Debugging GTM Code on Virtual Prototypes and Real Microcontrollers

Matthias Noack, Jens Braunes
PLS Programmierbare Logik & Systeme GmbH

PLS Development Tools

- Debug, test and trace tools for Microcontrollers & Embedded Processors
- Made in Germany
- Since 1990

PLS Development Tools

Germany

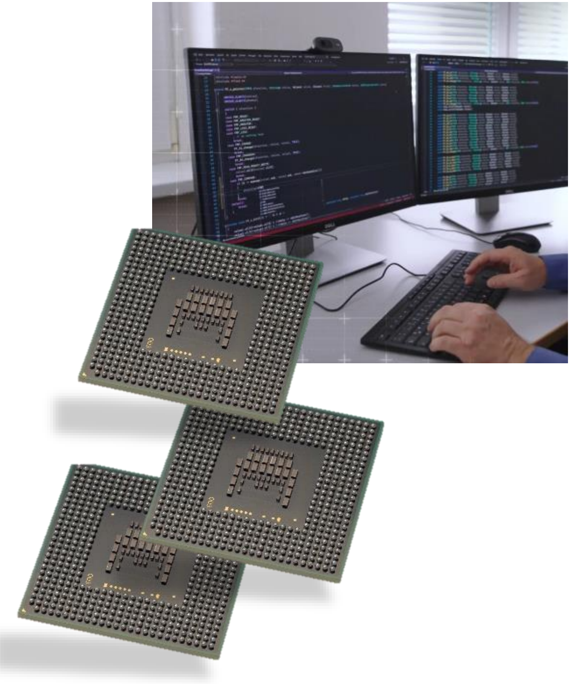
PLS Programmierbare Logik & Systeme GmbH
Technologiepark
D-02991 Lauta
Germany

PLS Programmierbare Logik & Systeme GmbH
TechnologieZentrumDresden
D-01099 Dresden
Germany

North America

PLS Development Tools
10080 N. Wolfe Rd., Suite SW3-200
Cupertino, CA 95014
USA

PLS Development Tools – Part of the Electronic Ecosystem



We Help to Develop

- Safe
- Secure
- Reliable
- Performant

Embedded Software





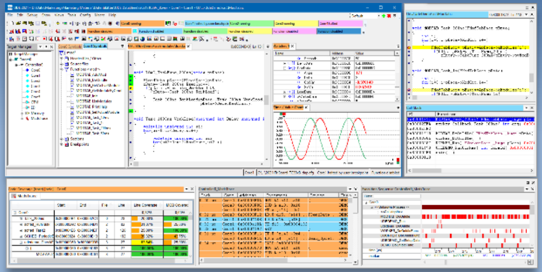

Universal Debug Engine UDE®

Complete tool set for debugging and trace

- On real target hardware
  - Microcontrollers, embedded systems, multicore SoCs
- On virtual prototypes
  - Simulators

- Real multicore debugging
  - System centric debugger environment, not core centric
  - One common user interface
  - Integrated Core Debuggers instead of separate debugger UI instances for cores

Software Debug environment on PC



Hardware Target access

Support for different Debug I/F

Support for trace I/F

pls Development Tools

Architecture and Controller Support

Infineon	ST	NXP	arm	Renesas	Texas Instruments	RISC-V	Synopsys
AURIX TC4x, TC3xx, TC2xx	Stellar automotive microcontroller family	S32G Vehicle Network Processors	Cortex-M (M7, M4, M4F, M3, M1, M0)	RH850/U, RH850/E, RH850/C	Jacinto 7 TDA4, DRA8	SiFive	ARC EM
Traveo T2G	SPC58xx, SPC57xx, SPC56xx	S32Z, S32E Real-Time Processors	Cortex-M23, Cortex-M33	RH850/F1, RH850/P1, RH850/D1	Sitara AM64x	GreenWaves	ARC EV
XMC7000	STM32 series	S32V S32V2 Processors for Vision	Cortex-R52, Cortex-R4, Cortex-R5	R-Car	Sitara AM243x	GigaDevice	ARC HS
XMC4xxx, XMC1xxx		S32K1, S32K3, S32M	Cortex-A8, Cortex-A9 Cortex-A53	RZ/T, RZ/N			
TriCore TC17xx, TC11xx		S32R Radar Processing	ARM7 / ARM9 /	SH-2A			
XC2000 / XE166		i.MX-RT					
		MPC57xx, MPC56xx					
		Kinetis, LPC					
		KW45 Wireless MCU					

GTM IP implemented

UDE GTM Support
- Basic debug support for GTM version < 4.1
- MCS run-control for GTM 4.1
- Trace support

UDE Support for Virtual Prototypes / Virtual Targets
- Synopsys VDK incl. GTM

- **COSEDA COSIDE®**

pls Development Tools

GTM Debugging on Real Targets

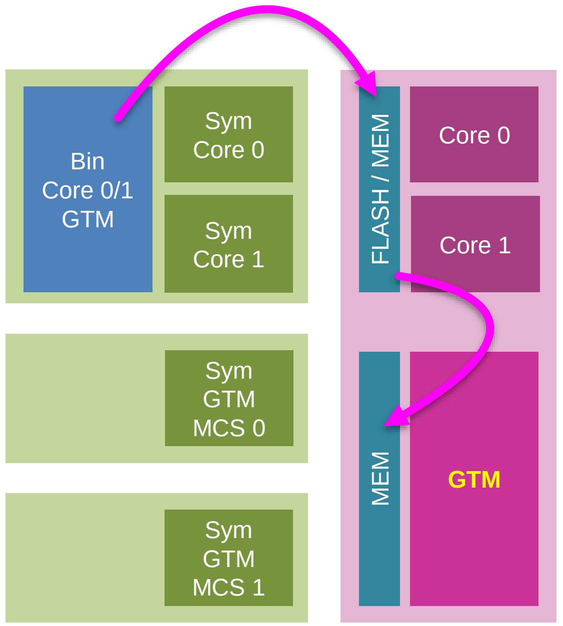
- GTM is part / is HW module of a heterogeneous SoC
  - Main cores (e.g. TriCore, Arm Cortex, etc.
  - GTM special core
- GTM needs to be enabled and initialized by host application

GTM application

- GTM MCS channel code
- Created by assembler or C compiler (HighTec, Tasking)
- Binary cannot be loaded directly into GTM RAM
  - Copied by host application
- ➘ Only debug symbols need to be loaded into GTM core debugger

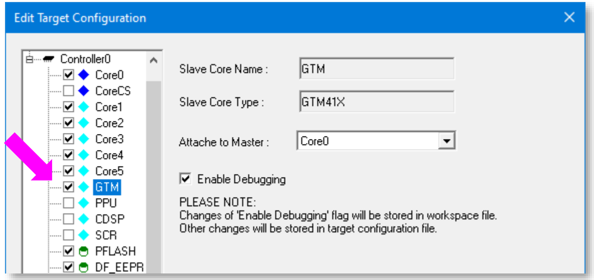
Host application

- Application code for host cores (e.g. TriCore, Cortex-R, etc.)
- GTM MCS binary (C-array)
- Initialization and start-up of GTM
  - Enable GTM clock
  - Copy GTM binary into GTM RAM
- ➘ Binaries and debug symbols need to be loaded into core debuggers of host

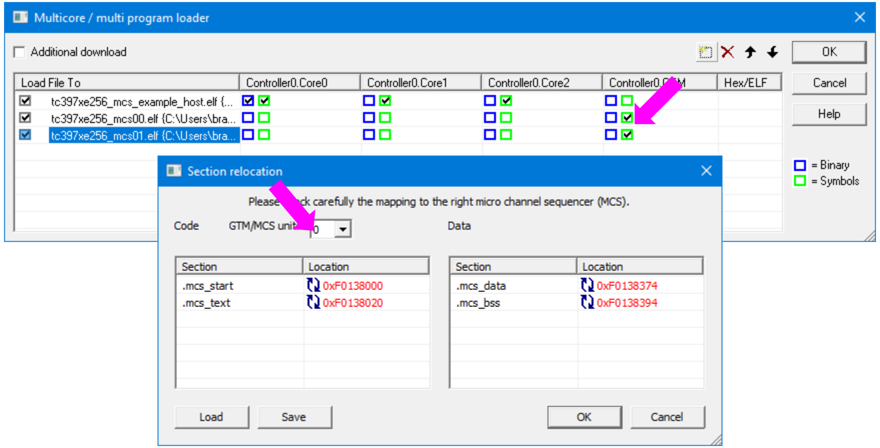


GTM Debugging in a Multicore SoC

- UDE – multicore debug and trace tool
- Integrated Core Debuggers for all cores of SoC
- GTM is visible as core

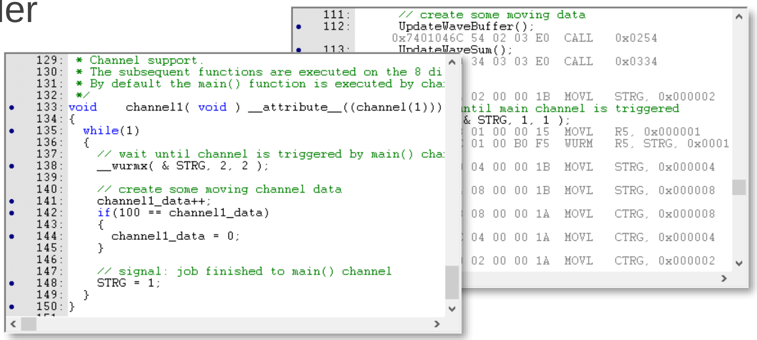


- Loading GTM Application
  - Contains GTM debug symbols
  - No binaries, binaries provided by host application (copied)
  - For debugging GTM host application always needed
- GTM memory is mapped into host memory
  - Needs mapping of local MCS addresses to global addresses
  - Semi-automatic in UDE, but manual check required

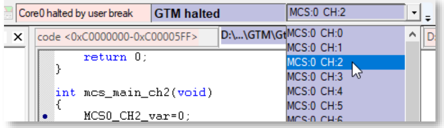


Basic Debugging

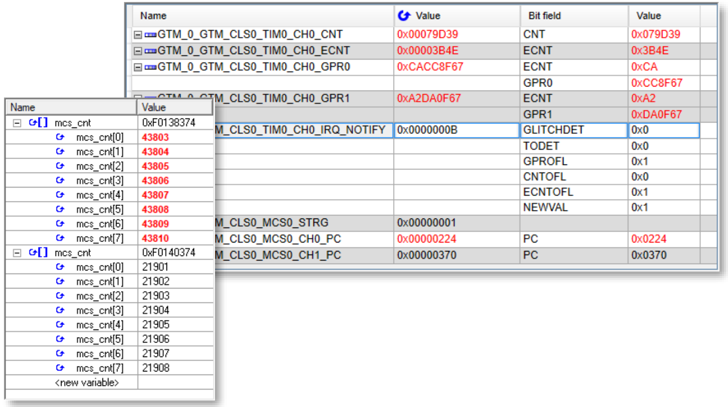
- Display of assembler code executed by MCS channels
- Display of C sources if MCS code is compiled by C-compiler



- Navigate to channel code by debugger toolbar



- Display of channel registers and module registers (TIM, TOM, SPE, etc.)
- Watch variables
- Real-time watch of variables, registers, memory
- Modifiable variables, registers, memory



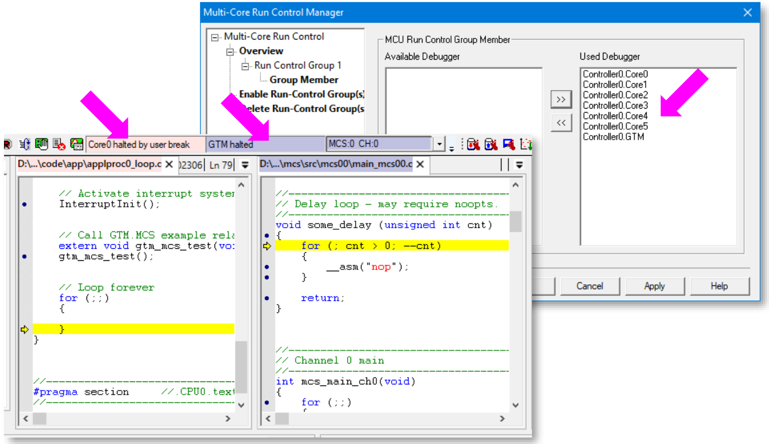
pls Development Tools

GTM Debugging on Real Targets – Run Control

GTM prior version 4.1

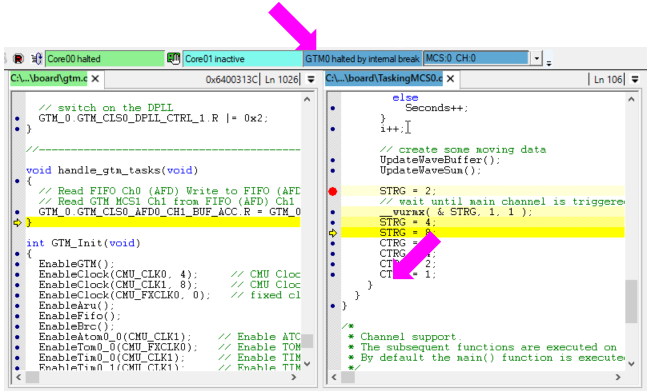
- No HW debug support for breakpoints
- ➘ No breakpoints, no single stepping

- Suspend / release synchronized with run-control of host cores



GTM version 4.1 (and higher)

- Two hardware breakpoints introduced with GTM v4.1
- ➘ "Normal" debugging of MCS code now possible
  - Breakpoints + Single stepping



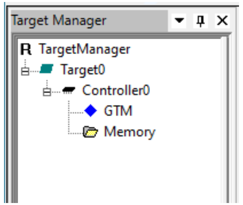
- Break options for breakpoint or manual break
  - Only MCS channel in focus
  - Entire GTM

pls Development Tools

GTM Debugging on COSIDE® Simulation

Stand-alone GTM

- → Host application by COSIDE®
- → No enabling and initializing needed
- → GTM is ready right from the beginning



GTM Simulation Model

- → Virtual target in UDE
- → No debug probe
- → Entire simulation model is halted on break

No HW limitations for debugging

- → Unlimited number of breakpoints

UDE as frontend for simulator

- → Same look and feel
- → Comparable functionality
- → Seamless switch from pre-silicon development to real HW

Evolution of GTM Debugging

Customer understanding

- Focus on MCS code
- C and assembler
- Breakpoints, single stepping
- Access to variables (read/write)
- Access to registers (read/write)

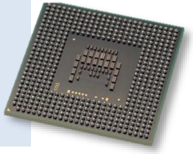
GTM (MCS Instruction Set) Simulator

- Debugging of MCS code
  - Unlimited breakpoints, single stepping
  - Unlimited access to memory and registers

COSEDA technologies

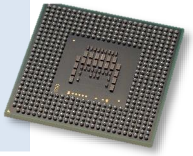
GTM v4.1

- HW debug support → HW breakpoints
  - "Normal" debugging of MCS code now possible
  - Breakpoints, single stepping
- Access to memory and registers
- Trace support
  - Trace based post-mortem debugging of executed MCS code
  - Data trace
  - Signal trace



GTM GEN 3

- No HW debug support for breakpoints
  - No breakpoints, no single stepping
  - Suspend / release synchronized with run-control of host cores
- Access to memory and registers
- Trace support
  - Trace based post-mortem debugging of executed MCS code
  - Data trace
  - Signal trace



pls Development Tools

UDE® as Frontend for COSIDE®

- ☑ Start development in pre-silicon phase
- ☑ Better project partitioning → development of pure GTM functionality independent from host SoC
- ☑ Continued use of the same tooling when switching to real hardware

Thank You!

PLS Programmierbare
Logik & Systeme GmbH
Technologiepark
02991 Lauta
GERMANY

+49 - 35722 - 384 - 0

info@pls-mc.com
www.pls-mc.com