

UDE Code Coverage

Proving the Quality of Tests Through Non-Intrusive Code Coverage

Assuring high software quality in electronic control units is a challenging task. Extensive and, above all, suitable tests are essential to achieve this.

Code coverage is generally considered a very meaningful metric for assessing test quality. Relevant standards for functional safety of electronic systems, e.g. ISO 26262, therefore require evidence and documentation of the code coverage achieved by the software test.

In particular, for real-time critical multicore systems, typical control-flow oriented methods that use compiler-assisted code instrumentation for coverage measurement reach their limits very quickly.

UDE Code Coverage support is a trace-based, non-intrusive method for determining statement coverage (C0 coverage) and branch coverage (C1 coverage) even with optimized code. The measurement is based on instruction traces provided by hardware trace channels.

Code Coverage Analysis

Different views are provided by UDE to enable interpretation of code coverage results.

The **Code Coverage Window** provides details of the coverage on 4 levels:

- Core minimum-coverage
- Function coverage
- Source line coverage
- Machine instruction coverage

For C0 and C1 coverage the results are displayed as a percentage bar chart.

Additional functions include

- Sorting functions by name, start address, coverage value
- Pre-filtering the trace for analysis of specific functions
- Accumulation of coverage data over multiple trace recordings and sessions

Highlights

- Non-intrusive, no change of run-time behavior
- No code-instrumentation required
- Compact presentation of coverage results
- Meaningful code coverage reports
- Export of reports into different formats
- Scripting and automation support by UDE Object Model

The **Program Window** displays per-core statement coverage information as line markers for completely covered, partially covered, and uncovered source lines and machine instructions.

| | Start | End | File | Line | Line Coverage | MCB Coverage |
|--------------------------------------|------------|------------|--------------|------|---------------|--------------|
| Core0 | | | | | 0,32% | 0,00% |
| Task_200ms | 0x800002B4 | 0x800002C1 | main.c | 28 | 20,00% | 100,00% |
| sched_RunTask | 0x800003AC | 0x800004D1 | ched_swirq.i | 82 | 31,07% | 25,00% |
| sched_Task2 | 0x800004E6 | 0x800004EF | ched_swirq.i | 120 | 25,00% | 100,00% |
| SCHED_PeriodicExec | 0x80000824 | 0x80000B13 | ched_swirq.i | 192 | 39,32% | 43,75% |
| { | 0x80000824 | 0x80000827 | ched_swirq.i | 192 | 100,00% | 100,00% |
| if(sched_Data.Running) | 0x80000828 | 0x80000839 | ched_swirq.i | 193 | 100,00% | 50,00% |
| sched_Data.CycleCnt++; | 0x8000083A | 0x80000859 | ched_swirq.i | 196 | 100,00% | 100,00% |
| for(i=0;i<SCHED_TASK_COUNT;i++) | 0x8000085A | 0x80000861 | ched_swirq.i | 197 | 100,00% | 100,00% |
| if(sched_Data.aTaskData[i].CntDown>0 | 0x80000862 | 0x8000087F | ched_swirq.i | 199 | 100,00% | 50,00% |
| MOVH d15, 0xD000 | 0x80000862 | 0x80000865 | ched_swirq.i | 199 | 100,00% | 100,00% |
| ADDI d2, d15, 0x300 | 0x80000866 | 0x80000869 | ched_swirq.i | 199 | 100,00% | 100,00% |
| LD.W d15, [a14]-0x4 | 0x8000086A | 0x8000086D | ched_swirq.i | 199 | 100,00% | 100,00% |
| MUL d15, d15, 0x2C | 0x8000086E | 0x80000871 | ched_swirq.i | 199 | 100,00% | 100,00% |
| ADD d15, d2 | 0x80000872 | 0x80000873 | ched_swirq.i | 199 | 100,00% | 100,00% |
| ADDI d15, d15, 0x14 | 0x80000874 | 0x80000877 | ched_swirq.i | 199 | 100,00% | 100,00% |

```
U:\UdeDemo\src\worker.c 0x8000282E | Ln 124 | X
void WORKER_TaskFunc_200ms(void* pvDa
{
  // forward task interrupt to work
  PWorkerData pData=(PWorkerData)pv
  switch(pData->CoreId)
  {
    case 1:
      SFR_SRC_GPSR12|=(1<<26);
      break;
    case 2:
      ...
  }
}

U:\UdeDemo\src\ModTab.c 0x8000180C | Ln 107 | X
void MODTAB_Task_200ms(PModTabData pD
{
  0x8000180C 40 AE MOV AA
  0x8000180E 20 10 SUB A
  0x80001810 B5 E4 F4 FF ST A
  int i;
  for(i=0;i<pData->ModCnt;i++)
  0x80001814 82 0F MOV
  0x80001816 59 EF FC FF ST.W
  0x8000181A 3C 22 J
  ...
}
```

Trace Data Collection for Code Coverage

The code coverage measurement is based on instruction trace data. The trace data are generated by commonly used hardware trace systems: Infineon MCDS, NEXUS 5001™ class 3, and Arm® CoreSight™ ETM, PTM. Trace systems with on-chip trace buffers are supported as well as high bandwidth trace interfaces (e.g. HSSTP, AURORA, NEXUS 5001™ and Arm® parallel trace).

UDE Code Coverage

Code Coverage Reports and Documentation

To meet the requirements for evidence of the performed code coverage measurements within the overall software quality assurance process, complete reports with all details must be generated. These reports are essential for the subsequent traceability of the performed measurements and their interpretation. The reports include:

- Used target application
- Date of measurement
- Overview about function coverage results
- Line based coverage of source files
- Address based coverage of machine instructions
- List of uncovered instructions and branches.

Report Export

- Formats: HTML, XML, CSV and plain text files
- XML and text reports contain the same information as the HTML report
- CSV reports are user configurable in terms of different levels of details, ranging from function level only to detailed machine instruction level

Automation Support

The code coverage analysis is available via the UDE Object Model, as trace stream-based service. The interfaces provide fine grained control over each aspect of the coverage analysis, including configuration, control, access to results and creating of reports. User-controlled or automatic start of analysis or creating of reports and storage exports are provided.

Coverage overview about function ranges:

| Range or function name | Source name | Statement coverage in % | MCB coverage in % | Remarks |
|------------------------|---------------|-------------------------|-------------------|---------|
| Task_10ms | main.c | 100 | 100 | |
| sched_RunTask | sched_swirq.c | 75 | 50 | |
| sched_Task0 | sched_swirq.c | 100 | 100 | |
| UDEDEMO_Task_10ms | UdeDemo.c | 100 | 100 | |
| MODTAB_Task_10ms | ModTab.c | 100 | 100 | |
| MODTAB_Task_200ms | ModTab.c | 78 | 75 | |
| WORKER_TaskFunc_10ms | worker.c | 88 | 50 | |
| COUNTER_TaskFunc_10ms | counter.c | 100 | 100 | |
| DEMO_TaskFunc_10ms | demo.c | 83 | 50 | |
| DEMO_TaskFunc_200ms | demo.c | 87 | 50 | |
| Task_10ms_WorkLoad | load.c | 100 | 100 | |
| LOAD_TaskFunc_10ms | load.c | 100 | 50 | |
| Task_200ms_WorkLoad | load.c | 85 | 75 | |
| LOAD_TaskFunc_200ms | load.c | 86 | 50 | |

Code Coverage Function Range Task_10ms

| Root source module path | Root source name | Overall number of source lines | Start address | Length of range | Statement coverage in % | MCB coverage in % |
|---|------------------|--------------------------------|---------------|-----------------|-------------------------|-------------------|
| u:\ude-test-and-verify\samples\pls\TriCore3\TriBoard_TC39x\UdeDemo\src\main.c | main.c | 3 | 0x80000298 | 0xE | 100 | 100 |

Overview about included Source Modules:

| Source module index | Source name | Source module path | Number of source lines |
|---------------------|-------------|---|------------------------|
| 0 | main.c | u:\ude-test-and-verify\samples\pls\TriCore3\TriBoard_TC39x\UdeDemo\src\main.c | 3 |

Coverage Overview about Source Line Ranges:

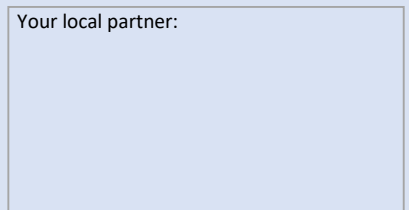
| Source module index | Line number | Start address | Range length | Unreached instructions | Partly covered instructions | Statement coverage in % | MCB coverage in % | Source Line(s) | Comment |
|---------------------|-------------|---------------|--------------|------------------------|-----------------------------|-------------------------|-------------------|----------------------|---------|
| 0 | 18 | 0x80000298 | 0x8 | 0 | 0 | 100 | 100 | | |
| 0 | 19 | 0x800002A0 | 0x4 | 0 | 0 | 100 | 100 | code00c_task_10ms(); | |
| 0 | 20 | 0x800002A4 | 0x2 | 0 | 0 | 100 | 100 | | |

If you have any questions about our products, please feel free to contact us:

PLS Programmierbare Logik & Systeme GmbH
 Technologiepark Lauta
 D-02991 Lauta
 Germany
 Phone: + 49 35722 384 - 0

PLS Development Tools
 10080 N. Wolfe Rd., Suite SW3-200
 Cupertino, CA 95014
 USA
 Phone: +1-949-863-0327
 Toll Free: +1-877-77-DEBUG

Your local partner:



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

