Using UDE for Debugging and Tracing of GTM

Jens Braunes | PLS Development Tools

GTM Techday 2022
September 22-23

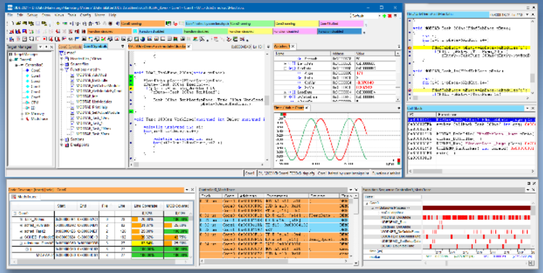
PLS Development Tools

- Debug, test and trace tools for microcontrollers and multicore SoCs
- Made in Germany
- Since more than 30 years on the market

Universal Debug Engine® UDE

- Complete workbench for debugging and trace, including
  - Debugger software
  - Debugger hardware
- Debugging and trace on real target hardware
  - Microcontrollers, embedded systems, multicore SoCs
- … on virtual prototypes / simulators
- Real Multi-Core Debugger
  - 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

Architecture and Controller Support

Infineon

- AURIX TC4x, TC3xx, TC2xx
- TriCore TC17xx, TC11xx
- XMC4500 / XMC1000
- XC2000 / XE166

NXP

- S32Z, S32E Real-Time Processors Octa-Cortex-R52
- S32G, S32V Quad Cortex-A53
- S32S Quad Cortex-R52
- S32R, MPC57xx, MPC56xx Multicore Power Architecture (e200 cores
- i.MX-RT
- S32K, Kinetis, LPC

ST life.augmented

- Stellar Cortex®-R52 automotive microcontroller family
- SPC58xx, SPC57xx, SPC56xx Multi-core Power Architecture (e200 cores)
- STM32 series (Cortex-M)

RENESAS

- R-Car
- RH850
- SH-2A

arm

- Cortex-M (M7, M4, M4F, M3, M1, M0)
- Cortex-M23, Corte
- Corte R4, C
- Corte Cortex-A53
- ARM7 / ARM9 / ARM11

RISC-V

- SiFive
- GreenWaves
- GigaDevice

SYNOPSYS

- ARC EM
- ARC EV

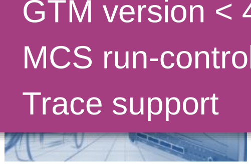
UDE Support for Virtual Prototypes / Virtual Targets

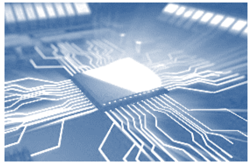
- Synopsys VDK incl. GTM

GTM IP implemented

UDE GTM Support

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



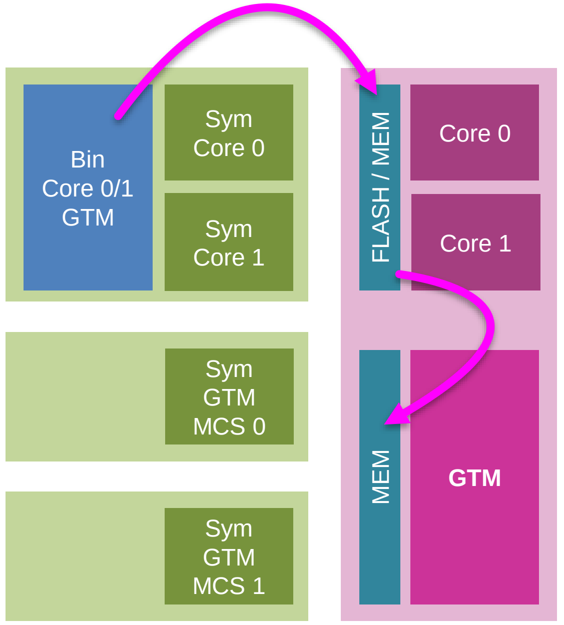


ANALOG DEVICES · DIGI · infineon · MICROCHIP · Atmel · NXP · ST life.augmented · TEXAS INSTRUMENTS

Software Architecture for GTM Applications

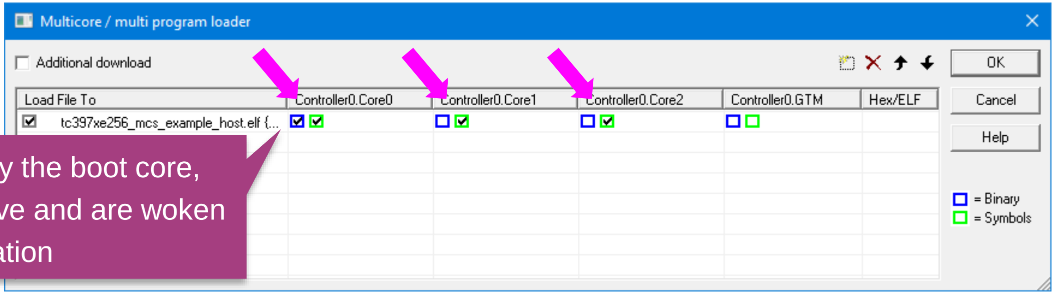
- GTM application
  - GTM MCS channel code
  - Created by assembler or C compiler (HighTec, Tasking)
  - Binary cannot be loaded directly into GTM RAM
    - Copied by host application
  - ➘ Only debug symbols need to be loaded into GTM core debugger

- 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



Loading GTM Applications

- Load host application into FLASH / RAM
- Load debug symbols for host application into host core debuggers



Multicore / multi program loader

☐ Additional download

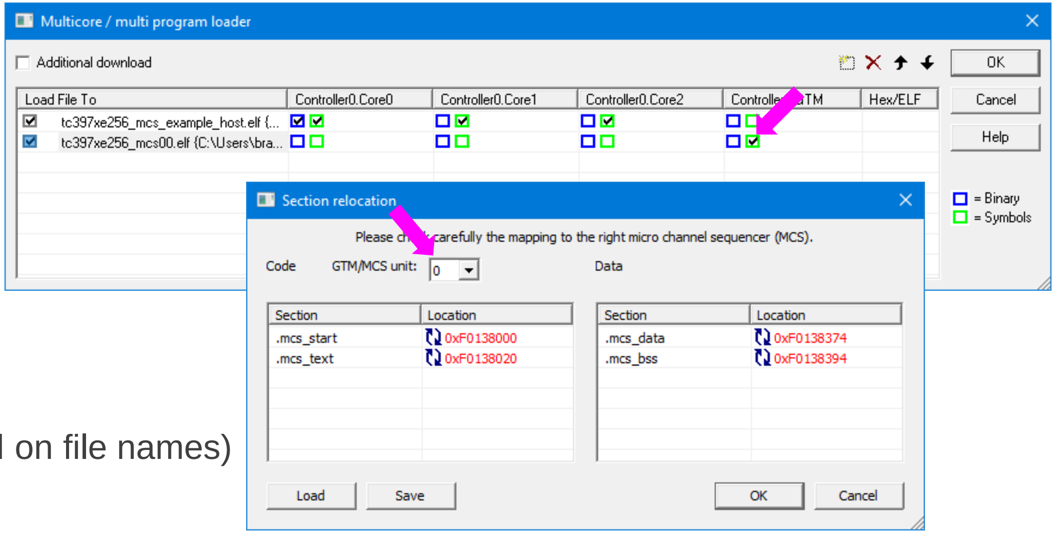
Load File To	Controller0.Core0	Controller0.Core1	Controller0.Core2	Controller0.GTM	Hex/ELF
☑ tc397xe256_mcs_example_host.elf {...	☑ ☑	☐ ☑	☐ ☑	☐ ☐	

☐ = Binary
☐ = Symbols

Binaries are loaded by the boot core, other cores are inactive and are woken up by the host application

Loading GTM Applications

- Load host application into FLASH / RAM
- Load debug symbols for host application into host core debuggers
- Load GTM Application
  - Contains GTM debug symbols
  - No binaries
- UDE tries to assign automatically the correct MCS for the ELF and to map GTM local addresses to system global addresses (based on file names)
- User needs to check address mapping of code + data sections to MCSx


Multicore / multi program loader

☐ Additional download

Load File To	Controller0.Core0	Controller0.Core1	Controller0.Core2	Controller0.GTM	Hex/ELF
☑ tc397xe256_mcs_example_host.elf {...	☑ ☑	☐ ☑	☐ ☑	☐ ☐	
☑ tc397xe256_mcs00.elf {C:\Users\bra...	☐ ☐	☐ ☐	☐ ☐	☐ ☑	

🟦 = Binary
🟩 = Symbols

Section relocation

Please check carefully the mapping to the right micro channel sequencer (MCS).

Code GTM/MCS unit: 0

Section	Location
.mcs_start	0xF0138000
.mcs_text	0xF0138020

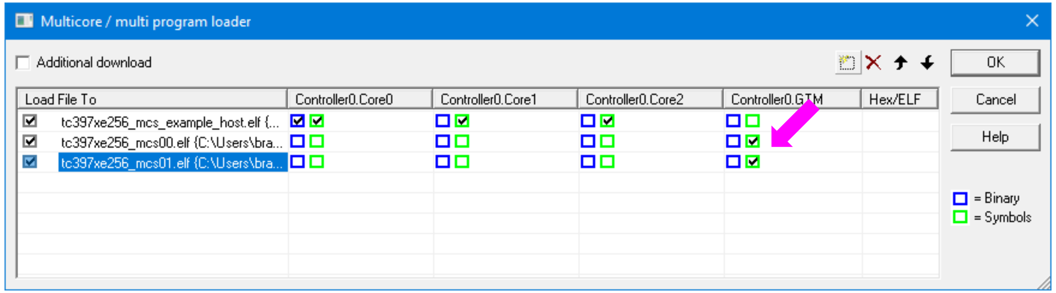
Data

Section	Location
.mcs_data	0xF0138374
.mcs_bss	0xF0138394

pls Development Tools

Loading GTM Applications

- Load host application into FLASH / RAM
- Load debug symbols for host application into host core debuggers
- Load GTM Application
  - Contains GTM debug symbols
  - No binaries
- UDE tries to assign automatically the correct MCS for the ELF and to map GTM local addresses to system global addresses (based on file names)
- User needs to check address mapping of code + data sections to MCSx
- Multiple ELF files possible (e.g. separately for MCSx)

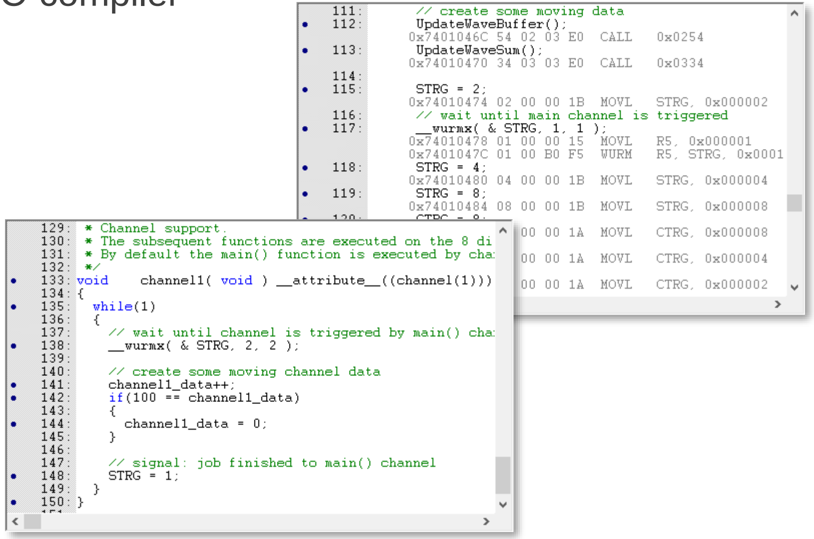

Multicore / multi program loader

☐ Additional download

Load File To	Controller0.Core0	Controller0.Core1	Controller0.Core2	Controller0.GTM	Hex/ELF
✓ tc397xe256_mcs_example_host.elf {...	✓ ✓	☐ ✓	☐ ✓	☐ ☐	
✓ tc397xe256_mcs00.elf {C:\Users\bra...	☐ ☐	☐ ☐	☐ ☐	☐ ✓	
✓ tc397xe256_mcs01.elf {C:\Users\bra...	☐ ☐	☐ ☐	☐ ☐	☐ ✓	

Debugging GTM

- Display of assembler code executed by MCS channels
- Display of C sources if MCS code is compiled by C-compiler
- Display of channel registers and module registers (TIM, TOM, SPE, etc.)
- Watch variables
- Real-time watch of variables, registers, memory
- Modifiable variables, registers, memory



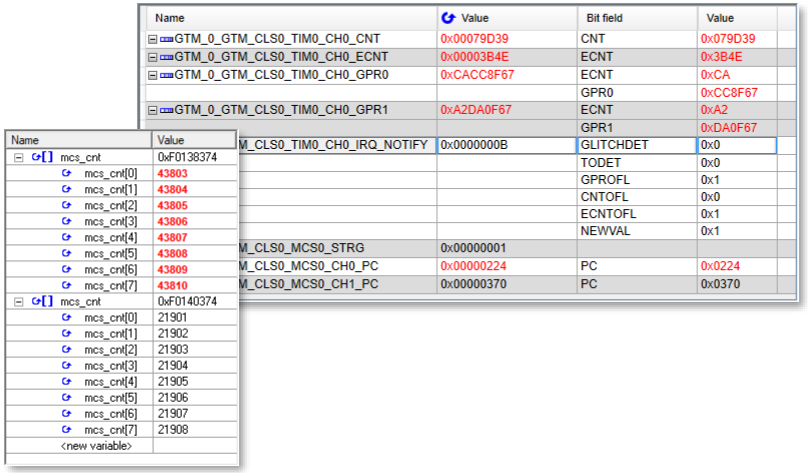
```
    // create some moving data
    UpdateWaveBuffer();
   0x7401046C 54 02 03 E0  CALL   0x0254
    UpdateWaveSum();
   0x74010470 34 03 03 E0  CALL   0x0334

    STRG = 2;
   0x74010474 02 00 00 1B  MOVL   STRG, 0x000002
    // wait until main channel is triggered
    __wurmx( & STRG, 1, 1 );
   0x74010478 01 00 00 15  MOVL   R5, 0x000001
   0x7401047C 01 00 B0 F5  WURM   R5, STRG, 0x0001
    STRG = 4;
   0x74010480 04 00 00 1B  MOVL   STRG, 0x000004
    STRG = 8;
   0x74010484 08 00 00 1B  MOVL   STRG, 0x000008
                00 00 1A  MOVL   CTRG, 0x000008
                00 00 1A  MOVL   CTRG, 0x000004
                00 00 1A  MOVL   CTRG, 0x000002
```

```
 * Channel support.
 * The subsequent functions are executed on the 8 di
 * By default the main() function is executed by cha
 */
void    channel1( void ) __attribute__((channel(1)))
{
  while(1)
  {
    // wait until channel is triggered by main() cha
    __wurmx( & STRG, 2, 2 );

    // create some moving channel data
    channel1_data++;
    if(100 == channel1_data)
    {
      channel1_data = 0;
    }

    // signal: job finished to main() channel
    STRG = 1;
  }
}
```



Name	Value	Bit field	Value
GTM_0_GTM_CLS0_TIM0_CH0_CNT	0x00079D39	CNT	0x079D39
GTM_0_GTM_CLS0_TIM0_CH0_ECNT	0x00003B4E	ECNT	0x3B4E
GTM_0_GTM_CLS0_TIM0_CH0_GPR0	0xCACC8F67	ECNT	0xCA
		GPR0	0xCC8F67
GTM_0_GTM_CLS0_TIM0_CH0_GPR1	0xA2DA0F67	ECNT	0xA2
		GPR1	0xDA0F67
M_CLS0_TIM0_CH0_IRQ_NOTIFY	0x0000000B	GLITCHDET	0x0
		TODET	0x0
		GPROFL	0x1
		CNTOFL	0x0
		ECNTOFL	0x1
		NEWVAL	0x1
M_CLS0_MCS0_STRG	0x00000001		
M_CLS0_MCS0_CH0_PC	0x00000224	PC	0x0224
M_CLS0_MCS0_CH1_PC	0x00000370	PC	0x0370

Name	Value
mcs_cnt	0xF0138374
mcs_cnt[0]	43803
mcs_cnt[1]	43804
mcs_cnt[2]	43805
mcs_cnt[3]	43806
mcs_cnt[4]	43807
mcs_cnt[5]	43808
mcs_cnt[6]	43809
mcs_cnt[7]	43810
mcs_cnt	0xF0140374
mcs_cnt[0]	21901
mcs_cnt[1]	21902
mcs_cnt[2]	21903
mcs_cnt[3]	21904
mcs_cnt[4]	21905
mcs_cnt[5]	21906
mcs_cnt[6]	21907
mcs_cnt[7]	21908
<new variable>	

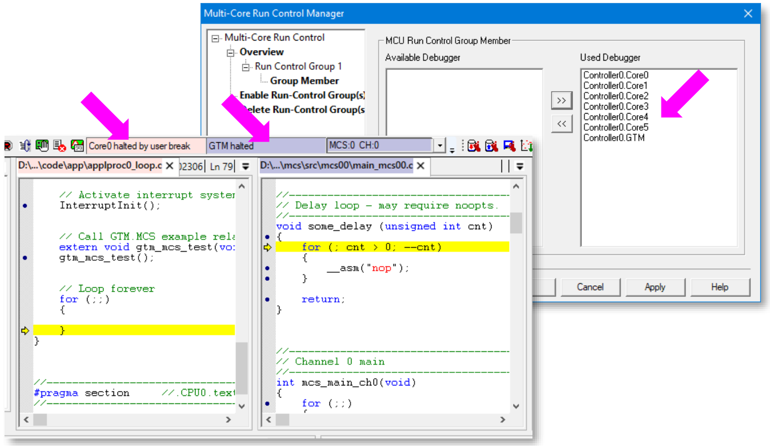
pls Development Tools

Debugging MCS Code – 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



Multi-Core Run Control Manager

- Multi-Core Run Control
  - Overview
    - Run Control Group 1
      - Group Member
  - Enable Run-Control Group(s)
  - Delete Run-Control Group(s

MCU Run Control Group Member

Available Debugger

Used Debugger

```
Controller0.Core0
Controller0.Core1
Controller0.Core2
Controller0.Core3
Controller0.Core4
Controller0.Core5
Controller0.GTM
```

Core0 halted by user break | GTM halted | MCS:0 CH:0

D:\...\code\app\applproc0_loop.c | 02306 | Ln 79

```
    // Activate interrupt system
    InterruptInit();

    // Call GTM.MCS example rela
    extern void gtm_mcs_test(vo
    gtm_mcs_test();

    // Loop forever
    for (;;)
    {

    }
}


//--------------------------------
#pragma section      //.CPU0.text
//--------------------------------
```

D:\...\mcs\src\mcs00\main_mcs00.c

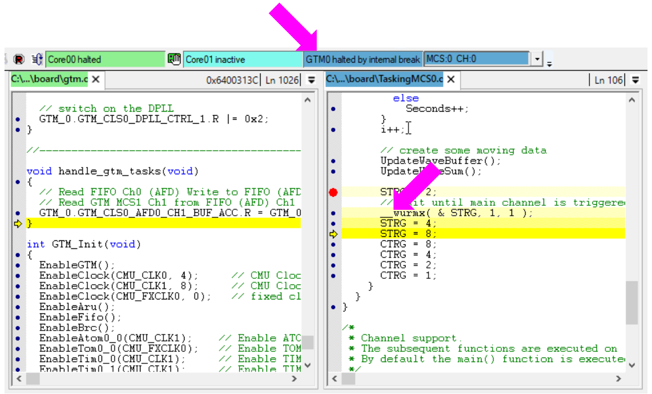
```
//---------------------------------------
// Delay loop - may require noopts.
//---------------------------------------
void some_delay (unsigned int cnt)
{
    for (; cnt > 0; --cnt)
    {
        __asm("nop");
    }

    return;
}


//---------------------------------------
// Channel 0 main
//---------------------------------------
int mcs_main_ch0(void)
{
    for (;;)
```

GTM version 4.1 (and higher)

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



Core00 halted | Core01 inactive | GTM0 halted by internal break | MCS:0 CH:0

C:\...\board\gtm.c | 0x6400313C | Ln 1026

```
  // switch on the DPLL
  GTM_0.GTM_CLS0_DPLL_CTRL_1.R |= 0x2;
}

//----------------------------------------

void handle_gtm_tasks(void)
{
  // Read FIFO Ch0 (AFD) Write to FIFO (AFD
  // Read GTM MCS1 Ch1 from FIFO (AFD) Ch1
  GTM_0.GTM_CLS0_AFD0_CH1_BUF_ACC.R = GTM_0
}

int GTM_Init(void)
{
  EnableGTM();
  EnableClock(CMU_CLK0, 4);      // CMU Cloc
  EnableClock(CMU_CLK1, 8);      // CMU Cloc
  EnableClock(CMU_FXCLK0, 0);    // fixed cl
  EnableAru();
  EnableFifo();
  EnableBrc();
  EnableAtom0_0(CMU_CLK1);     // Enable ATC
  EnableTom0_0(CMU_FXCLK0);    // Enable TOM
  EnableTim0_0(CMU_CLK1);      // Enable TIM
  EnableTim0_1(CMU_CLK1);      // Enable TIM
```

C:\...\board\TaskingMCS0.c | Ln 106

```
          else
            Seconds++;
        }
        i++;

        // create some moving data
        UpdateWaveBuffer();
        UpdateWaveSum();

        STRG = 2;
        // wait until main channel is triggere
        __wurmx( & STRG, 1, 1 );
        STRG = 4;
        STRG = 8;
        CTRG = 8;
        CTRG = 4;
        CTRG = 2;
        CTRG = 1;
      }
    }
}

/*
 * Channel support.
 * The subsequent functions are executed on
 * By default the main() function is execute
 */
```

pls Development Tools

GTM Trace

- Because of specific hardware characteristics and real-time characteristics of the applications breaking GTM is often not a good idea.

Debugging use cases for GTM

- Monitoring
  - Program execution of MCS channels
  - Data transfers (MCS, DPLL RAMs, ARU)
  - Module signals
- Debugging MCS execution
  - Monitoring program flow
- Debug GTM / host core(s) interaction

Typical Trace Use Cases

Run-time observation without influencing the run-time behavior

GTM Trace Features

- MCS trace
  - Program trace (fetch trace)
  - Data trace (r/w) for RAM accesses
  - Parallel trace of single channel / multi channels
  - Address compare and triggers for code addresses
- ARU data trace
  - Two trace channels → trace of two ARU debug channels in parallel
- TIM / TOM / ATOM trace
  - Two trace channels → trace of two modules in parallel
- SPE watchpoints
- DPLL data trace
  - Trace of one DPLL memory module
- TBU trace
  - Trace of timestamp of one TBU*

- GTM trace is integrated into device trace system
  - MCDS for AURIX
  - CoreSight for Arm based devices (Stellar, S32)
  - Nexus for PowerArchitecture (MPC57xx, SPC5)
- Parallel trace of GTM and main cores
  - Global timestamp *
  - Cross-triggers for signaling GTM ↔ Cores (e.g. trace start / stop) *

* Device specific

pls Development Tools

Trace Support in UDE

Trace Window

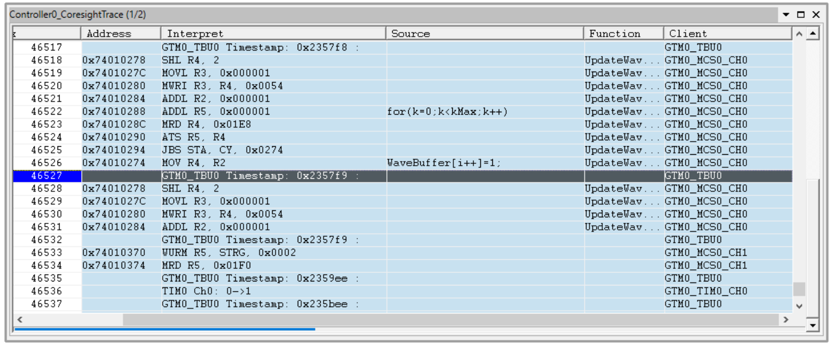
- Sequential list of recorded trace data
  - Program flow
  - Data transfers
  - Signals
  - Timestamps
  - Etc.

Execution Sequence Chart

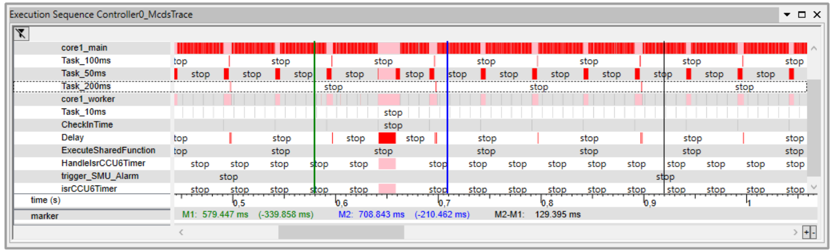
- Graphical visualization of MCS channel code execution over time

Code Coverage

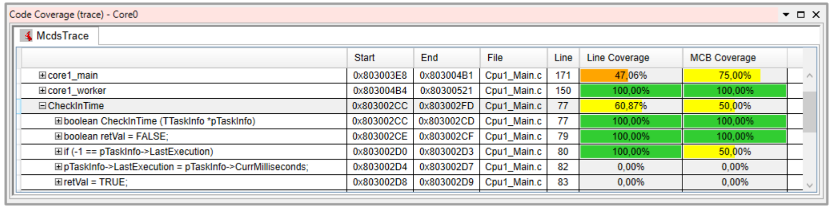
- Statement coverage (object code)
- Branch coverage (object code)


Controller0_CoresightTrace (1/2)

	Address	Interpret	Source	Function	Client
46517		GTM0_TBU0 Timestamp: 0x2357f8 :			GTM0_TBU0
46518	0x74010278	SHL R4, 2		UpdateWav...	GTM0_MCS0_CH0
46519	0x7401027C	MOVL R3, 0x000001		UpdateWav...	GTM0_MCS0_CH0
46520	0x74010280	MWRI R3, R4, 0x0054		UpdateWav...	GTM0_MCS0_CH0
46521	0x74010284	ADDL R2, 0x000001		UpdateWav...	GTM0_MCS0_CH0
46522	0x74010288	ADDL R5, 0x000001	for(k=0;k<kMax;k++)	UpdateWav...	GTM0_MCS0_CH0
46523	0x7401028C	MRD R4, 0x01E8		UpdateWav...	GTM0_MCS0_CH0
46524	0x74010290	ATS R5, R4		UpdateWav...	GTM0_MCS0_CH0
46525	0x74010294	JBS STA, CY, 0x0274		UpdateWav...	GTM0_MCS0_CH0
46526	0x74010274	MOV R4, R2	WaveBuffer[i++]=1;	UpdateWav...	GTM0_MCS0_CH0
46527		GTM0_TBU0 Timestamp: 0x2357f9 :			GTM0_TBU0
46528	0x74010278	SHL R4, 2		UpdateWav...	GTM0_MCS0_CH0
46529	0x7401027C	MOVL R3, 0x000001		UpdateWav...	GTM0_MCS0_CH0
46530	0x74010280	MWRI R3, R4, 0x0054		UpdateWav...	GTM0_MCS0_CH0
46531	0x74010284	ADDL R2, 0x000001		UpdateWav...	GTM0_MCS0_CH0
46532		GTM0_TBU0 Timestamp: 0x2357f9 :			GTM0_TBU0
46533	0x74010370	WURM R5, STRG, 0x0002			GTM0_MCS0_CH1
46534	0x74010374	MRD R5, 0x01F0			GTM0_MCS0_CH1
46535		GTM0_TBU0 Timestamp: 0x2359ee :			GTM0_TBU0
46536		TIM0 Ch0: 0->1			GTM0_TIM0_CH0
46537		GTM0_TBU0 Timestamp: 0x235bee :			GTM0_TBU0


Execution Sequence Controller0_McdsTrace

M1: 579.447 ms (-339.858 ms) M2: 708.843 ms (-210.462 ms) M2-M1: 129.395 ms


Code Coverage (trace) - Core0

McdsTrace

	Start	End	File	Line	Line Coverage	MCB Coverage
⊞ core1_main	0x803003E8	0x803004B1	Cpu1_Main.c	171	47,06%	75,00%
⊞ core1_worker	0x803004B4	0x80300521	Cpu1_Main.c	150	100,00%	100,00%
⊟ CheckInTime	0x803002CC	0x803002FD	Cpu1_Main.c	77	60,87%	50,00%
⊞ boolean CheckInTime (TTaskInfo *pTaskInfo)	0x803002CC	0x803002CD	Cpu1_Main.c	77	100,00%	100,00%
⊞ boolean retVal = FALSE;	0x803002CE	0x803002CF	Cpu1_Main.c	79	100,00%	100,00%
⊞ if (-1 == pTaskInfo->LastExecution)	0x803002D0	0x803002D3	Cpu1_Main.c	80	100,00%	50,00%
⊞ pTaskInfo->LastExecution = pTaskInfo->CurrMilliseconds;	0x803002D4	0x803002D7	Cpu1_Main.c	82	0,00%	0,00%
⊞ retVal = TRUE;	0x803002D8	0x803002D9	Cpu1_Main.c	83	0,00%	0,00%

Example: GTM / Core Interaction

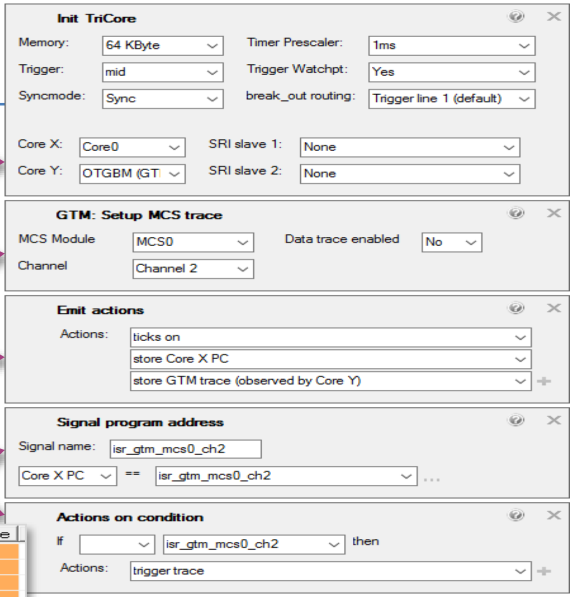
- Using MCDS trace of AURIX
  - “Graphical” trace configuring with building blocks
- Monitoring GTM interrupt
  - Parallel trace of MCS channel and main core
  - Exact timing of interrupt handling

Enable monitoring of GTM and Core0

Select MCS module and channel

Capture program trace

Trace recording around Core0 interrupt handler



Init TriCore

Memory:	64 KByte	Timer Prescaler:	1ms
Trigger:	mid	Trigger Watchpt:	Yes
Syncmode:	Sync	break_out routing:	Trigger line 1 (default)
Core X:	Core0	SRI slave 1:	None
Core Y:	OTGBM (GT	SRI slave 2:	None

GTM: Setup MCS trace

MCS Module	MCS0	Data trace enabled	No
Channel	Channel 2		

Emit actions

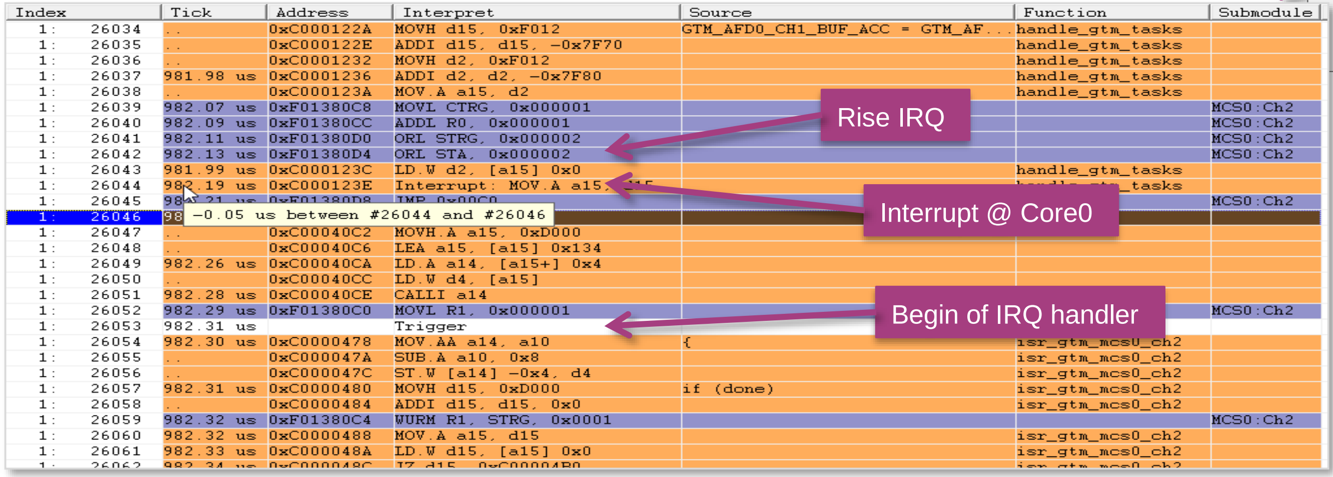
Actions: ticks on
store Core X PC
store GTM trace (observed by Core Y)

Signal program address

Signal name: isr_gtm_mcs0_ch2
Core X PC == isr_gtm_mcs0_ch2 ...

Actions on condition

If isr_gtm_mcs0_ch2 then
Actions: trigger trace



Index	Tick	Address	Interpret	Source	Function	Submodule
1: 26034	..	0xC000122A	MOVH d15, 0xF012	GTM_AFD0_CH1_BUF_ACC = GTM_AF...	handle_gtm_tasks	
1: 26035	..	0xC000122E	ADDI d15, d15, -0x7F70		handle_gtm_tasks	
1: 26036	..	0xC0001232	MOVH d2, 0xF012		handle_gtm_tasks	
1: 26037	981.98 us	0xC0001236	ADDI d2, d2, -0x7F80		handle_gtm_tasks	
1: 26038	..	0xC000123A	MOV.A a15, d2		handle_gtm_tasks	
1: 26039	982.07 us	0xF01380C8	MOVL CTRG, 0x000001			MCS0:Ch2
1: 26040	982.09 us	0xF01380CC	ADDL R0, 0x000001			MCS0:Ch2
1: 26041	982.11 us	0xF01380D0	ORL STRG, 0x000002			MCS0:Ch2
1: 26042	982.13 us	0xF01380D4	ORL STA, 0x000002			MCS0:Ch2
1: 26043	981.99 us	0xC000123C	LD.W d2, [a15] 0x0		handle_gtm_tasks	
1: 26044	982.19 us	0xC000123E	Interrupt: MOV.A a15, d15		handle_gtm_tasks	
1: 26045	982.21 us	0xF01380D8	JMP 0x00C0			MCS0:Ch2
1: 26046	98					
1: 26047	..	0xC00040C2	MOVH.A a15, 0xD000			
1: 26048	..	0xC00040C6	LEA a15, [a15] 0x134			
1: 26049	982.26 us	0xC00040CA	LD.A a14, [a15+] 0x4			
1: 26050	..	0xC00040CC	LD.W d4, [a15]			
1: 26051	982.28 us	0xC00040CE	CALLI a14			
1: 26052	982.29 us	0xF01380C0	MOVL R1, 0x000001			MCS0:Ch2
1: 26053	982.31 us		Trigger			
1: 26054	982.30 us	0xC0000478	MOV.AA a14, a10	{	isr_gtm_mcs0_ch2	
1: 26055	..	0xC000047A	SUB.A a10, 0x8		isr_gtm_mcs0_ch2	
1: 26056	..	0xC000047C	ST.W [a14] -0x4, d4		isr_gtm_mcs0_ch2	
1: 26057	982.31 us	0xC0000480	MOVH d15, 0xD000	if (done)	isr_gtm_mcs0_ch2	
1: 26058	..	0xC0000484	ADDI d15, d15, 0x0		isr_gtm_mcs0_ch2	
1: 26059	982.32 us	0xF01380C4	WURM R1, STRG, 0x0001			MCS0:Ch2
1: 26060	982.32 us	0xC0000488	MOV.A a15, d15		isr_gtm_mcs0_ch2	
1: 26061	982.33 us	0xC000048A	LD.W d15, [a15] 0x0		isr_gtm_mcs0_ch2	
1: 26062	982.34 us	0xC000048C	JZ d15, 0xC00004B0		isr_gtm_mcs0_ch2	

pls Development Tools

Thank you!

PLS Programmierbare
Logik & Systeme GmbH
Technologiepark
02991 Lauta
GERMANY

+49 - 35722 - 384 - 0

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