AUTOSAR™

Why AUTOSAR Fails so Often

A report from a firefighter for Embedded Software Timing

Version 3

Peter Gliwa

12th June 2024

15th AUTOSAR Open Conference

Tokyo

Excuse me …?

Why AUTOSAR Fails so Often ???

- Wait a minute, we **are** at an AUTOSAR Open Conference.
- Is this an offense?
- A revolution?
- No, with this talk, I would like to
  1. Explain common reasons for timing problems and memory problems
  2. Point out solutions

Source: Wikipedia

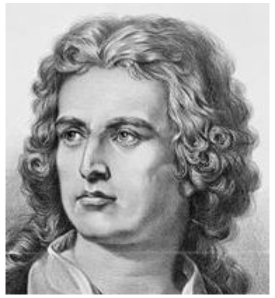
This talk is a report from someone who saw many projects struggle with AUTOSAR.

Let’s talk about German culture

With ‘German culture’,
I do *not* mean

- Johann Sebastian Bach
- Ludwig van Beethoven
- Friedrich Schiller
- Johann Wolfgang von Goethe

Source: Wikipedia

Source: Wikipedia

Source: Wikipedia

Source: Wikipedia

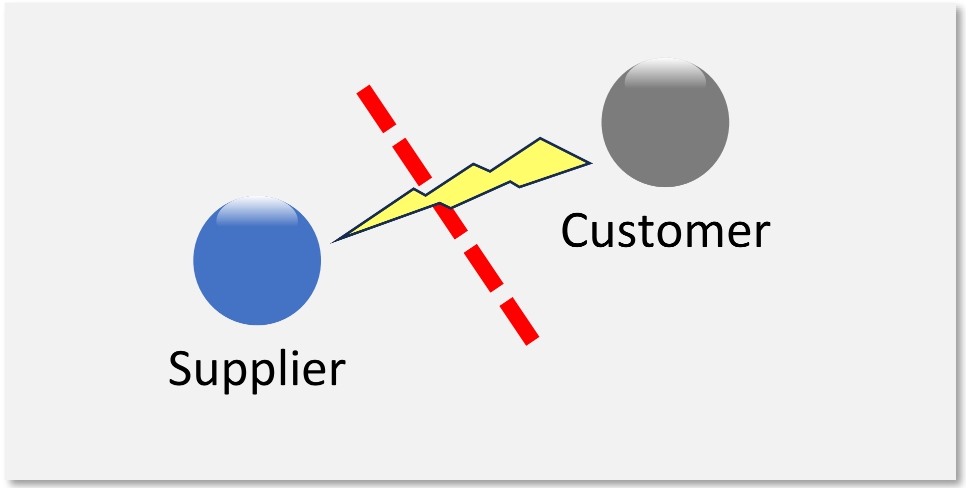
Let’s talk about the German culture of

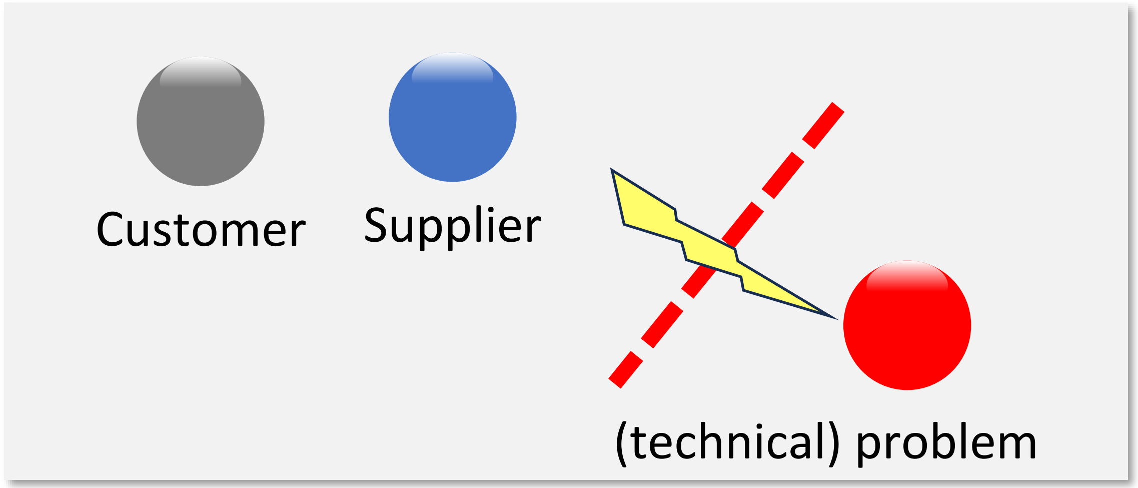
- working together
- communicating

The following slide describes a certain tendency, please do not take it too seriously!

Creator: Dan Dalton | Credit: Getty Images/Caiaimage

Where is the opponent?





<u>Scenario seen around the globe</u>

Customer is superior

- Has the power
- Must not be criticized

Sometimes Supplier and Customer act like opponents

<u>Scenario with German engineers</u>

Customer and Supplier work ***together*** to solve the problem.

If honest and very direct communication helps solving the problem, so be it.

Contents

20 years GLIWA

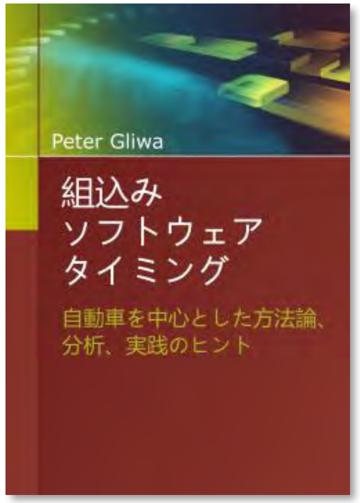
AUTOSAR
GLIWA is a development member

Timing/stack/memory analysis

German company, **~60** empl., **20%** annual growth

Peter Gliwa (owner & CEO)

- Actively coaching international automotive OEMs and Tier-1s
- AUTOSAR work-package leader of work-package "ARTI" and document owner of AUTOSAR TR "Timing Analysis"
- Previously with ETAS / BOSCH
- Author of book "Embedded Software Timing" (DE, EN, CN, KR, JP)



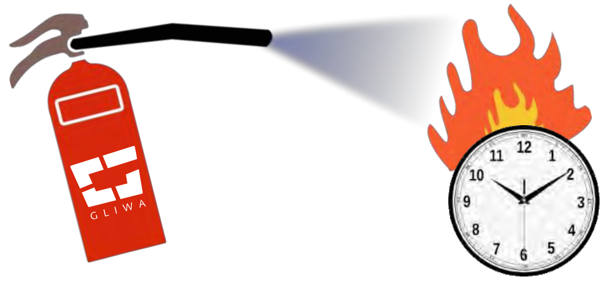
BMW Group, TRW, DIAS 联创汽车电子有限公司, DAIMLER, ZF, PORSCHE, UAES, HELLA, DELPHI, Audi Vorsprung durch Technik, GEELY, JLR, HYUNDAI, DENSO, 上汽集团 SAIC MOTOR, 蔚来, iav, Mando, Panasonic, METTLER TOLEDO, HITACHI, 潍柴动力 WEICHAI POWER, BOSCH, 中国一汽, EvoBus, infineon, HYUNDAI AUTRON, MAGNA POWERTRAIN, 长城汽车, Inovance, Continental, NUCELLSYS, EADS, ZF SACHS, nexteer AUTOMOTIVE, 宁德时代, BAIC BJEV 北汽新能源, ZF Lenksysteme

So what exactly is the problem?

Many projects run into timing issues

- System overloaded, yet functionality to add
- Sporadic crashes
- Weird functional behavior
- Communication issues
- …

It is often not clear that a timing problem is causal!



Root-cause No.1 is an inefficient AUTOSAR configuration.

The trend is: it is getting worse (despite all the education and many task forces with successful outcome)

Common AUTOSAR problem I: RTE overhead

1

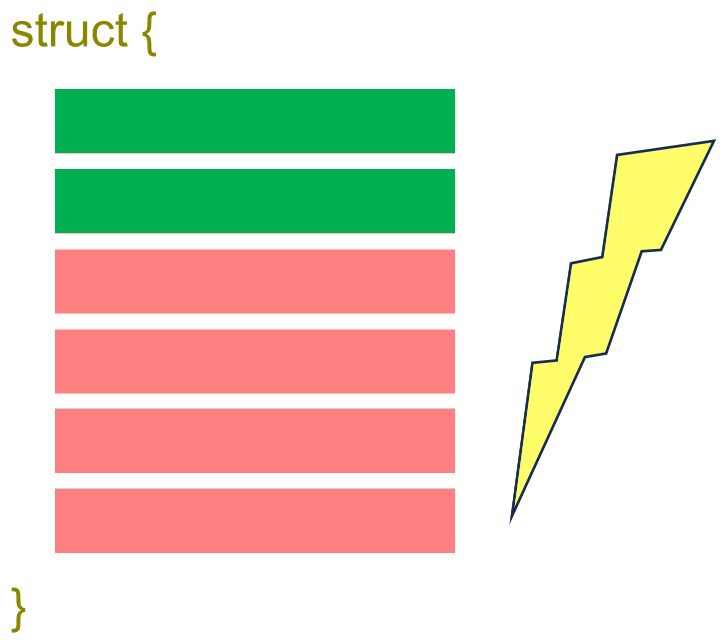
Data consistency in preemptive systems

Example: sharing data between preemptive tasks

Step 1: Task A starts reading data from a struct.

Step 2: Task B preempts Task A and updates the data structure.

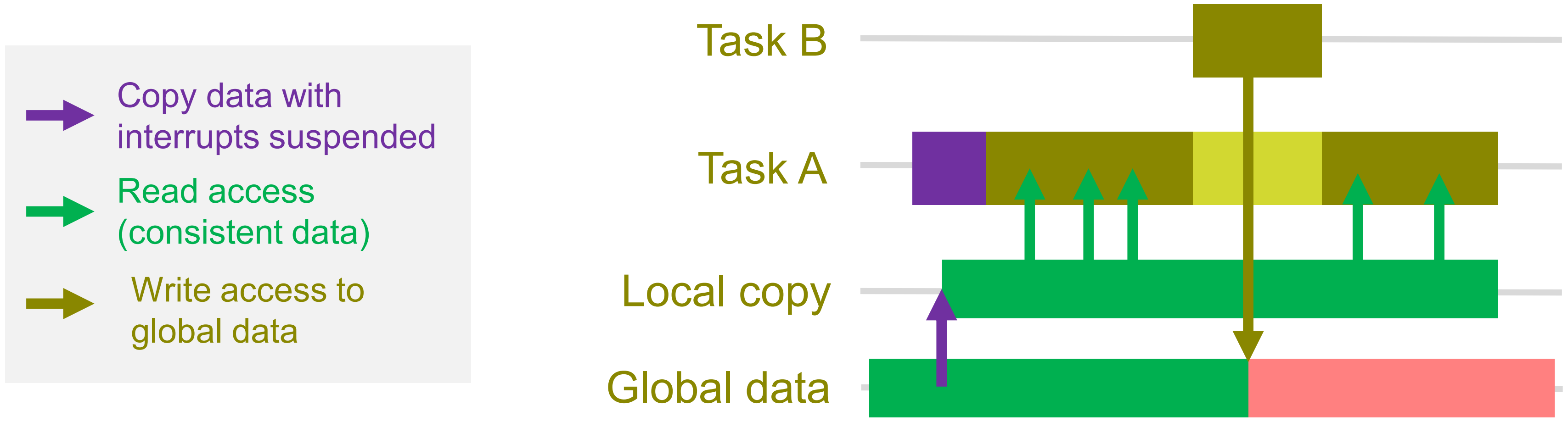
Step 3: When Task A resumes execution, it uses inconsistent data (partly old, partly new)



Common AUTOSAR problem I: RTE overhead

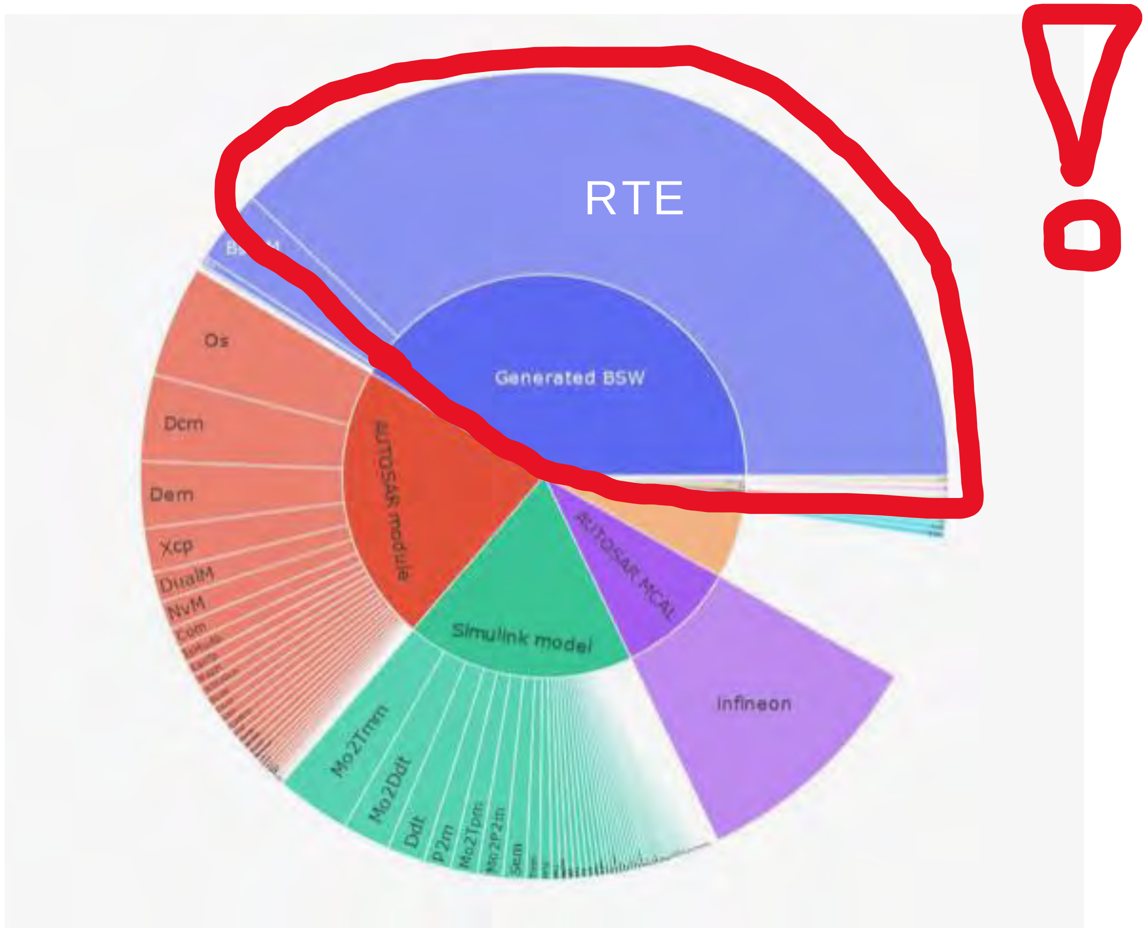
Data consistency through copies (RTE implicit com.)

- At the beginning of Task A, copies of all critical data are created.
  → 'critical' means here: data gets accessed from code with higher priority, such as Task B.
- The actual copy process is protected typically through interrupt suspension or even spinlocks.
- Task A uses the copy of the data only.
- If Task A also *writes* shared data, it is copied back (not shown below) with protection.


Copy data with interrupts suspended
Read access (consistent data)
Write access to global data
Task B
Task A
Local copy
Global data

Common AUTOSAR problem I: RTE overhead

AUTOSAR RTE: in many projects a high-consumer!



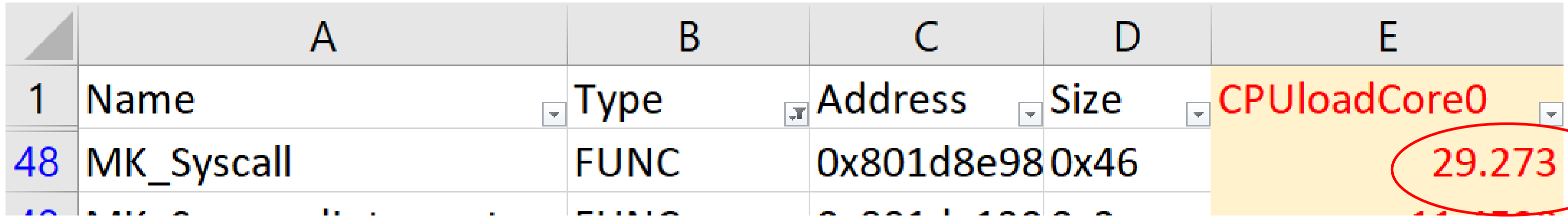
Example I
Customer project (platform)
19 vehicle models affected

Status today: platform optimized, all good now

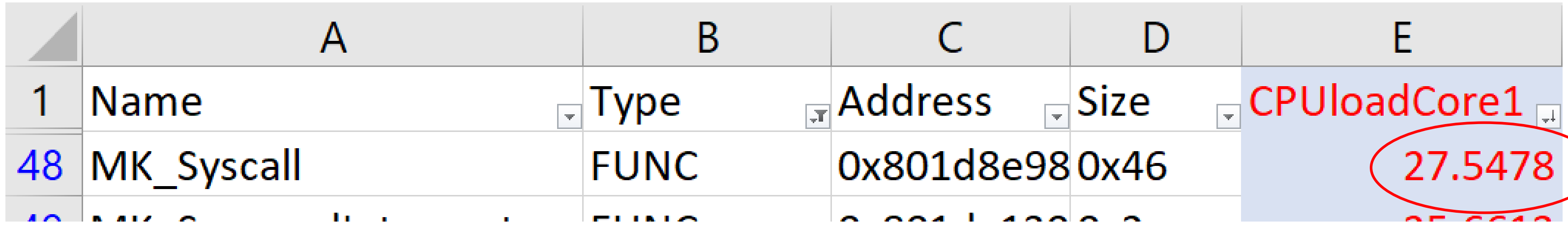
Common AUTOSAR problem I: RTE overhead

AUTOSAR RTE: in many projects a high-consumer!

Example II
Customer project (platform)
6 vehicle models affected



	A	B	C	D	E
1	Name	Type	Address	Size	CPUloadCore0
48	MK_Syscall	FUNC	0x801d8e98	0x46	29.273



	A	B	C	D	E
1	Name	Type	Address	Size	CPUloadCore1
48	MK_Syscall	FUNC	0x801d8e98	0x46	27.5478

Nearly 30% CPU-load on each core is spent for disabling/enabling interrupts!

223,312 syscalls per second (!) on core 0 and 343,284 on core 1.

Status today: optimization ongoing

Common AUTOSAR problem I: RTE overhead

AUTOSAR RTE: in many projects a high-consumer!

Example III
Customer project (platform)
29 vehicle models affected

Multicore AURIX overloaded.
Option: use more powerful AURIX device.

Estimated cost: 160m €

<u>**€ 160,000,000.00 !**</u>
¥ 26,001,600,000.00

Status today: platform optimized, all good now

Common AUTOSAR problem I: RTE overhead

Thoughts leading to a solution

Preemptions can cause inconsistent data (cf. earlier slide)
→ no preemption = no problem

AUTOSAR RTE specification R20-11:

Copy semantic
Copy semantic means, that the accessing entities are able to read or write the 'copied' data from their execution context in a non concurrent and non preempting manner. If all accessing entities are in the same preemption area this might not require a real physical data copy.

Example: Task A and Task B use implicit communication. When assigned the same priority, no copy is required for the data only these two tasks access.

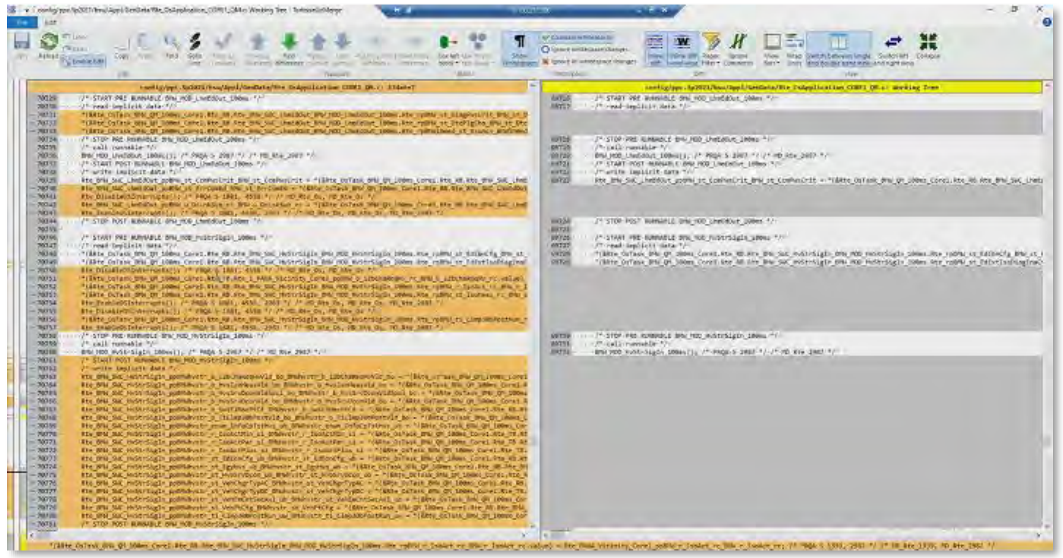
Common AUTOSAR problem I: RTE overhead

What is the solution?

Use as few priorities as possible.

Even better: use internal resources (cf. Priority Ceiling Protocol) to avoid preemption.
This allows to control the order in which several tasks in ‘ready’ state tasks are started through their priority.

Hint: assign all tasks same priority to evaluate the optimization potential (do not flash/use this software).



Starting point of the optimization: thousands of copies!

After the optimization: lots of code/data vanishes!

2

Common AUTOSAR problem II: ECC Tasks

Yet another customer example

```
void ErrorHook(StatusType status)
{
  switch(status) {
    case E_OS_LIMIT:
      /* failed task activation
       * as a result of an overload
       * situation  */
      SystemReset();
      break;
    default:
      break;
  }
}
```

Customer:	"We face sporadic communication issues."
GLIWA:	"Is the system overloaded?"
Customer:	"No. If it was, the ErrorHook would trigger a reset."
GLIWA:	"Are you using the standard RTE set-up?"
Customer:	"Yes."
GLIWA:	"Ouch!"

User's intention: Reset when system is overloaded

BUT: ErrorHook does not get called when an event is re-triggered

ErrorHook: called by the OS, implemented by the user (of the OS)

Common AUTOSAR problem II: ECC Tasks

What is defined in the RTE specification?

```c
TASK(Task_B)
{
  EventMaskType ev;
  for(;;)
  {
    (void)WaitEvent(    Rte_Ev_Cyclic2_Task_B_0_10ms |
                        Rte_Ev_Cyclic2_Task_B_0_5ms );

    (void)GetEvent(Task_B, &ev);

    (void)ClearEvent(ev & ( Rte_Ev_Cyclic2_Task_B_0_10ms |
                            Rte_Ev_Cyclic2_Task_B_0_5ms ));

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_10ms) != (EventMaskType)0)
    {
      CanNm_MainFunction();
      CanSM_MainFunction();
    }

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_5ms) != (EventMaskType)0)
    {
      CanTp_MainFunction();
      CanXcp_MainFunction();
    }
}
```

10ms runnables

5ms runnables

RunnableEntity RE6 is activated by DataReceivedEvent DR3.
RunnableEntity RE7 is activated by DataReceivedEvent DR4.
DataReceivedEvent DR2, DR3 and DR4 references the same dataElement. Evaluation order of the RTEEvents shall be T1, T2, T3, DR1, DR2, DR3 then DR4.
RE2 and RE3 shall be monitored as a group.
RE6 and RE7 shall be monitored as a group.

Possible RTE configuration:
RE1 is mapped to OsTask TaskA with a reference to OsEvent EvtA and RtePositionInTask equal to 1.
RE2 is mapped to OsTask TaskB but virtually mapped to TaskA with a reference to OsEvent EvtA and RtePositionInTask equal to 2.
RE3 is mapped to OsTask TaskB but virtually mapped to TaskA with a reference to OsEvent EvtA and RtePositionInTask equal to 3.
RE4 is mapped to OsTask TaskA with a reference to OsEvent EvtB and RtePositionInTask equal to 4.
RE5 is mapped to OsTask TaskA with a reference to OsEvent EvtC and RtePositionInTask equal to 5.
RE6 is mapped to OsTask TaskC but virtually mapped to TaskA with a reference to OsEvent EvtC and RtePositionInTask equal to 6.
RE7 is mapped to OsTask TaskC but virtually mapped to TaskA with a reference to OsEvent EvtC and RtePositionInTask equal to 7.

Possible RTE implementation:
RTE starts cyclic OsAlarm with 100ms period.
This OsAlarm is configured to set EvtA.
RTE set EvtB and EvtC according to the callbacks from COM.
Full preemptive scheduling is configured for Task A.
TaskB priority = TaskC priority = TaskA priority + 1

```c
void TaskA(void)
{
   EventMaskType Event;

   while(1)
   {
     WaitEvent(EvtA | EvtB | EvtC);
     GetEvent(TaskA, &Event);
     if (Event & EvtA)
     {
        ClearEvent(EvtA);
        RE1();
        ActivateTask(TaskB);
     }
     else if (Event & EvtB)
     {
        ClearEvent(EvtB);
        RE4();
     }
     else if (Event & EvtC)
     {
        ClearEvent(EvtC);
```

AUTOSAR RTE specification

Common AUTOSAR problem II: ECC Tasks

Interpreting RTE scheduler

```
TASK(Task_B)
{
  EventMaskType ev;
  for(;;)
  {
    (void)WaitEvent(    Rte_Ev_Cyclic2_Task_B_0_10ms |
                        Rte_Ev_Cyclic2_Task_B_0_5ms );

    (void)GetEvent(Task_B, &ev);

    (void)ClearEvent(ev & ( Rte_Ev_Cyclic2_Task_B_0_10ms |
                            Rte_Ev_Cyclic2_Task_B_0_5ms ));

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_10ms) != (EventMaskType)0)
    {
      CanNm_MainFunction();
      CanSM_MainFunction();
    }

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_5ms) != (EventMaskType)0)
    {
      CanTp_MainFunction();
      CanXcp_MainFunction();
    }
}
```

10ms runnables

5ms runnables

Non terminating ECC task

This is how you implement a thread in POSIX, not TASKs in OSEK!

AUTOSAR RTE adds another layer of scheduling violating the OSEK idea and the 'keep it simple' paradigm.

If you now add a regular OSEK event, it **REALLY** gets messy!

Common AUTOSAR problem II: ECC Tasks

Back to customer example

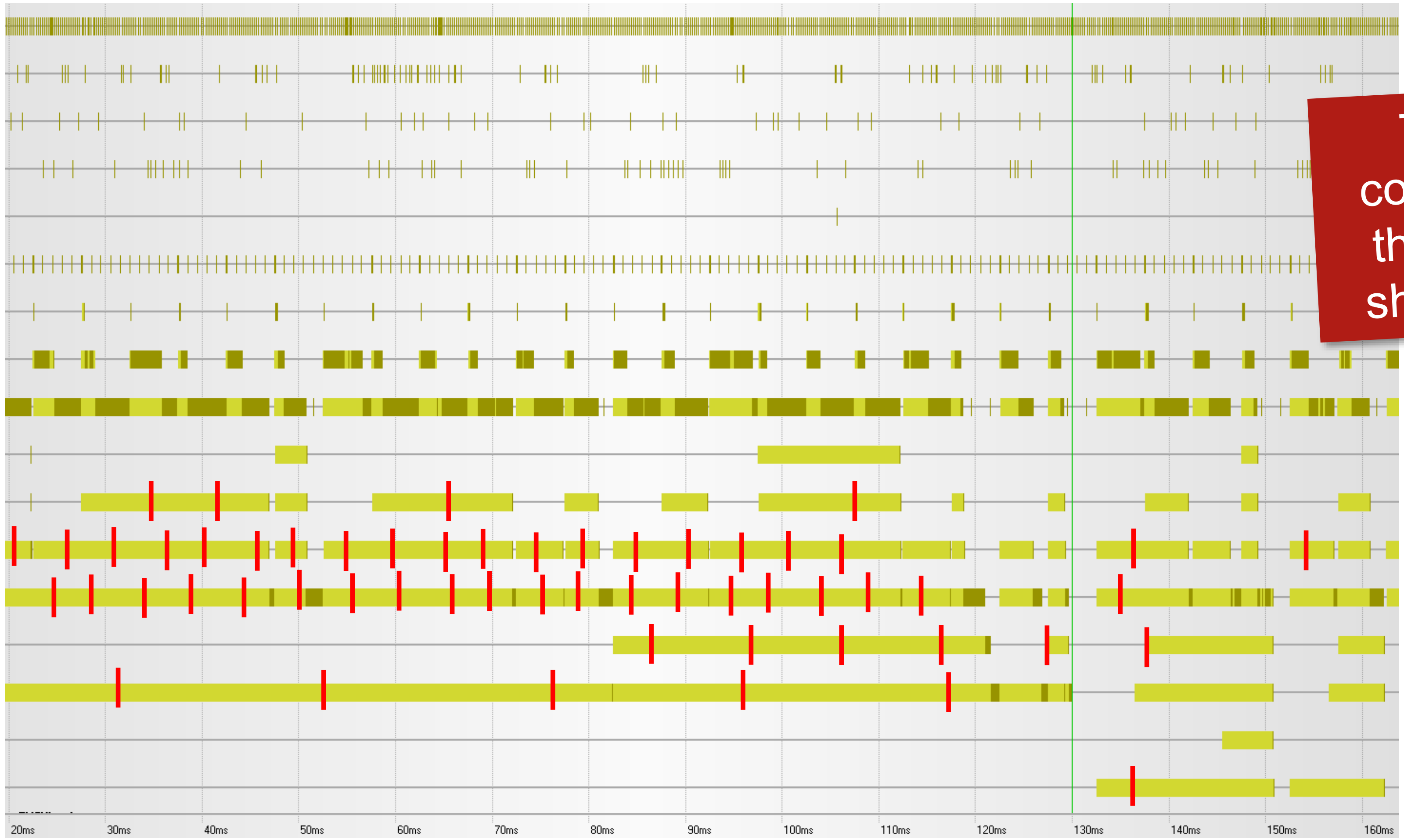
```
TASK(Task_B)
{
  EventMaskType ev;
  for(;;)
  {
    (void)WaitEvent(    Rte_Ev_Cyclic2_Task_B_0_10ms |
                        Rte_Ev_Cyclic2_Task_B_0_5ms );

    (void)GetEvent(Task_B, &ev);

    (void)ClearEvent(ev & ( Rte_Ev_Cyclic2_Task_B_0_10ms |
                            Rte_Ev_Cyclic2_Task_B_0_5ms ));

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_10ms) != (EventMaskType)0)
    {
      CanNm_MainFunction();      10ms runnables
      CanSM_MainFunction();
    }

    if ((ev & Rte_Ev_Cyclic2_Task_B_0_5ms) != (EventMaskType)0)
    {
      CanTp_MainFunction();      5ms runnables
      CanXcp_MainFunction();
    }
}
```

Let's assume this runnable with a desired period of 10ms runs for 26ms (which it sometimes did in the customer's project).

Event `Rte_Ev_Cyclic2_Task_B_0_10ms` gets re-triggered, ErrorHook does *not* get called.

Common AUTOSAR problem II: ECC Tasks

And here the impact…



The customer was completely unaware of the overload scenario shown in this T1 trace.

Red lines: here the `ErrorHook` was expected to fire.

Status today: project optimized, all good now

Common AUTOSAR problem II: ECC Tasks

Simple solution: use BCC1

```
TASK(Task_B_10ms)
{
  CanNm_MainFunction();
  CanSM_MainFunction();
  TerminateTask();
}

TASK(Task_B_5ms)
{
  CanTp_MainFunction();
  CanXcp_MainFunction();
  TerminateTask();
}
```

BCC1: Straight forward and simple

`ErrorHook` works as expected.

Perfect basis for cooperative multitasking

By the way: never use multiple task activations (BCC2). It tends to be a dirty workaround for an overloaded system.

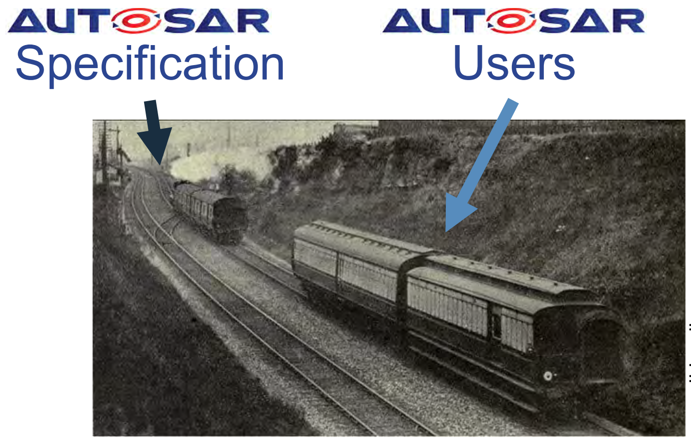
Summary

Far too many projects using AUTOSAR are inefficient.

My recommendation: do not unlink from the real world!

More (AUTOSAR features) is not naturally *better*!

Introduce safety/efficiency subset/conformance class?
Remove some of the problematic features?

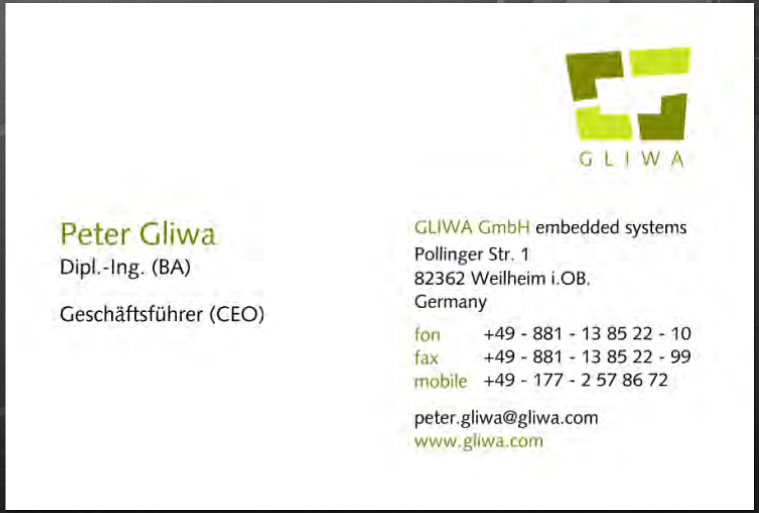


en.wikipedia.org

Source: BSE-Galerie

AUTOSAR™

ありがとう
Thank you!



GLIWA

Peter Gliwa
Dipl.-Ing. (BA)

Geschäftsführer (CEO)

GLIWA GmbH embedded systems
Pollinger Str. 1
82362 Weilheim i.OB.
Germany

fon	+49 - 881 - 13 85 22 - 10
fax	+49 - 881 - 13 85 22 - 99
mobile	+49 - 177 - 2 57 86 72

peter.gliwa@gliwa.com
www.gliwa.com

BOSCH Continental Ford gm STELLANTIS TOYOTA VOLKSWAGEN GROUP