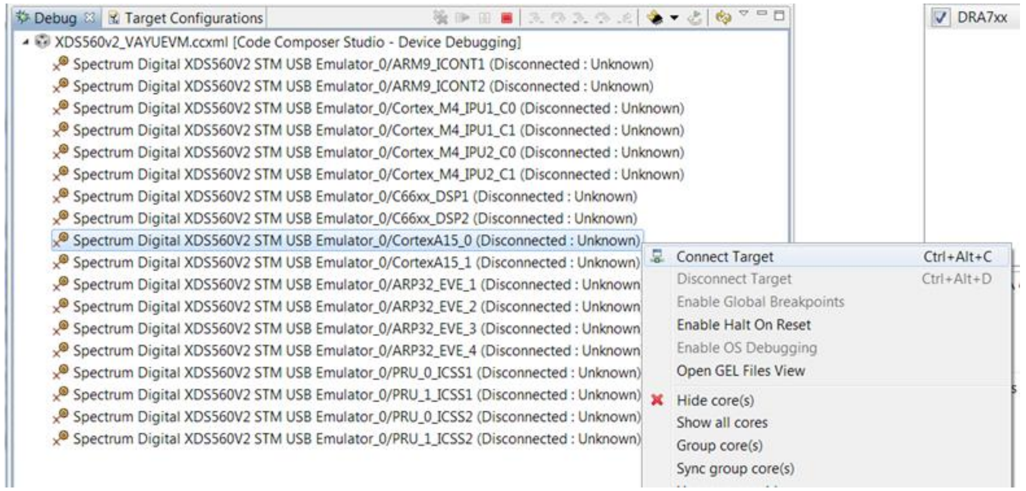
This section describes the general configuration about the target.

Connection	Spectrum Digital XDS560V2 STM USB Emulator
Board or Device	DRA7xx

[✓] DRA7xx

DRA7xx NDA device

Note: Support for more devices may be available from the update manager.



```
XDS560v2_VAYUEVM.ccxml [Code Composer Studio - Device Debugging]
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT2 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C0 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU2_C0 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU2_C1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/C66xx_DSP1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/C66xx_DSP2 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/CortexA15_0 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/CortexA15_1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARP32_EVE_1 (Disconnected : Unknown
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARP32_EVE_2 (Disconnected : Unknown
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARP32_EVE_3 (Disconnected : Unknown
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARP32_EVE_4 (Disconnected : Unknown
  Spectrum Digital XDS560V2 STM USB Emulator_0/PRU_0_ICSS1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/PRU_1_ICSS1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/PRU_0_ICSS2 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/PRU_1_ICSS2 (Disconnected : Unknown)
```

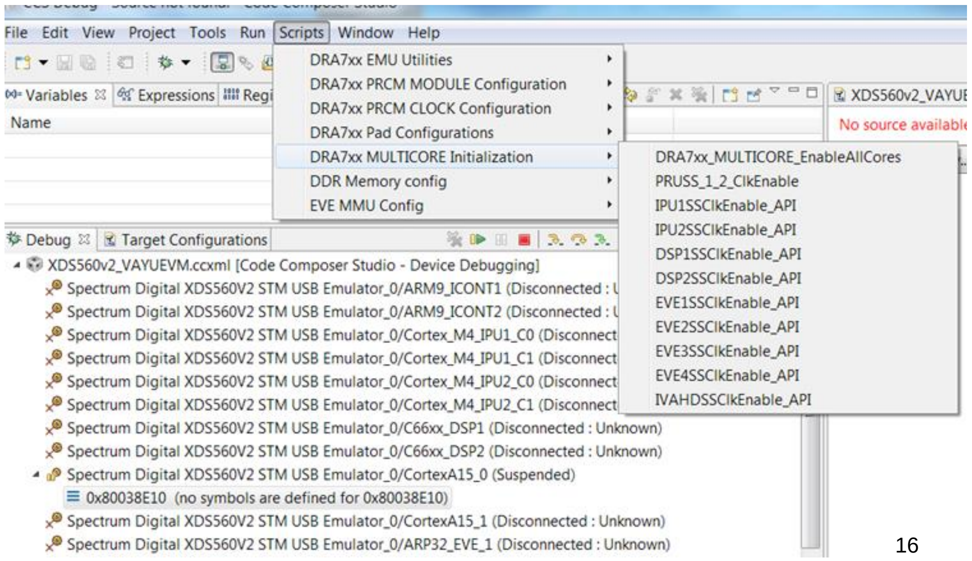
General Debug Features

GEL Files

- GEL (General Extension Language):

  The General Extension Language (GEL) is an interpretive language similar to C that lets you create functions to extend CCS IDE's usefulness. You create your GEL functions using the GEL grammar and then load them into the IDE. With GEL, you can access actual/simulated target memory locations and add options to the IDE's GEL menu. You can also add GEL functions to the Watch window so they execute at every breakpoint. Details can be found in CCS Help.

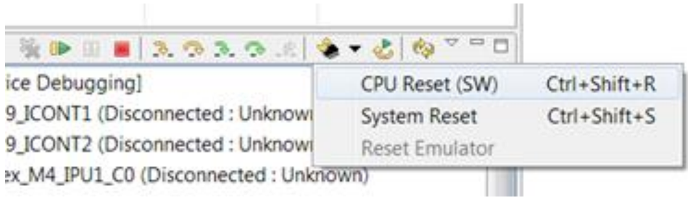
- GEL files included from Vayu CSP (Download via CDDS)
  - GEL file default location: C:\ti\ccsv5\ccs_base\emulation\gel\DRA7xx
  - DRA7xx_prcm_config.gel
    - PRCM functions: DPLLs, Power & Clocks Initialization.
  - DRA7xx_ddr_config.gel
    - Configure EMIFs for DDR3 at 532Mhz or 400 Mhz.
  - DRA7xx_pad_config.gel
    - Initialize Padconf registers based on Vayu EVM.
  - DRA7xx_multicore_reset.gel
    - Provide options to enable/reset Multi-core.



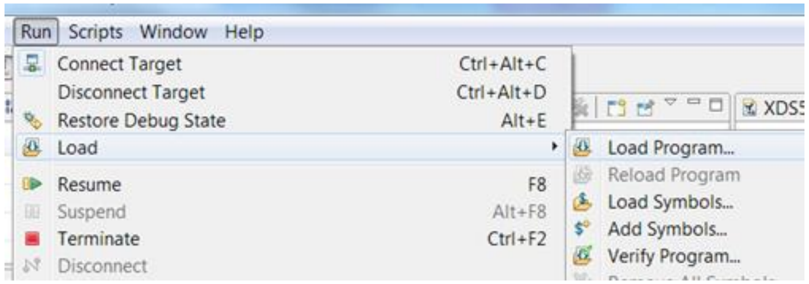
```
XDS560v2_VAYUEVM.ccxml [Code Composer Studio - Device Debugging]
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT1 (Disconnected : U
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT2 (Disconnected : U
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C0 (Disconnect
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C1 (Disconnect
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU2_C0 (Disconnect
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU2_C1 (Disconnect
  Spectrum Digital XDS560V2 STM USB Emulator_0/C66xx_DSP1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/C66xx_DSP2 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/CortexA15_0 (Suspended)
    0x80038E10  (no symbols are defined for 0x80038E10)
  Spectrum Digital XDS560V2 STM USB Emulator_0/CortexA15_1 (Disconnected : Unknown)
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARP32_EVE_1 (Disconnected : Unknown)
```

Connection and Register View

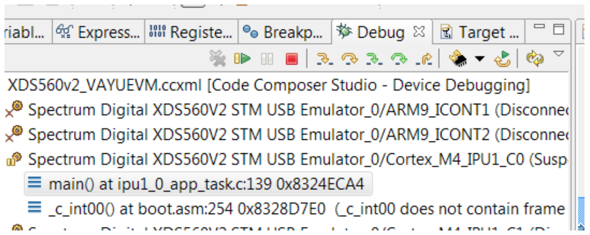
- Download program or symbol only to target cores
  - Click “CPU Reset (SW)” before loading program



ice Debugging]
9_ICONT1 (Disconnected : Unknown
9_ICONT2 (Disconnected : Unknown
ex_M4_IPU1_C0 (Disconnected : Unknown)
CPU Reset (SW) Ctrl+Shift+R
System Reset Ctrl+Shift+S
Reset Emulator



- Debug execution controls (run, halt step, reset etc) are available from the debug view



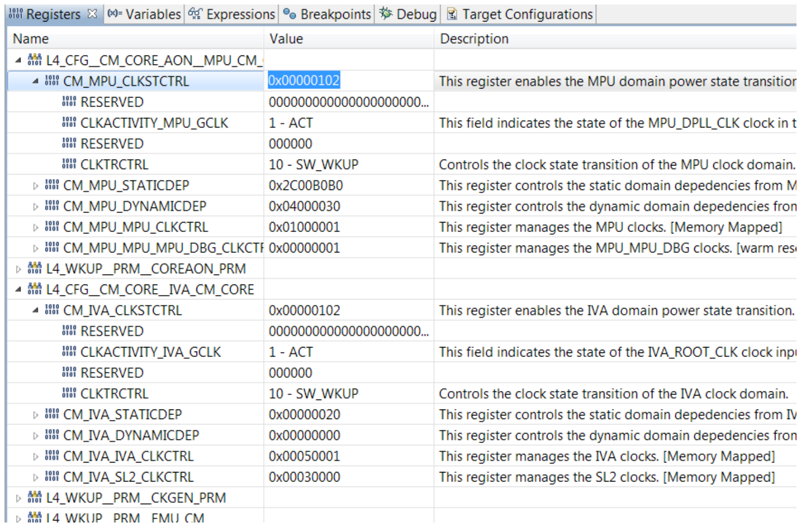
```
XDS560v2_VAYUEVM.ccxml [Code Composer Studio - Device Debugging]
Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT1 (Disconne
Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT2 (Disconne
Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C0 (Susp
    main() at ipu1_0_app_task.c:139 0x8324ECA4
    _c_int00() at boot.asm:254 0x8328D7E0  (_c_int00 does not contain frame
```

- Register view (View → Registers) includes

  all the core registers including CP15.

  A rich set of peripheral register is also available

  when connecting to A15.



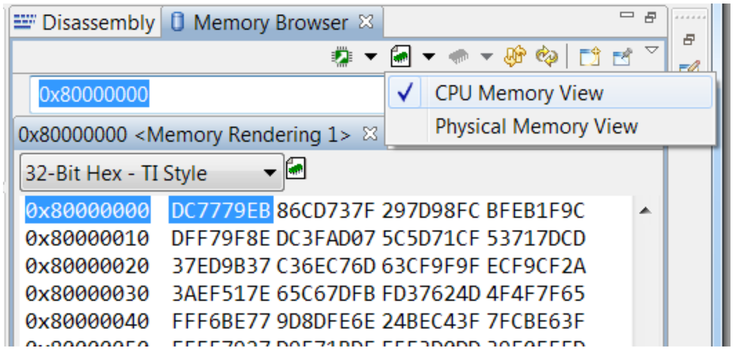
Registers Variables Expressions Breakpoints Debug Target Configurations

Name	Value	Description
L4_CFG__CM_CORE_AON__MPU_CM_		
CM_MPU_CLKSTCTRL	0x00000102	This register enables the MPU domain power state transitior
RESERVED	00000000000000000000...	
CLKACTIVITY_MPU_GCLK	1 - ACT	This field indicates the state of the MPU_DPLL_CLK clock in t
RESERVED	000000	
CLKTRCTRL	10 - SW_WKUP	Controls the clock state transition of the MPU clock domain.
CM_MPU_STATICDEP	0x2C00B0B0	This register controls the static domain depedencies from M
CM_MPU_DYNAMICDEP	0x04000030	This register controls the dynamic domain depedencies fron
CM_MPU_MPU_CLKCTRL	0x01000001	This register manages the MPU clocks. [Memory Mapped]
CM_MPU_MPU_MPU_DBG_CLKCTI	0x00000001	This register manages the MPU_MPU_DBG clocks. [warm res
L4_WKUP__PRM__COREAON_PRM		
L4_CFG__CM_CORE__IVA_CM_CORE		
CM_IVA_CLKSTCTRL	0x00000102	This register enables the IVA domain power state transition.
RESERVED	00000000000000000000...	
CLKACTIVITY_IVA_GCLK	1 - ACT	This field indicates the state of the IVA_ROOT_CLK clock inpu
RESERVED	000000	
CLKTRCTRL	10 - SW_WKUP	Controls the clock state transition of the IVA clock domain.
CM_IVA_STATICDEP	0x00000020	This register controls the static domain depedencies from IV
CM_IVA_DYNAMICDEP	0x00000000	This register controls the dynamic domain depedencies fron
CM_IVA_IVA_CLKCTRL	0x00050001	This register manages the IVA clocks. [Memory Mapped]
CM_IVA_SL2_CLKCTRL	0x00030000	This register manages the SL2 clocks. [Memory Mapped]
L4_WKUP__PRM__CKGEN_PRM		
L4_WKUP__PRM__EMU_CM		

Memory view

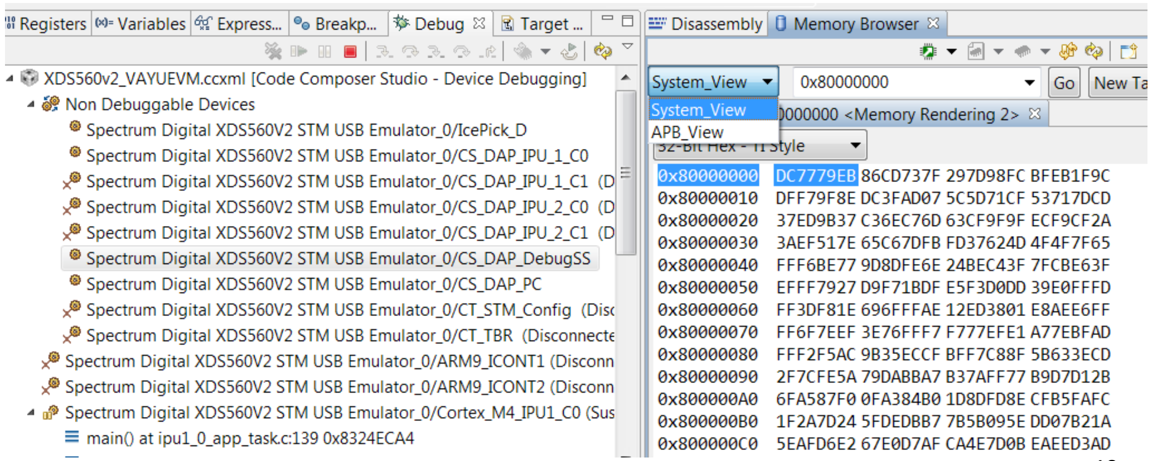
- Memory view (View → Memory Browser) in the context of Cortex A15.

  Ability to simultaneously view physical memory and the memory as visible to the CPU.



```
0x80000000
0x80000000 <Memory Rendering 1>
0x80000000  DC7779EB 86CD737F 297D98FC BFEB1F9C
0x80000010  DFF79F8E DC3FAD07 5C5D71CF 53717DCD
0x80000020  37ED9B37 C36EC76D 63CF9F9F ECF9CF2A
0x80000030  3AEF517E 65C67DFB FD37624D 4F4F7F65
0x80000040  FFF6BE77 9D8DFE6E 24BEC43F 7FCBE63F
```

- Cortex DAP memory view
  - Right click on the connection in the debug view and click on Show All Cores
  - Select debug context for CS_DAP_DebugSS
  - Memory view (system view) shows system view of the memory (same as what you saw from the Cortex A15 context in the memory view)
  - View can be changed to APB view from the memory windows

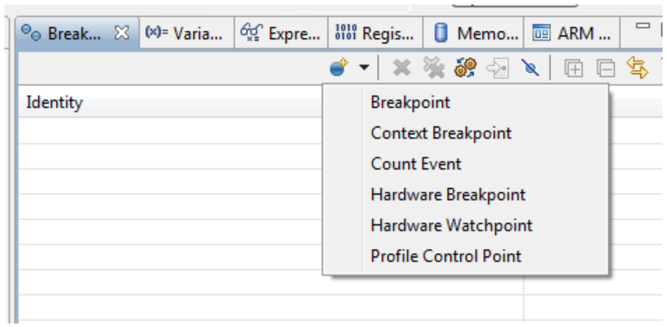


```
XDS560v2_VAYUEVM.ccxml [Code Composer Studio - Device Debugging]
  Non Debuggable Devices
    Spectrum Digital XDS560V2 STM USB Emulator_0/IcePick_D
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_IPU_1_C0
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_IPU_1_C1  (D
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_IPU_2_C0  (D
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_IPU_2_C1  (D
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_DebugSS
    Spectrum Digital XDS560V2 STM USB Emulator_0/CS_DAP_PC
    Spectrum Digital XDS560V2 STM USB Emulator_0/CT_STM_Config  (Disc
    Spectrum Digital XDS560V2 STM USB Emulator_0/CT_TBR  (Disconnecte
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT1 (Disconn
  Spectrum Digital XDS560V2 STM USB Emulator_0/ARM9_ICONT2 (Disconn
  Spectrum Digital XDS560V2 STM USB Emulator_0/Cortex_M4_IPU1_C0 (Sus
    main() at ipu1_0_app_task.c:139 0x8324ECA4
```

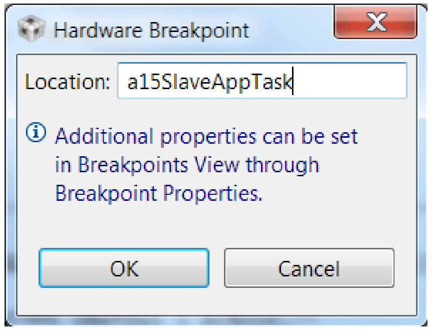
```
0x80000000
0x80000000 <Memory Rendering 2>
0x80000000  DC7779EB 86CD737F 297D98FC BFEB1F9C
0x80000010  DFF79F8E DC3FAD07 5C5D71CF 53717DCD
0x80000020  37ED9B37 C36EC76D 63CF9F9F ECF9CF2A
0x80000030  3AEF517E 65C67DFB FD37624D 4F4F7F65
0x80000040  FFF6BE77 9D8DFE6E 24BEC43F 7FCBE63F
0x80000050  EFFF7927 D9F71BDF E5F3D0DD 39E0FFFD
0x80000060  FF3DF81E 696FFFAE 12ED3801 E8AEE6FF
0x80000070  FF6F7EEF 3E76FFF7 F777EFE1 A77EBFAD
0x80000080  FFF2F5AC 9B35ECCF BFF7C88F 5B633ECD
0x80000090  2F7CFE5A 79DABBA7 B37AFF77 B9D7D12B
0x800000A0  6FA587F0 0FA384B0 1D8DFD8E CFB5FAFC
0x800000B0  1F2A7D24 5FDEDBB7 7B5B095E DD07B21A
0x800000C0  5EAFD6E2 67E0D7AF CA4E7D0B EAEED3AD
```

Breakpoints

- Breakpoint view (View → Breakpoints) has SW Breakpoints, HW Breakpoints, Watchpoints, and PMU counters (count events).

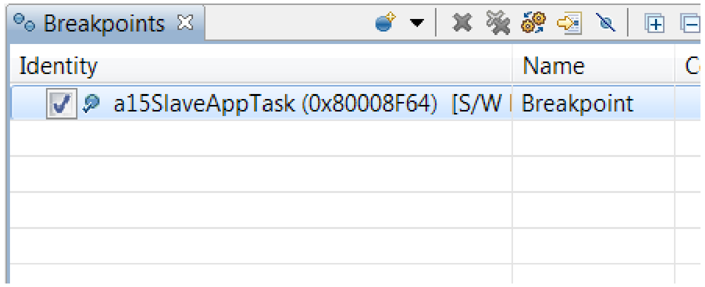


- A breakpoint or Watchpoint can be added from the breakpoint view (or right click source view context).
  - Once a breakpoint is set, it is populated in the breakpoint view.



Location: a15SlaveAppTask

Additional properties can be set in Breakpoints View through Breakpoint Properties.

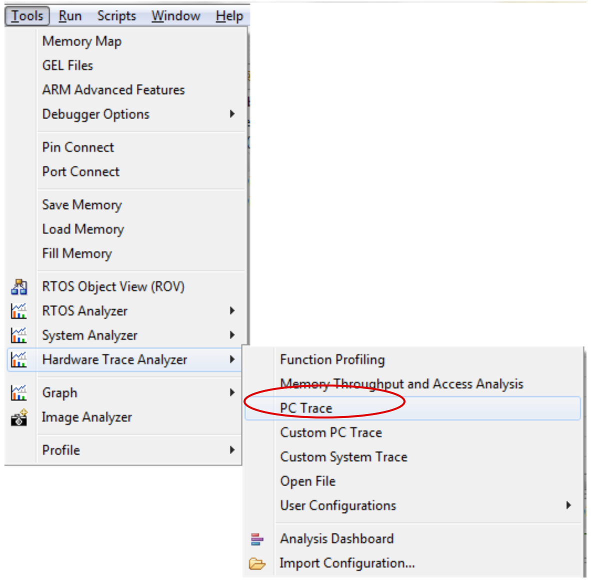


Identity	Name	C
✓ a15SlaveAppTask (0x80008F64) [S/W	Breakpoint	

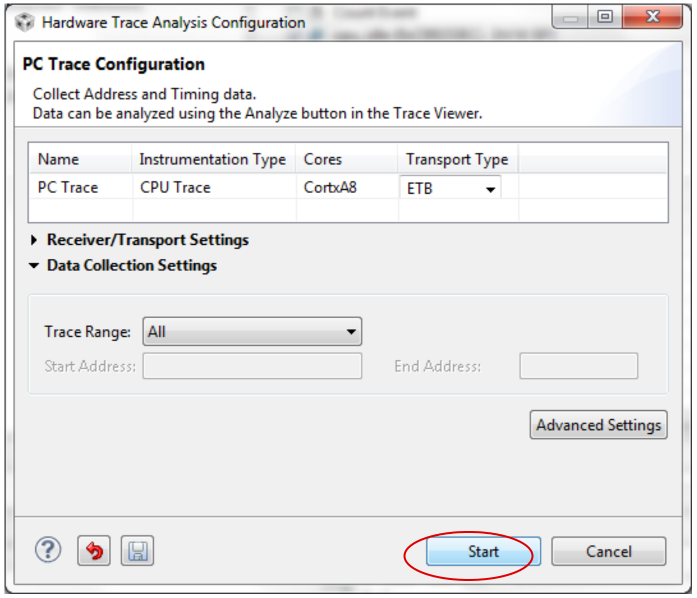
  - Run the target and the breakpoint will be hit on a15SlaveAppTask.

Trace Use Cases

- In menu select *Tools -> Hardware Trace Analyzer -> PC Trace* to start PC Trace



- Click on Start to setup the trace and open Trace Viewer.



Hardware Trace Analysis Configuration

PC Trace Configuration

Collect Address and Timing data.
Data can be analyzed using the Analyze button in the Trace Viewer.

Name	Instrumentation Type	Cores	Transport Type
PC Trace	CPU Trace	CortxA8	ETB

Receiver/Transport Settings

Data Collection Settings

Trace Range: All

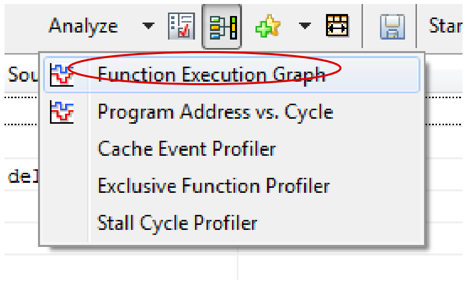
Start Address:

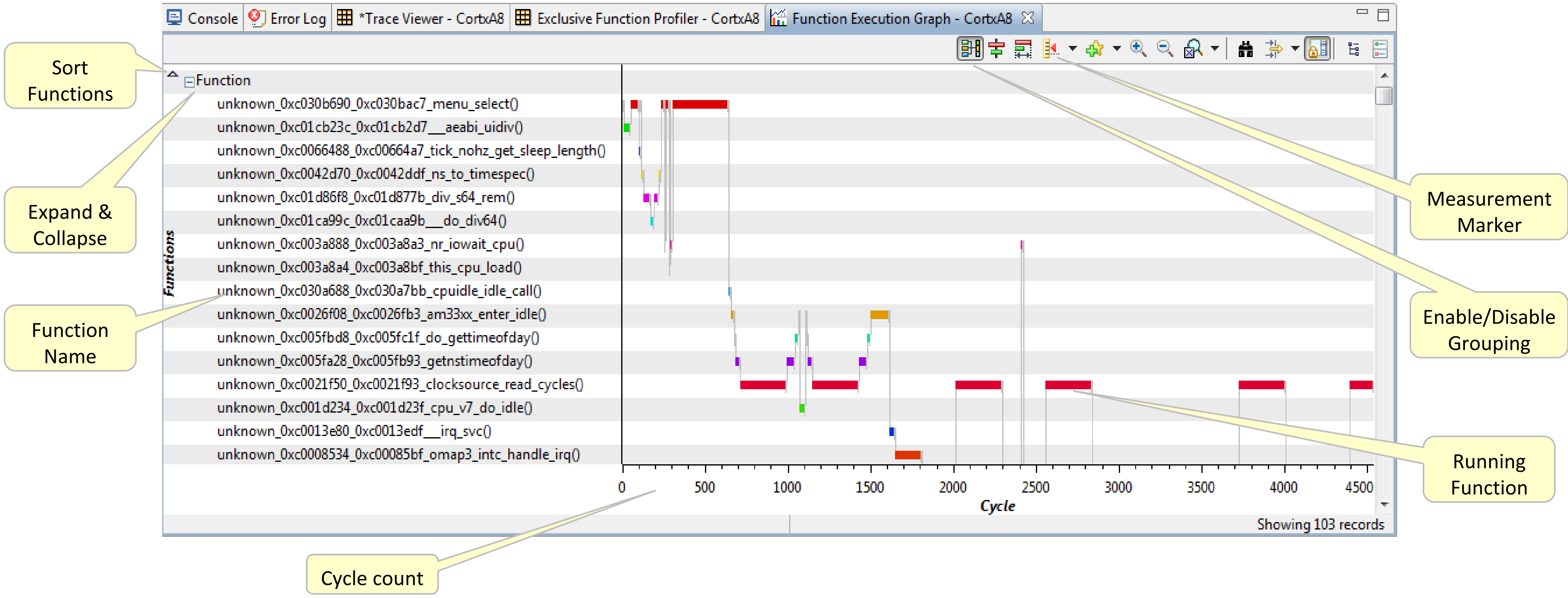
End Address:

Texas Instruments

Execution flow graph

Function execution graph can be launched by clicking Analyze -> Function Execution Graph

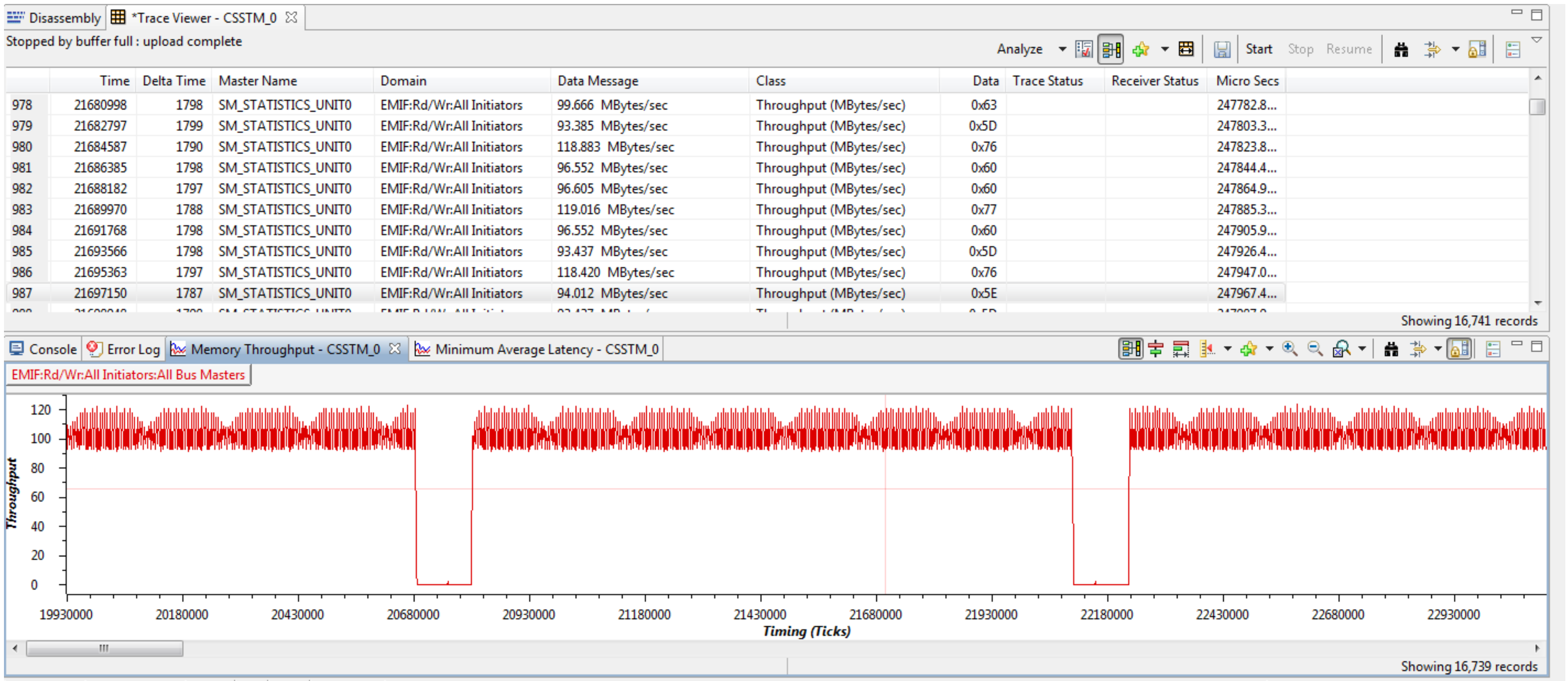




Console
Error Log
*Trace Viewer - CortxA8
Exclusive Function Profiler - CortxA8
Function Execution Graph - CortxA8
Function
Functions
unknown_0xc030b690_0xc030bac7_menu_select()
unknown_0xc01cb23c_0xc01cb2d7__aeabi_uidiv()
unknown_0xc0066488_0xc00664a7_tick_nohz_get_sleep_length()
unknown_0xc0042d70_0xc0042ddf_ns_to_timespec()
unknown_0xc01d86f8_0xc01d877b_div_s64_rem()
unknown_0xc01ca99c_0xc01caa9b__do_div64()
unknown_0xc003a888_0xc003a8a3_nr_iowait_cpu()
unknown_0xc003a8a4_0xc003a8bf_this_cpu_load()
unknown_0xc030a688_0xc030a7bb_cpuidle_idle_call()
unknown_0xc0026f08_0xc0026fb3_am33xx_enter_idle()
unknown_0xc005fbd8_0xc005fc1f_do_gettimeofday()
unknown_0xc005fa28_0xc005fb93_getnstimeofday()
unknown_0xc0021f50_0xc0021f93_clocksource_read_cycles()
unknown_0xc001d234_0xc001d23f_cpu_v7_do_idle()
unknown_0xc0013e80_0xc0013edf__irq_svc()
unknown_0xc0008534_0xc00085bf_omap3_intc_handle_irq()
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Cycle
Showing 103 records
Sort Functions
Expand & Collapse
Function Name
Cycle count
Measurement Marker
Enable/Disable Grouping
Running Function

View EMIF bandwidth utilization

- Select *Memory Throughput – CSSTM_0* tab to see EMIF throughput over a period of time the trace was captured
- The tabular view and graphs are correlated for easy navigation
- Capabilities like zooming, measurement market, grouping, find, and filtering etc are available from the tabular and graphical views


Disassembly
*Trace Viewer - CSSTM_0
Analyze
Start
Stop
Resume
Console
Error Log
Memory Throughput - CSSTM_0
Minimum Average Latency - CSSTM_0
EMIF:Rd/Wr:All Initiators:All Bus Masters
Throughput
Timing (Ticks)

Stopped by buffer full : upload complete

	Time	Delta Time	Master Name	Domain	Data Message	Class	Data	Trace Status	Receiver Status	Micro Secs
978	21680998	1798	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	99.666 MBytes/sec	Throughput (MBytes/sec)	0x63			247782.8...
979	21682797	1799	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	93.385 MBytes/sec	Throughput (MBytes/sec)	0x5D			247803.3...
980	21684587	1790	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	118.883 MBytes/sec	Throughput (MBytes/sec)	0x76			247823.8...
981	21686385	1798	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	96.552 MBytes/sec	Throughput (MBytes/sec)	0x60			247844.4...
982	21688182	1797	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	96.605 MBytes/sec	Throughput (MBytes/sec)	0x60			247864.9...
983	21689970	1788	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	119.016 MBytes/sec	Throughput (MBytes/sec)	0x77			247885.3...
984	21691768	1798	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	96.552 MBytes/sec	Throughput (MBytes/sec)	0x60			247905.9...
985	21693566	1798	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	93.437 MBytes/sec	Throughput (MBytes/sec)	0x5D			247926.4...
986	21695363	1797	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	118.420 MBytes/sec	Throughput (MBytes/sec)	0x76			247947.0...
987	21697150	1787	SM_STATISTICS_UNIT0	EMIF:Rd/Wr:All Initiators	94.012 MBytes/sec	Throughput (MBytes/sec)	0x5E			247967.4...

Showing 16,741 records

Showing 16,739 records

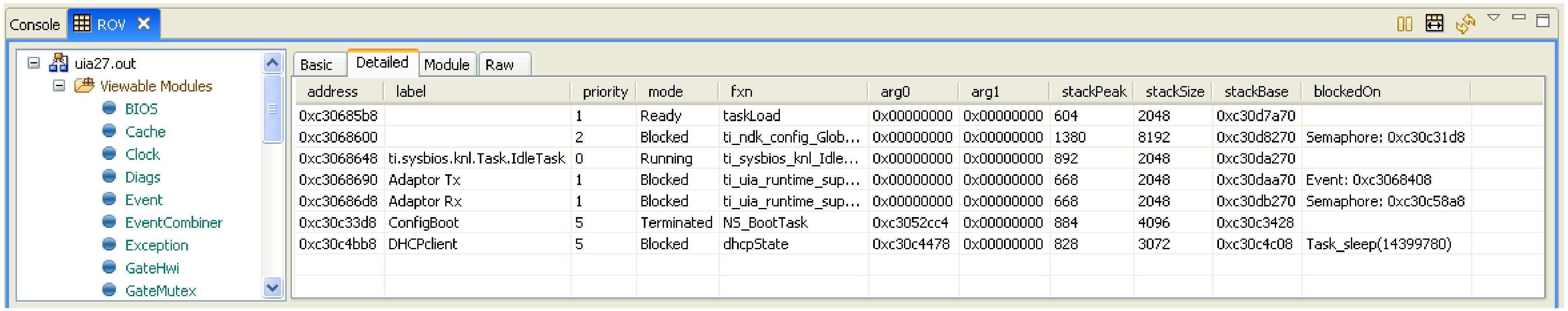
Texas Instruments

CCS Debug Support for SYS/BIOS

- Object Viewer (ROV) provides higher-level view of internal state of objects like Task, Swi, and Semaphore
  - “Smart” Watch Window – much more than a raw C struct view
  - Example: stack and heap usage computed and displayed
- Instrumentation (System Analyzer & UIA)
  - Interesting events are efficiently logged from SYS/BIOS and application code
  - Host tools decode and display: context-switch graph, CPU load, status messages
- Exceptions
  - Hardware-generated exceptions are caught and registers state is saved (device specific)
- Stack checking at runtime
- Debug kernel
  - Parameter checking via asserts
  - Internal state asserts
  - Optional via configuration – no API differences
- Source code

Object Viewer (ROV)

- ROV provides four different views
  - Module: module-wide (global) state
  - Basic: simpler view of state for each object (instance)
  - Detailed: advanced view of state for each object
  - Raw: all the state in an unfiltered view (much like a C struct view)
- Runs in the same Debug perspective as C source debugger



address	label	priority	mode	fxn	arg0	arg1	stackPeak	stackSize	stackBase	blockedOn
0xc30685b8		1	Ready	taskLoad	0x00000000	0x00000000	604	2048	0xc30d7a70	
0xc3068600		2	Blocked	ti_ndk_config_Glob...	0x00000000	0x00000000	1380	8192	0xc30d8270	Semaphore: 0xc30c31d8
0xc3068648	ti.sysbios.knl.Task.IdleTask	0	Running	ti_sysbios_knl_Idle...	0x00000000	0x00000000	892	2048	0xc30da270	
0xc3068690	Adaptor Tx	1	Blocked	ti_uia_runtime_sup...	0x00000000	0x00000000	668	2048	0xc30daa70	Event: 0xc3068408
0xc30686d8	Adaptor Rx	1	Blocked	ti_uia_runtime_sup...	0x00000000	0x00000000	668	2048	0xc30db270	Semaphore: 0xc30c58a8
0xc30c33d8	ConfigBoot	5	Terminated	NS_BootTask	0xc3052cc4	0x00000000	884	4096	0xc30c3428	
0xc30c4bb8	DHCPclient	5	Blocked	dhcpState	0xc30c4478	0x00000000	828	3072	0xc30c4c08	Task_sleep(14399780)

ROV Setup

- Connect target and load symbols
- Go to Window->preferences->Code Composer Studio->RTSC
- Pick device family as “ARM”,
- Remove all unnecessary paths and the click Add
- Ensure you add package path till packages folder e.g. <my_path>\bios_6_37_01_24\packages – This is under ti_components
- Add all packages that you need typically (ipc, bios and xdc)
- Click Apply then ok
- Copy the “*.rov.xs” file from \vision_sdk\binaries\obj\vision_sdk\tda2xx-evm\ipu1_0\release\vision_sdk_configuro\package\cfg to \vision_sdk\binaries\vision_sdk\bin\tda2xx-evm (where you have the binaries).
  - For IPU1_0 – copy MAIN_APP_ipu1_0_pem4.rov.xs
  - For DSP1 – Copy MAIN_APP_c6xdsp1_pe66.rov.xs
- This is a very important step, without this ROV will not work
- Close and re-open CCS
- Launch/Re-Launch the ROV through Tools-> RTOS object View (ROV)
- Navigate through “Viewable Modules” to debug/analysis

CCS Trace Debug

- L3 Statistic Collectors
- OCP Watch Point
- EVE SMSET Trace
- DSP Processor Trace

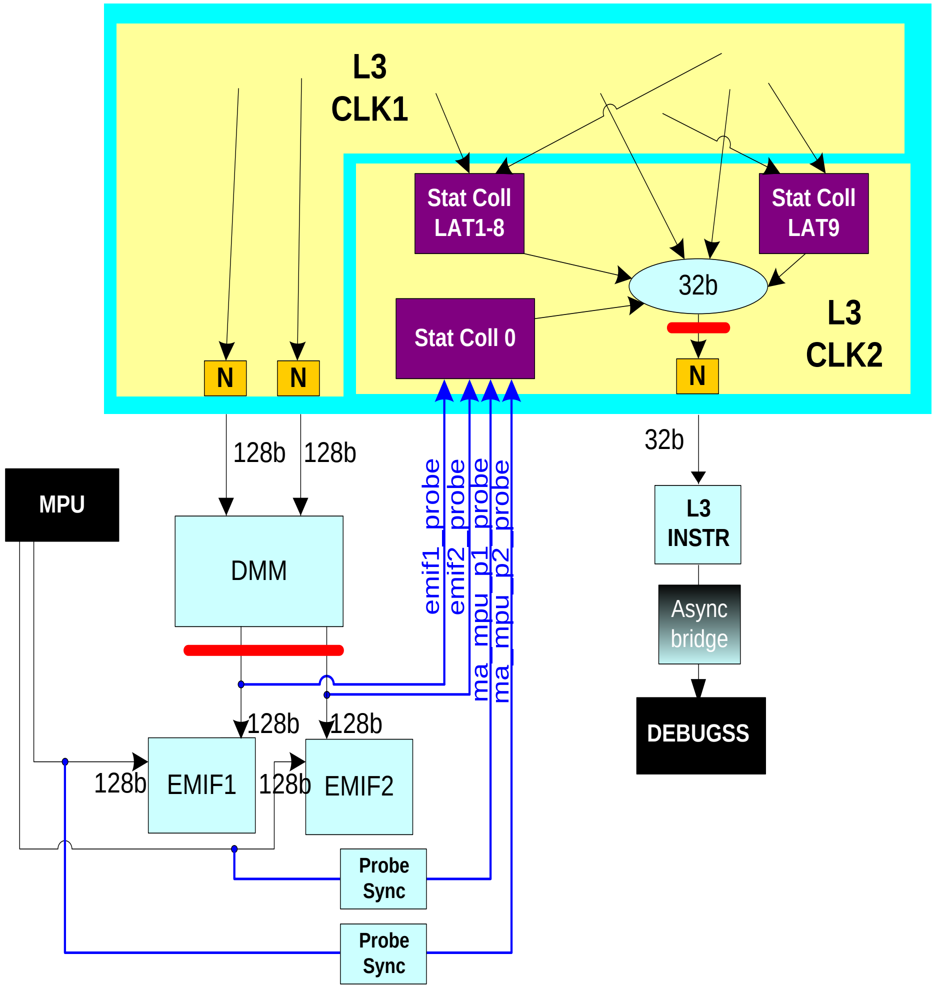
L3 StatColl Overview

- Provides ability to probe OCP (Open Core Bus Protocol) or NTTP (Arteris L3 Interconnect Packet Protocol) links.
- Transmitting results to a debug unit through a dedicated NTTP link.
- Software controlled at run time through the service network.
- Non intrusive monitoring
- Up to 8 probes for monitoring NTTP or OCP links.
- Programmable filters and counters.
- Collect results at programmable time interval
- Provides metrics such as throughput and latency on some data flows.
- Additional filtering capabilities to focus on certain initiators or targets.

L3 Statistic Collectors (StatColl)

- 10 statcoll instances available on Vayu to measure traffic statistics of different subsystems.
- Static Mapping from subsystems to statistic collector.

StatColl_0			
Probe #	Description	Link	Port #
0	EMIF1_SYS	OCP REQ	0
		OCP RSP	1
1	EMIF2_SYS	OCP REQ	2
		OCP RSP	3
2	MA_MPU_P1	OCP REQ	4
		OCP RSP	5
3	MA_MPU_P2	OCP REQ	6
		OCP RSP	7



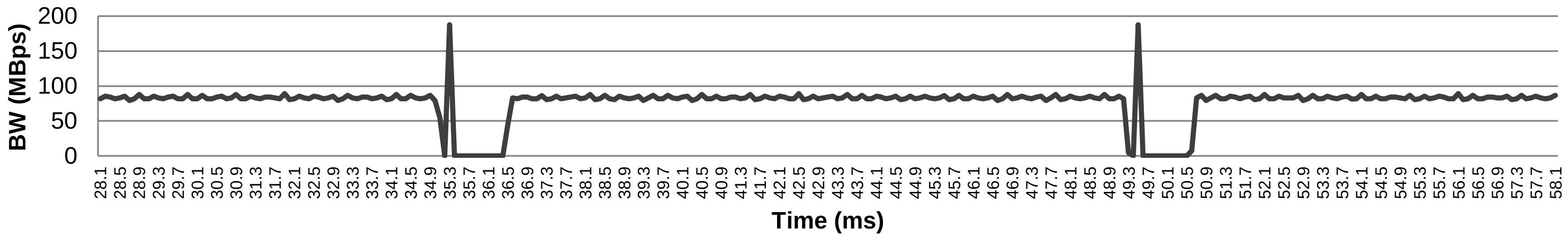
Refer the Vayu TRM to know the complete Initiator to statcoll mapping.

L3 Statcoll - A Small Caveat

- If CPU has cache enabled, the data read from the L3 Statcoll and actual bytes transferred may differ.
- This is because the L3 statcoll reads bytes at the L3/EMIF interface and lot of data may be cached already.
- ARM performs speculative reads. Not all reads are used by the CPU.

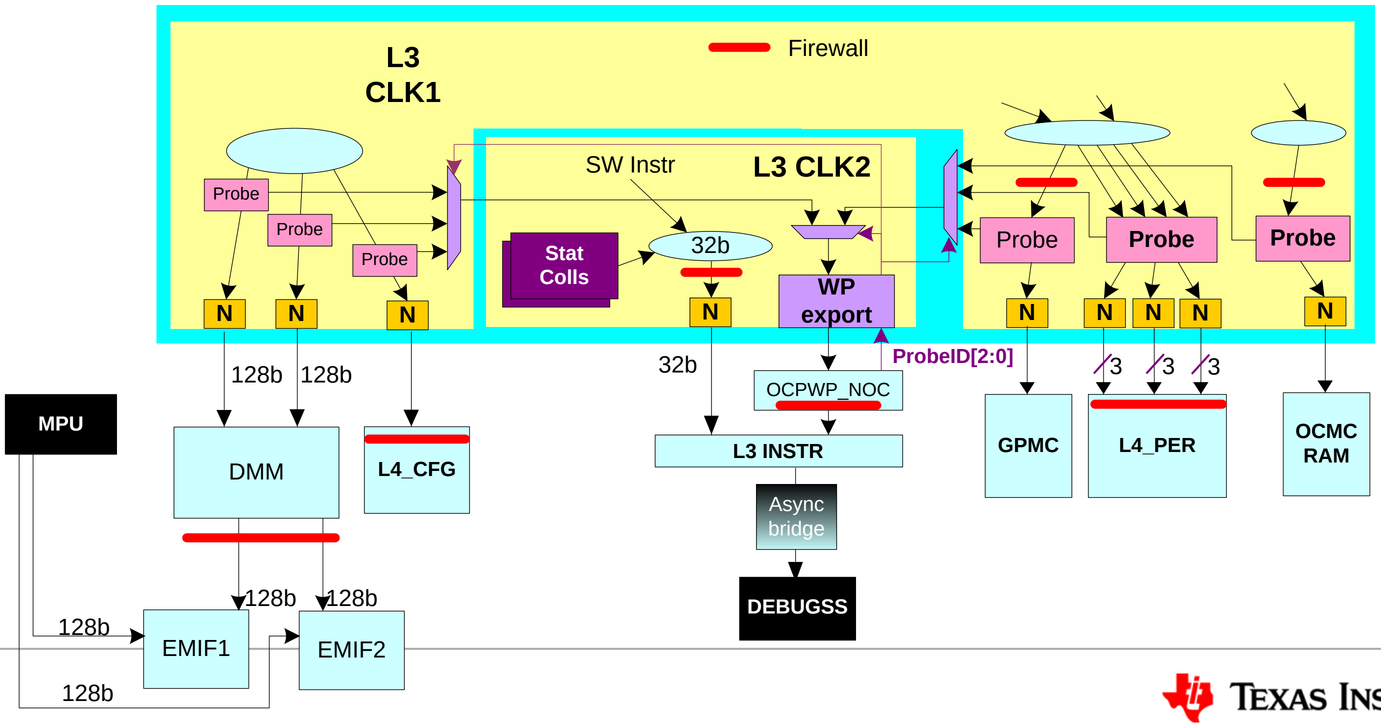
L3 Statistic Collectors from Software

- In a production board where DEBUG connectivity is difficult to achieve, a spare CPU core and spare timer can be used to read the StatColl registers at regular intervals.
- Eg. To capture the BW profile of a given initiator use the following steps:
  - Set up the StatColl for the initiator to capture number of bytes transferred.
  - Configure a timer to maintain ‘x’ us time gap.
  - Read the number of bytes transferred every ‘x’ us at the initiator ports (read + write) or the destination memory from the StatColl register.
  - The number of bytes obtained is divided by ‘x’ us to get the average BW within the ‘x’ us.
  - The process is repeated multiple times to generate a bandwidth profile.
- x=100 us is found to give good granularity for analyzing the traffic.



OCP Watch Point Profiling

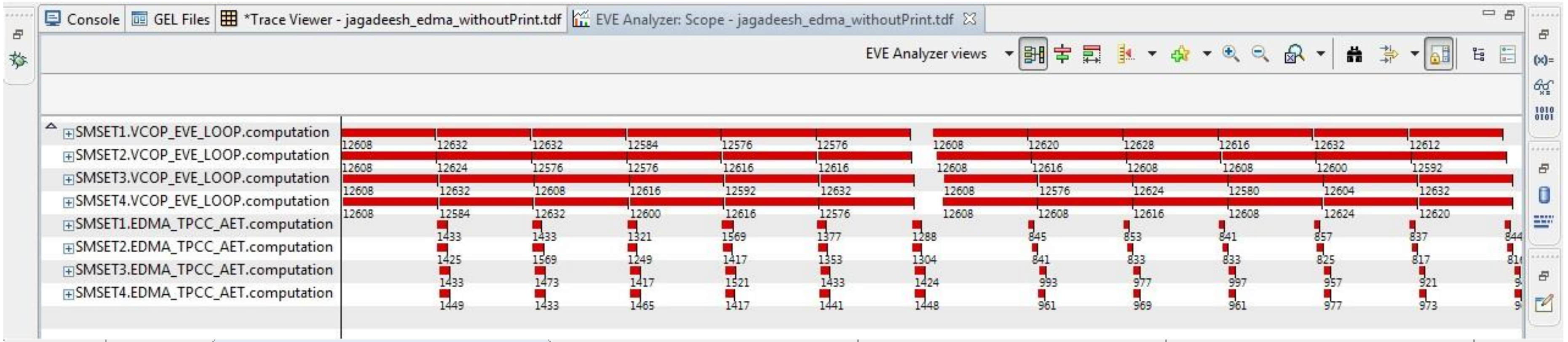
- Functional debug capability.
- Internal probes embedded for DMM_P1-2, L4_CFG, L4_PER_P0-1-2-3, GPMC and OCMC_RAM
- Capture time stamped Address, Burst Length Trace filtered on a Single Initiator/All Initiators/Group of Initiators.
- Trace sent directly to the Debug SS.

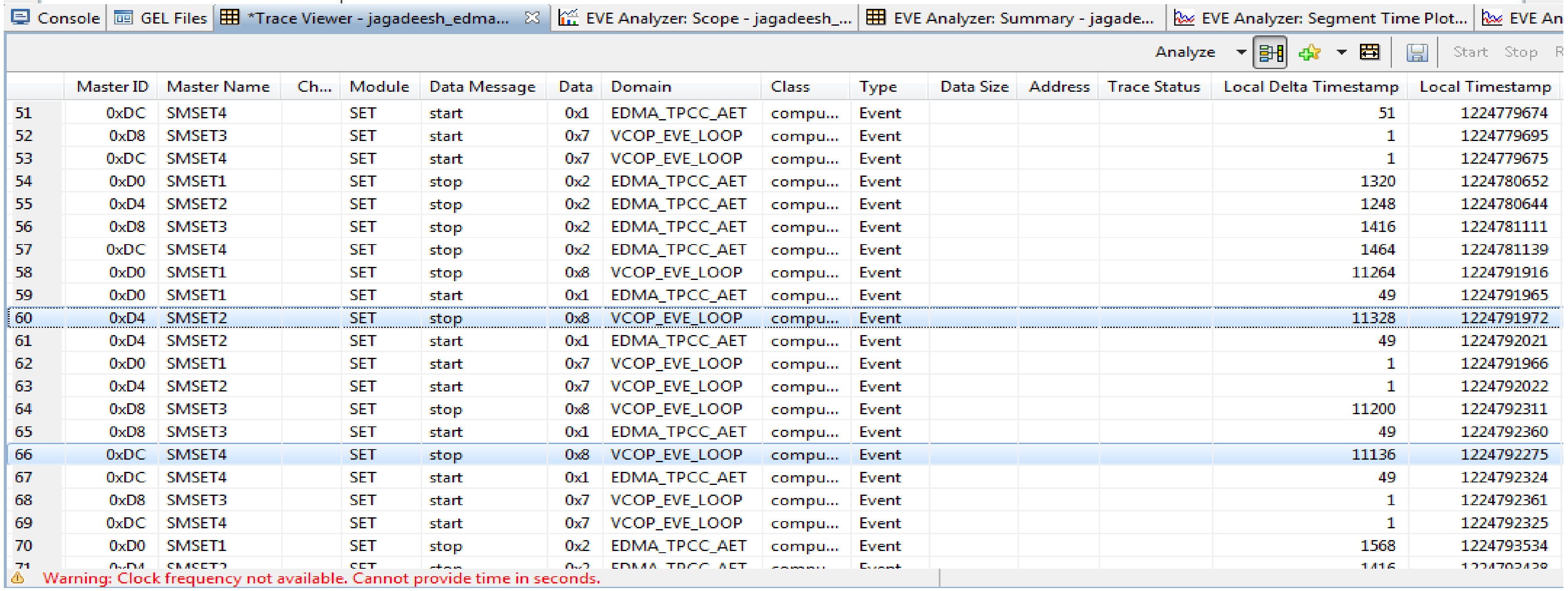


EVE SMSET Trace

- Example:
  - 7x7 Gaussian filter of a 768x512 image divided into 128x64 blocks and apportioned across 4 EVE cores,
  - 4 EVE's working horizontally with first EVE being allocated 1/4th of the data, next EVE being allocated next 1/4th and so on.
  - From a compute we would expect at least: 128 x 64 x 49 / 16 x 1/2 = 12544 SMSET (ARP32) cycles with 19 cycles of pipeline overhead + 6-7 cycles of parameter decode + 6 cycles of command decode = 12544 + 32 = 12576.
  - From a DMA we need to bring in a 130 x 66 and produce a 128 x 64 = 16,772 bytes which at @ 5.88 bytes per VCOP cycle should be = 16772 * 1/5.88 * 1/2 = 1445 cycles.

EVE SMSET Trace (cont.)



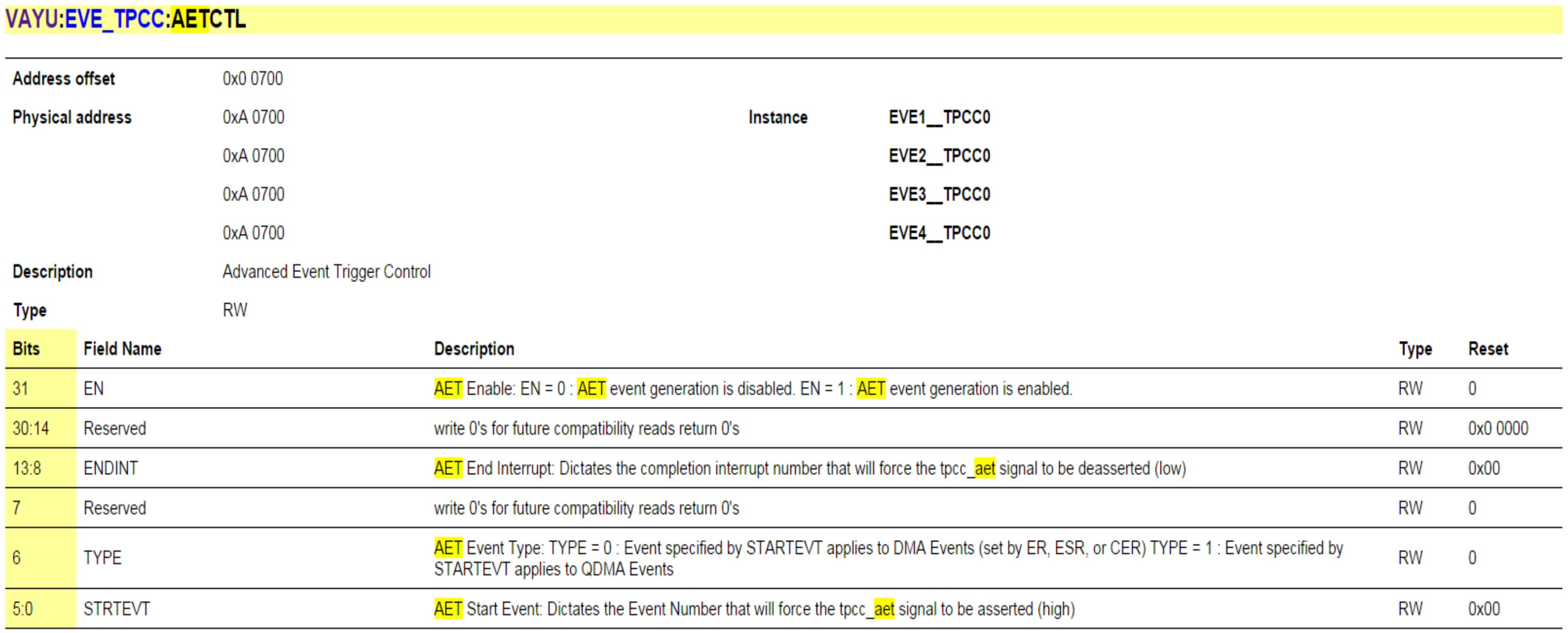

Console
GEL Files
*Trace Viewer - jagadeesh_edma...
EVE Analyzer: Scope - jagadeesh_...
EVE Analyzer: Summary - jagade...
EVE Analyzer: Segment Time Plot...
EVE An
Analyze
Start
Stop

	Master ID	Master Name	Ch...	Module	Data Message	Data	Domain	Class	Type	Data Size	Address	Trace Status	Local Delta Timestamp	Local Timestamp
51	0xDC	SMSET4		SET	start	0x1	EDMA_TPCC_AET	compu...	Event				51	1224779674
52	0xD8	SMSET3		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224779695
53	0xDC	SMSET4		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224779675
54	0xD0	SMSET1		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1320	1224780652
55	0xD4	SMSET2		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1248	1224780644
56	0xD8	SMSET3		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1416	1224781111
57	0xDC	SMSET4		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1464	1224781139
58	0xD0	SMSET1		SET	stop	0x8	VCOP_EVE_LOOP	compu...	Event				11264	1224791916
59	0xD0	SMSET1		SET	start	0x1	EDMA_TPCC_AET	compu...	Event				49	1224791965
60	0xD4	SMSET2		SET	stop	0x8	VCOP_EVE_LOOP	compu...	Event				11328	1224791972
61	0xD4	SMSET2		SET	start	0x1	EDMA_TPCC_AET	compu...	Event				49	1224792021
62	0xD0	SMSET1		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224791966
63	0xD4	SMSET2		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224792022
64	0xD8	SMSET3		SET	stop	0x8	VCOP_EVE_LOOP	compu...	Event				11200	1224792311
65	0xD8	SMSET3		SET	start	0x1	EDMA_TPCC_AET	compu...	Event				49	1224792360
66	0xDC	SMSET4		SET	stop	0x8	VCOP_EVE_LOOP	compu...	Event				11136	1224792275
67	0xDC	SMSET4		SET	start	0x1	EDMA_TPCC_AET	compu...	Event				49	1224792324
68	0xD8	SMSET3		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224792361
69	0xDC	SMSET4		SET	start	0x7	VCOP_EVE_LOOP	compu...	Event				1	1224792325
70	0xD0	SMSET1		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1568	1224793534
71	0xD4	SMSET2		SET	stop	0x2	EDMA_TPCC_AET	compu...	Event				1416	1224793438

⚠ Warning: Clock frequency not available. Cannot provide time in seconds.

EVE SMSET EDMA Trace

- Write to the AETCTL the STRTEVT and the ENDINT the DMA channel numbers being used to be able to capture the beginning of the series of the EDMA transactions and the end of the EDMA transactions.



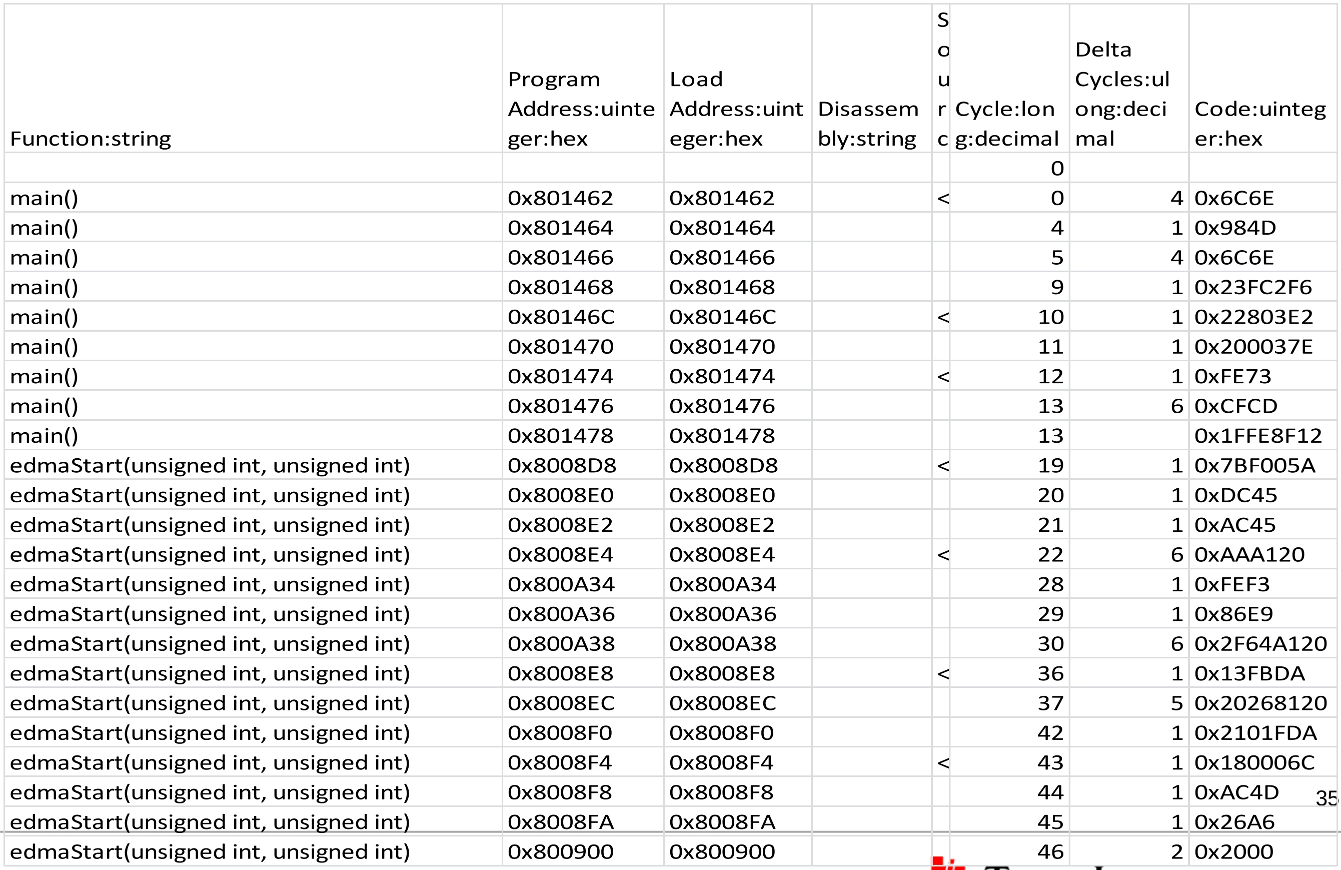
VAYU:EVE_TPCC:AETCTL

Address offset	0x0 0700		
Physical address	0xA 0700	**Instance**	**EVE1__TPCC0**
	0xA 0700		**EVE2__TPCC0**
	0xA 0700		**EVE3__TPCC0**
	0xA 0700		**EVE4__TPCC0**
Description	Advanced Event Trigger Control		
Type	RW		

Bits	**Field Name**	**Description**	**Type**	**Reset**
31	EN	AET Enable: EN = 0 : AET event generation is disabled. EN = 1 : AET event generation is enabled.	RW	0
30:14	Reserved	write 0's for future compatibility reads return 0's	RW	0x0 0000
13:8	ENDINT	AET End Interrupt: Dictates the completion interrupt number that will force the tpcc_aet signal to be deasserted (low)	RW	0x00
7	Reserved	write 0's for future compatibility reads return 0's	RW	0
6	TYPE	AET Event Type: TYPE = 0 : Event specified by STARTEVT applies to DMA Events (set by ER, ESR, or CER) TYPE = 1 : Event specified by STARTEVT applies to QDMA Events	RW	0
5:0	STRTEVT	AET Start Event: Dictates the Event Number that will force the tpcc_aet signal to be asserted (high)	RW	0x00

DSP Processor Trace

- Example:
  - DSP Programming the DSP EDMA to start and then waiting for EDMA completion of 8MB DDR to DDR transfer.
  - Processor PC Trace/Function profiling and Write address trace enabled for the DSP.



Function:string	Program Address:uinteger:hex	Load Address:uinteger:hex	Disassembly:string	Source	Cycle:long:decimal	Delta Cycles:ulong:decimal	Code:uinteger:hex
					0		
main()	0x801462	0x801462		<	0	4	0x6C6E
main()	0x801464	0x801464			4	1	0x984D
main()	0x801466	0x801466			5	4	0x6C6E
main()	0x801468	0x801468			9	1	0x23FC2F6
main()	0x80146C	0x80146C		<	10	1	0x22803E2
main()	0x801470	0x801470			11	1	0x200037E
main()	0x801474	0x801474		<	12	1	0xFE73
main()	0x801476	0x801476			13	6	0xCFCD
main()	0x801478	0x801478			13		0x1FFE8F12
edmaStart(unsigned int, unsigned int)	0x8008D8	0x8008D8		<	19	1	0x7BF005A
edmaStart(unsigned int, unsigned int)	0x8008E0	0x8008E0			20	1	0xDC45
edmaStart(unsigned int, unsigned int)	0x8008E2	0x8008E2			21	1	0xAC45
edmaStart(unsigned int, unsigned int)	0x8008E4	0x8008E4		<	22	6	0xAAA120
edmaStart(unsigned int, unsigned int)	0x800A34	0x800A34			28	1	0xFEF3
edmaStart(unsigned int, unsigned int)	0x800A36	0x800A36			29	1	0x86E9
edmaStart(unsigned int, unsigned int)	0x800A38	0x800A38			30	6	0x2F64A120
edmaStart(unsigned int, unsigned int)	0x8008E8	0x8008E8		<	36	1	0x13FBDA
edmaStart(unsigned int, unsigned int)	0x8008EC	0x8008EC			37	5	0x20268120
edmaStart(unsigned int, unsigned int)	0x8008F0	0x8008F0			42	1	0x2101FDA
edmaStart(unsigned int, unsigned int)	0x8008F4	0x8008F4		<	43	1	0x180006C
edmaStart(unsigned int, unsigned int)	0x8008F8	0x8008F8			44	1	0xAC4D
edmaStart(unsigned int, unsigned int)	0x8008FA	0x8008FA			45	1	0x26A6
edmaStart(unsigned int, unsigned int)	0x800900	0x800900			46	2	0x2000