
TWENTIETH ANNUAL
TestConX™
March 3 - 6, 2019
Hilton Phoenix / Mesa Hotel
Mesa, Arizona
Archive

COPYRIGHT NOTICE

www.testconx.org

Standardized Closed-Chassis Debug Solution for Platform and System Debug

Rolf Kühnis
Intel Corporation

TestConX™
Mesa, Arizona • March 3 - 6, 2019

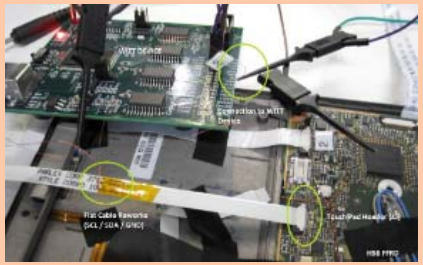
Disclaimer

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Agenda

- Terminologies
  - Closed versus Open-Chassis Debug
- The Beginning
  - Mobile phone driving MIPI® NIDnT℠
- Scaling onto USB
- Conclusions and Call to Action

Terminologies



Open Chassis/ Validation Platform

Debug Card

Wing Design

[1] Presented by Nokia in 2009 at IP & ESC-panel

Closed Chassis (Form Factor Design)

[2] Source:https://9to5mac.com/2016/06/28/review-minix-neo-c-usb-c-multiport-adapter-video/

TestConX™

Standardized Closed-Chassis Debug Solution for Platform and System Debug 4

2000-2019 20 YEARS ANNIVERSARY

The Beginning

- Mobile Phones
  - Cost pressure
    - Cannot afford additional I/F for debug
  - Increased design constraints
    - Accessibility/ Audio/ RF
- Started with SD Card Slot
  - Ubiquitous mobile I/F

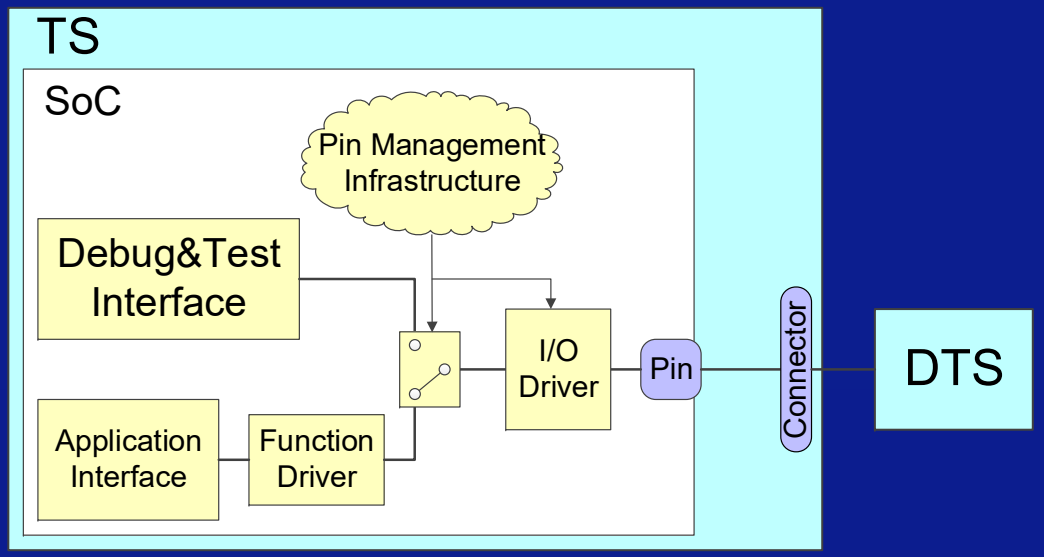
Nokia N95

Nokia N8

Motorola RAZR i

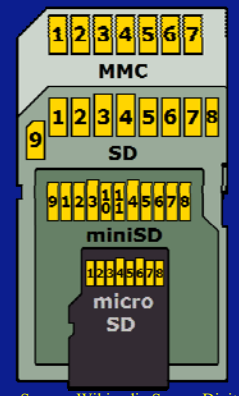
MIPI NIDnT SM a Conceptual View

- Narrow Interface for Debug and Test (NIDnT)
- NIDnT is temporary taking over pins or interfaces for debug
- MIPI Specification includes graceful interface switching methodology

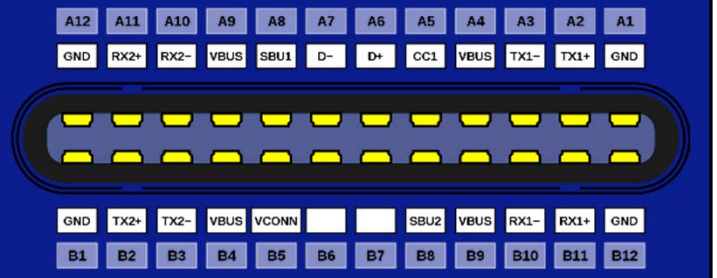


The Beginning: Evolution

- MIPI Narrow Interface for Debug and Test
  - From MIPI White Paper (2007) to Specification (2013)
  - Added HDMI, USB Type-C™ Receptacle (2017)



Source: Wikipedia Secure Digital
CC BY-SA 3.0



Source: Wikipedia USB-C; CC BY-SA 4.0

TestConX™

Standardized Closed-Chassis Debug Solution for Platform and System Debug 7

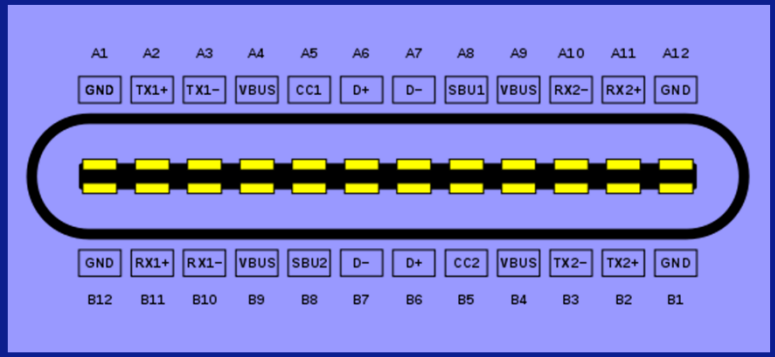
2000-2019 20 YEARS ANNIVERSARY

USB Type-C

Flippable

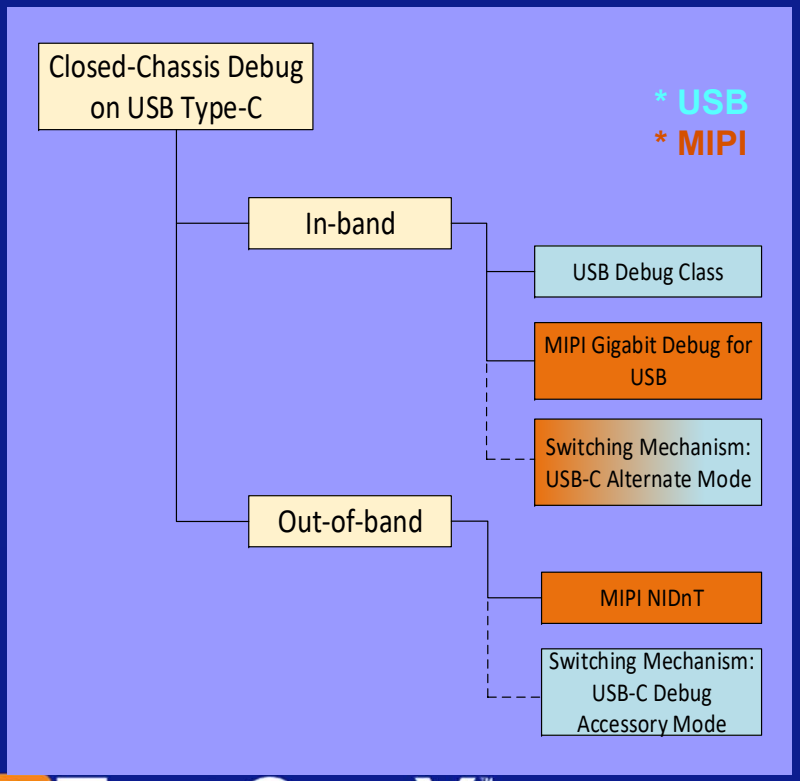
24 Signals/ Versatile

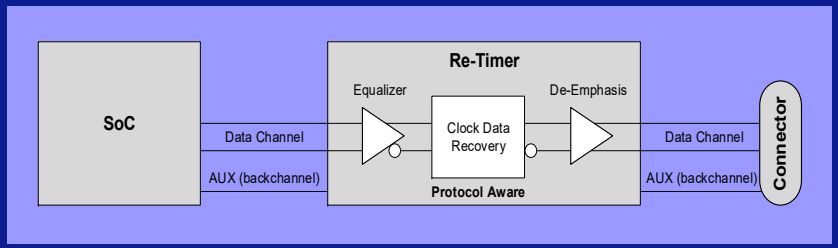
- USB D+/D-
- USB SS Tx/Rx
- Sideband Use (SBU)
- Configuration Channel (CC)
- VBus, GND



Source: Wikipedia USB-C; CC BY-SA 4.0

Solution Space Overview





TestConX

Standardized Closed-Chassis Debug Solution for Platform and System Debug 9

2000-2019 20 YEARS ANNIVERSARY

Standardizing Debug over USB Type-C™

Out-of-Band Solution

- MIPI® NIDnT℠ → MIPI.ORG
  - Reuse of receptacle for debug → Debug Alternate Mode in USB Type-C

NIDnT USB Type-C Overlay Modes

Overlay Mode Num.	0 (OFM)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 (VOM)
Descr.	Min-Pin + USB 2	Min-Pin + USB 2 + USB 3.1	Min-Pin + USB 2 + HS Trace	Min-Pin + USB 2 + USB 3.1 + HS Trace	Min-Pin Only	Min-Pin + USB 3.1	Min-Pin + HS Trace	Min-Pin + USB 3.1 + HS Trace	Min-Pin + USB 3.1 + Display x2	Min-Pin + Display x4	HS Debug + Audio + Display x2	Min-Pin + Display x2 + HS Trace	HS Debug + USB 3.1 + Audio	HS Debug + USB 2 + USB 3.1	HS Debug + USB 2 + Display x2	Vendor Overlay Mode
A2, A3 (SSTX1)		USB 3.1 Lane 0 TX	HTI Lane 1	USB 3.1 Lane 0 TX		USB 3.1 Lane 0 TX	HTI Lane 1	USB 3.1 Lane 0 TX	USB 3.1 Lane 0 TX	Display Lane 2 TX	HS Debug TX	HTI Lane 1	USB 3.1 Lane 0 TX	USB 3.1 Lane 0 TX	HS Debug TX	Implementation-defined
B11, B10 (SSRX1)		USB 3.1 Lane 0 RX	HTI Lane 2	USB 3.1 Lane 0 RX		USB 3.1 Lane 0 RX	HTI Lane 2	USB 3.1 Lane 0 RX	USB 3.1 Lane 0 RX	Display Lane 3 TX	HS Debug RX	HTI Lane 2	USB 3.1 Lane 0 RX	USB 3.1 Lane 0 RX	HS Debug RX	Implementation-defined
B2, B3 (SSTX2)		USB 3.1 Lane 1 TX	HTI Lane 3	HTI Lane 1		USB 3.1 Lane 1 TX	HTI Lane 3	HTI Lane 1	Display Lane 1 TX	Display Lane 1 TX	Display Lane 1 TX	Display Lane 1 TX	HS Debug TX	HS Debug TX	Display Lane 1 TX	Implementation-defined
A11, A10 (SSRX2)		USB 3.1 Lane 1 RX	HTI Lane 4	HTI Lane 2		USB 3.1 Lane 1 RX	HTI Lane 4	HTI Lane 2	Display Lane 0 TX	Display Lane 0 TX	Display Lane 0 TX	Display Lane 0 TX	HS Debug RX	HS Debug RX	Display Lane 0 TX	Implementation-defined
B6, B7 (D+/D-)	USB 2.0	USB 2.0	USB 2.0	USB 2.0	2-Pin Debug	2-Pin Debug	2-Pin Debug	2-Pin Debug	2-Pin Debug	2-Pin Debug	Audio L/R	2-Pin Debug	Audio L/R	USB 2.0	USB 2.0	Implementation-defined
A8, B8 (SBU1/2)	2-Pin Debug	2-Pin Debug	2-Pin Debug	2-Pin Debug					Display AUX	Display AUX	Audio Mic	Display AUX	Audio Mic		Display AUX	Implementation-defined

* High Speed Trace Interface (MIPI HTI℠)

Debugging through USB Type-C™

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1
GND	RX2+	RX2-	VBUS	SBU1	D-	D+	CC1	VBUS	TX1-	TX1+	GND
GND	TX2+	TX2-	VBUS	CC2	D+	D-	SBU2	VBUS	RX1-	RX1+	GND
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12

Two main mechanism have been defined:

1. USB Type-C Debug Accessory Mode
   - Using CC voltage level to identify Accessory Modes
   - R_p/R_p or R_d/R_d over CC-pins to identify DFP/UFP
2. MIPI NIDnT defines USB Type-C Debug Alternate Mode
   - Using a predefined SVID to switch between different overlay modes. Similar to VESA® Display Mode. Using a PD VDM

* Configuration Channel (CC)
Power Delivery (PD)
Vendor Defined Message (VDM)

TestConX™

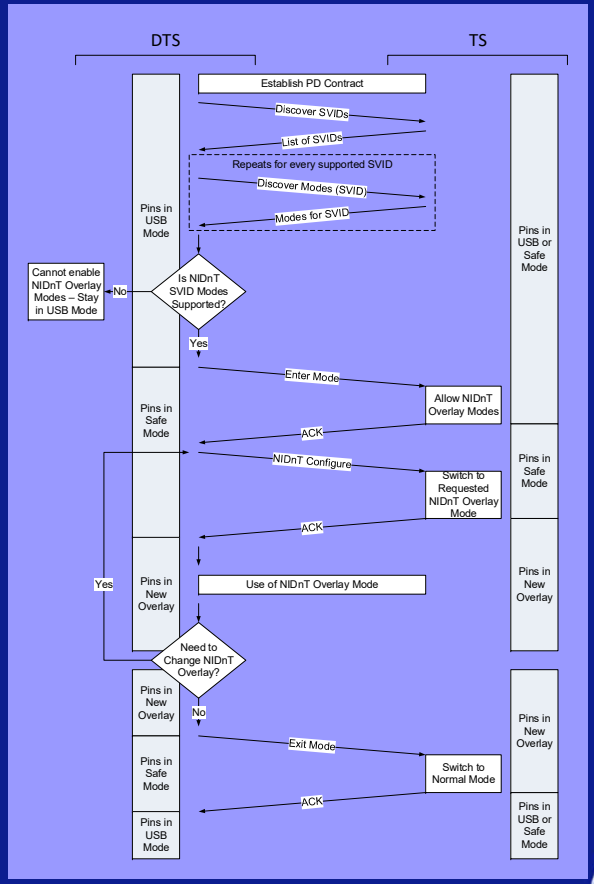
Standardized Closed-Chassis Debug Solution for Platform and System Debug 12

2000-2019 20 YEARS ANNIVERSARY

MIPI NIDnT VDM Flow

- Following the USB Type-C PD communication flow to enter USB alternate modes:
  - USB PD 0xFF00
  - Display Mode 0xFF01
  - MHL® 0xFF02
  - MIPI Debug 0xFF03
  - HDMI 0xFF04

…used for debug overall detection and configuration

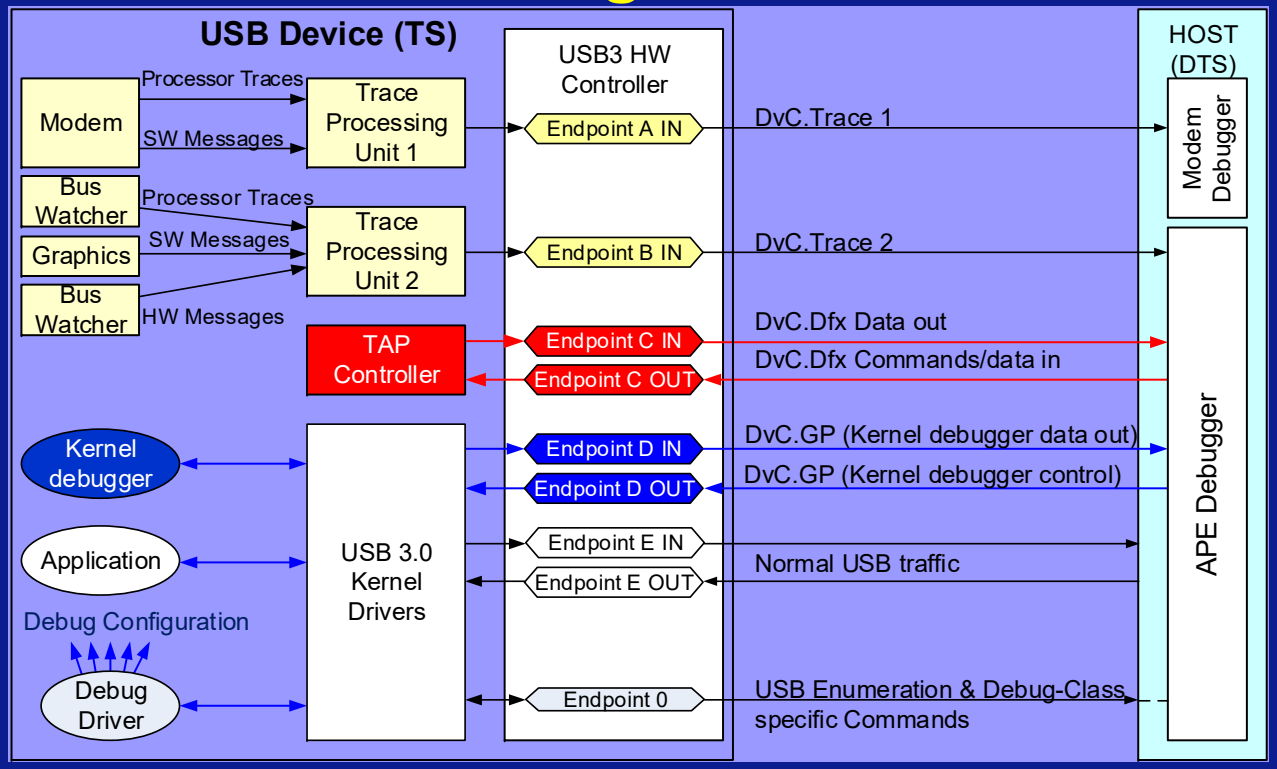


Standardizing Debug over USB

In-Band Solution

- USB Debug Class → USB.ORG
  - In-band debug traffic
  - Multiple end-points for debug/ concurrency with functional traffic

Scaling to USB



TestConX

Standardized Closed-Chassis Debug Solution for Platform and System Debug

15

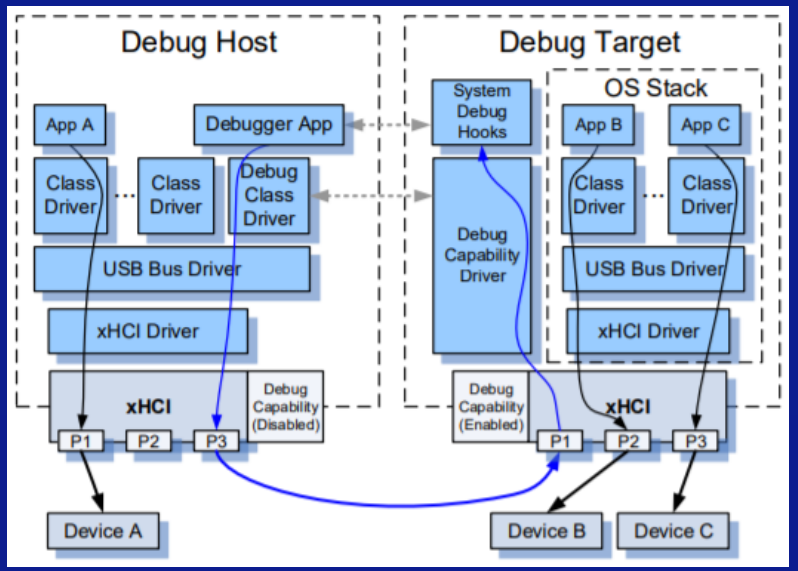
Scaling to USB (cont.)

- Need for high-speed link (mainly for tracing)
  - USB Debug Class standardized as part of USB3.1 (2015)

bInterface Class	bInterface Sub-Class	Description of Typical usage
Diagnostic Class (0xDC)	DbC.GP	General-Purpose Software debug function (e.g., GNU debugger)
	DbC.Dfx	Access to hardware Dfx hooks within the host (e.g., TAP, Memory access, debug trace, etc.)
	DbC.Trace	Debug traces
	DvC.GP	General-Purpose Software debug function (e.g., GNU debugger)
	DvC.Dfx	Access to Dfx hooks within the device (e.g., TAP, Memory access, debug trace1, etc.)
	DvC.Trace	Debug traces
	Debug Control	Control interface applicable to DxC.Trace, DxC.Dfx, DvC.GP (and optional for DbC.GP)

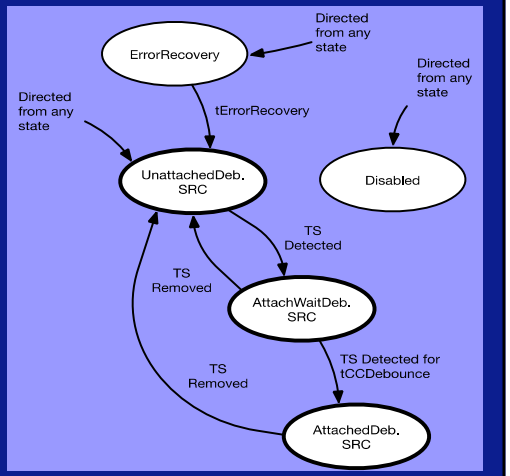
USB Debug Class

- Debug infrastructure behind one or several USB Endpoints
- Attached to a functional or dedicated USB controller
- Supports different USB debug roles (host/device)
- Implementation in HW or SW
- Examples:
  - MSFT Windbg. DbC as a USB3 xHCI debug capability; activated via USB A-to-A cable



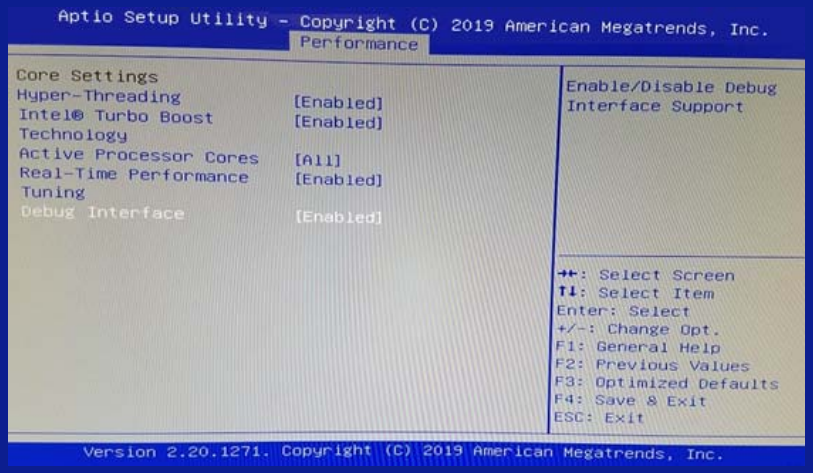
USB Type-C Debug Accessory Mode

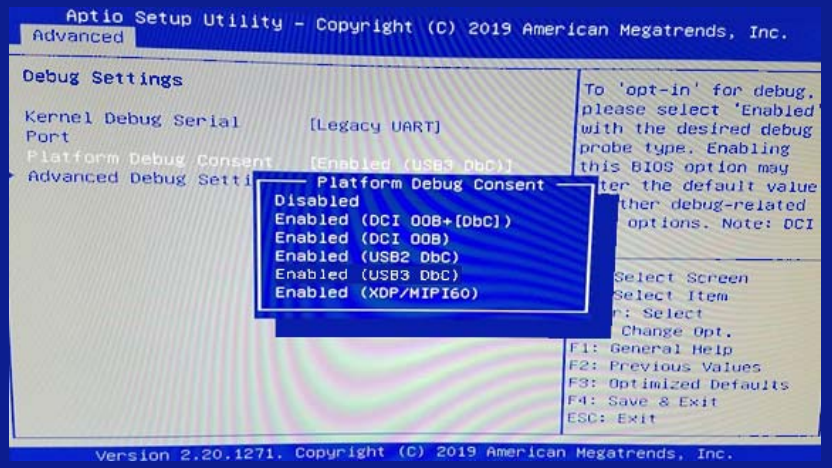
- Defines switching mechanism
  - FSM flow defined
  - Detection of R_p/R_p or R_d/R_d on CC lines
  - Definition of orientation (asymmetric R_p values or removing of one R_d)
  - Definition of maximal charging current (values of different R_p)



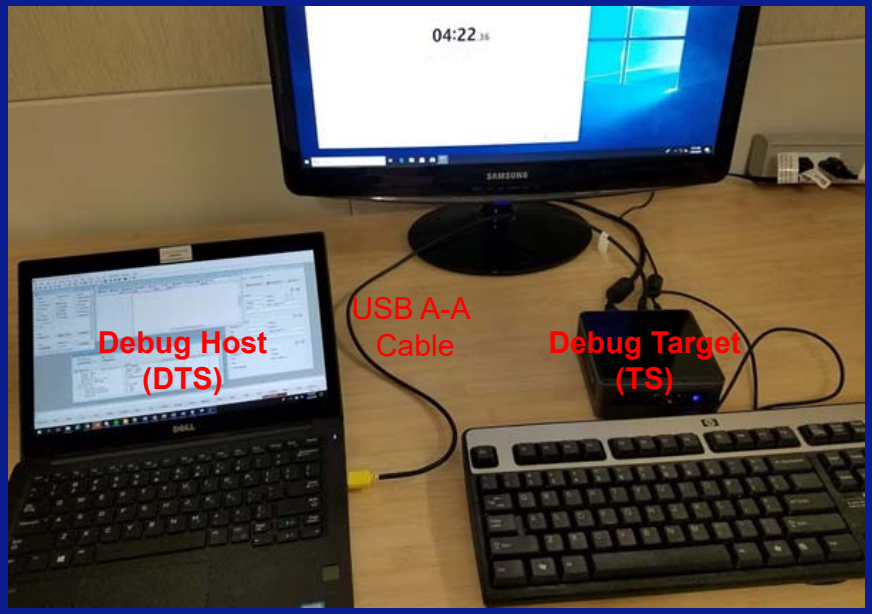
Mode of Operation	CC1	CC2
Default USB Power	Rp3A	Rp1.5A
USB Type-C Current @ 1.5A	Rp1.5A	RpDefault
USB Type-C Current @ 3A	Rp3A	RpDefault

Prepare Platform for Debug





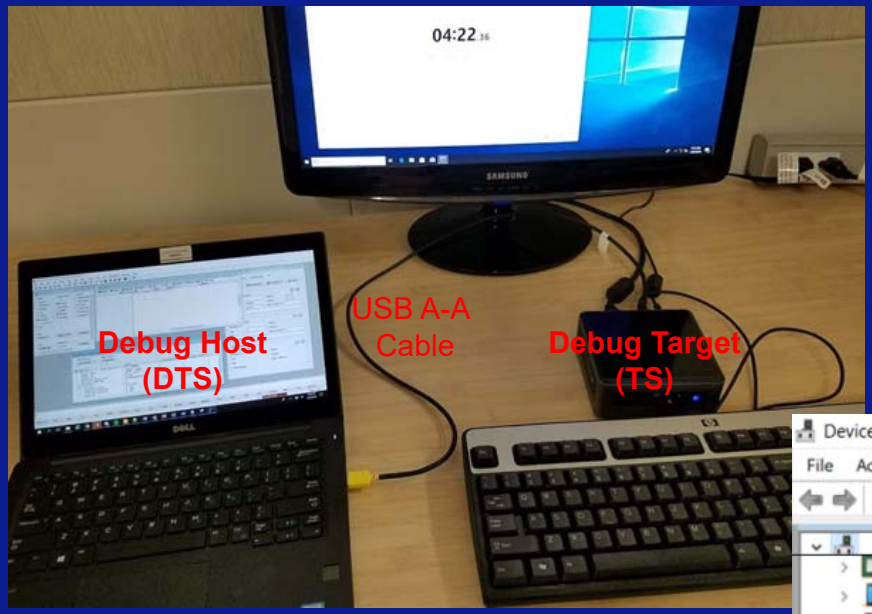
After access is granted (privacy/ security), configure debug infrastructure and enable interface (e.g. via BIOS)



Connect USB Debug Cable

Windows Host DbC enumerates

Debug Endpoints visible to debugger



Connect USB Debug Cable

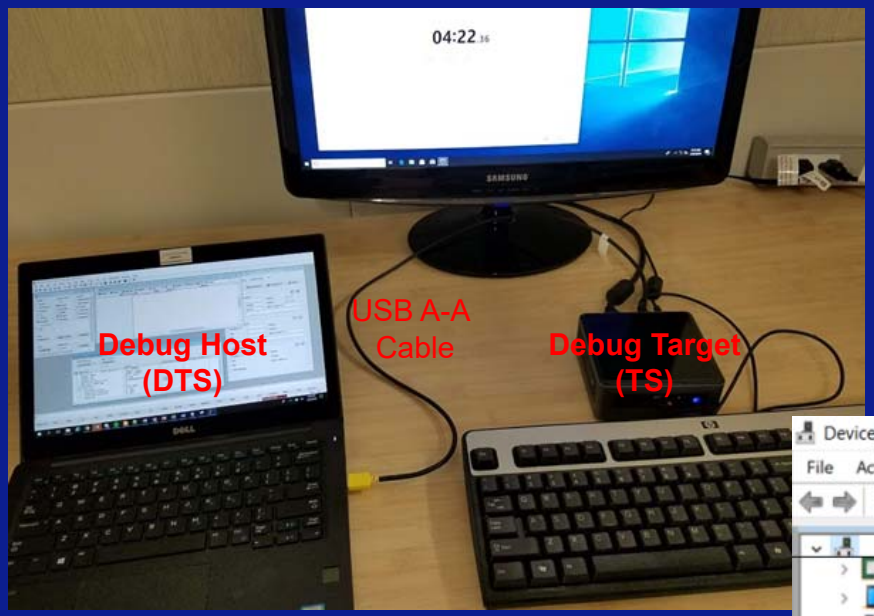
Windows Host DbC enumerates

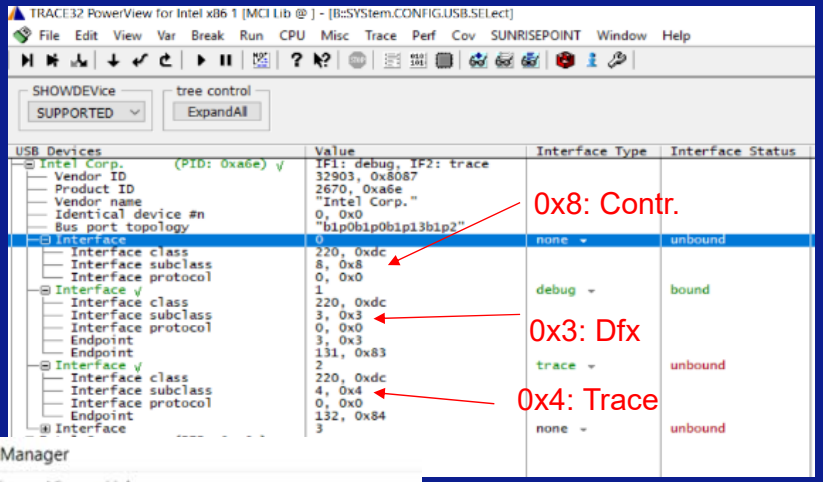
Debug Endpoints visible to debugger

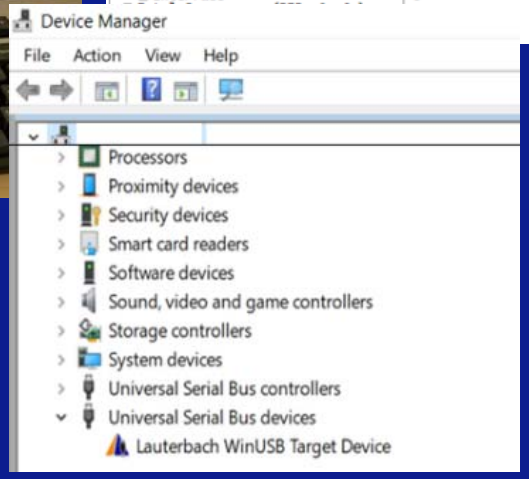
TestConX™

Standardized Closed-Chassis Debug Solution for Platform and System Debug 21

2000-2019 20 YEARS ANNIVERSARY







Connect USB Debug Cable
Windows Host DbC enumerates
Debug Endpoints visible to debugger

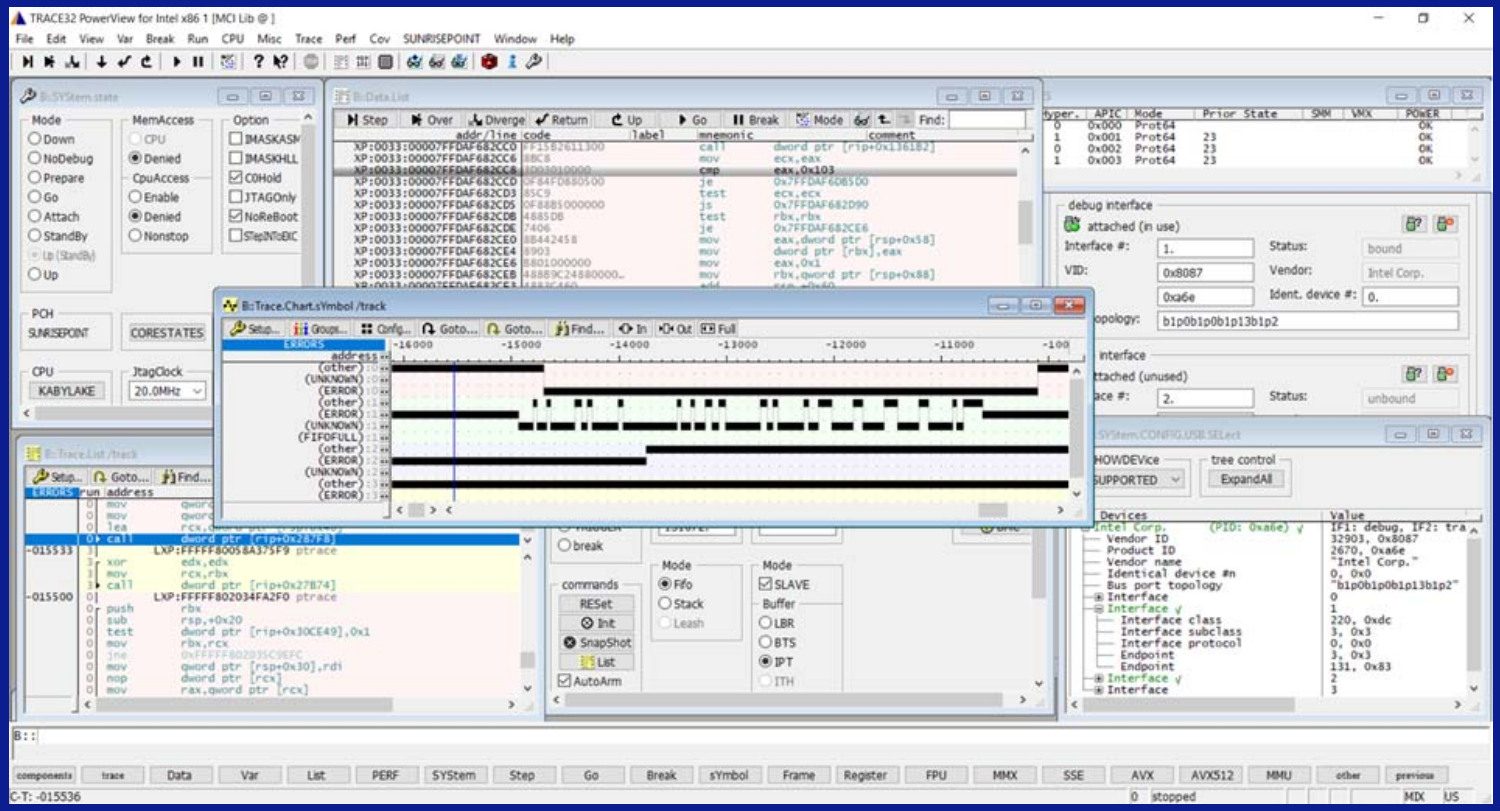
Debug Interface Sub-Class Code	
SC_DbC	0x02
SC_DbC_DFX	0x03
SC_DbC_TRACE	0x04
SC_DvC_GP	0x05
SC_DvC_DFX	0x06
SC_DvC_TRACE	0x07
SC_DEBUG_CONTROL	0x08

TestConX™

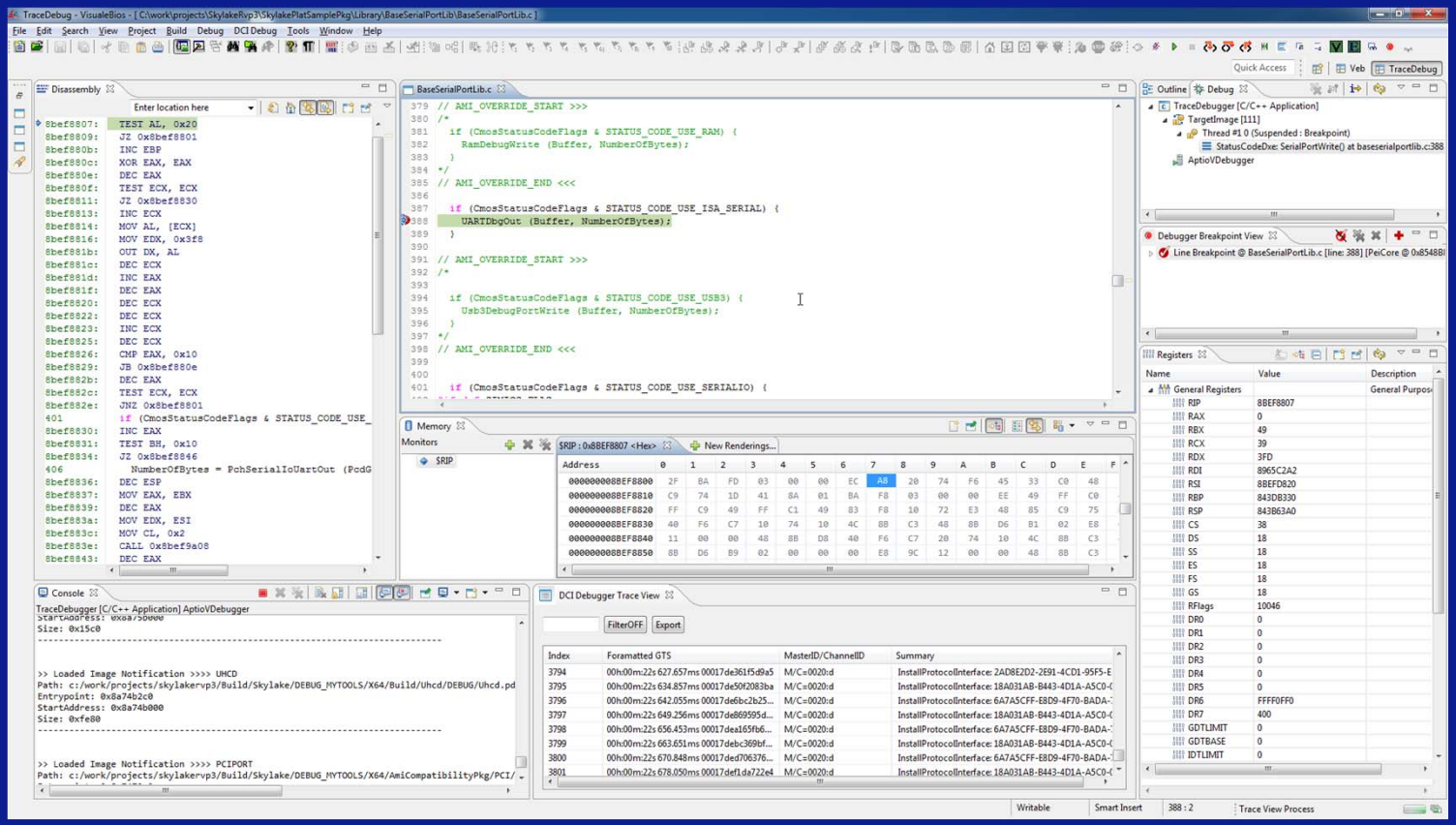
Standardized Closed-Chassis Debug Solution for Platform and System Debug 22

2000-2019 20 YEARS ANNIVERSARY

Ready to Debug



BIOS Debugger



Conclusions

- OEMs demand Closed-Chassis Debug and Test
  - Lower cost and faster Time to Market
  - Use-case focused
- USB Debug provides a comprehensive debug/ test solutions
  - Meets High Bandwidth requirements
  - Can be implemented in Hardware or SW (e.g. to setup TRBs) or a mixture of both
  - Ease of use on functional interfaces

Call to Action

- Adopt, implement and use Closed-Chassis Debug solutions
- Join the MIPI Debug WG
  - Currently participants are SoC, IP, Tools vendors
  - Requires MIPI Contributor membership
  - Telcos every other Tuesday at 9am pacific time
    - 90-120 minutes depending on the agenda
  - Three F2F sessions with all MIPI WGs
    - Generally meet for 4 days

Further Information

- MIPI Debug WG Public Page
- MIPI Architecture Overview for Debug
- MIPI Debug on Wikipedia
- MIPI NIDnT
- USB Debug Class
- USB Type-C Cable and Connector Spec

Sources unless otherwise noted

- MIPI®
  - MIPI Narrow Interface for Debug and Test v1.2
  - MIPI Architecture Overview for Debug v1.2
- USB
  - USB Type-C™ Cable and Connector Specification Revision 1.3
  - USB3.1 Debug Class v1.0
- Wikipedia
  - CC BY-SA 3.0
  - Public domain
  - Creative Commons
  - Wikimedia Commons
- Intel Corporation
- American Megatrends Inc.
- Lauterbach GmbH