Verteidigung der Diplomarbeit

„Untersuchung von Funktionsabläufen für das Remote-Thread-Debugging in eingebetteten Systemen“

Thomas Werner
thomas.werner@email.de

Technische Universität Dresden
Institut für Technische Informatik

Betreuer: Dipl.-Inf. Steffen Köhler
Betreuender Hochschullehrer: Prof. Dr.-Ing. habil. Rainer G. Spallek

Inhalt

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Aufgabenstellung

1 Aufgabenstellung

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Aufgabenstellung

- Literaturrecherche zu grundlegenden Themen
- Analyse repräsentativer Remote-Debug-Szenarien am Beispiel einer selbstgewählten Multithread-Applikation unter Linux
- Konzeption und Implementierung einer Multithread-Erweiterung der bereits existierenden Debug-Agentsoftware für die eingebettete eTW-ARM-eval Plattform (FTZ Leipzig)
- Erweiterung der entsprechenden Target-Interface-Komponente für die UDE (PLS)
- Funktionsnachweis und Dokumentation der Ergebnisse

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

2 Motivation

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Motivation (1)

Stand der Technik:

- Wachsende Zahl eingebetteter Systeme in industrieller Verwendung
- Hohe Anpassungsfähigkeit durch Umsetzung der jeweiligen Aufgabe in Software
- Für Entwicklung oft Emulation des einbettenden Systems oder angeschlossener Elektronik erforderlich
- Steigende Leistungsfähigkeit ermöglicht Einsatz von Betriebssystemen (z.B. Embedded Linux)

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Motivation (2)

Verbesserungen durch Betriebssysteme:

- Übernahme des direkten Hardwarezugriffs durch Betriebssystemfunktionen
- Quasiparallele Abarbeitung mehrerer Aufgaben durch Multithreading

Verbesserungen durch Remote-Debugging:

- Test eines Komplettsystems aus Hard- und Software in realer Einsatzumgebung möglich
- Emulation kann entfallen
- Entwicklungsarbeit ausschließlich auf komfortablem Hostsystem

⇒ **Anforderungen an Debugger im „*embedded*“-Bereich**

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

3 Grundlagen

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debugger (1)

„Debuggers are the magnifying glass, the microscope, the logic analyzer, the profiler, and the browser with which a programm can be examined.“

Jonathan B. Rosenberg [Rose96]

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debugger (2)

Begriff:

- Werkzeug zur Fehlersuche in Applikationen
- Bug, aus dem Englischen → „Wanze“

Einsatz:

- Steuerung und Überwachung des Programmablaufs einer Testanwendung
- Fehlerlokalisierung und -beseitigung
- Meist Bestandteil einer Entwicklungsumgebung
- Analyse der Funktionsweise unbekannter Programmmabschnitte
- *closed source*, *reverse engineering*

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debugger (3)

Funktionen: Einige Anforderungen sind zwingend notwendig, um den Ablauf eines Programms zu überwachen, zu steuern und über den Zustand des Systems bzw. der Anwendung informiert zu werden.

- **Ablaufsteuerung:** Unterbrechen, Fortsetzen, Einzelschritt
- **Haltepunkte:** bedingte Unterbrechung, Hardware- bzw. Software-Breakpoints
- **Zustandsabfrage:** Register und Speicher lesen
- **Zustandsnotifikation:** Register und Speicher schreiben
- **Überwachung:** Ursache für Programmabsturz
- **erweiterte Funktionen:** Hochsprachen-Debugging

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Eingebettete Systeme (1)

Begriff:

- Computer bzw. Mikrocomputer, der in ein technisches System eingebunden ist
- Aufgaben: Steuerung, Regelung, Überwachung
- Übernimmt Teilaufgabe in einem grösseren Kontext
- Bestandteile der Unterhaltungselektronik, Küchengeräte, mobiler Kommunikation, Automobil- und Flugzeugbau
- Vernetzung autonomer, eingebetteter Systeme zu Gesamtsystem

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Eingebettete Systeme (2)

Aufbau:

- Anpassung an Bestimmungsort und Zweck
- Implementierung der Funktionalität in Hard- und Software
- Besondere Anforderung an Stromverbrauch, Platzbedarf, Ausfallsicherheit (Verfügbarkeit)
- Prozessoren der ARM, MIPS- und PowerPC-Architektur
- Flüchtiger Speicher (SRAM) und nichtflüchtiger Speicher (Flash)
- Peripherie-Module (Steuerung, Sensordaten)
- Kommunikationsschnittstellen (RS232, Ethernet)
- Ein- und Ausgabegeräte
- Betriebsnotwendige Software (Firmware oder Betriebssystem)

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Eingebettete Systeme (3)

Softwareentwicklung:

- Entwicklung der Firmware oder der Anwendungen am Arbeitsplatz des Programmierers
- Cross-Compiler für jeweilige Zielarchitektur
- Hochsprache (C, C++, . . .) und Assembler
- Test der Software in Simulator
- **Test auf der Zielhardware:**
  - In-Circuit-Emulator
  - Debugging mittels JTAG
  - Remote-Software-Debugger
- Unterstützung dieser Verfahren durch UDE

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Eingebettete Systeme (4)

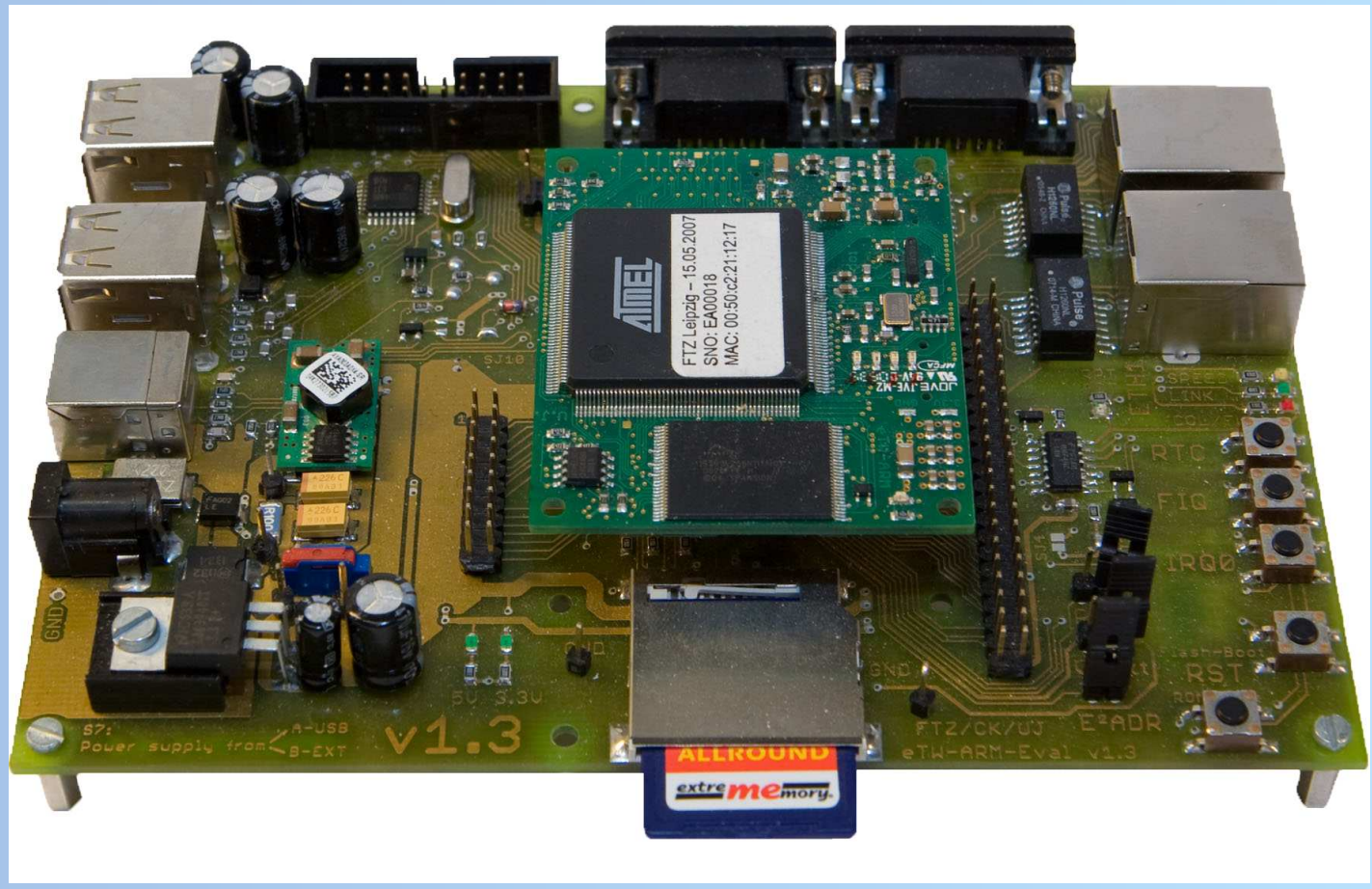


Abbildung 1: eTW-ARM-eval Plattform

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Remote-Debugging (1)

Begriff:

- Fehlersuche aus der „Entfernung“
- Interaktion mit dem zu analysierenden Programm auf dessen System (*remote system*)
- Interaktion mit dem Debugger über System des Entwicklers (*local system*)
- Trennung von Kernfunktionalität und Benutzerschnittstelle
- Kommunikation z.B. über serielle Schnittstelle oder Netzwerk, ermöglicht räumliche Trennung
- **Verwendungsgründe:** Einsatzumgebung, Grafische Benutzeroberfläche, Zielarchitektur

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Remote-Debugging (2)

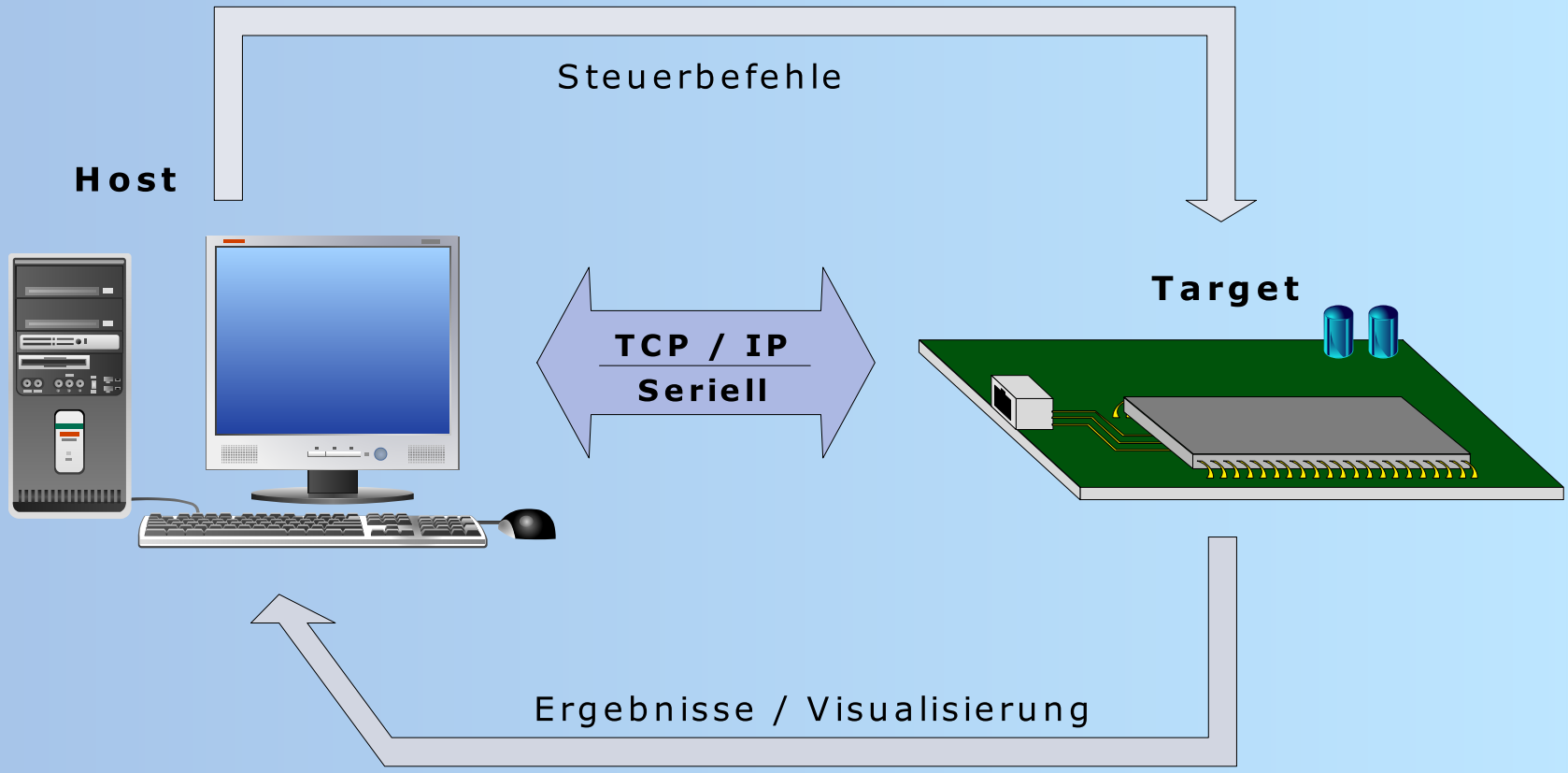


Abbildung 2: Schematische Darstellung zum Remote-Debugging

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Remote-Debugging (3)

Remote-Debugging mit dem GNU Debugger (kurz GDB):

- Keine grafische Oberfläche (aber z.B. Eclipse oder DDD)
- Remote-Debugging über *stub* oder GDB-Server
- **gdbserver:**
  - Reduzierte Variante des GDB
  - Fernsteuerung über *remote serial protocol*

Remote-Debugging mit der UDE:

- Universal Debug Engine der Firma PLS
- Windows-Software, Unterstützung verschiedenster Mikrocontroler und Architekturen
- Modularer Aufbau

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Remote-Debugging (4)

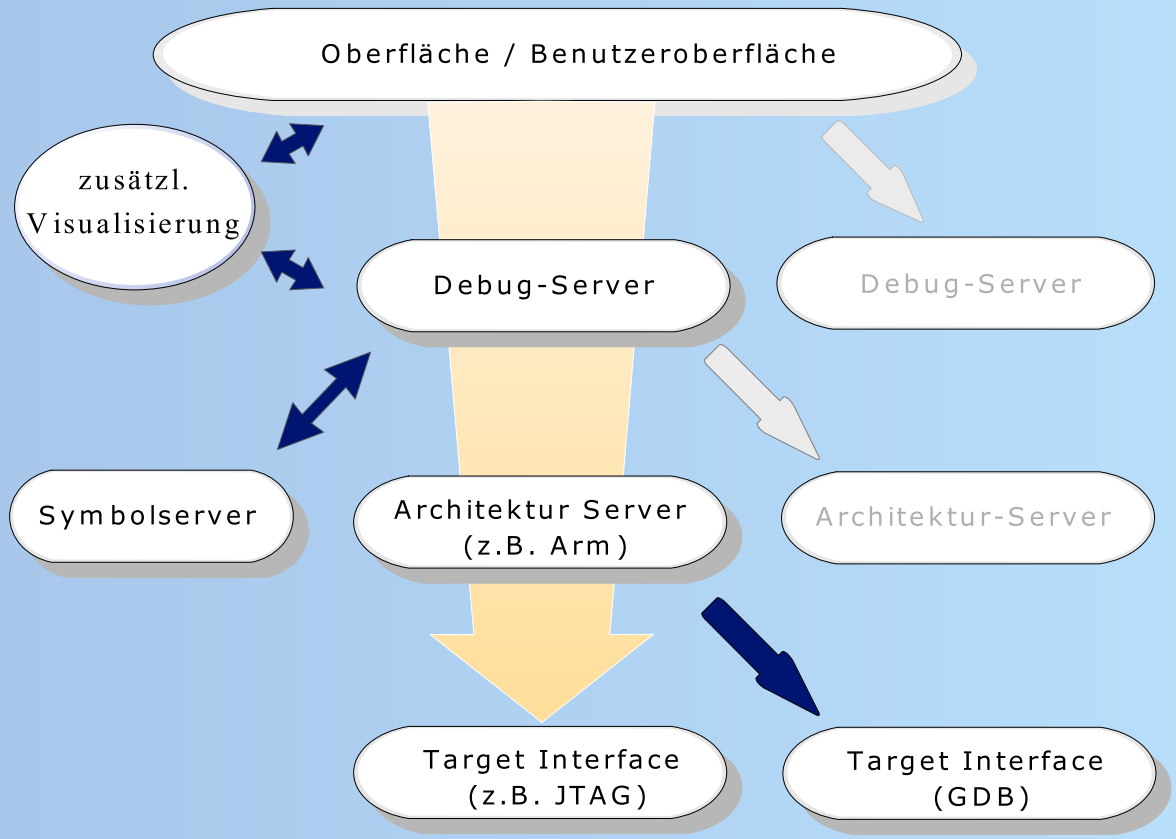


Abbildung 3: Modularer Aufbau der *Universal Debug Engine*

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (1)

Prozesse:

- Betrachtung eines Programms in Ausführung als Prozess
- Zusammenfassung sequenzieller Befehle, getrennte Bearbeitung
- Eigener Befehlzähler, eigener Speicherbereich, eigener Stack
- Gleichzeitige Ausführung auf nur einer CPU nicht möglich
- Ablaufsteuerung durch Scheduler $\Rightarrow$ Quasiparallelität
- Abwicklung mehrerer Teilaufgaben eines Programms nur sequentiell möglich

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (2)

Threads:

- Erweiterung des Prozessmodells
- Gruppierung der Anweisung zu Ausführungseinheiten (Fäden, engl. Threads)
- Quasiparalleler Ablauf analog zu Prozessen
- Aufgliederung eines Prozesses in mehrere „Ausführungsfäden“ $\Rightarrow$ Multithreading
- Gemeinsamer Zugriff auf Adressraum und Ressourcen des Prozesses
- Eigener Kellerspeicher, eigener Registersatz (inkl. Befehlszähler)
- Einführung „kritischer Abschnitte“ notwendig

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (3)

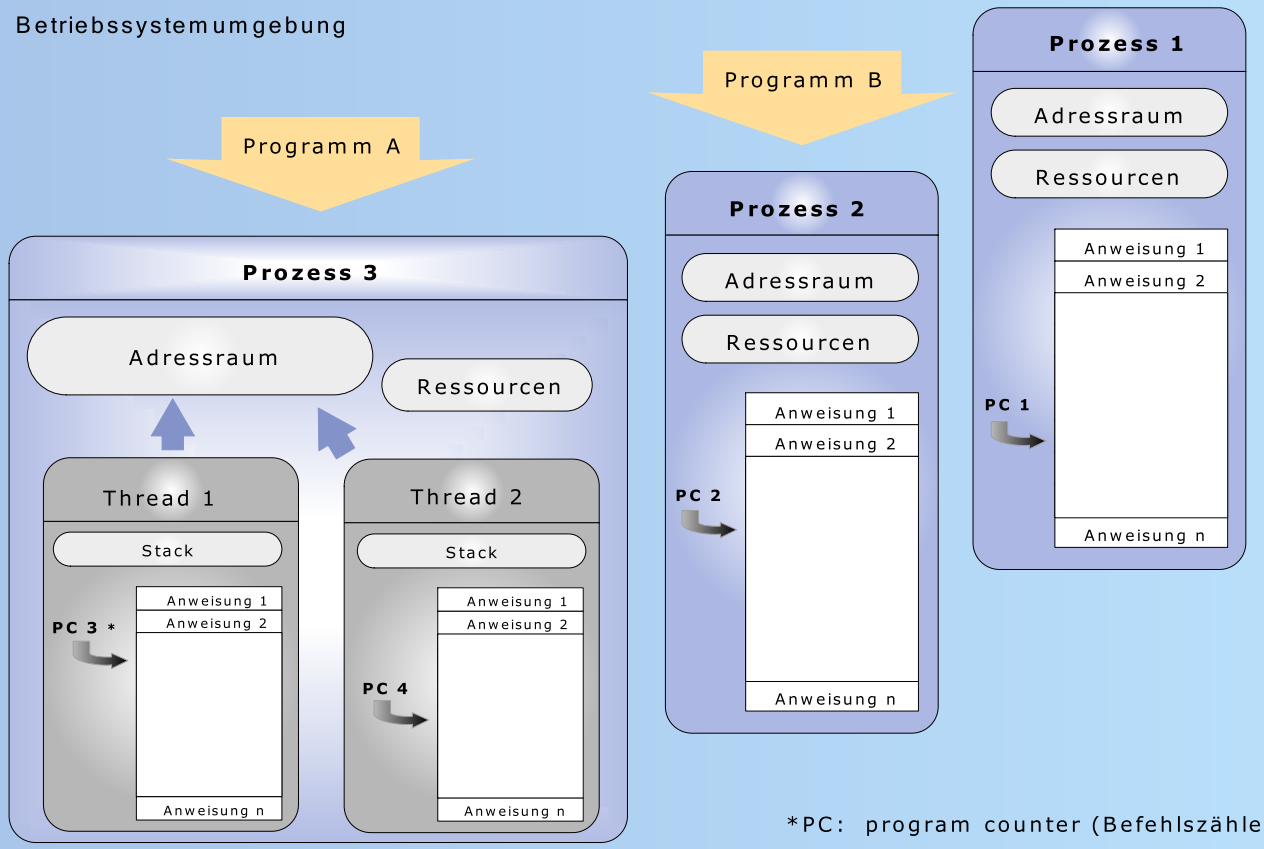


Abbildung 4: Abgrenzung zwischen Prozess und Thread

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (4)

Threadzustände: Der gemeinsame Zugriff auf begrenzte Ressourcen, der wechselseitige Ausschluss aus kritischen Abschnitten und eine bedingte Reihenfolge der Threadabarbeitung führt zur Unterscheidung folgender Zustände.

- Rechnend
- Blockiert
- Rechenbereit
- Beendet
- Verwaist

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (5)

User-Thread

- komplett im Adressraum eines Programms
- Realisierung durch Laufzeitbibliothek
- Verwaltung in Thread-Tabelle im Prozess-Adressraum
- Kooperatives Scheduling

Kernel-Thread

- Verwaltung in Thread-Tabelle im Kern-Adressraum
- Native Unterstützung durch Betriebssystem
- Präemtives Scheduling möglich
- Hybride Realisierung mit Userthreads $\Rightarrow$ Fiber

Multithreading (6)

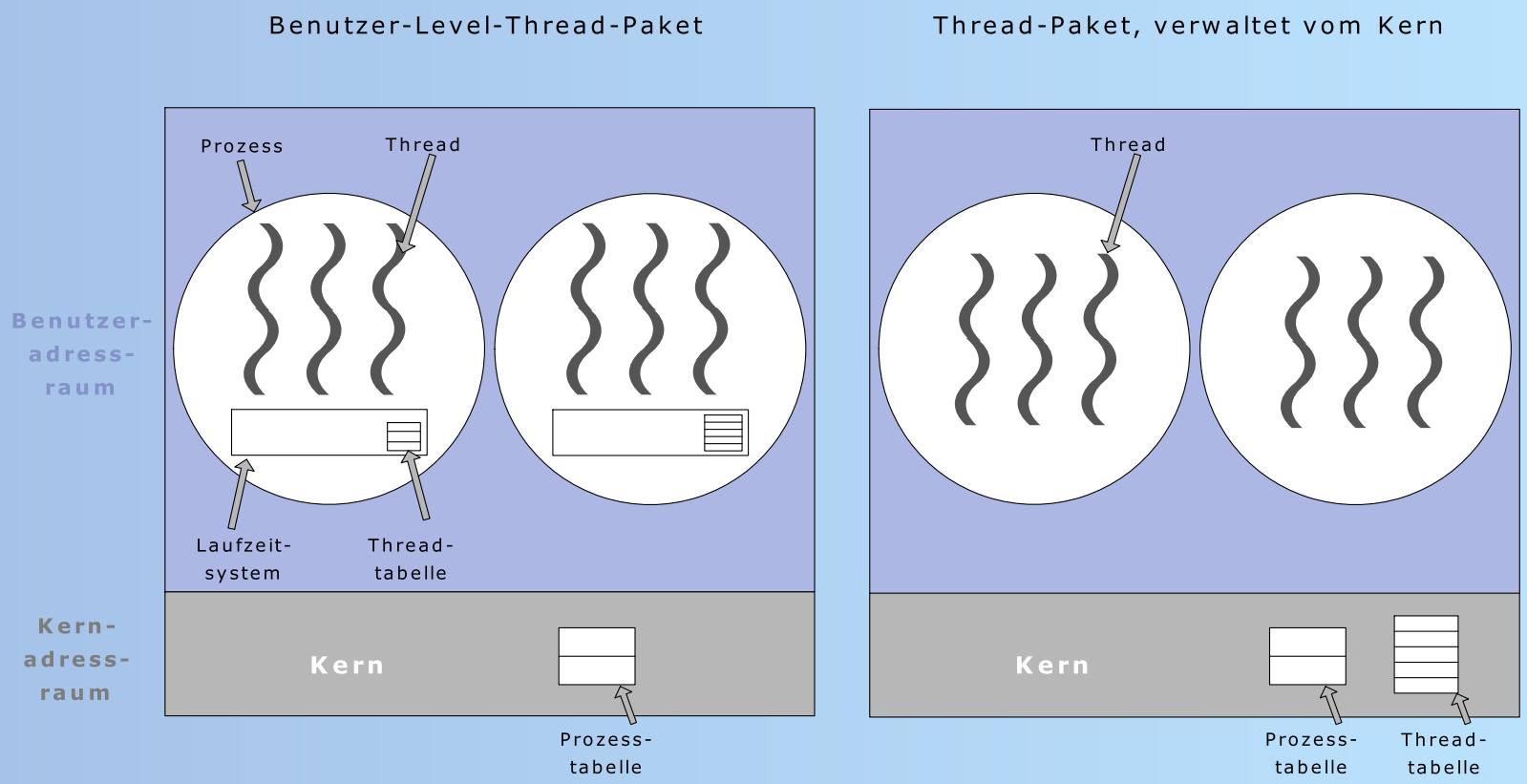


Abbildung 5: Verwaltung bei Kernel- und User-Threads [Tane03]

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Multithreading (7)

Multithreading mit der Pthreads Bibliothek

- Laufzeitbibliothek unter Unix-basierten Betriebssystemen
- Funktionen zur Erzeugung, Manipulation und Synchronisierung
- Programmierschnittstelle nach Standard IEEE POSIX 1003.1c

LinuxThreads

- Prozesskopie durch Systemaufruf `clone()`
- Synchronisierung über Signalsystem des Betriebssystems

Native POSIX Thread Library (NPTL)

- Direkte Kernel-Unterstützung, Systemaufrufe für Synchronisation
- Abbildung jedes Ausführungsfadens auf einen Prozess (1:1)

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Analyse und Entwurf

4 Analyse und Entwurf

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

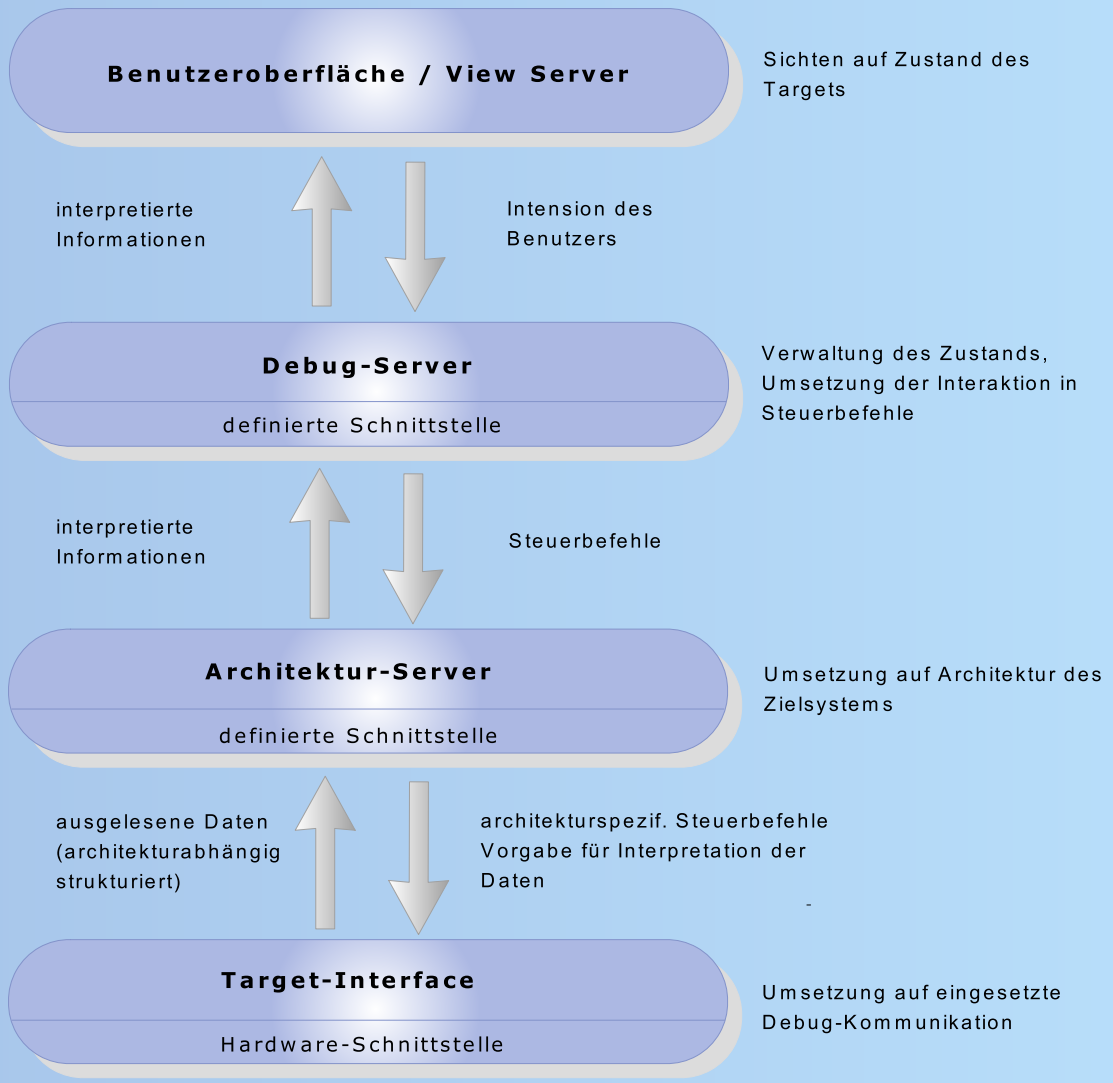
UDE Debug-Komponenten (1)

UDE Debug-Komponenten

- Graphische Benutzeroberfläche
  - Interaktion mit Benutzer
  - Informationsdarstellung
- Debug-Server
  - Steuerung der Debug-Vorgänge
- Architekturserver
  - Anpassung an Prozessor-Architektur des Targets
  - Kontextverwaltung, Disassembler
- Target-Interface
  - Schnittstelle zum Zielsystem

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

UDE Debug-Komponenten (2)



Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

GDB-Target-Interface (1)

Gliederung des GDB-Target-Interface in 3 Schichten

CoGDBTargIntf:

- Schnittstellen-Implementierung
- `Connect()`, `StartUserApp()`, `CheckUserApp()`, `...`

TCP_functions:

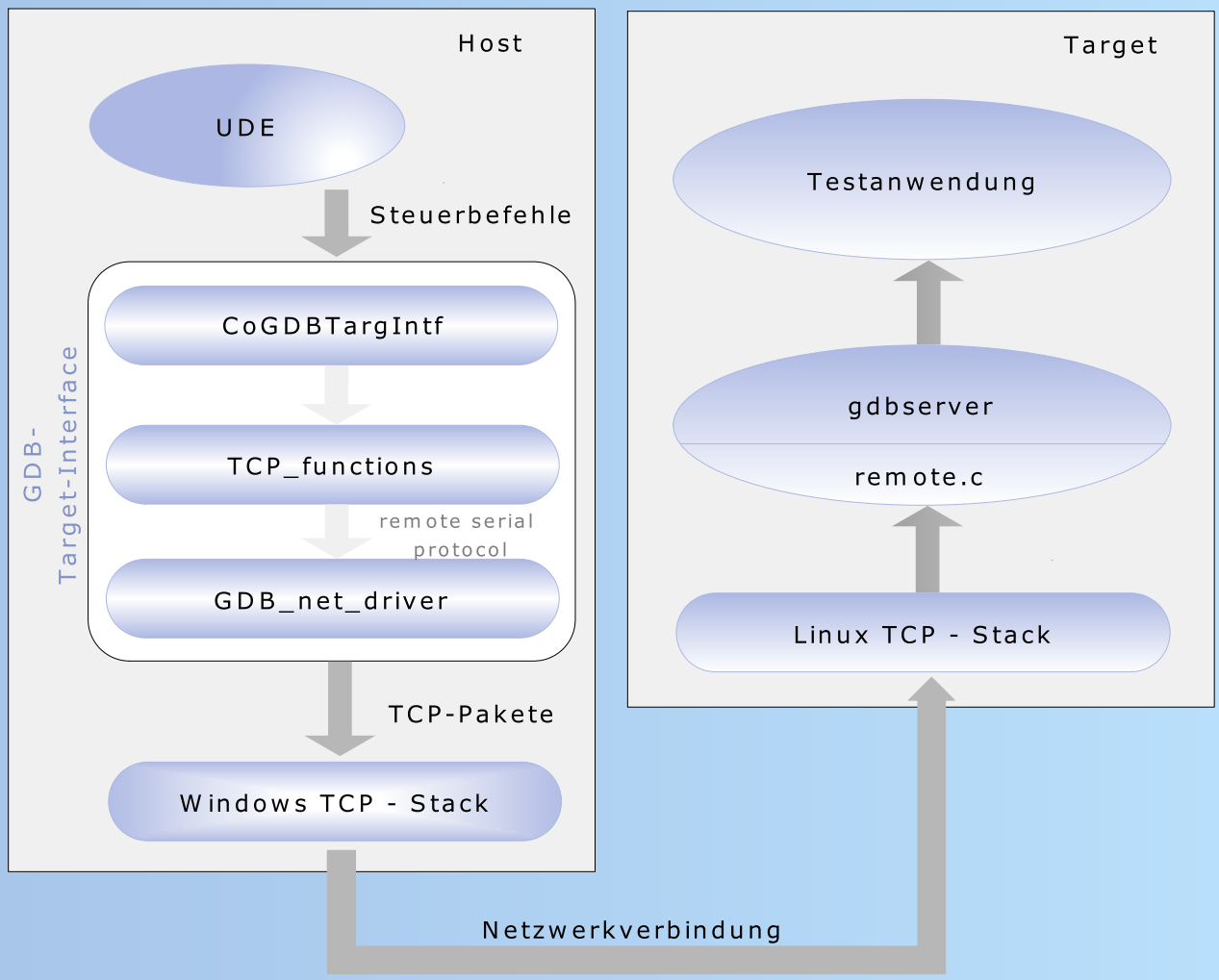
- Umsetzung des GDB-Protokolls
- `IsConnected()`, `Run()`, `ReadContext()`, `...`

GDB_net_driver:

- Netzwerkkommunikation (TCP/IP)
- `SentGBDPacket()`, `ReceiveGDBPacket()`, `...`

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

GDB-Target-Interface (2)



Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debug-Kommunikation (1)

GDB Remote Serial Protocol

- Kommunikation zwischen Debug-Umgebung und GDB-Server
- Überführung einer Debug-Anfrage in ein oder mehrere Kommandos
- Strukturiert in Nachrichten, paketorientiert
- Austausch über Netzwerkverbindung
- Anfragen nur vom Host, meist beantwortet
- korrekte Übertragung durch TCP und Bestätigungspakete
- eine Debug-Anweisung pro Nachricht (!, ?, g, k, vCont;c, Z0,*addr,length*, qSymbol)

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debug-Kommunikation (2)

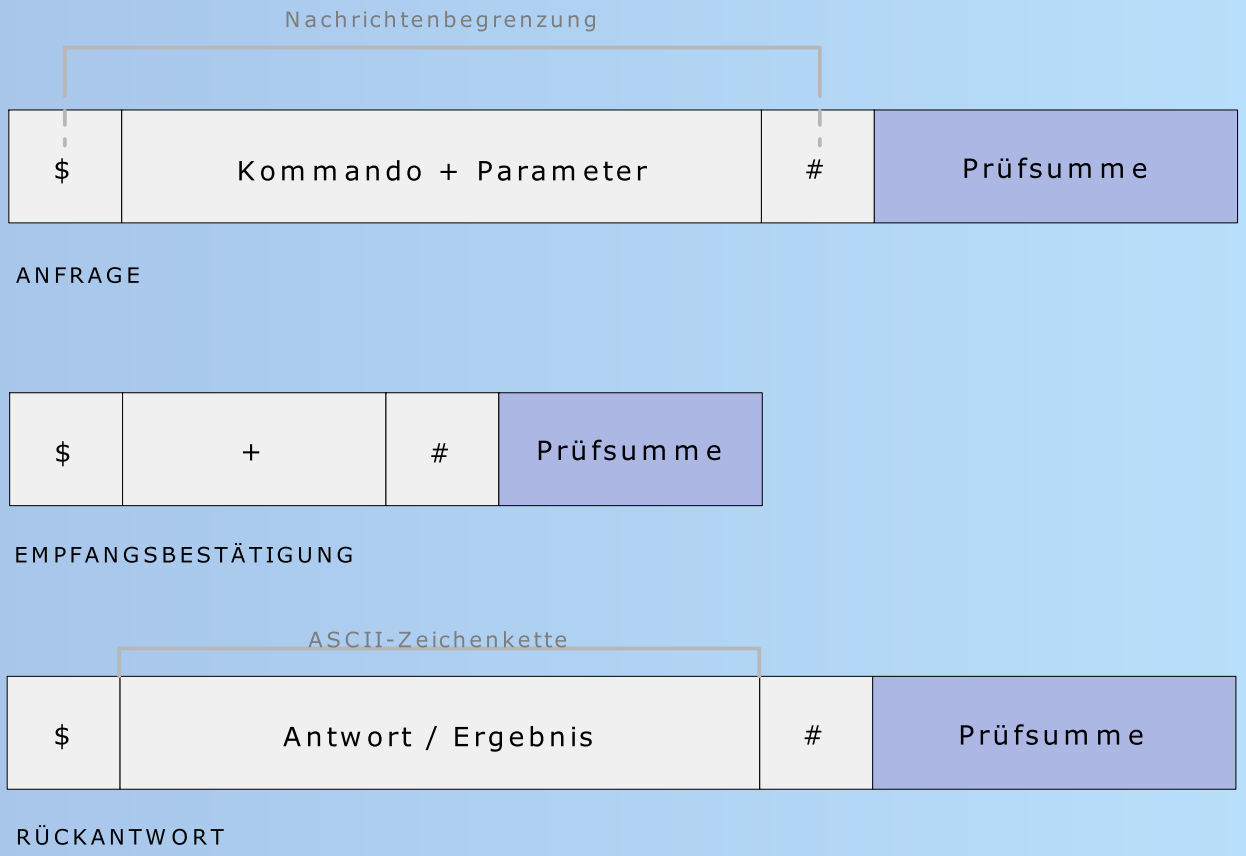


Abbildung 6: Paketstruktur des GDB Remote Serial Protocols

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Erweiterungen (1)

- Verwendung eines angepassten GDB-Servers (Version 6.6)
- Anpassung der Methoden zur Ablaufsteuerung in CoGDBTargIntf: `StartUserApp()`, `BreakUserApp()`, `CheckUserApp()`
- Hinzufügen neuer Methoden zur Generierung von Thread-Informationen: `GetNumCores()`, `GetActiveCores()`, `SetActiveCores()`, `GetFirstCoreID()`, `GetNextCoreID()`
- Erweiterung der Schnittstellenbeschreibung zwischen Architektur-Server und Target-Interface (idl)

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Erweiterungen (2)

Anpassung der Methode StartUserApp()

- Symbolaustausch
- Unterscheidung zwischen normalem Start und Single-Step
- Beseitigung des blockierenden Verhaltens

Anpassung der Methode CheckUserApp()

- Überprüfung auf Unterbrechung der Ausführung durch neue Methode `hasStopped()` (nicht blockierend)
- Sammlung von Informationen über laufende Threads
- Aktualisierung der Anzeige

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Beispiel-Applikation

```
void *thread_function(void *arg) {
    printf("Hello World from %d!\n", arg);
    sleep(2);
    pthread_exit(NULL),
}
int main(int argc, char *argv[]) {
    pthread_t threads[NUM_THREADS]; int i;
    for (i=0; i<NUM_THREADS, i++)
        pthread_create(&threads[i], NULL, \
            thread_function, (void*)i);
    for (i=0; i<NUM_THREADS, i++)
        pthread_join(threads[i], NULL);
    return 0;
}
```

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

5 Implementierung

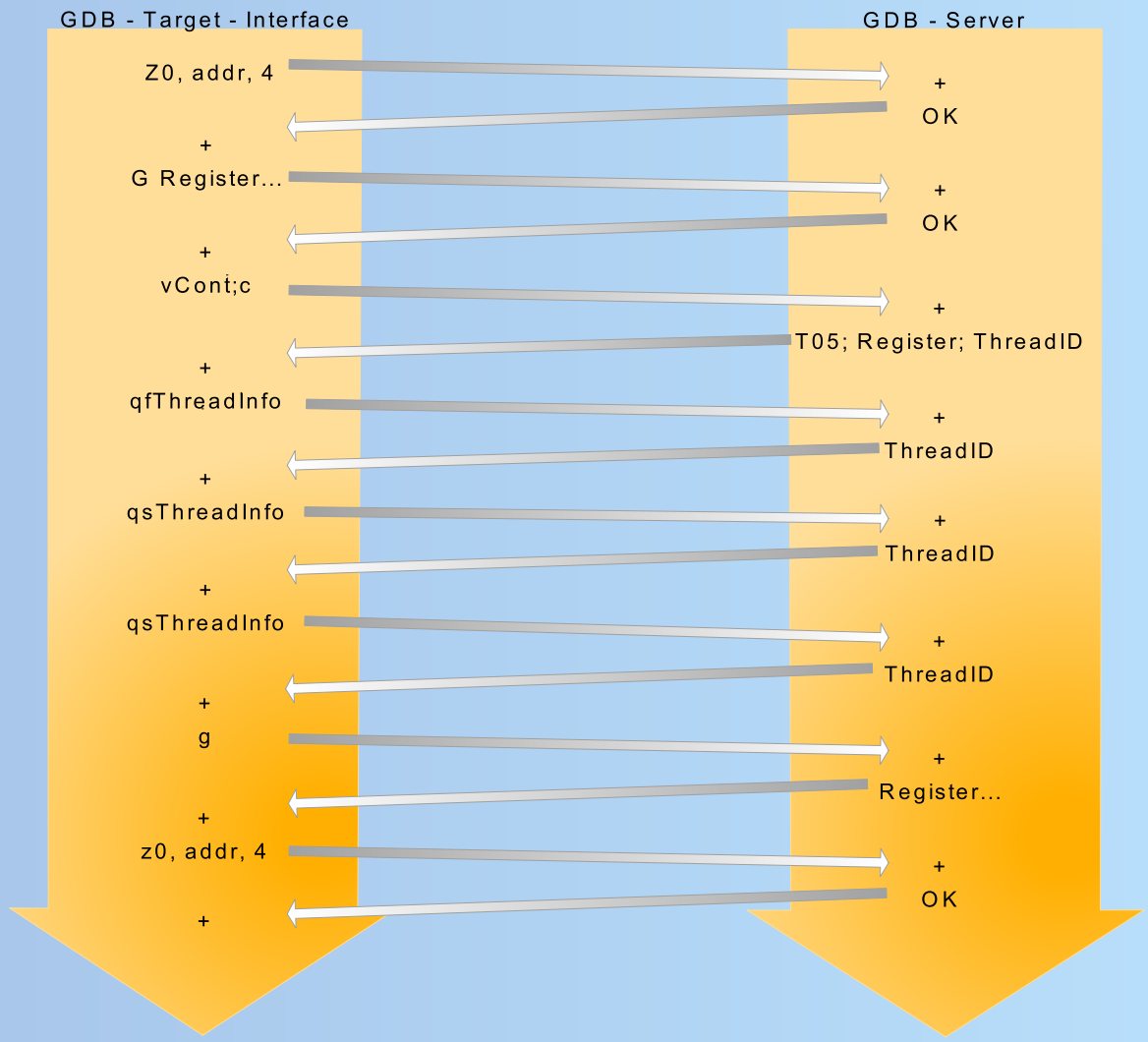
Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Implementierung (1)

- Keine weitere Anpassung des GDB-Servers notwendig
- Für Single-Step Unterscheidung kleine Änderung am Architektur-Server notwendig, `GetBrpObjectByDescription("Single Step", ...)`
- Für Symbolsaustausch Zugriff auf Debug-Server notwendig, `GetDbgSrvUnk()`
- Symbolabfrage über `SymbolEngine.GetLabels()`
- Symbolaustausch über qSymbol-Nachricht

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Implementierung (2)



Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

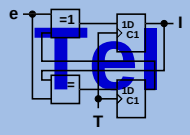
Implementierung (3)

- Problem bei `HasStopped()` (blockierende TCP-Verbindung)
- Umgestaltung der GDB_net_driver-Schicht
- Timing-Problem bei Single-Step $\Rightarrow$ Begrenzung auf aktiven Thread
- Abfrage der laufenden Threads durch qfThreadInfo-Nachrichten
- Bestimmung der Threadzustände durch T*threadid*-Nachricht

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Leistungsbewertung

6 Leistungsbewertung



Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debug-Szenario (1)

- gdbserver und Testanwendung auf eTW-ARM-eval Plattform
- Verbindung über lokales Netzwerk
- UDE-Projekt mit ARM7Arch-Server und GDB-Target-Interface
- Testanwednung und Quelltext in UDE geladen
- Debug-Verbindung hergestellt
- Beliebig gesetzte Haltepunkte

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Debug-Szenario (2)

C:\Uni\thread.c

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 5

void *thread_function(void *arg) {
    printf("Hello World from %d!\n", arg);
    sleep(2);
    pthread_exit(NULL);
}

int main(int argc, char *argv[]) {

    pthread_t threads[NUM_THREADS];
    int i;

    for (i = 0; i < NUM_THREADS; i++)
        pthread_create(&threads[i], NULL, thread_function, (void *)i);

    for (i = 0; i < NUM_THREADS; i++)
        pthread_join(threads[i], NULL);

    return 0;
}
```

Abbildung 7: Screenshot vom Debuggen der Beispielanwendung

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Ergebnis (1)

- Erfolgreiche Unterbrechung an alle Haltepunkten
- Funktionierendes Single-Stepping
- Nach Ausführungsende automatischer Neustart
- Manuelle Unterbrechung und Fortsetzung jederzeit möglich
- Anzeige der laufenden Threads und deren Zustände in seperatem Fenster

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Ergebnis (2)

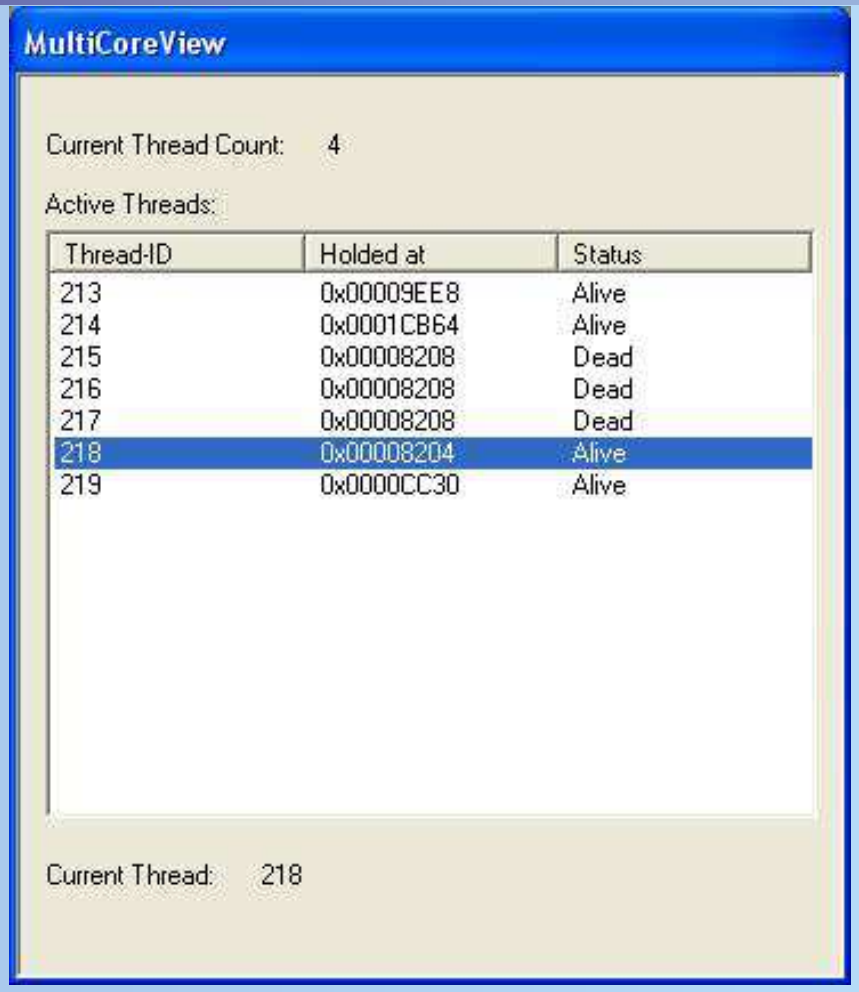


Abbildung 8: Anzeige der Thread-Information

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Ausblick

7 Ausblick

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Ausblick

- Automatischer Download der Testanwendung auf das Target
- Verwendung dynamisch gelinkter Bibliotheken
- Starten des GDB-Servers über Netzwerkverbindung
- Aufbereitung weiterer Thread-Informationen
- Implementierung der Thread-Anzeige als UDE-Komponente

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Literaturverzeichnis (1)

Literatur

[Zell05] **Zeller, Andreas**
Why Programs Fail: A Guide to Systematic Debugging;
Morgan Kaufmann; 2005

[Rose96] **Rosenberg, Jonathan B.**
How Debuggers Work: Algorithms, Data Structures, and Architecture;
Wiley; 1996

[Stal02] **Richard M. Stallman, Roland Pesch, Stan Shebs, et al.**
Debugging with GDB;
Free Software Foundation; 2002

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de

Literaturverzeichnis (2)

Literatur

[Tane03] **Andrew S. Tanenbaum**
Moderne Betriebssysteme, 2. überarbeitete Auflage;
Pearson Studium; 2003

[NBPF96] **Bradford Nichols, Dick Buttlar, Jacqueline Proulx Farrell**
Pthreads Programming;
O'Reilly & Associates, Inc.; 1996

[Appl07] **Debugging with gdb - gdb Remote Serial Protocol**
http://developer.apple.com/documentation/DeveloperTools/gdb/gdb/gdb_33.html;
Apple; 2007
Datum des letzten Zugriffs: 12.03.2008

Abschluss

Vielen Dank für Ihre Aufmerksamkeit!

Thomas Werner
Technische Universität Dresden
Institut für Technische Informatik
thomas.werner@email.de