TASKING®

COMPILER TOOL CHAIN FOR BOSCH GTM-IP BASED SOCS WITH FOCUS ON SAFETY CRITICAL APPLICATIONS

Joachim Hampp

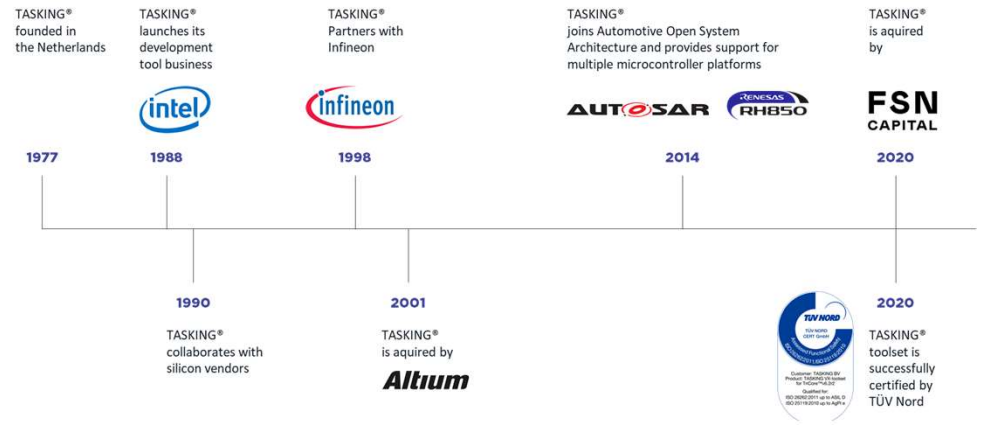
Product Architect

Stuttgart

September 2022

TASKING Company Overview
Key Facts

- Headquarters: Munich, Germany
- TASKING® supplies safe, secure and high performing embedded software development tools specializing in multi-core safety critical applications
- Global company with over 86 employees located in 6 countries
- 40+ years of experience in automotive industry
- Field-proven tools used by most OEM and Tier-1 suppliers
- Development processes in accordance with ASPICE CL 2
- Acquired by FSN Capital V in December 2020
- For FY 2022, TASKING® is projecting revenues of c. EUR 24m



TASKING®

Agenda

1. **Compiler Tool Chain for SoCs**
2. GTM-MCS Compiler Tool Chain integration
3. ISO26262 Tool Qualification

Complex SoC
AURIX TC4x

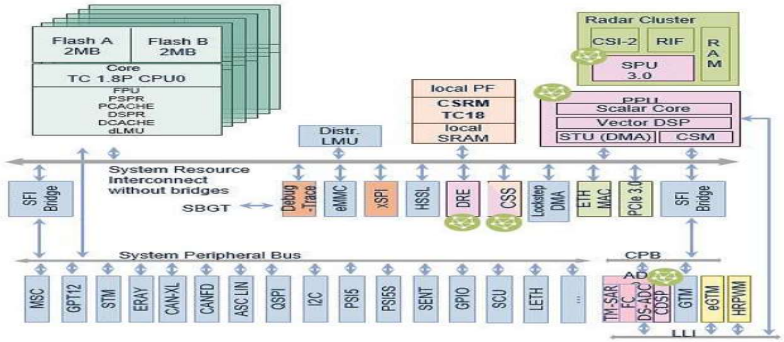
TASKING

TriCore™ ASIL-D Performance
- Enhanced TriCore™ v1.8
- Up to 6CPUs, upto 500MHz

On-Chip Memory
- Upto 25MB on chip-flash
- Full A/B Swap support for SOTA
- More Tightly Coupled SRAM

AURIX ™ Accelerator Suite
- **Parallel Processing Unit (PPU)** Programmable SIMD Vector DSP
- **DRE- Data Routing Engine** communications accelerator
- **cDSP** dedicated ADC accelerator
- **SPU 3.0** dedicated radar accelerator
- **CSS:** Provides security acceleration

xSPI
- External Memory Interface



Debug and Trace
- Safe and secure in the field

ADC
- Enhanced ADCs plus additional DSP accelerator

Radar Cluster
- 4 x TC3xx performance, 800M Samples/s
- Radar DMA
- CSI-2 Interface

Real-time Control
- New eGTM timers, High Resolution PWMs
- Low latency interconnect (LLI) to PPU for fast processing

Security
- **CSRM:** High performance Security Modules with ASIL-B support
- **CSS:** Dedicated communication security satellites

High-speed Comm I/F
- PCIe 3.0
- 100Mb- 5 Gbps Ethernet

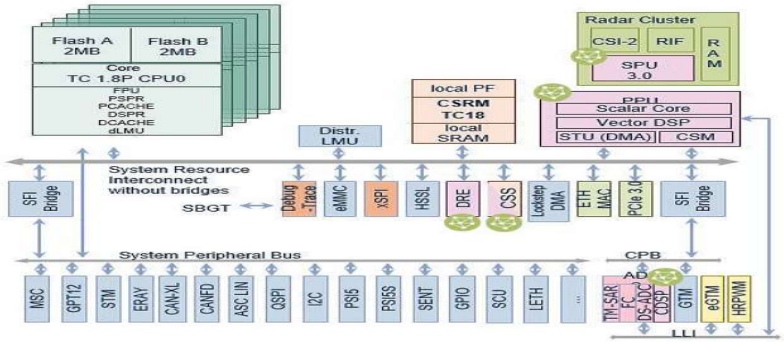
Future proof Comm I/F
- CAN-XL
- 10Mb – 5Gbs Ethernet

TriCore™ ASIL-D Performance
- Enhanced TriCore™ v1.8
- Up to 6CPUs, upto 500MHz

On-Chip Memory
- Upto 25MB on chip-flash
- Full A/B Swap support for SOTA
- More Tightly Coupled SRAM

AURIX ™ Accelerator Suite
- **Parallel Processing Unit (PPU)** Programmable SIMD Vector DSP
- **DRE- Data Routing Engine** communications accelerator
- **cDSP** dedicated ADC accelerator
- **SPU 3.0** dedicated radar accelerator
- **CSS:** Provides security acceleration

xSPI
- External Memory Interface



Debug and Trace
- Safe and secure in the field

ADC
- Enhanced ADCs plus additional DSP accelerator

Radar Cluster
- 4 x TC3xx performance, 800M Samples/s
- Radar DMA
- CSI-2 Interface

Real-time Control
- New eGTM timers, High Resolution PWMs
- Low latency interconnect (LLI) to PPU for fast processing

Security
- **CSRM:** High performance Security Modules with ASIL-B support
- **CSS:** Dedicated communication security satellites

High-speed Comm I/F
- PCIe 3.0
- 100Mb- 5 Gbps Ethernet

Future proof Comm I/F
- CAN-XL
- 10Mb – 5Gbs Ethernet

source : https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc4x/

TASKING SmartCode
Product Version, Evaluation, Licensing

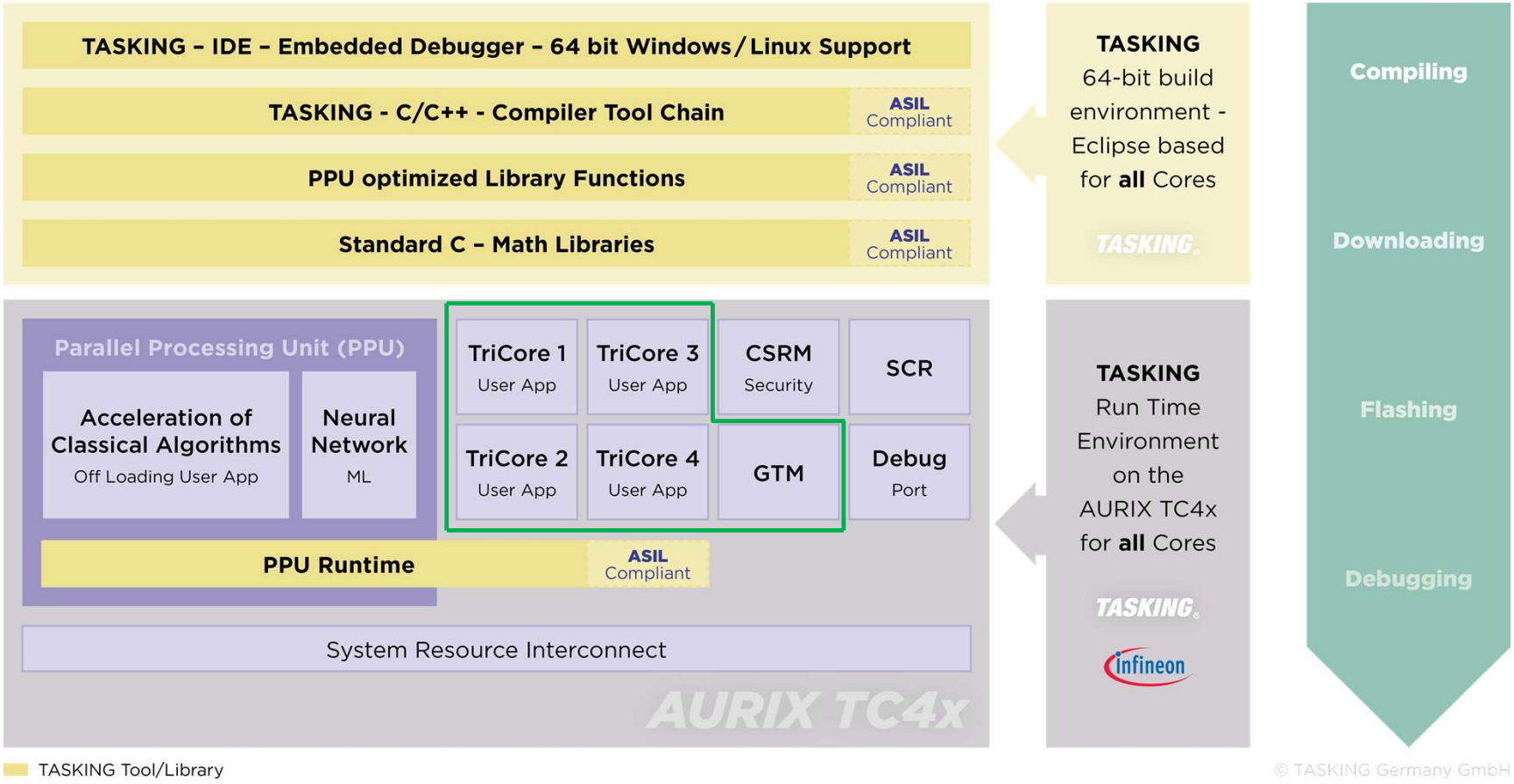
Product Name	Description
TASKING SmartCode v10.1r1	Eclipse based 64-bit application software design environment for Infineon next generation AURIX™ TC4xx Microcontrollers : - Supports Infineon TriCore v1.8 and PPU (Synopsys ARC EV7xFS) instruction set simulator - Includes support for Infineon`s AURIX™ TC4xx Virtual Development Kit (VDK) - ISO – C/C++ compiler toolset for TriCore TC v1.8 real time cores including CSRM - ISO – C compiler toolset for the standby control unit (SCR) - ISO – C compiler toolset for BOSCH GTM IP v4.1 and standard simulation debugger - ISO – C compiler toolset for AURIX™ PPU (ARC EV7xFS) - *Hardware debugger for TriCore TC v1.8 via IFX miniWiggler ** - *ISO – C++ compiler toolset for AURIX™ PPU (ARC EV7xFS) ** - *PPU hardware debugger via IFX miniWiggler ** - *Qualified PPU runtime library according to ISO26262 ** - *PPU optimized library functions ** - *Pin Mapper for TC4x ** **) will be part of a later release*

Customers who purchase SmartCode v10.1r1 **with commercial maintenance** are entitled to get all features which will be released with upcoming releases

Lates Advancements for Safety and Security
TASKING SmartCode is developed according to Automotive SPICE® Level 2 standards and is currently in the process of being certified for the development of safety critical software applications by safety experts from TÜV-Nord (ISO 26262:2018, ISO 25119:2018, EN 50657:2017, IEC 61508:2010, ISO/SAE 21434:2021). The combined Safety/Security Manual is included in the license, no additional cost for a Qualification Kit.

License Type	Term	Support / Maintenance	Mode
Evaluation License	30 days	Free	Node-locked
Time based License	6 months, 1 year	Included in license fee	Node-locked , Floating

TASKING SmartCode
Overview



Compiler Tool Chain for SoCs
Main Benefits

- **Software platform that fully supports** the unique combination of architectures and microprocessor cores integrated in the TC4x.
- Support for the lifetime of your product with **long-term support** from the **experts at TASKING** with commercial maintenance.
- Easy to use across the project.
- Tool Chain contains **mechanism to load, initialize and debug slave cores** ready to use for custom applications

TASKING®

Agenda

1. Compiler Tool Chain for SoCs
2. **GTM-MCS Compiler Tool Chain integration**
3. ISO26262 Tool Qualifcation

TASKING GTM-MCS Compiler
Overview

- TASKING GTM-MCS Compiler Tool Chain includes
  - GTM-MCS C Compiler
  - GTM-MCS Assembler
    - Assembler offers macro support and provides automatic conversion of GTM legacy code
  - GTM-MCS Linker
    - Multi-core Linker-Locator allows linking multiple MCS cores, swapping and borrowing memory among MCS channels and supports references between TriCore and GTM-MCS memories
  - GTM-MCS Simulator Debugger
    - Full featured C and assembly level debugger includes a simulator for the MCS Core
  - MISRA-C and CERT-C static code analysis are an integral part of the compiler
- Supports all GTM hardware features*. TASKING's GTM-MCS assembler supports 1st, 2nd, 3rd and 4th generation GTM cores. Special Function Register (SFR) files provide access to SFR's from C code.
- The compiler supports multi-core design and generates highly optimized time-deterministic code (Built-in code optimization techniques).
- Supports In-line assembly

*MCS features not supported in ISO-C99 have language extensions or intrinsic functions

TASKING GTM Compiler
Overview

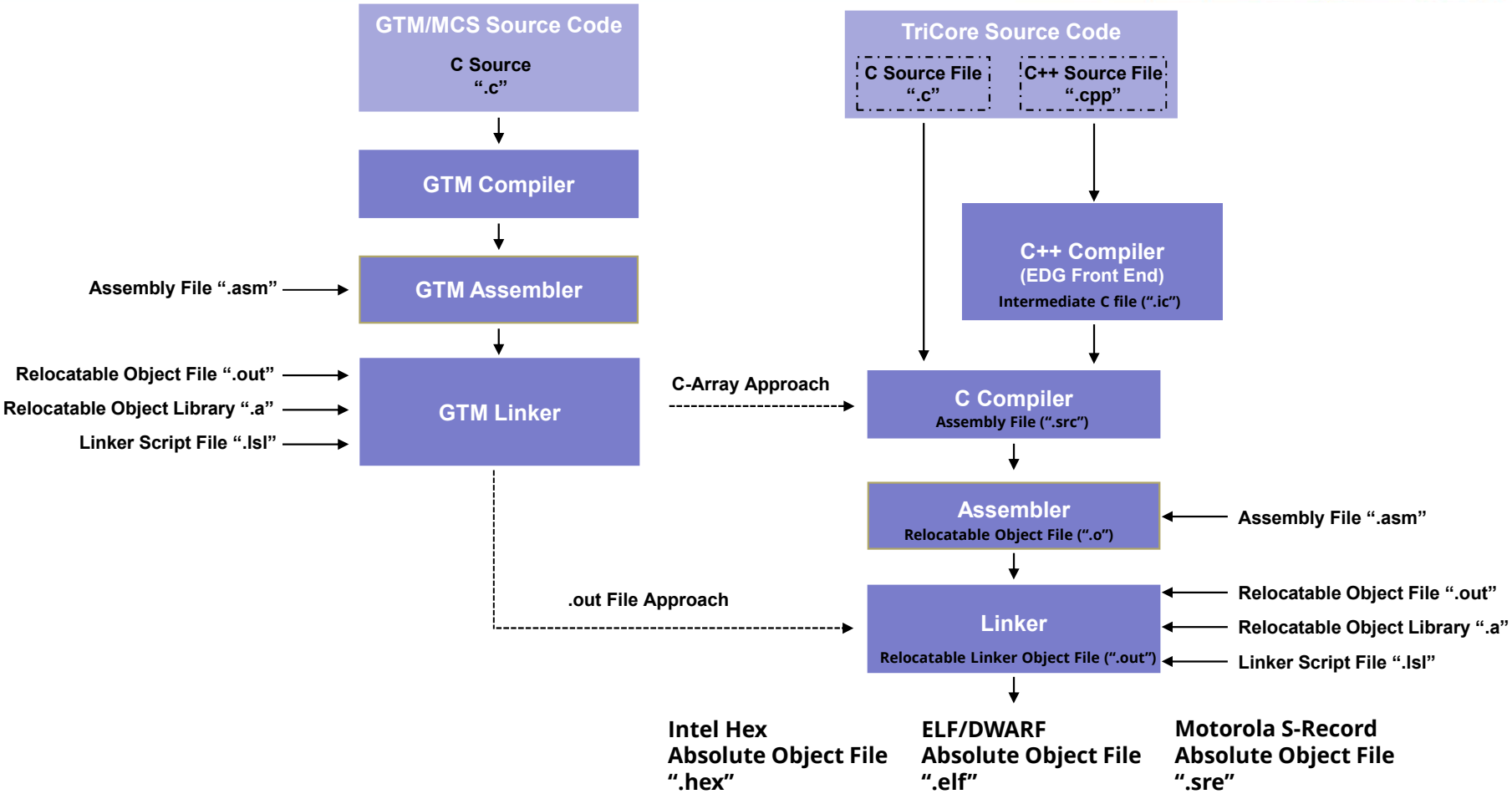
- The TASKING GTM-MCS C compiler fully supports the ISO C99 standard and supports all mandatory language features of the C11 standard and adds extra possibilities to program the special functions of the target.

- The complete Tool Chain is developed according ASPICE level 2 processes. Simplifies ISO 26262 compiler qualification up to ASIL-B
  - Easily meet and exceed industry-standard tests for compiler qualification
  - Full support for all common and compiler relevant C, ISO, IEEE and EABI standards.

- Compatible with 3rd Party debuggers and supports multiple output formats (ELF/DWARF) for symbolic debugging, Hex for programming flash memory and BOSCH C-Array format
  - Comprehensive integrated debugging tools for C and assembly.

- Compiler-independent export feature utilizing the classic C-Array approach allows use of another compiler for the main microcontroller (Standalone GTM-MCS Compiler Tool Chain Version).

TASKING Build Flow
Supports multiple build strategies



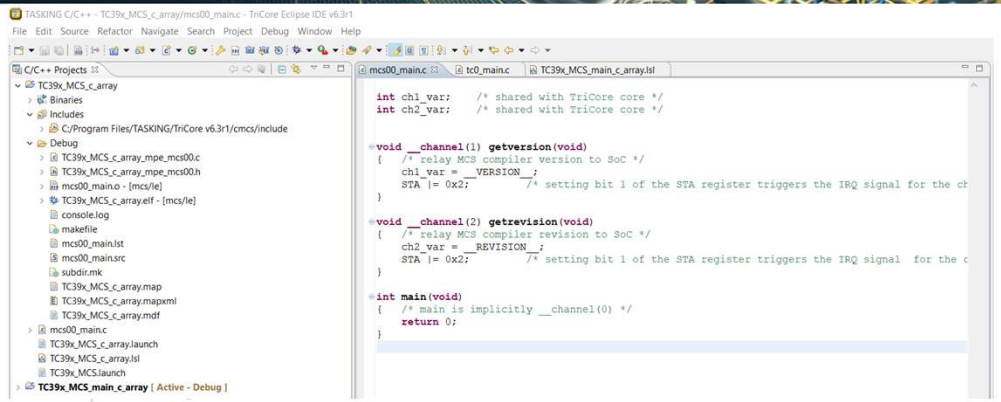
TASKING Build Flow
C-Array Approach

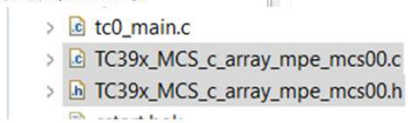
- Two separate Eclipse projects are needed
  - Tricore -> TCxx_MCS_main_c_array
  - MCS-GTM -> TCxx_MCS_c_array

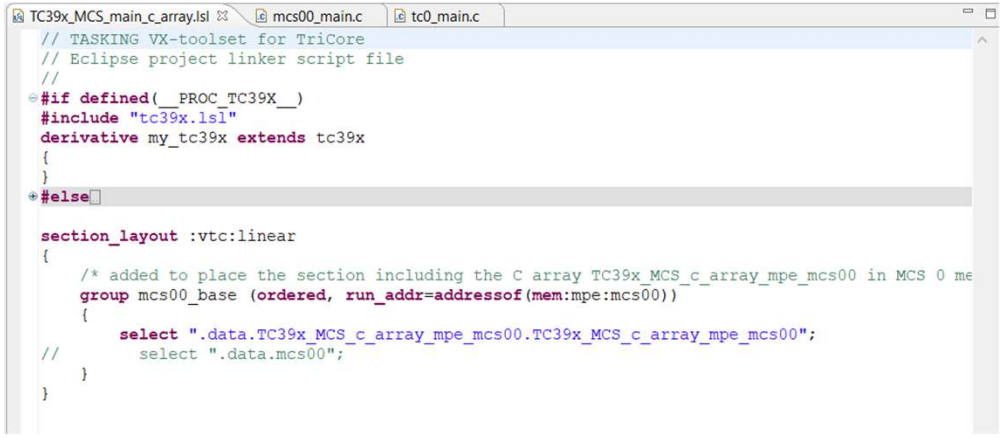
- Building the GTM project 'TCxx_MCS_c_array' creates the following .c and .h files which consist of an initialized array containing the code for a single MCS core.
  - TCxx_MCS_c_array_mpe_mcs00.c
  - TCxx_MCS_c_array_mpe_mcs00.h

- The GTM-MCS source file with initialized array is referenced in the TriCore main project

- The main TriCore project .lsl file includes an entry to place the initialized array in the MCS core RAM Memory







TASKING Build Flow
C Array Approach

- The startup code of the AURIX application copies the MCS code from flash memory to MCS core RAM memory -> _c_init()
- An optional header file can be created, which includes the addresses of global MCS project symbols: **ch1_var** and **ch2_var.**

Aurix Main Source Code

```
/* macro to define the base address of the GTM memory */
#define GTM_BASE_ADDR        0xf0100000
/* macro to define the offset of the MCS00 memory within the GTM memory */
#define GTM_MCS00_OFFSET     0x38000

/* calculating addresses of ch1_var and ch2_var defined in the MCS project for MCS core 0 */
#define mcs00_ch1_var_address (GTM_BASE_ADDR+GTM_MCS00_OFFSET + (TCxx_MCS_C_ARRAY_MPE_MCS00_ch1_var * 4))
#define mcs00_ch2_var_address (GTM_BASE_ADDR+GTM_MCS00_OFFSET + (TCxx_MCS_C_ARRAY_MPE_MCS00_ch2_var * 4))
```

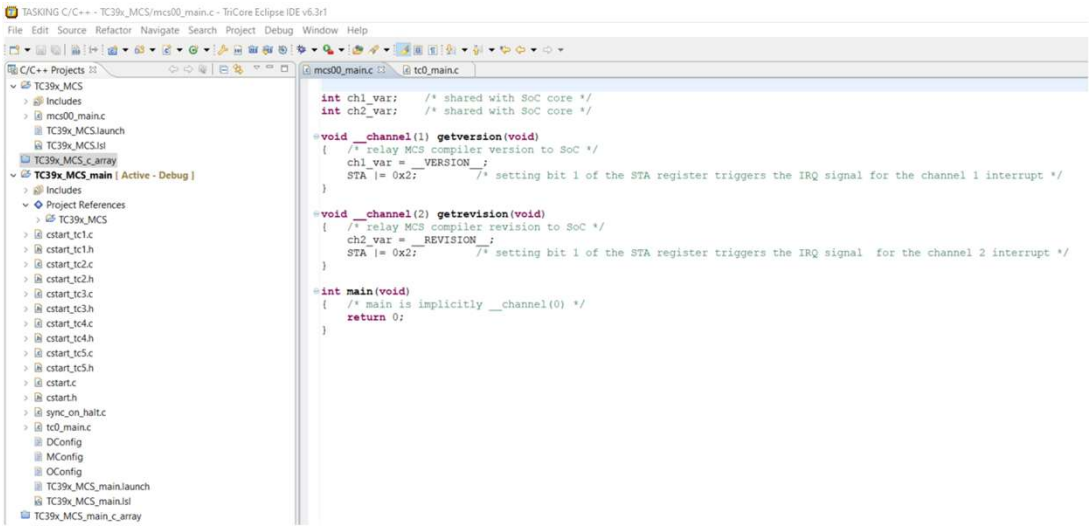
.h file contains GTM addresses. Need to *4 for byte offset of absolute address in TriCore Memory

User responsible for supplying address and offset

Defined in TCcx_MCS_c_array_mpe_mcs00.h
(TCxX_MCS_C_ARRAY_MPE_MCS00_ch1_var)
contains variable offset in MCS memory in words

TASKING Build Flow
Integrated Approach

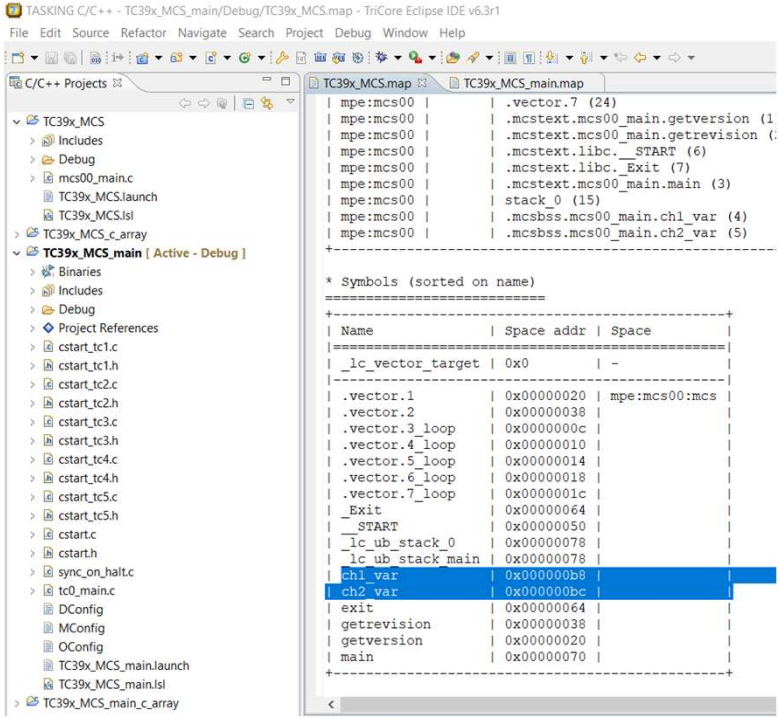
- The MCS core application is a sub project of the AURIX core main project. The MCS source is identical in both approaches.
- The Aurix C startup code initializes the sections belonging to the MCS application in MCS RAM.
- Variables defined in the MCS core project are declared as extern variables in the Aurix application using the '__mcsram' language extension keyword (tool adds mcs core number prefix to symbol name).



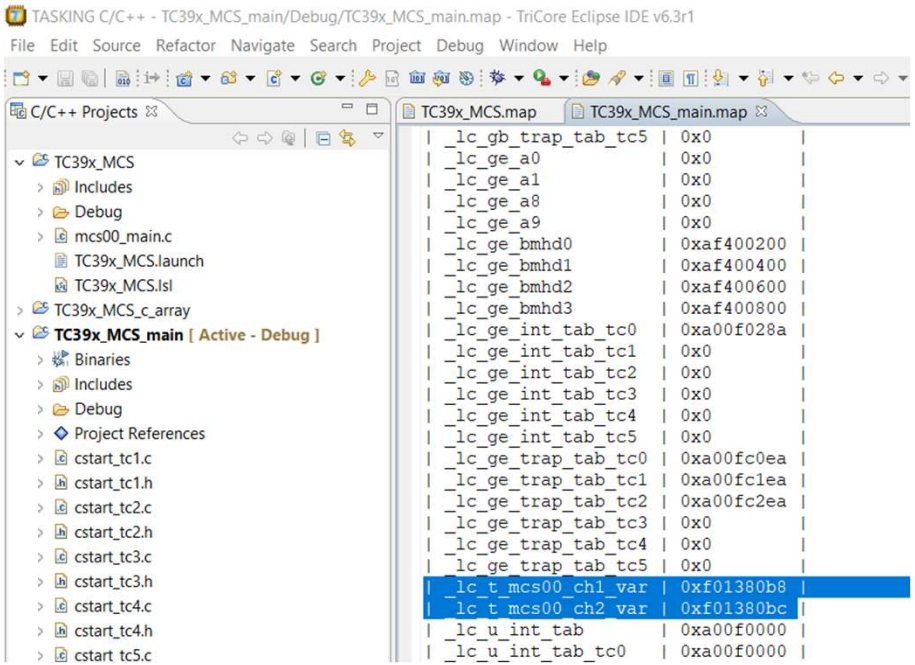

```
/* Global variable definition in the MCS core 0 application */
int ch1_var;
int ch2_var;

/* extern declarations for those variables in the AURIX application */
extern volatile int __mcsram mcs00_ch1_var;
extern volatile int __mcsram mcs00_ch2_var;
```

TASKING Build Flow
Integrated Approach

- Very easy to use, No need for macros. No knowledge of Aurix memory space needed!



- **Two global variables in absolute Address space of GTM/MCS**



- **Per user manual, _lc_t is for __mcsram external variables.**
- **The MCS core number is also provided in the label.**

GTM_BASE_ADDR (0xf0100000) + GTM_MCS00_OFFSET (0x38000) + MCS Address (0x..b8 & 0x..bc)

TASKING GTM Compiler
Why program in C?

- The GTM is initialized/configured by the CPU, this code is (typically) written in C. No need to change to assembler for MCS development resulting in addition complexity.

- Even though the amount of RAM available for the GTM-MCS is limited, you can have up to 12 clusters (i.e., MCSs) resulting in small programs requiring small amounts of RAM.

- The C compiler supports all hardware features of the GTM-MCS. The code produced by the C compiler is on par with handcrafted assembly. (No need to write assembler)

- The software needs to be time deterministic. The compiler creates a list file that shows the exact number of cycles for each assembly instruction that is generated.

TASKING®

Agenda

1. Compiler Tool Chain for SoCs
2. GTM-MCS Compiler Tool Chain integration
3. **ISO26262 Tool Qualifcation**

TASKING GTM-MCS Compiler
ISO26262 Implications

VX-toolset for GTM

- TASKING GTM-MCS Compiler Tool Chain is developed in accordance with ASPICE CL2.
- The GTM-MCS Compiler Tool Chain can be considered as qualified for the development of safety critical software up to and including ASIL B.
- TASKING does not offer a Qualification Kit for GTM-MCS Compiler Tool Chain, because it is typically not needed.
  Rationale:
  - The GTM-MCS memory is limited in size and programs that run on GTM are small.
  - Higher ASIL's require rigorous testing (100% MC/DC coverage) of application software.
  - Therefore, a high probability that software errors(User error or compiler) are detected by the user's test procedures.
  - Per ISO26262-8:11, GTM toolset has a TD1 and TCL1 -> Conclusion: Compiler Qualification may not required.

TASKING GTM-MCS Compiler Tool Chain
Key Takeaways

- TASKING's GTM-MCS Tool Chain supports 1st, 2nd, 3rd and 4th generation of BOSCH GTM-IP.
- Supports all GTM & MCS features from C-Level (No need for assembly level programming)
- Compiler list file shows the generated assembly code annotated with timing info (No need to deduce execution time from analyzing assembly code)
- Compiler toolset includes an instruction set simulator-based C-level debugger
- GTM-MCS Compiler Tool Chain is integrated in the TASKING Development Environment for TriCore3x/4x.
- TASKING offers Stand alone GTM-MCS Compiler Tool Chain as well
- The TASKING GTM-MCS Compiler can be used in combination with any third-party toolset for CPU code development (Standalone GTM-MCS Compiler Tool Chain Version).
- TASKING GTM-MCS Compiler Tool Chain is developed in accordance with ASPICE CL2.

VX-toolset for GTM

TASKING

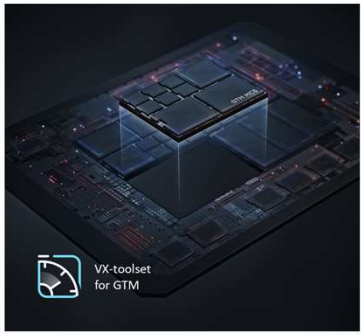
STAY CONNECTED

- **Contact us directly**
  www.tasking.com/contact
- **Follow us on LinkedIn**
  www.linkedin.com/company/tasking-inc
- **Follow us on WeChat**
- **Sign up to get the latest news from TASKING®**
  www.tasking.com/subscribe
- **Upcoming webinars, events and exhibitions**
  www.tasking.com/events

MEET US AT EXHIBITION

Demo at TASKING table:

TASKING presenting its SmartCode development environment for Infineon TC4x including BOSCH GTM-MCS



TASKING®

THANK YOU