# [TriCore] Debugging Safety-Critical Applications

2026-08-19 - Comments (0) - TriCore

When debugging with SMU enabled, unwanted security/safety alerts may be reported. To understand the cause of a reported alarm, please consult the Infineon documentation for your chip. Nevertheless, the most common issues are discussed below.

## 1. Bus Errors

If the SMU is set up strict, some debugger accesses might cause SMU Alarms (bus errors). The reason is that the chip can not differentiate between illegal and debugger accesses.

The following situations are to be considered:

### 1.1. Bus Errors Due to Debugger Access to Non-Mapped Address Range or Non-Initialized Peripheral Modules

The solution is to use the command `MAP.DenyAccess <addressrange>` to protect the address ranges from debugger access.

### 1.2. Bus Errors Due to Debugger Access to Erased PFLASH

When the debugger tries to access an erased PFLASH range, a bus error is returned due to the incorrect ECC of erased PFLASH.

In this situation, the user can also use the command `MAP.DenyAccess <addressrange>` to protect the erased PFLASH address ranges from debugger access.

An alternative solution is to program the unused PFLASH ranges with a pattern. This results in programming the correct ECCs for the unused PFLASH ranges.

Infineon recommends filling unused PFLASH with an invalid opcode. This will ensure that an IOPC (Illegal Opcode) trap is generated if the target attempts to execute from PFLASH sectors that are supposed to be empty. For example, AURIXTM TC3xx User's Manual suggests filling the empty PFLASH regions with the pattern 0x36363636 (see AURIXTC3XX_um_part1_v2.0.pdf, section 5.3.4.7 Invalid Opcode).

The following example shows how to achieve this on a TC36x device with a 4 MB PFLASH:

```
; 4 MB PFLASH - TC36x 64F
&PFLASH_RANGE="0xA0000000--0xA04FFFFF"

; Flash declaration
DO ~~/demo/tricore/flash/tc36x.cmm PREPAREONLY

; Enable FLASH programming
FLASH.ReProgram ALL /Erase

; Fill PFLASH with illegal opcode pattern
Data.Set &PFLASH_RANGE %Long 0x36363636

; Load the application binary
Data.LOAD.auto <your_program_file>
```

```
; Program the changes to the target
FLASH.ReProgram off
```

### 1.3. Debugger Write Access Attempts to Read-Only Memory

Bus error is also returned if the debugger attempts a write access to read-only memory.

This occurs, e.g., when the user attempts to set a program breakpoint on PFLASH without specifying the breakpoint type. To preserve the onchip comparators, TRACE32 tries first to set software breakpoints by attempting to patch the address of the program breakpoint by a debug instruction. TRACE32 R.2019.09 or newer uses an improved FLASH address range model to decide whether a program breakpoint is to be implemented as an onchip breakpoint.

To prevent the debugger from trying to set software breakpoints on PFLASH consider the following alternatives:

• Use `Break.CONFIG` to configure the behavior of different breakpoint types as well as their scope.

• Use `MAP.BOnchip <addressrange>` to force the debugger to set onchip breakpoints in the respective memory range.

Note

It is to be considered that `List.auto` window also reads before the displayed code range for the correct functioning of the dis-assembler. This is often a problem at the reset vector given that its address is set at the start of PFLASH (following a non mapped memory range) or following an erased PFLASH range.

# 2. TriCore Watchdog Timer

By default, the TriCore watchdog timer is disabled when OCDS is enabled, i.e., when the debugger is attached. Some safety tests (e.g. SafeTLib) report this as a failure.

The solution is to keep the TriCore watchdog timer enabled and link it to the suspend bus for synchronous start/stop with the TriCore cores.

Set:

```
SYStem.Option.WDTSUS ON
SYStem.Option.PERSTOP ON
```

# 3. TRACE32 Cache Handling

Cache evaluation and cache inspection require that TRACE32 can read the cache tags and contents. AURIX CPUs allow reading this by mapping the cache into the CPU address space using the MTU. As soon as the cache is mapped by TRACE32, the CPU will clean the cache content due to security reasons. This might be reported as a security/safety alarm.

• Cache mapping can be disabled using the command `SYStem.Option.MAPCACHE OFF` in case of unwanted security/safety alerts.

• TRACE32 R.2020.09 or newer avoids safety alarms originating from mapping the cache into memory by temporarily disabling these alarms.