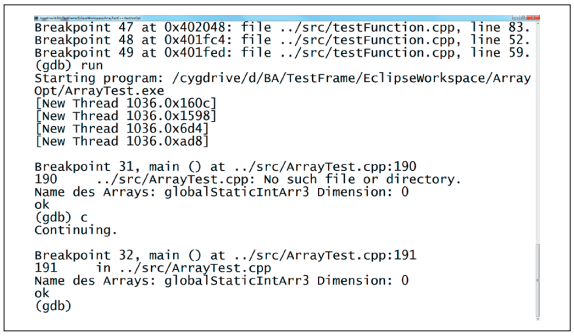
Daniel Gubser

Michael Meisser

Diplomanden	Daniel Gubser, Michael Meisser
Examinator	Prof. Reto Bonderer
Experte	Urs Reidt, Hamilton Medical AG, Bonaduz, GR
Themengebiet	Embedded Software Engineering

Bare Metal Test Framework

Laufzeitprüfungen bei Mikrocontroller-Software in C und C++

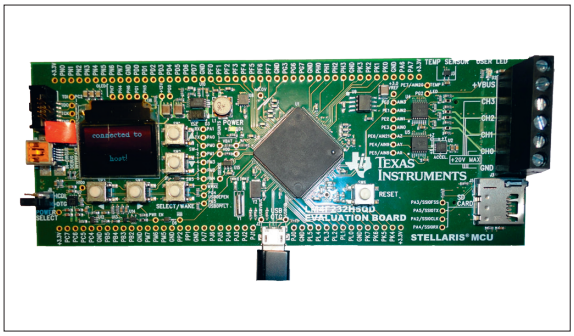


```
Breakpoint 47 at 0x402048: file ../src/testFunction.cpp, line 83.
Breakpoint 48 at 0x401fc4: file ../src/testFunction.cpp, line 52.
Breakpoint 49 at 0x401fed: file ../src/testFunction.cpp, line 59.
(gdb) run
Starting program: /cygdrive/d/BA/TestFrame/EclipseWorkspace/Array
Opt/ArrayTest.exe
[New Thread 1036.0x160c]
[New Thread 1036.0x1598]
[New Thread 1036.0x6d4]
[New Thread 1036.0xad8]

Breakpoint 31, main () at ../src/ArrayTest.cpp:190
190     ../src/ArrayTest.cpp: No such file or directory.
Name des Arrays: globalStaticIntArr3 Dimension: 0
ok
(gdb) c
Continuing.

Breakpoint 32, main () at ../src/ArrayTest.cpp:191
191     in ../src/ArrayTest.cpp
Name des Arrays: globalStaticIntArr3 Dimension: 0
ok
(gdb)
```

GNU Debugger (GDB) bei der Ausführung eines Testprogramms

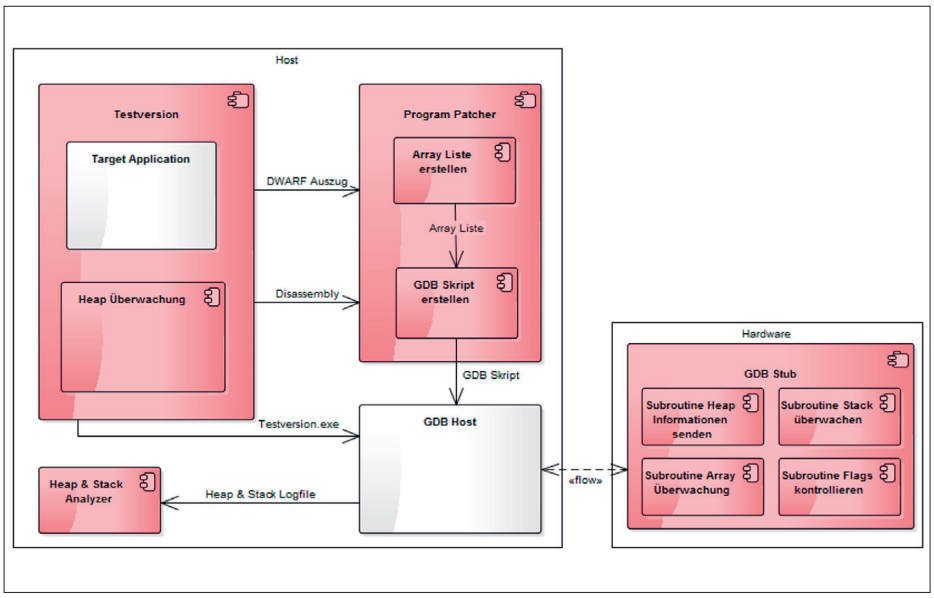


Hardware-Plattform Stellaris EKS-LM4F232

Ausgangslage: Die Programmiersprachen C und C++ verzichten aus Gründen der Performance auf Laufzeitprüfungen wie Integer-, Array-, Stack-Overflow und Heap-Überwachung. Dadurch können effiziente Programme erstellt werden, jedoch auf Kosten der Sicherheit. Um die Zuverlässigkeit eines Programms zu gewährleisten, muss dieses getestet werden können. Um ein Programm sicherer zu machen, können zusätzliche Kontrollmechanismen im erstellten Code eingefügt werden, das Programm wird dadurch aber ineffizienter. Da Embedded Systems beschränkte Ressourcen – z. B. Rechenleistung – haben, werden schlanke und massgeschneiderte Programme bevorzugt.

Ziel der Arbeit: Ziel dieser Bachelorarbeit ist es, ein Bare Metal Test Framework zu realisieren, welches unabhängig von Hardware und Integrated Development Environment ist. Das Framework soll die verwendeten Programmiersprachen C und C++ unterstützen, da diese im Embedded-Bereich weit verbreitet sind. Mit diesem Framework soll es möglich werden, Laufzeitprüfungen für ein beliebiges Programm durchzuführen. Diese Prüfungen sollen ohne allzu grosse Änderungen am Source Code des zu testenden Programms integriert werden. Das Framework soll möglichst wenig Speicherplatz benötigen und ohne grosse Umstände an die entsprechende Hardware angepasst werden können.

Ergebnis: Der Lösungsansatz besteht aus dem GNU Debugger (GDB), einem GDB-Stub und einem Programm, welches ein GDB-Skript erstellt. Der GDB wird auf einem Host Computer ausgeführt und steuert den Programmfluss des zu testenden Programms. Der Stub wird auf die Hardware geladen und muss dementsprechend an die eingesetzte Plattform angepasst werden. Der Programm-Patcher erstellt aus den Debug-Informationen der Zielapplikation ein Skript für den GDB, welches den normalen Programmablauf ändert, um die Laufzeitüberprüfung durchzuführen.



Design des Bare Metal Test Framework