LAUTERBACH
DEVELOPMENT TOOLS

TRACE32®
GTM Debug and Trace

1. TRACE32 brief introduction
2. GTM Debug
3. GTM Trace
4. Our role with GTM

AGENDA

AGENDA

1. TRACE32 brief introduction
2. GTM Debug
   1. Debug prerequisites
      1. Hardware interface
      2. define access path
      3. define debug registers
   2. Debug functions
      1. Go/Break/Step
      2. Breakpoints
      3. High-level language debugging
      4. Inline Assembler
      5. Var
      6. Call back
      7. Peripheral View of GTM registers
      8. Peripheral View of Memory
      9. GTM v4.1 new features
3. GTM Trace
   1. Trace prerequisites
      1. Hardware interface
      2. on-chip or off-chip trace
      3. define trace encoder
      4. define trace protocol
      5. define trace path
      6. define trace triggers
      7. how to identify the trace from main core or GTM
   2. Trace functions
      1. Multi Channel Sequencer (MCS)
      2. Advanced Routing Unit (ARU)
      3. I/O Channels (TIM, TOM, ATOM and TIO)
      4. Digital PLL Module (DPLL)
      5. Sensor Pattern Evaluation (SPE)
      6. Time Base Unit (TBU)
4. Our role with GTM

TRACE32 introduction

TRACE32 introduction

Lauterbach GmbH

1979

Company founded by
Lothar Lauterbach

1983

TRACE80 market launch

2009

New headquarter in
Hoehenkirchen/Germany
(near Munich)

2013

Global reach by
10 branch offices and
20+ distribution partners

Today

Well-established
international company
with 130+ employees

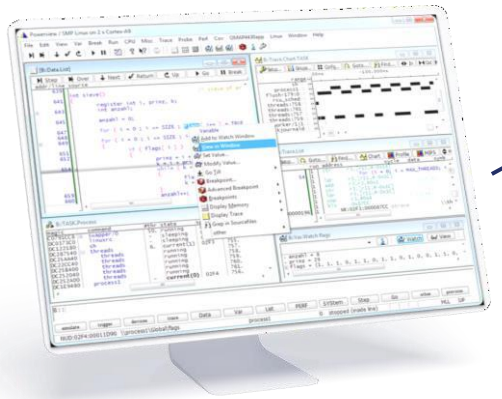
TRACE32 introduction

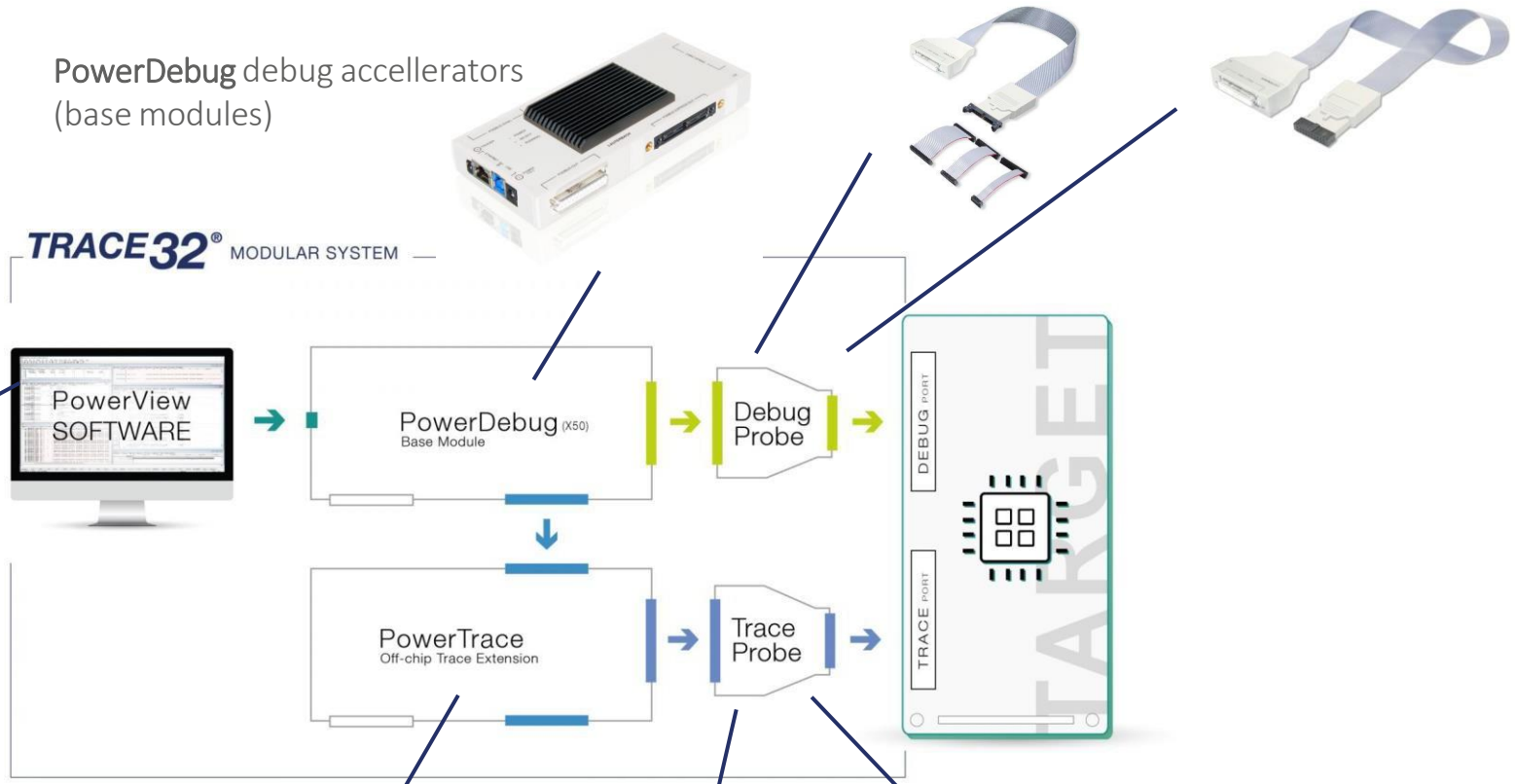
TRACE32® Modular System

Platform-specific
Debug Probes

PowerDebug debug accellerators
(base modules)

PowerView, feature-rich debug
and trace software solution





PowerTrace high-speed trace
extensions (parallel and serial)

Trace Probes, i.e. passive cables, trace
preprocessors (accellerators) and adapters

GTM 4.x China debug & trace open-source group

GTM Debug

Debug prerequisites

Hardware interface

define access path

define debug registers

Debug functions

Go/Break/Step

Breakpoints

High-level language debugging

Inline Assembler

Var

Call back

Peripheral View of GTM registers

Peripheral View of Memory

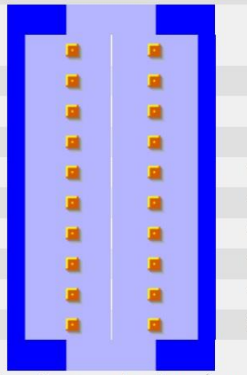
GTM v4.1 new features

GTM Debug

Debug prerequisites

- Hardware interface

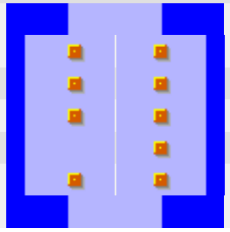
Signal	Pin		Pin	Signal
VREF-DEBUG	1		2	VSUPPLY (not used)
TRST-	3		4	GND
TDI	5		6	GND
TMS|TMSC|SWDIO	7		8	GND
TCK|TCKC|SWCLK	9		10	GND
RTCK	11		12	GND
TDO	13		14	GND
RESET-	15		16	GND
DBGRQ	17		18	GND
DBGACK	19		20	GND



Target connector, top-view

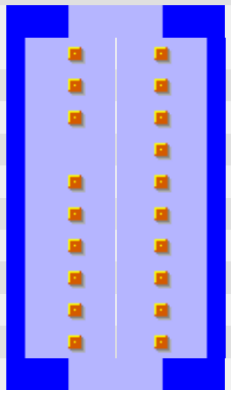
- JTAG-20
  - 2.54mm

Signal	Pin		Pin	Signal
VREF-DEBUG	1		2	TMS|TMSC|SWDIO
GND	3		4	TCK|TCKC|SWCLK
GND	5		6	TDO|- |SWO
GND (KEY)	7		8	TDI
GND	9		10	RESET-



- MIPI-10
  - 1.27 mm

Signal	Pin		Pin	Signal
VREF-DEBUG	1		2	TMS|TMSC|SWDIO
GND	3		4	TCK|TCKC|SWCLK
GND	5		6	TDO|-|SWO
GND (KEY)	7		8	TDI
GND	9		10	RESET-
GND	11		12	TRC CLK
GND	13		14	TRC DATA[0]
GND	15		16	TRC DATA[1]
GND	17		18	TRC DATA[2]
GND	19		20	TRC DATA[3]

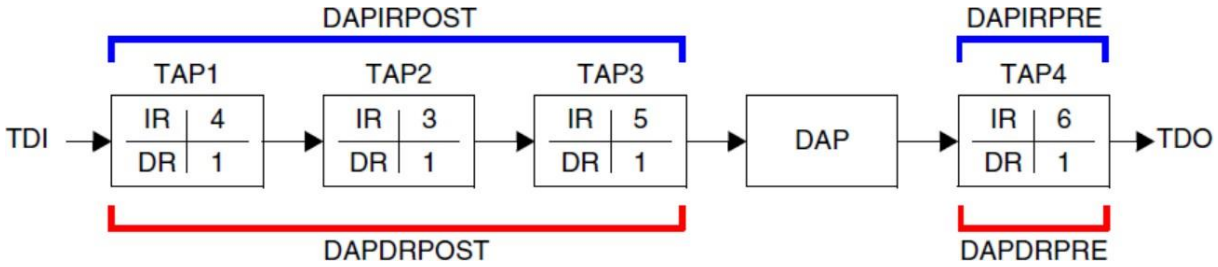


- MIPI-20T
  - 1.27 mm
  - Use JTAG signals only

GTM Debug in China

Debug prerequisites

- define access path
  - Arm CoreSight system access path
    - Romtable
  - JTAG chain access path



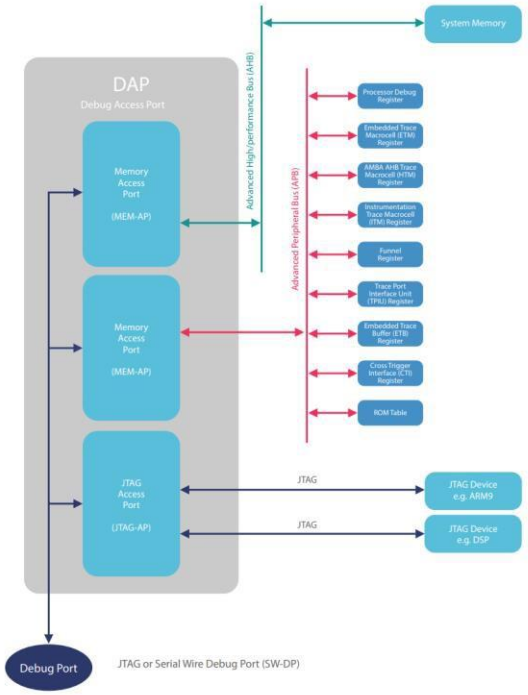


Figure: Debug Access Port (DAP) for SoC-400

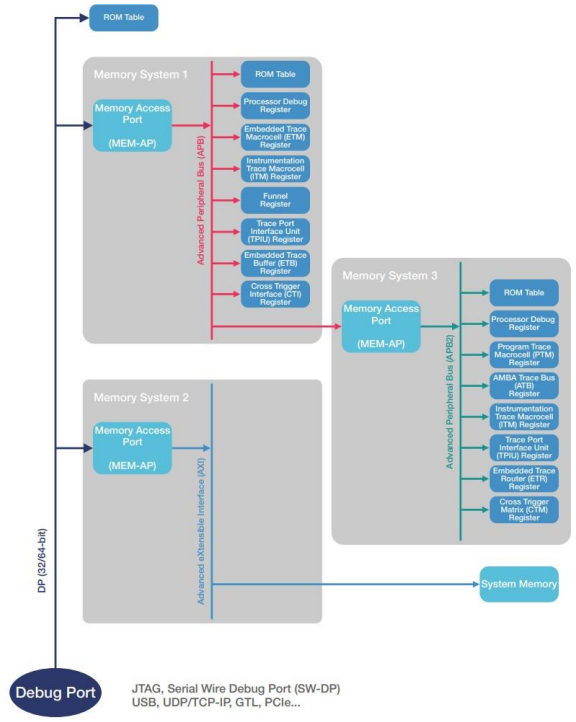
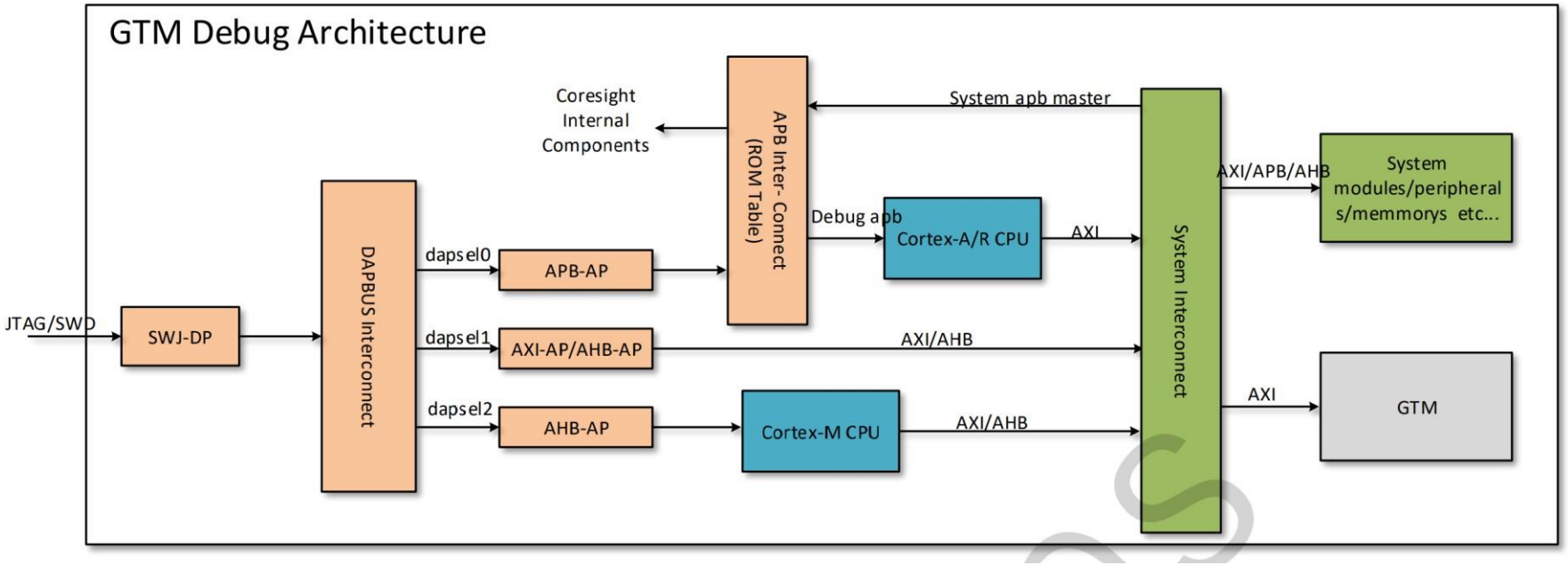


Figure: Debug Access Port (DAP) for SoC-600

GTM Debug Architecture diagram

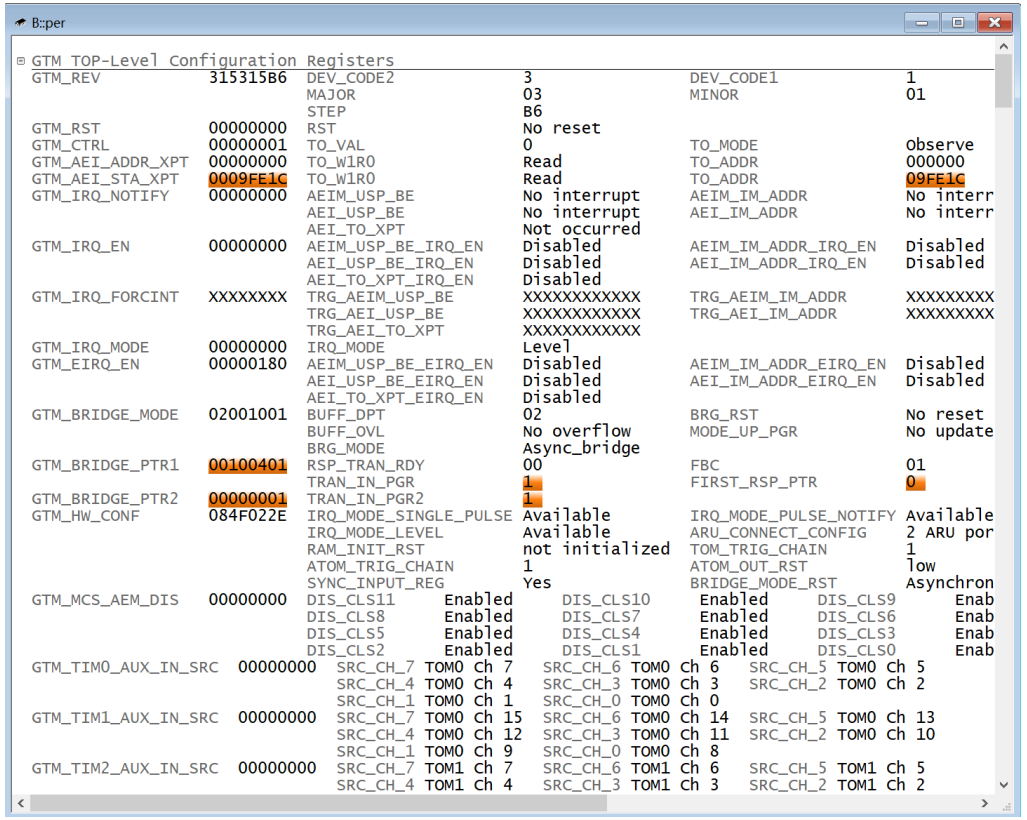
- GTM IP Registers can be accessed via the DAP AXI-AP and other main core (such as: arm)
- GTM Wrapper can be accessed via the APB-AP or AHB-AP or AXI-AP
- GTM CTI can be accessed via the DAP APB-AP



GTM Debug

Debug prerequisites

- define debug registers
  - The necessary debug control register
  - Optional register
  - Chinese customer define DBG register



GTM Wrapper Register

- Sample from XHSC
  - The CTI support up to 32 Trigger Inputs
  - if one customer's SoC has only 4 MCSx
    the CTI0 Trigger Input5—Input12 can only be reserved.

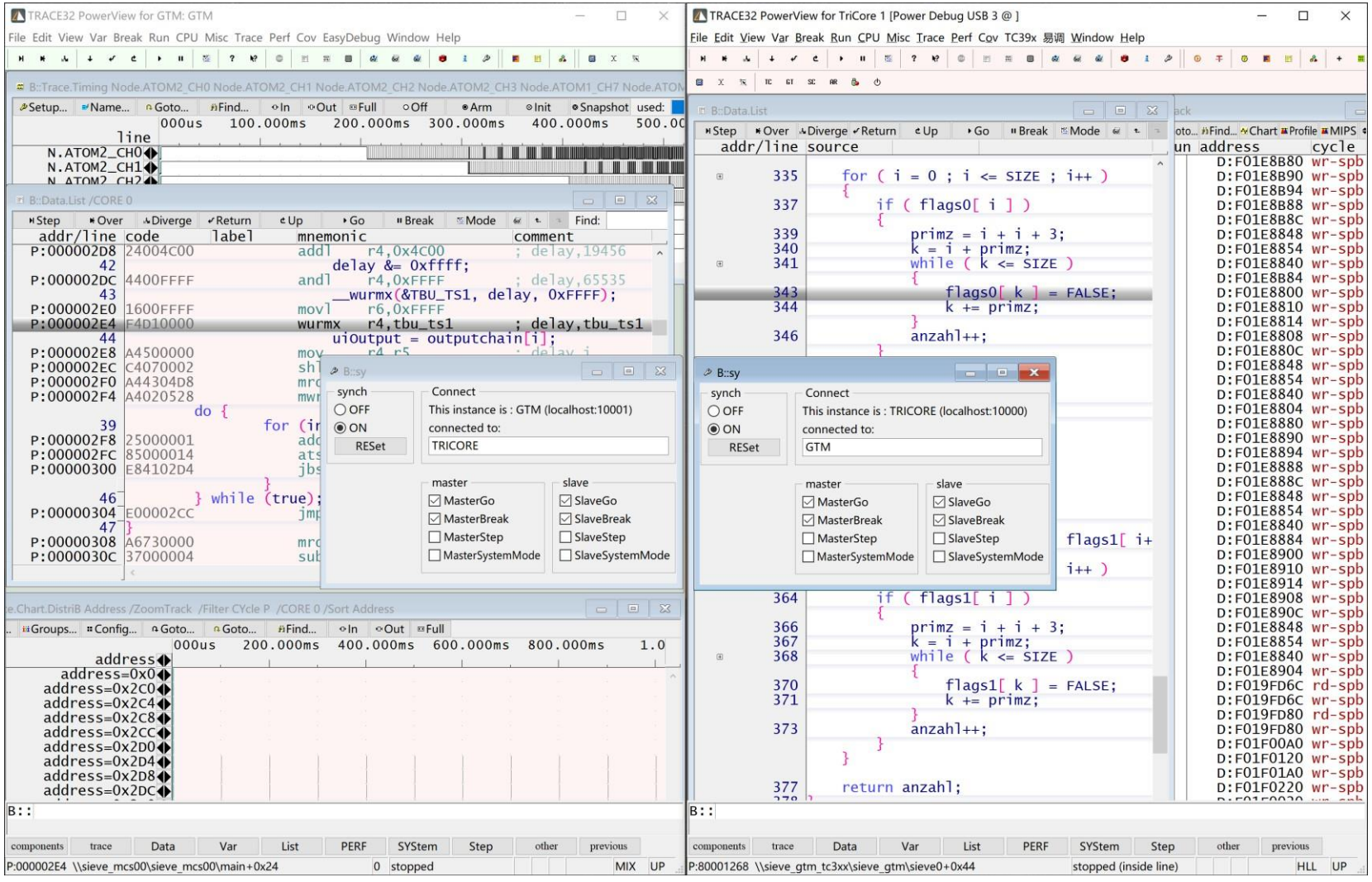
Symbol	Regiser Name
GTMD_GCR	GTM Debug Global Control Register
GTMD_HCR	GTM Debug Halt Control Register
GTMD_HSR	GTM Debug Halt Status Register
GTMC_MCSBPEN	MCS Break Point Enable Register

CTI0 Trigger Input	Connected to
Trigger Input0	GTM halted
Trigger Input1	MCS0 HBP EVENT
Trigger Input2	MCS1 HBP EVENT
Trigger Input3	MCS2 HBP EVENT
Trigger Input4	MCS3 HBP EVENT
Trigger Input5	Reserved
Trigger Input6	Reserved
Trigger Input7	Reserved
Trigger Input8	Reserved
Trigger Input9	Reserved
Trigger Input10	Reserved
Trigger Input11	Reserved
Trigger Input12	Reserved
CTI0 Trigger Output	Connected to
Trigger Output0	GTM Debug Request
Trigger Output1	GTM Restart
Trigger Output2	
Trigger Output3	

GTM Debug

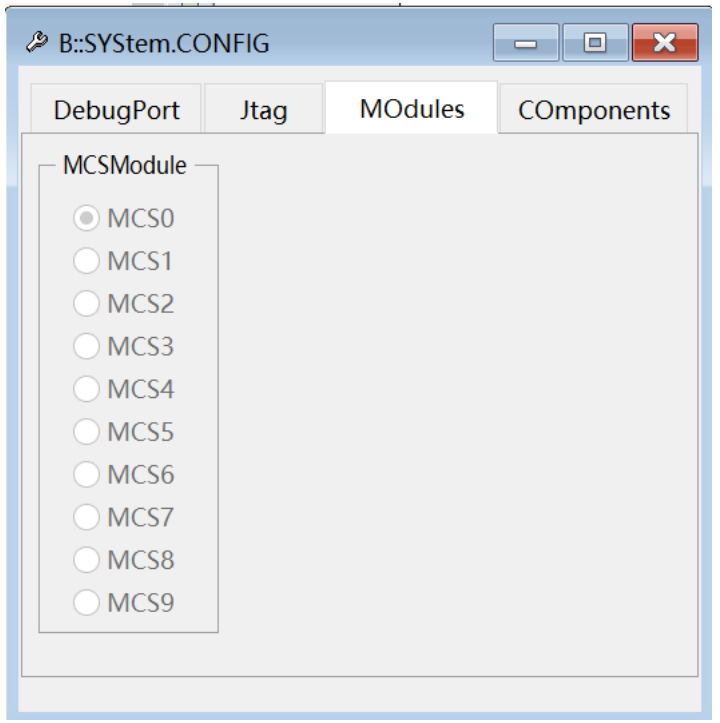
Debug Functions

- Go/Break/Step
- Multi core Debug
- SYNC



GTM Debug

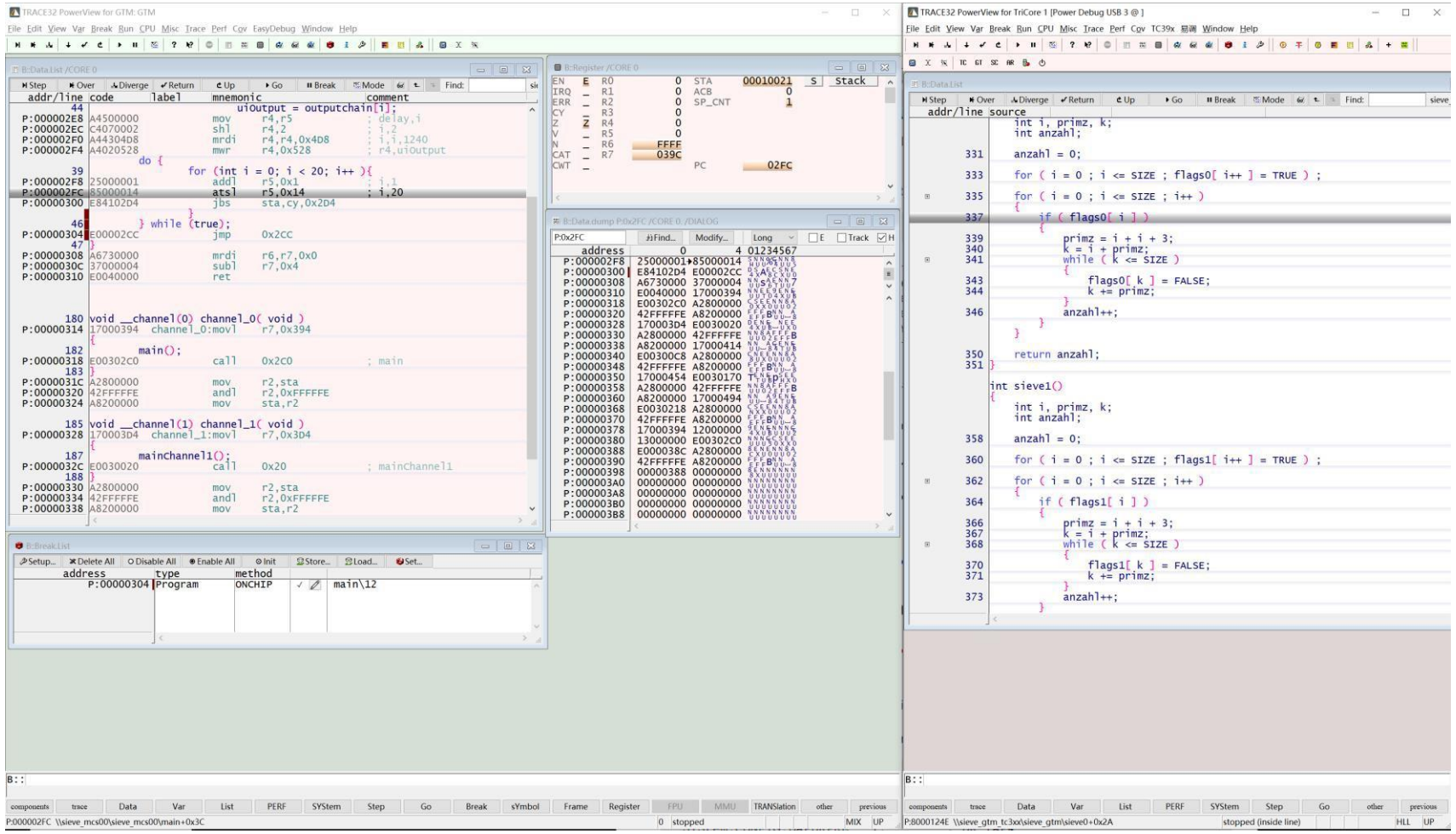
- MCS Module select
  - SYStem.CONFIG.MCSModule MCS
  - Support AMP Multi MCS



GTM Debug

Debug Functions

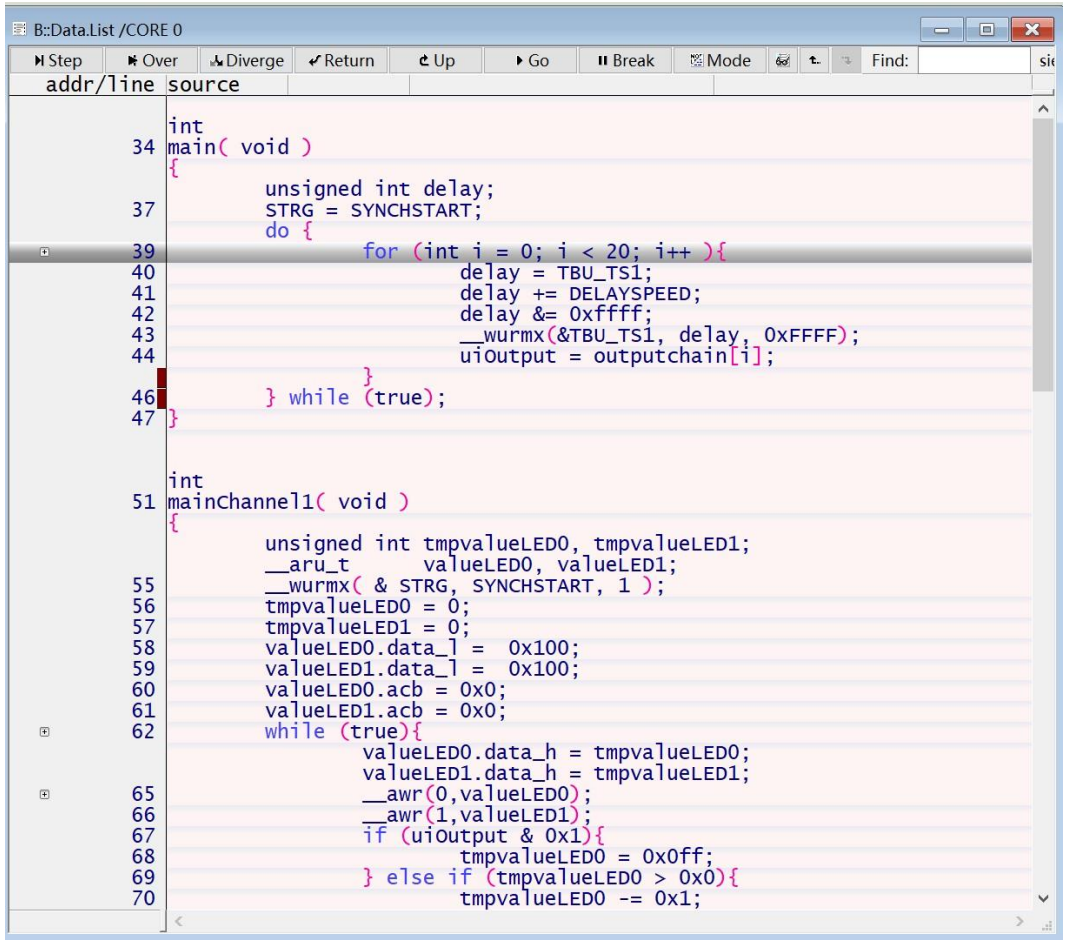
- Breakpoints



GTM Debug

Debug Functions

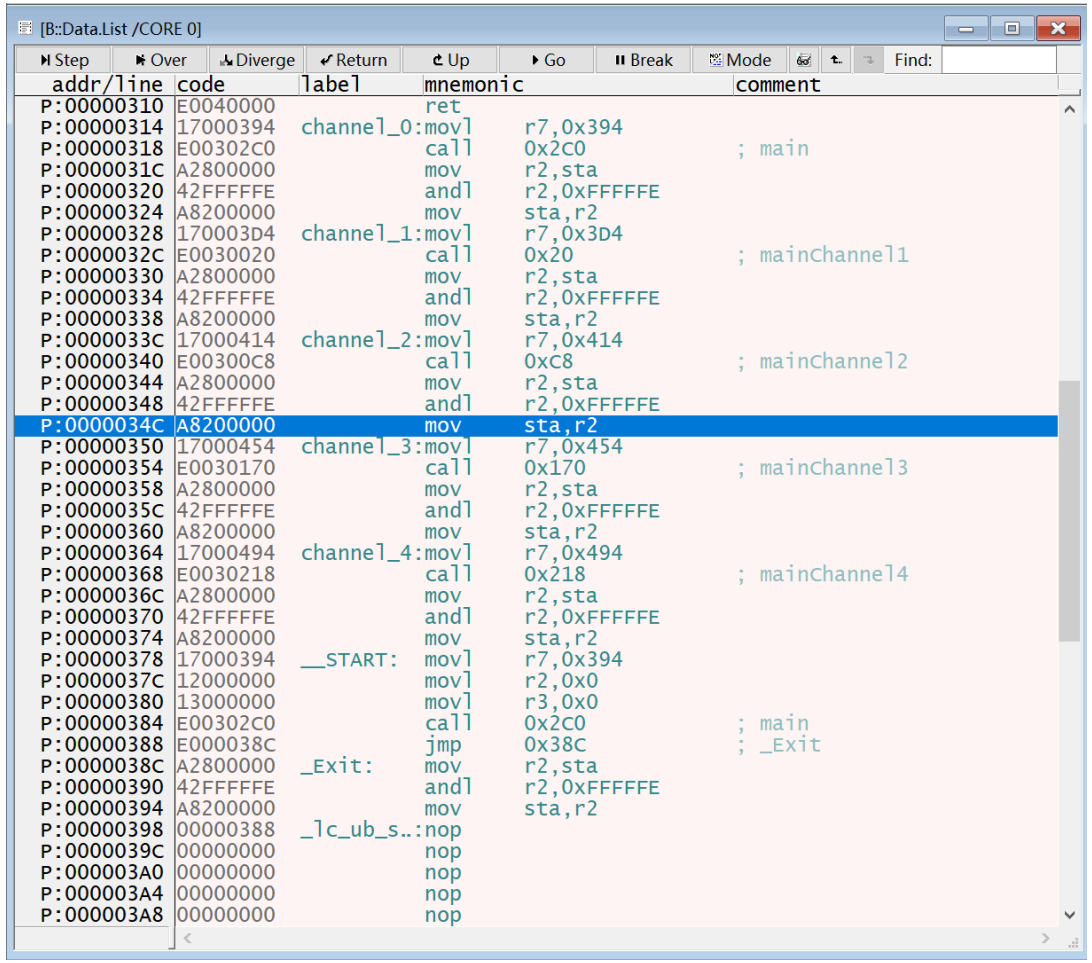
- High-level language debugging



GTM Debug

Debug Functions

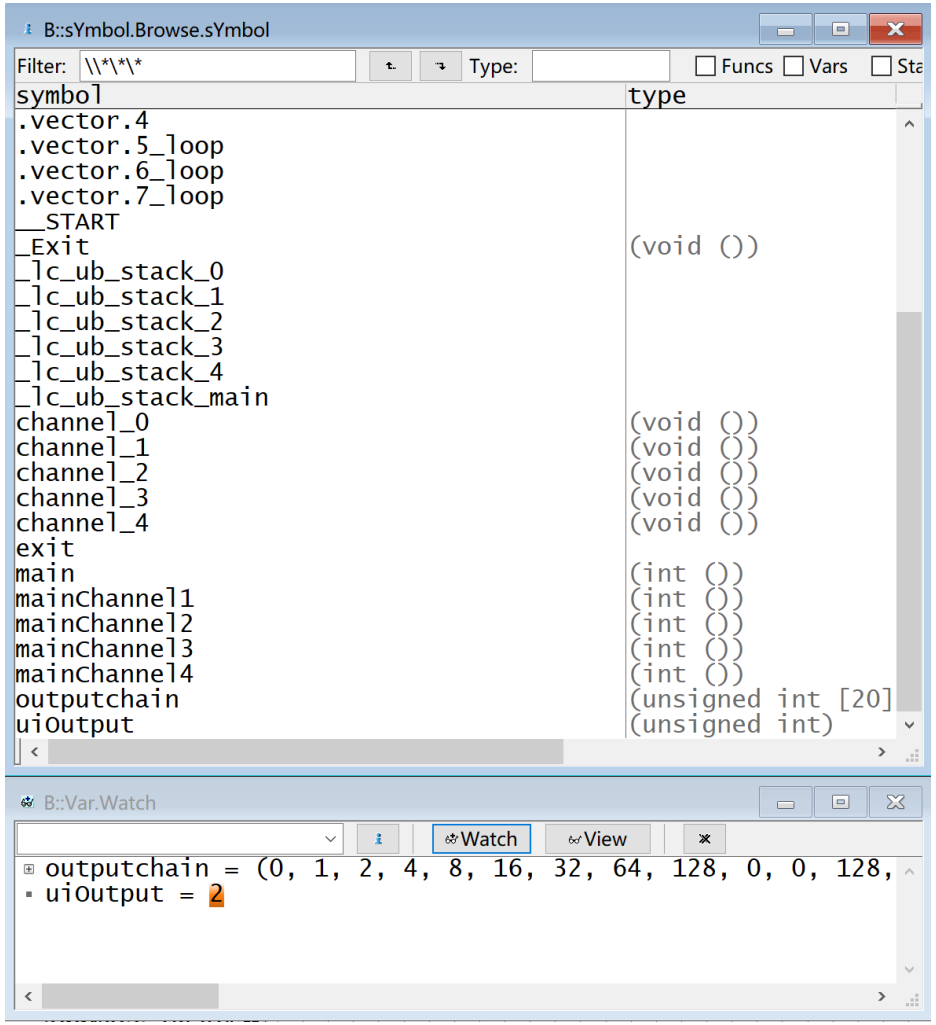
- Inline Assembler



GTM Debug

Debug Functions

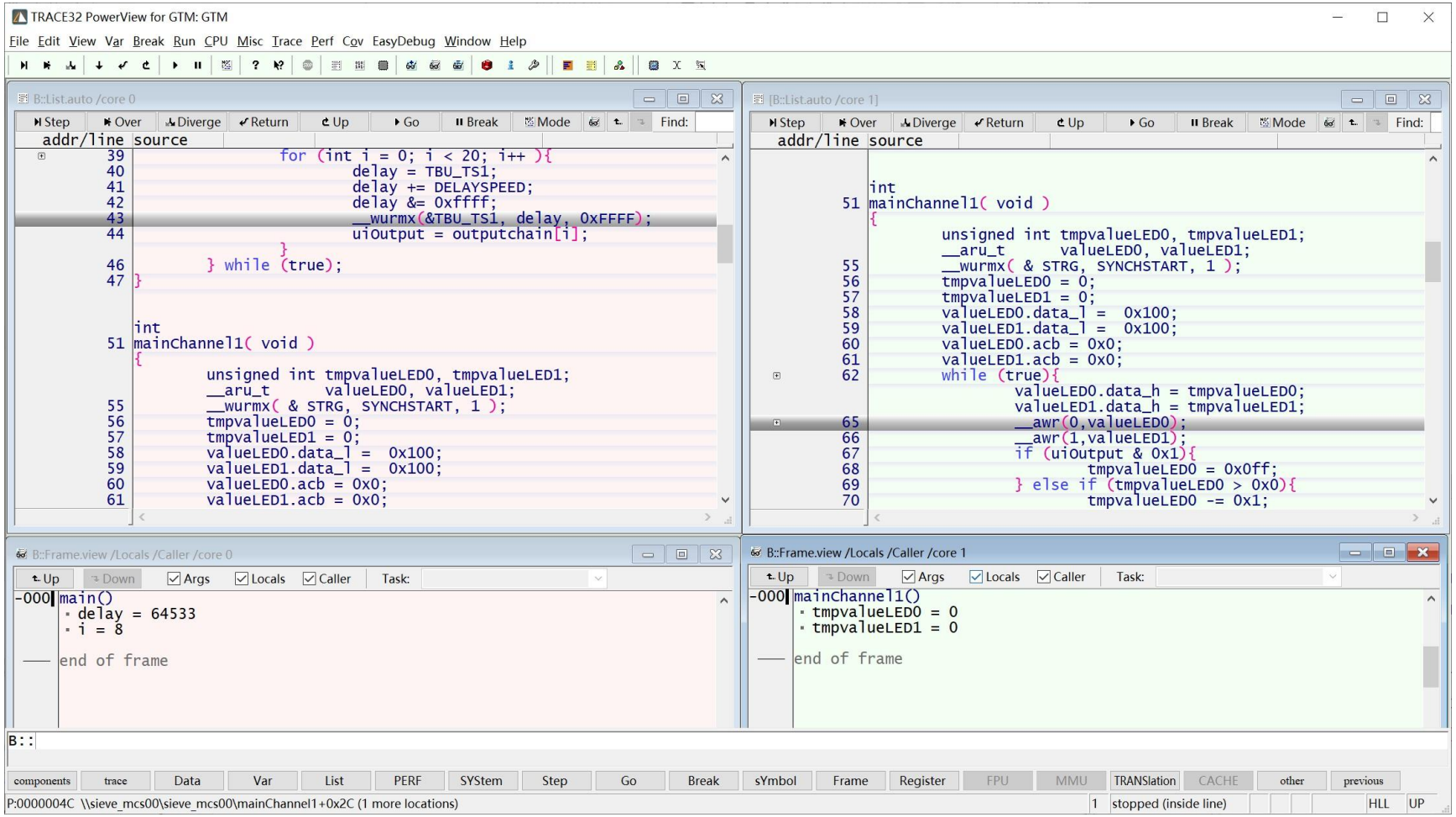
- Variable View (run time access)
  - SYStem.option Dualport on



GTM Debug

Debug Functions

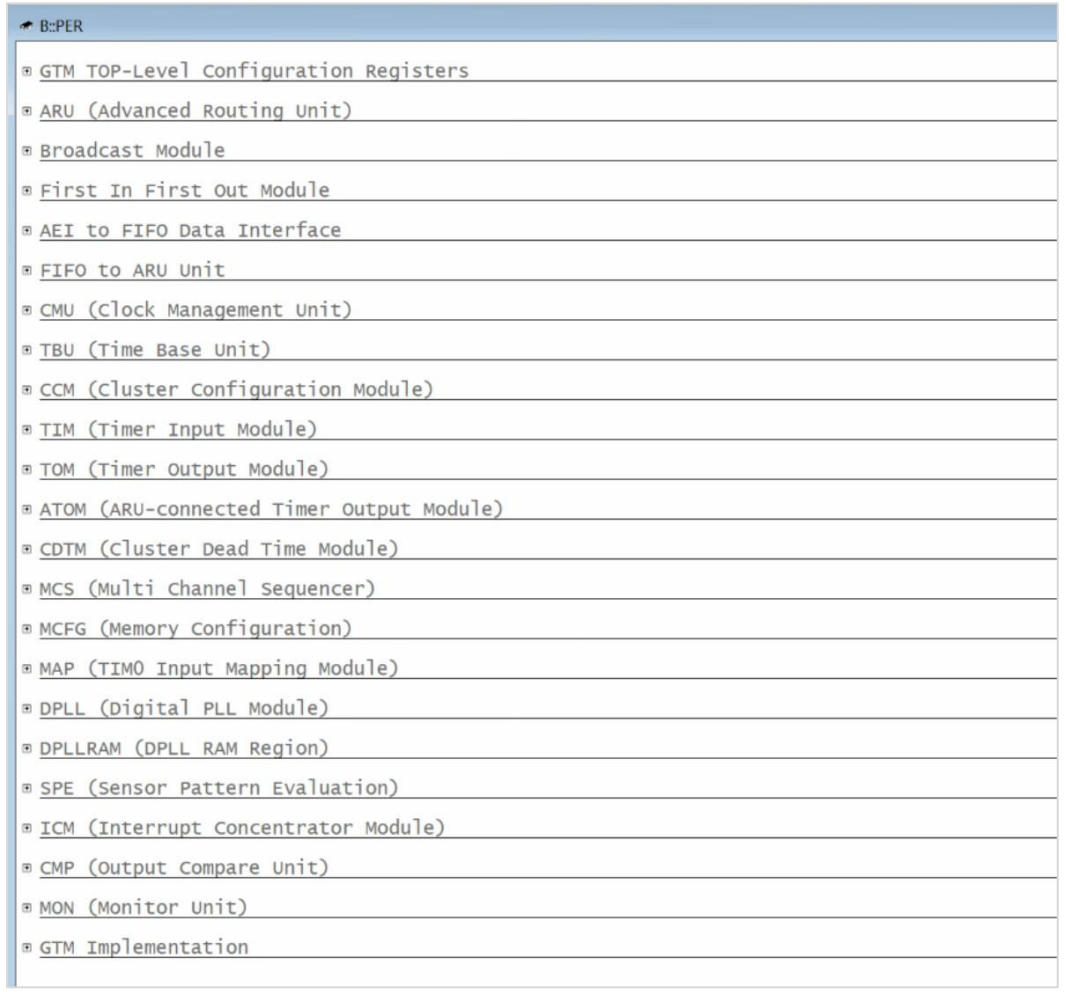
- Call back



GTM Debug

Debug Functions

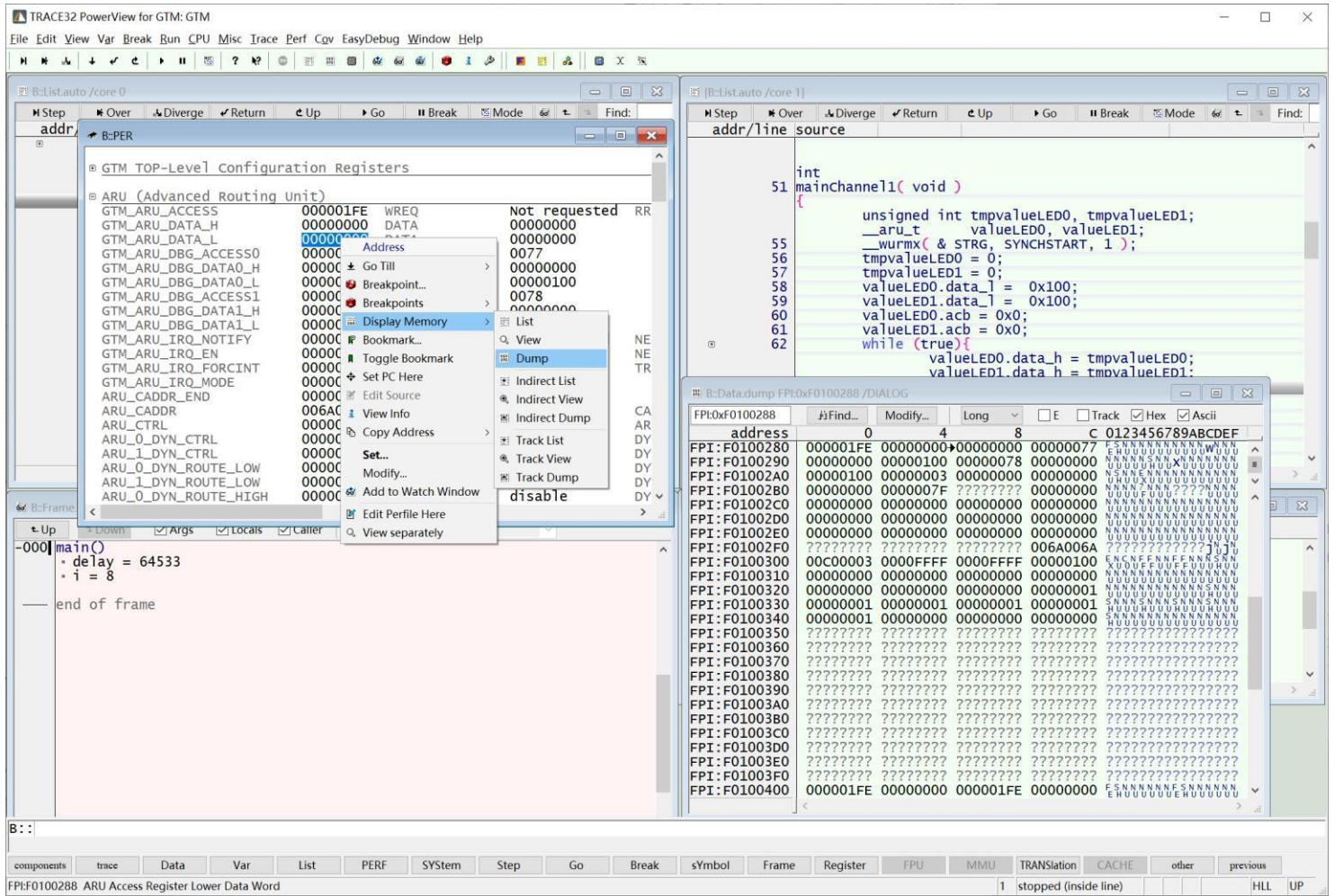
- Peripheral View of GTM registers



GTM Debug

Debug Functions

- Peripheral View of Memory



GTM Debug

Debug Functions

- GTM v4.1 new features
  - New Hardware Breakpoint Unit
  - Two Breakpoints (break before make) for every MCS Module
  - Used for Single Stepping or Breakpoints
  - Breakpoints on Program Address or Range
  - Breakpoints on Data Address or Range; Read or Write Access
  - Breakpoint on Data Address or Range and Data Value
  - Channel Selection possible

Virtual target support

- COSEDA virtual target has an adaptor via the MCD interface to the virtual GTM

MCD

The Multi-Core Debug API (MCD) is an interface between software development tools and simulated or real systems with multi-core SoCs. Lauterbach, together with other renowned companies, defined this interface. Start-ups to market leaders are using this API to connect their virtual or emulated target to our PowerView GUI.

Read more

GDB

Utilizing the Remote Serial Protocol (RSP), TRACE32 can connect to any GDB server implementation. Beyond Linux application and kernel debugging, TRACE32 PowerView can also control many simulator products like QEMU.

Vendor specific Interfaces (CADI, IRIS, ARCINT, etc.)

TRACE32 is the market-leading tool. TRACE32 supports vendor-specific interfaces of processor models and simulators from most providers. You will have your choice of vendors and models.

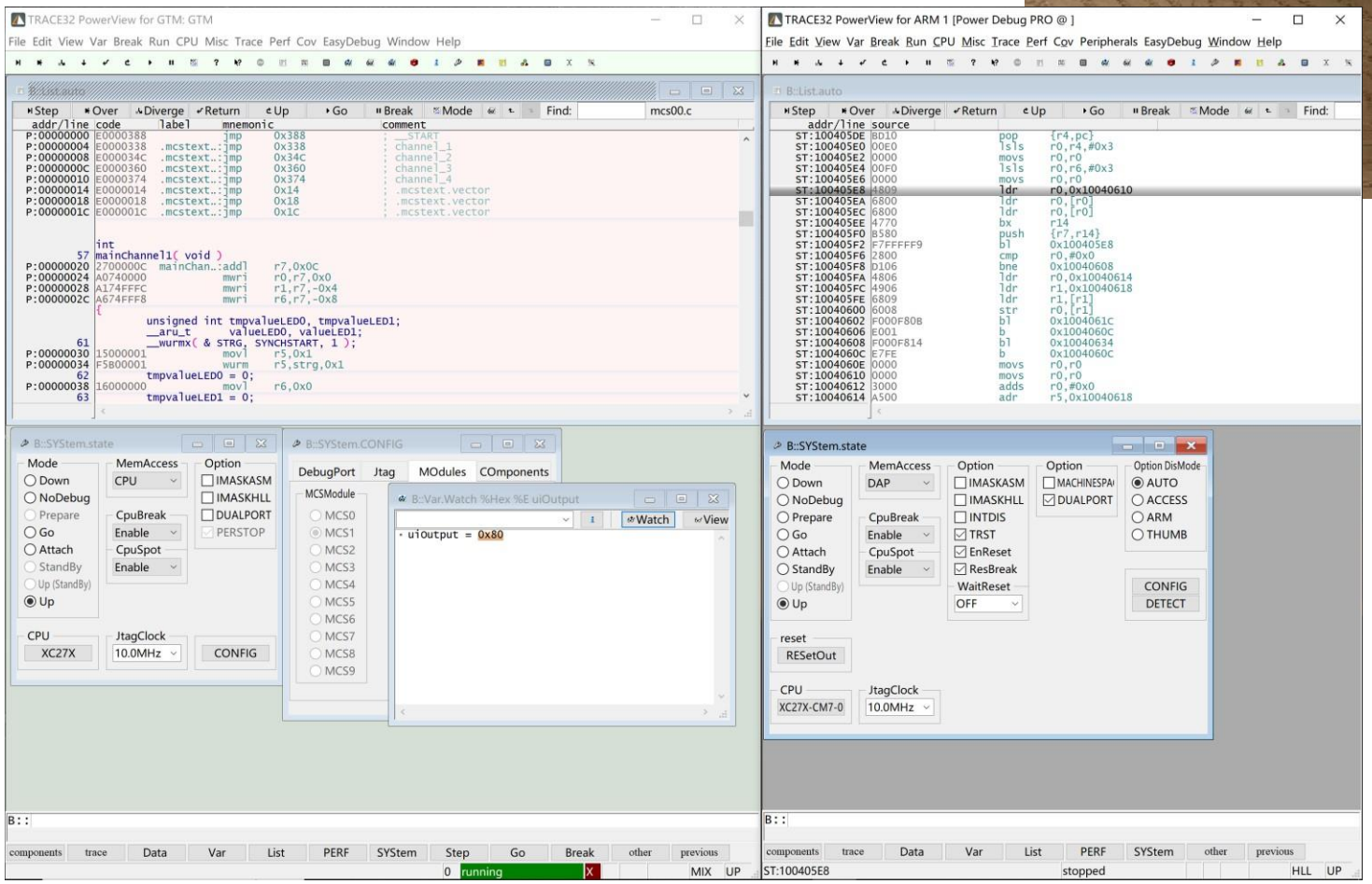
GTL

The Generic Transactor Library (GTL) is an API designed by Lauterbach. It connects the transactors of an emulation system to TRACE32's complete debug and trace software stack. The API provides interfaces for trace recording, bus level, and signal level transactors. This allows you to make trade-offs between detail and execution speed.

Read More

Live Demo

- Demo show with XHSC board



GTM Trace

Trace prerequisites

Hardware interface

on-chip or off-chip trace

define trace encoder

define trace protocol

define trace path

define trace triggers

how to identify the trace from main core or GTM

Trace functions

Multi Channel Sequencer (MCS)

Advanced Routing Unit (ARU)

I/O Channels (TIM, TOM, ATOM and TIO)

Digital PLL Module (DPLL)

Sensor Pattern Evaluation (SPE)

Time Base Unit (TBU)

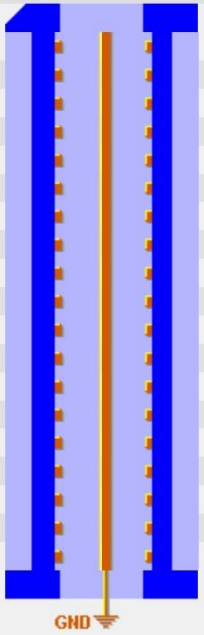
GTM Trace

Trace prerequisites

- Hardware interface

Connector 1

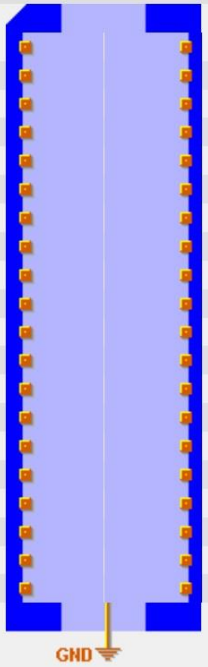
Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
GND	5	6	TRACECLK
DBGRQ	7	8	DBGACK
RESET-	9	10	EXTRIG
TDO	11	12	VREF-TRACE
RTCK	13	14	VREF-DEBUG
TCK|TCKC|SWCLK	15	16	TRACEDATA7
TMS|TMSC|SWDIO	17	18	TRACEDATA6
TDI	19	20	TRACEDATA5
TRST-	21	22	TRACEDATA4
TRACEDATA15	23	24	TRACEDATA3
TRACEDATA14	25	26	TRACEDATA2
TRACEDATA13	27	28	TRACEDATA1
TRACEDATA12	29	30	GND
TRACEDATA11	31	32	GND
TRACEDATA10	33	34	VCC
TRACEDATA9	35	36	TRACECTL
TRACEDATA8	37	38	TRACEDATA0



Connect Pin 39,40,41,42,43 to GND

■ Pinout of 40-pin Samtec Connector

Signal	Pin	Pin	Signal
TXP[4]	1	2	VREF-DEBUG
TXN[4]	3	4	TCK|TCKC|SWCLK
GND	5	6	GND
TXP[2]	7	8	TMS|TMSC|SWDIO
TXN[2]	9	10	TRST-
GND	11	12	GND
TXP[0]	13	14	TDI
TXN[0]	15	16	TDO
GND	17	18	GND
CLKP	19	20	RESET-
CLKN	21	22	DBGRQ
GND	23	24	GND
TXP[1]	25	26	DBGACK
TXN[1]	27	28	RTCK
GND	29	30	GND
TXP[3]	31	32	TRIGIN
TXN[3]	33	34	TRIGOUT
GND	35	36	RESERVED
TXP[5]	37	38	RESERVED
TXN[5]	39	40	RESERVED



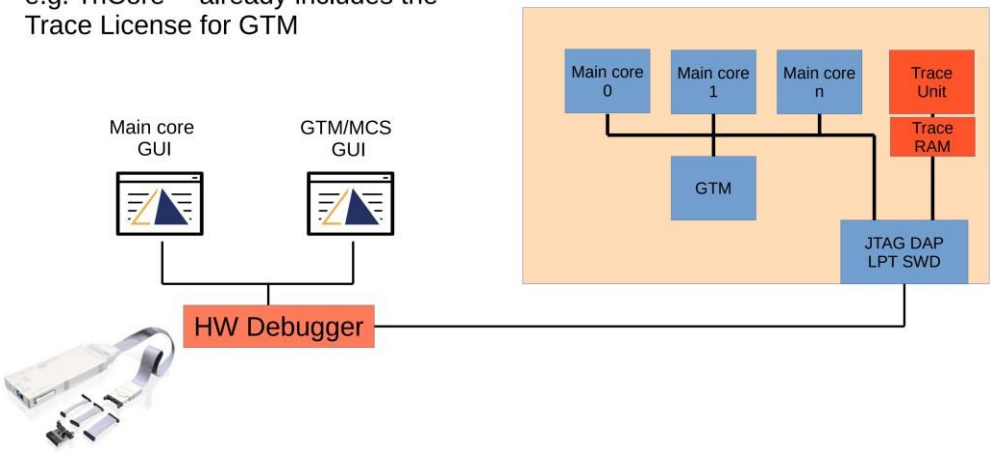
Target connector, top-view, connect the two latch pins to GND

GTM Trace

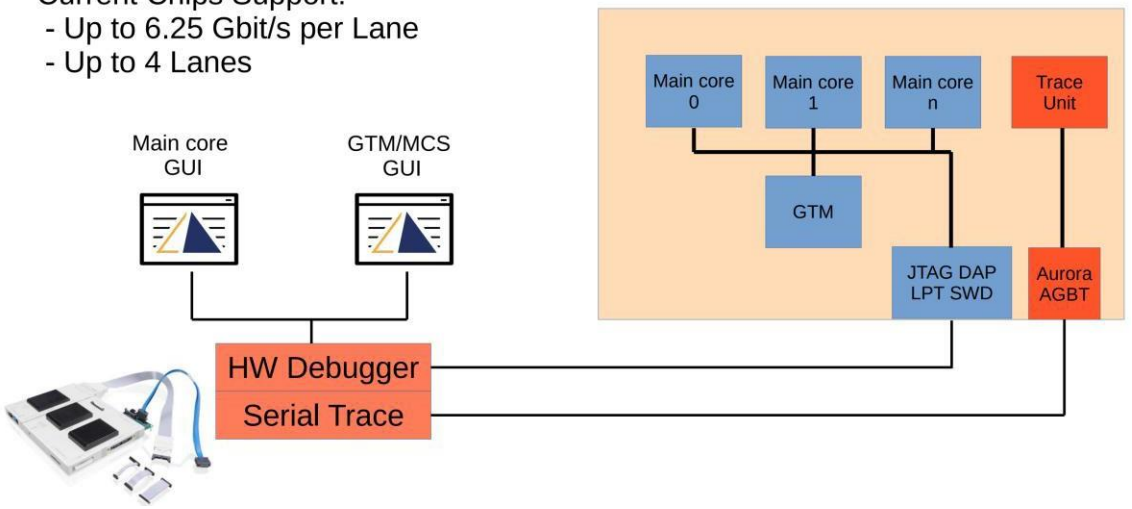
Trace prerequisites

- on-chip or off-chip trace

The Trace License for the main core
e.g. TriCore™ already includes the
Trace License for GTM



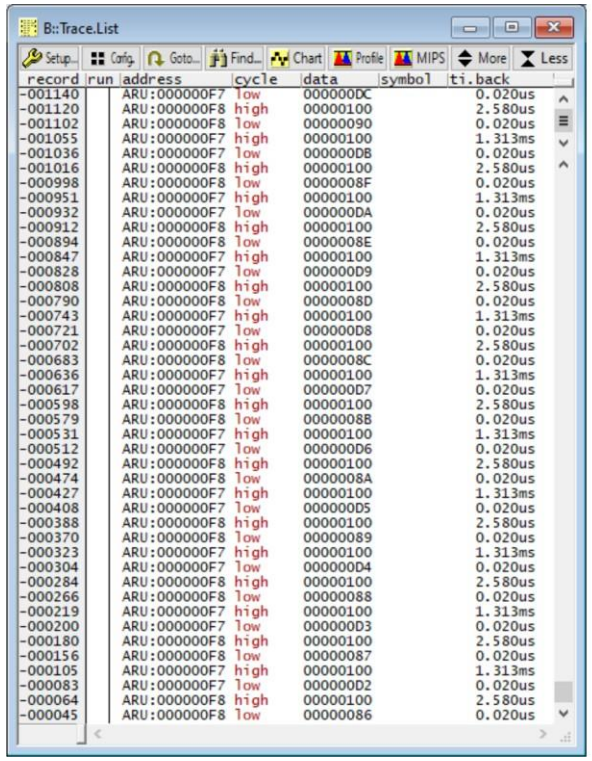
Aurora GigaBit Trace
Current Chips Support:
- Up to 6.25 Gbit/s per Lane
- Up to 4 Lanes

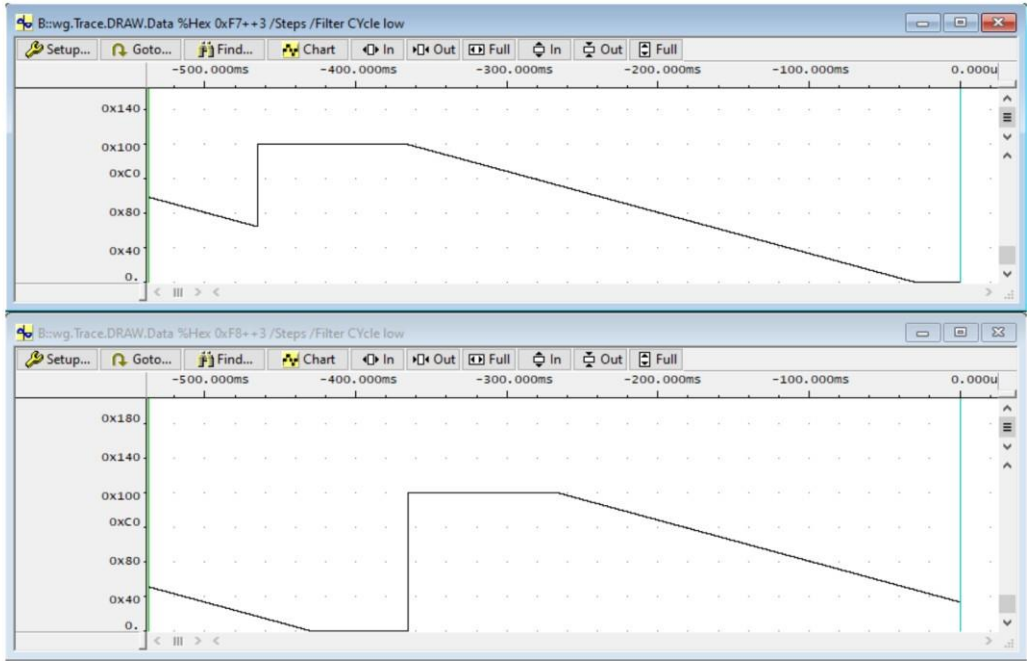


GTM Trace

Trace Functions

- Advanced Routing Unit (ARU)

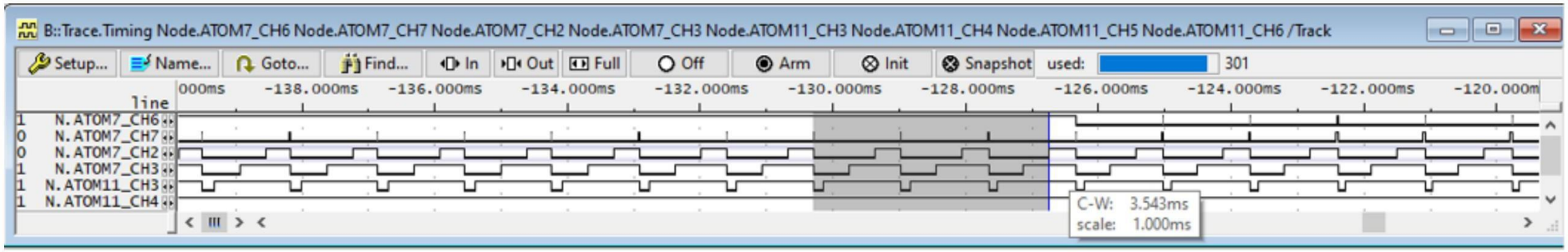


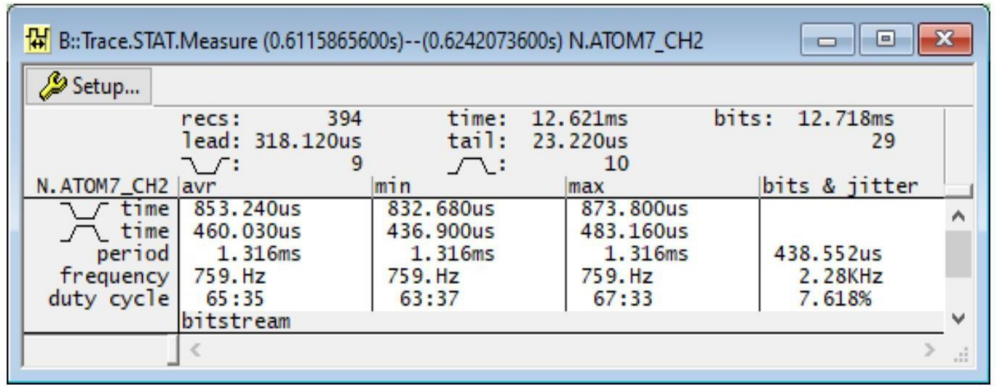


GTM Trace

Trace Functions

- I/O Channels (TIM, TOM, ATOM and TIO)

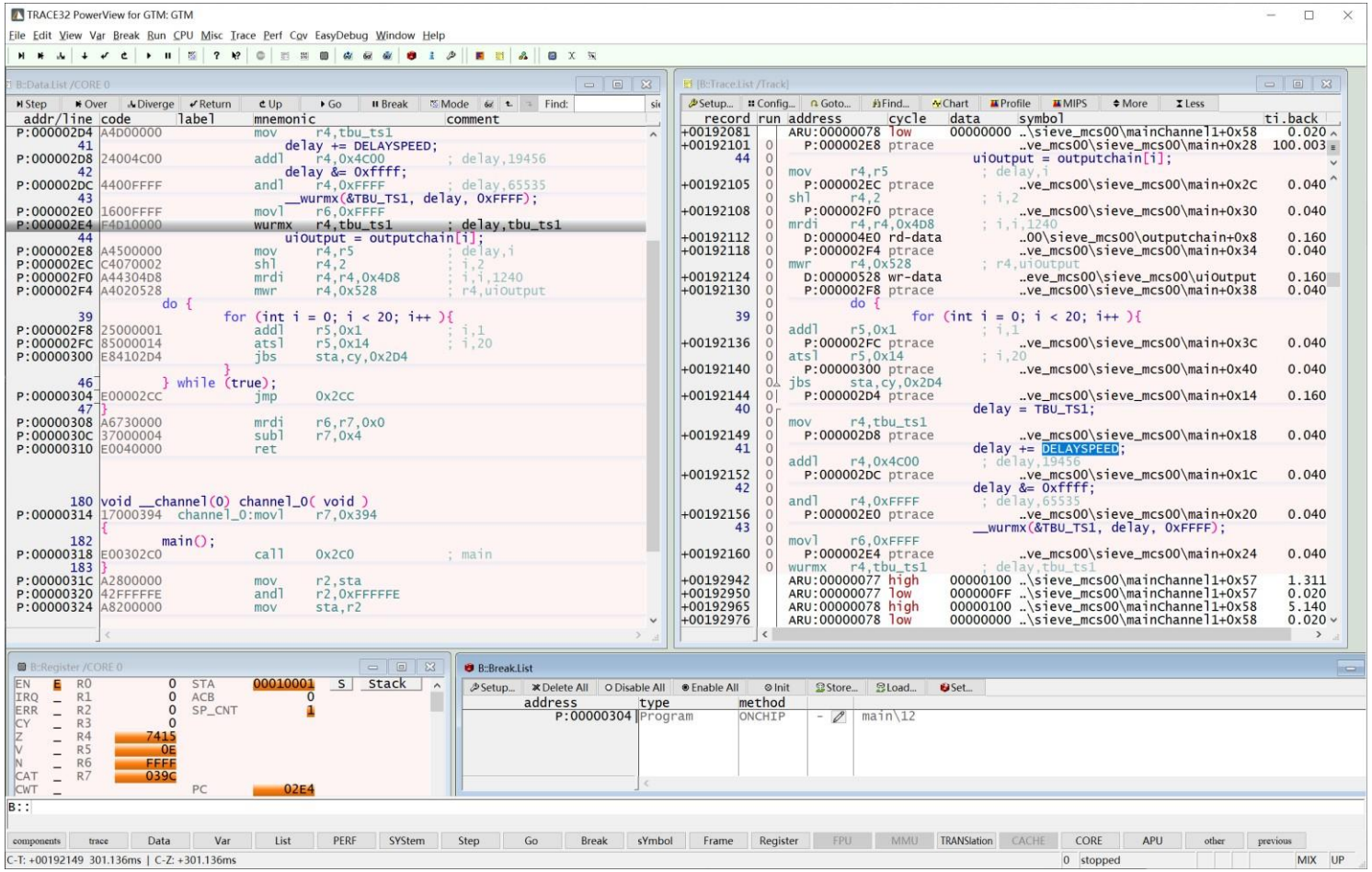




Logic analyzer like measurements

MCS Trace flow

- MCS Ptrace record

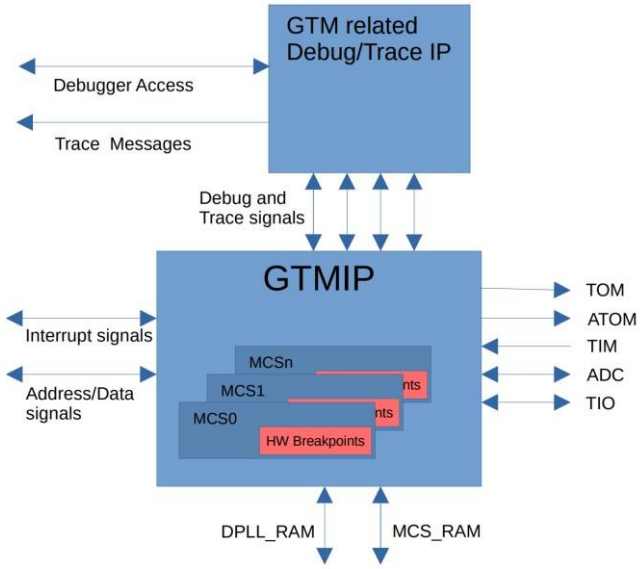


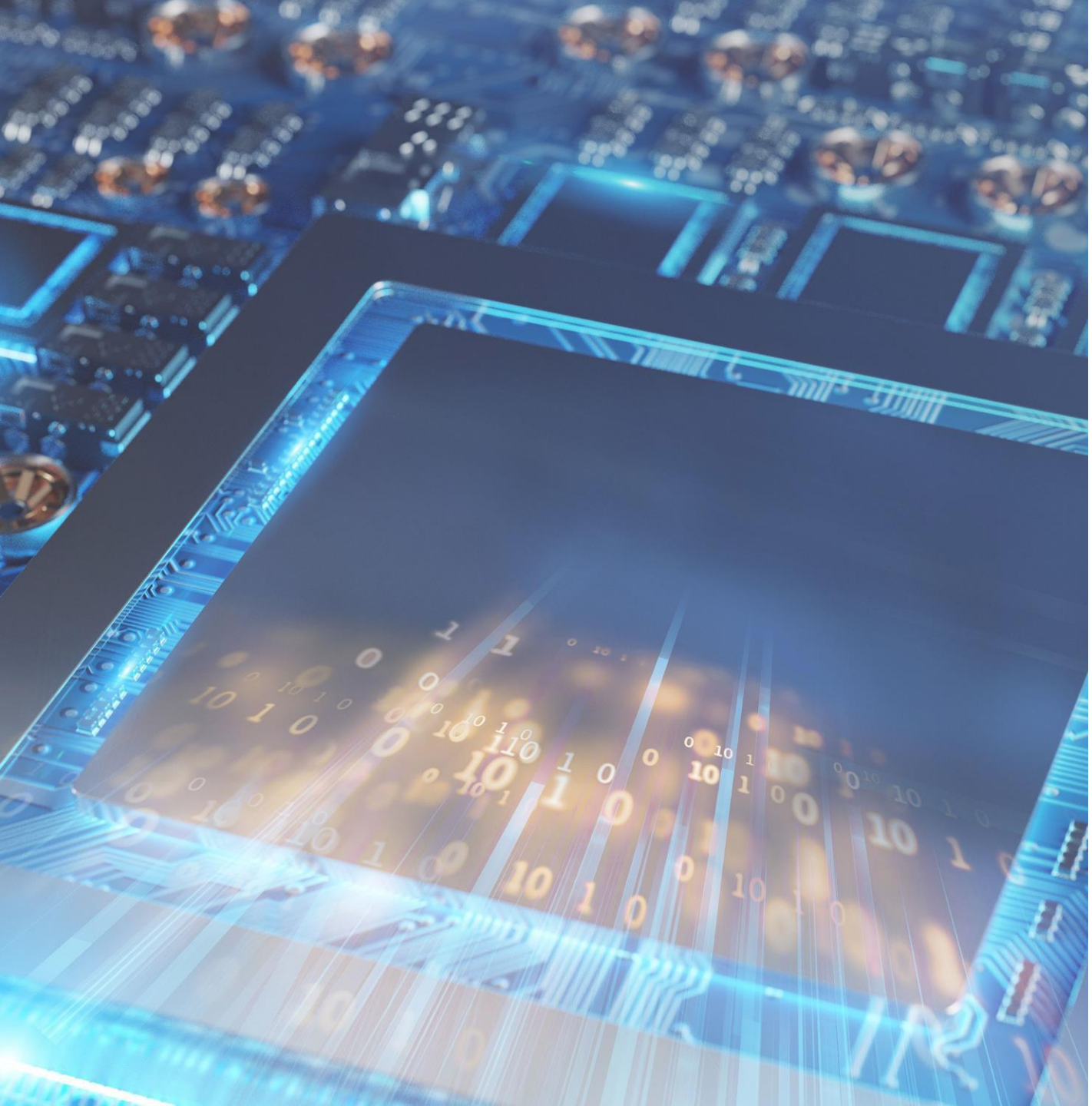
Our role with GTM

Our role with GTM

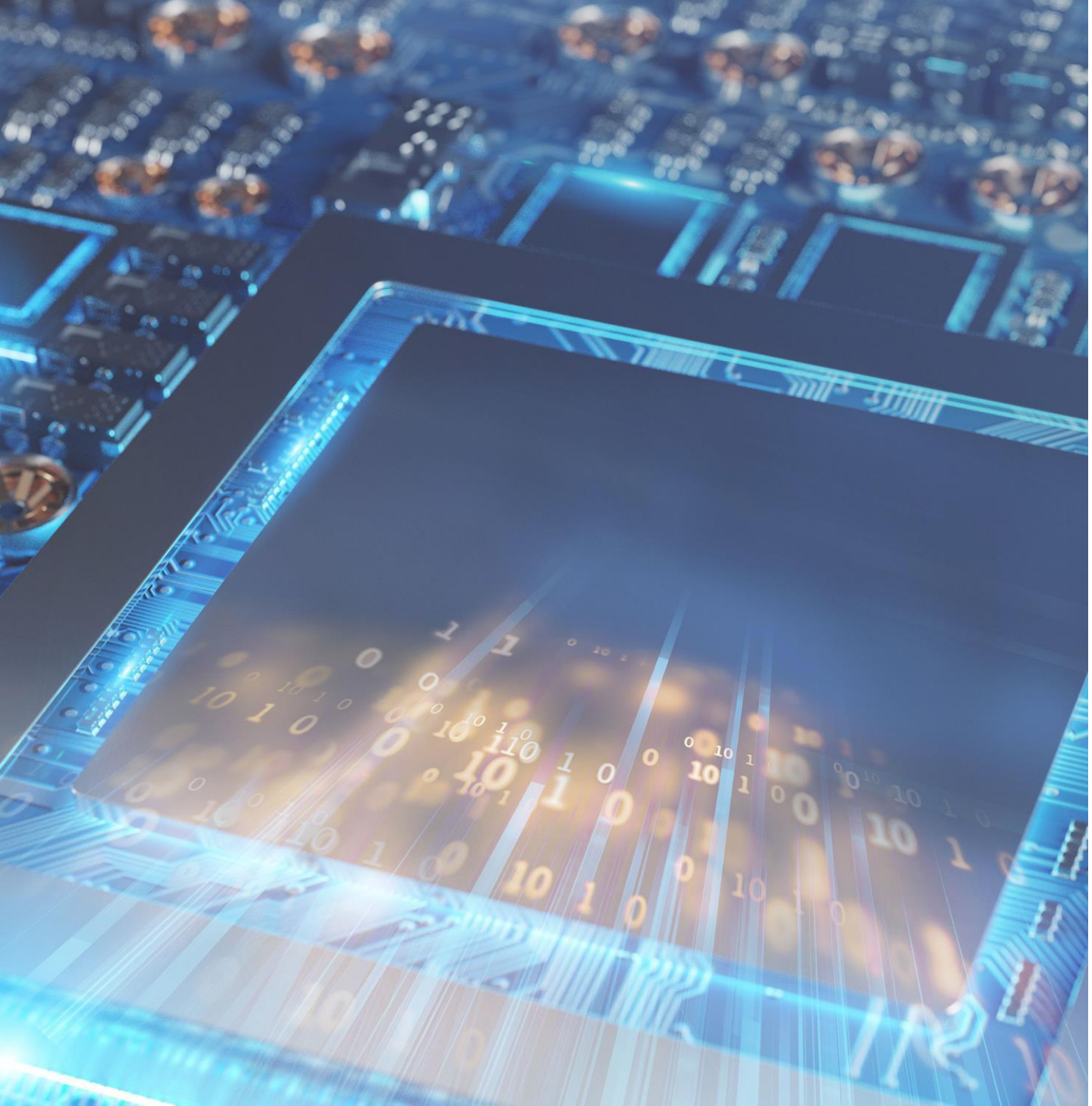
- We are honored to work with Bosch and relevant local semiconductor companies to define or standardize a way to debug and trace GTM.
  Hope our existing know-how will help the users to improve their development efficiency.

- We don't know which signals the GTMIP provides.
- We are really only a user of the Debug Interface.
- We are not allowed to be the initiator.
- We can comment on proposals and give recommendations.





QUESTIONS?



THANK YOU!