Ausführungszeiten

Übung zur Vorlesung EZS

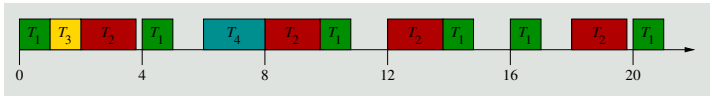
Florian Franzmann Martin Hoffmann Tobias Klaus
Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
http://www4.cs.fau.de

27. Oktober 2014

1. Rekapitulation: Worst-Case Execution-Time
2. Zeitmessungen
   - Zeitgeber
   - Oszilloskop
   - Diskussion
3. Statische Laufzeitanalyse
4. Handwerkszeug
   - libEZS
   - aiT

Worst-Case Execution-Time



- Statische Ablaufplanung
- Planbarkeitsanalyse
- Später: Übernahmeprüfung

- Worst-Case
  - ⇝ Obere Schranke für alle Fälle

1. Rekapitulation: Worst-Case Execution-Time
2. Zeitmessungen
   - Zeitgeber
   - Oszilloskop
   - Diskussion
3. Statische Laufzeitanalyse
4. Handwerkszeug
   - libEZS
   - aiT

Zähler in Mikrocontrollern

Zähler (*Counter*) zählen hardwarebasiert Ereignisse z.B. von:

- Externem Drehgeber (Radumdrehung)
- Externem Quarz (Real-Time Clock)
- Internem Prozessortakt (hohe Auflösung)

Äquidistante Ereignisse ermöglichen einen **Zeitgeber** (*Timer*) für

- Periodische Aktivierung
- **Messen von Zeitabständen**
- (Kontrolliertes Verbrennen von Prozessorzeit)

Zähler Betriebsmodi

Zähler bzw. Zeitgeber bieten zwei Betriebsmodi:

Abfragebetrieb (Polling) Aktives Auslesen des Zählers, bis zum Erreichen eines vorgegebenen Wertes.

Unterbrecherbetrieb (Interrupt) Der Zähler unterbricht das laufende System beim Erreichen eines vorkonfigurierten Zählerstandes.

Oszilloskop

- Tricore: Extrem guter Zeitgeber
  - ⇝ 13.3 ns-Raster
  - nicht selbstverständlich
- Oszilloskope
  - meist schon vorhanden
  - hohe Zeitauflösung
  - einfache Bedienung

Bedingung

Frei verfügbarer GPIO-Pin

Diskussion

Welche Probleme können bei messbasierter Bestimmung der Ausführungszeiten auftreten?

1 Rekapitulation: Worst-Case Execution-Time

2 Zeitmessungen
- Zeitgeber
- Oszilloskop
- Diskussion

3 Statische Laufzeitanalyse

4 Handwerkszeug
- libEZS
- aiT

Statische Laufzeitanalyse?

- Statisch?
  - ⇝ keine Ausführung des Programms
- Grundsätzliche Idee
  - Wie lange dauern die einzelnen Maschinenbefehle?
  - Wie sieht der längste Pfad durch das Programm aus?
  - ⇝ Addition aller Maschinenbefehle des längsten Pfades: WCET!
- Klingt doch einfach!?

Wie lang dauert ein Maschinenbefehl?

- Cache
- Pipeline
- Branch-Prediction

Hochkomplex!

Was ist der worst case?

- if/else
  - Einfach! Ein Pfad muss ja länger sein.
  - ABER: abhängig von Eingabe
- Schleifen
  - Wie oft ausgeführt? ⇝ abhängig von Eingabe
  - Wie lange ist ein Durchlauf? ⇝ abhängig von Eingabe

⇝ Wertanalyse: Welche Werte können Variablen annehmen?
- abstrakte Interpretation
- Ausrollen von Schleifen: aiT-Projektparameter: max-unroll
- oft manuelle Eingriffe erforderlich ⇝ Annotationen

⇝ Optimierungsproblem

1 Rekapitulation: Worst-Case Execution-Time

2 Zeitmessungen
- Zeitgeber
- Oszilloskop
- Diskussion

3 Statische Laufzeitanalyse

4 Handwerkszeug
- libEZS
- aiT

libEZS Überblick

Plattformunabhängige Hilfsfunktionen
- Timer-Zugriff (Zeitmessung)
- DAC-Zugriff
- GPIO-Zugriff
- ...

```
aufgabe2
|-- CMakeLists.txt
|-- app.c
|-- ecos
`-- libEZS
  |-- include
  | |-- ezs_dac.h
  | |-- ezs_gpio.h
  | `-- ezs_stopwatch.h
  |-- src
  | `-- ezs_stopwatch.c
  `-- drivers
    `- tc1796
       |-- ezs_dac.c
       |-- ezs_counter.c
       `-- ezs_gpio.c
```

Die *libEZS* wird ständig (auch von euch) erweitert.

Zeitmessung *ezs_stopwatch.c/.h*

Für die Zeitmessung sollen zwei Funktionen implementiert werden:

```
void ezs_watch_start(cyg_uint32 *state);
cyg_uint32 ezs_watch_stop(cyg_uint32 *state);
```

- Parameter: Zeiger auf (globale) Variable
  → unabhängige Messzeitpunkte
- `ezs_watch_stop(cyg_uint32 *state)` gibt die Zeitdifferenz in Ticks zurück

Hinweis

`ezs_counter_get()` in `drivers/include/ezs_counter.h`

GPIO

General Purpose Input/Output

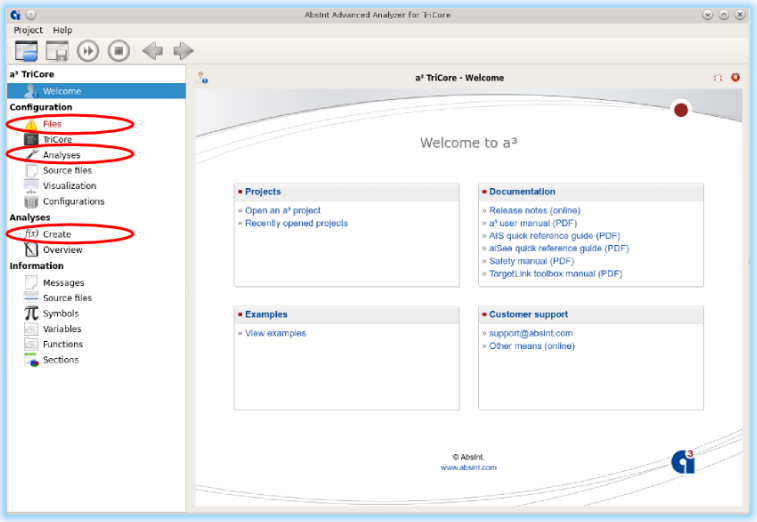
- Pins eines Mikrochips zur *freien Verwendung*
- Konfigurierbar als Ein-/Ausgang
- Oft auch Treiberstärke konfigurierbar
- Teilweise pegelfest bis 5 V
  ⇝ Mikrocontroller-Handbuch lesen ☺
- Zugriff über
  - spezielle Speicheradressen
  - Spezialanweisungen

Ansteuerung

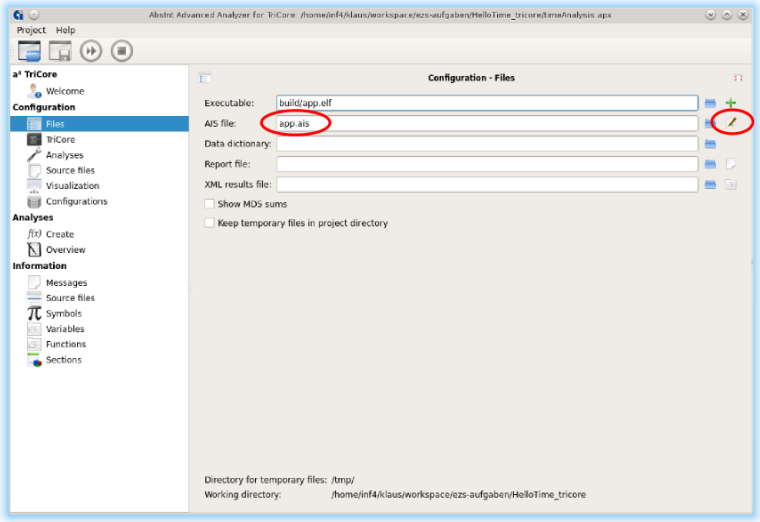
- Beim Tricore Ansteuerung per GPTA oder „von Hand“ möglich

⇝ `void ezs_gpio_set(bool)`

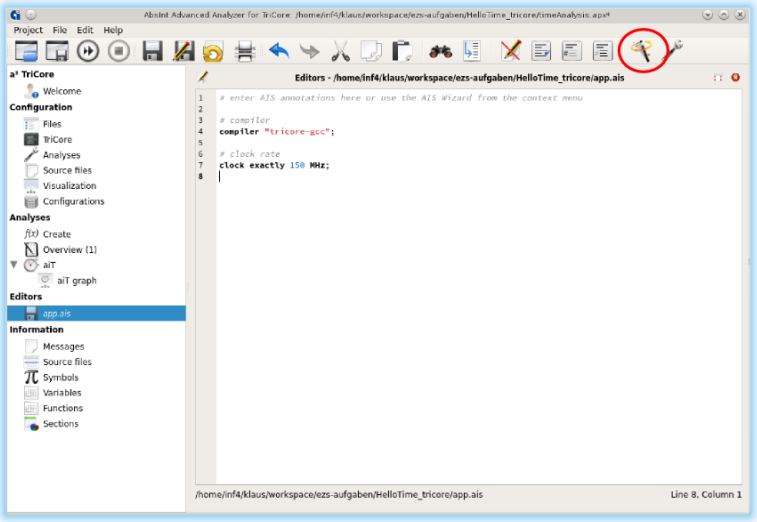
Übersicht



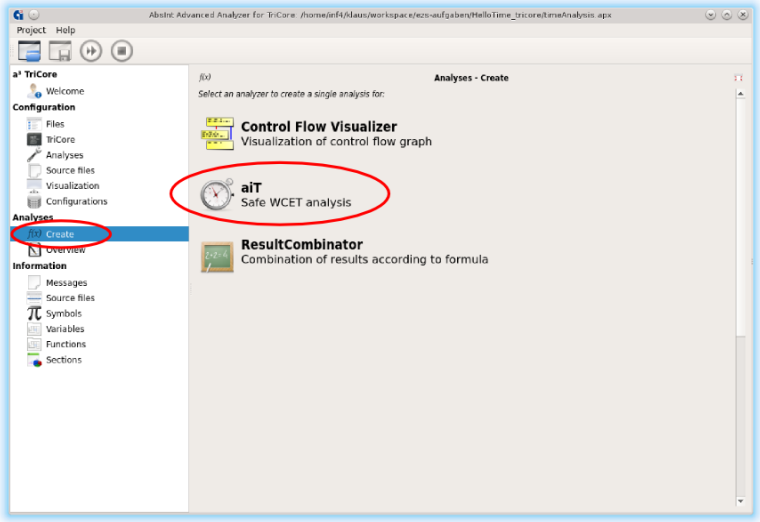
Projektdateien



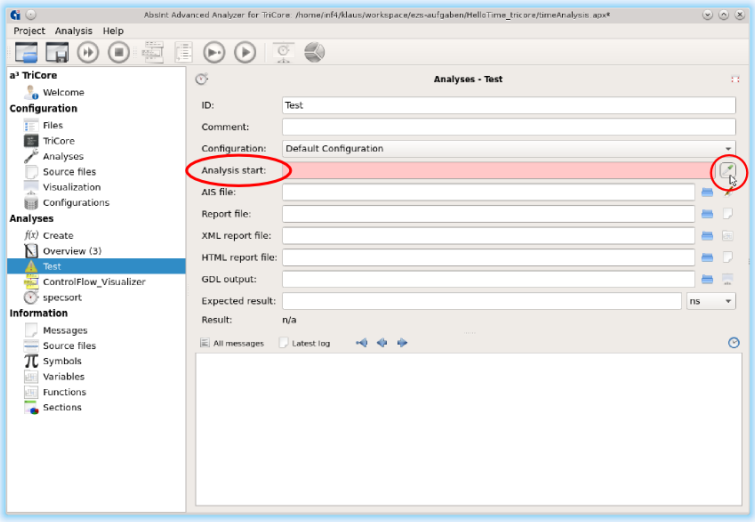
Projektdateien bearbeiten



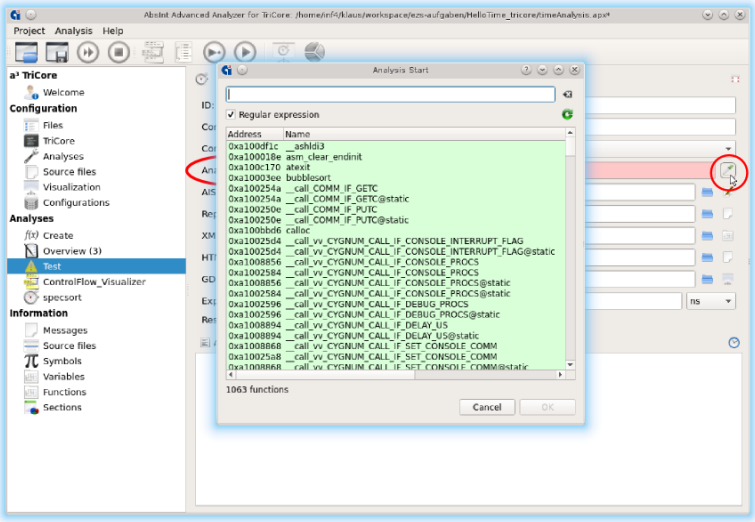
Neue Analyse anlegen



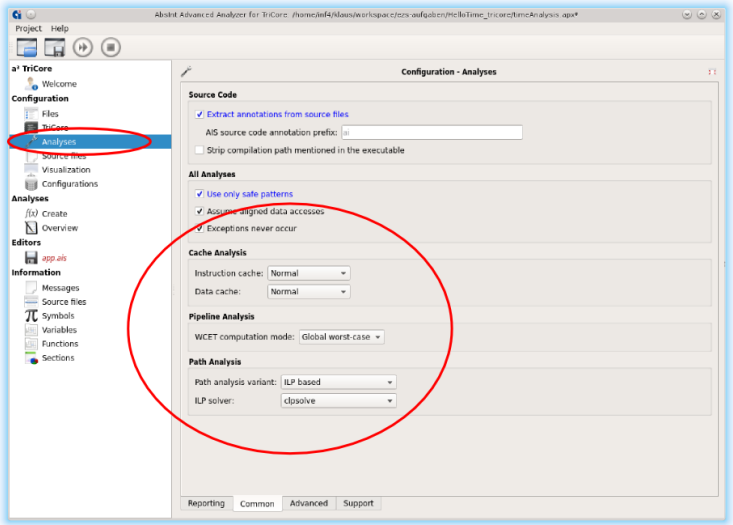
Neue Analyse anlegen



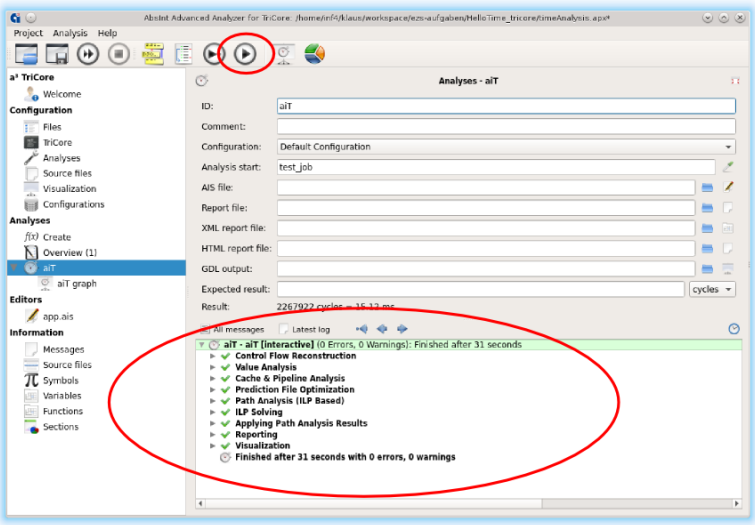
Neue Analyse anlegen



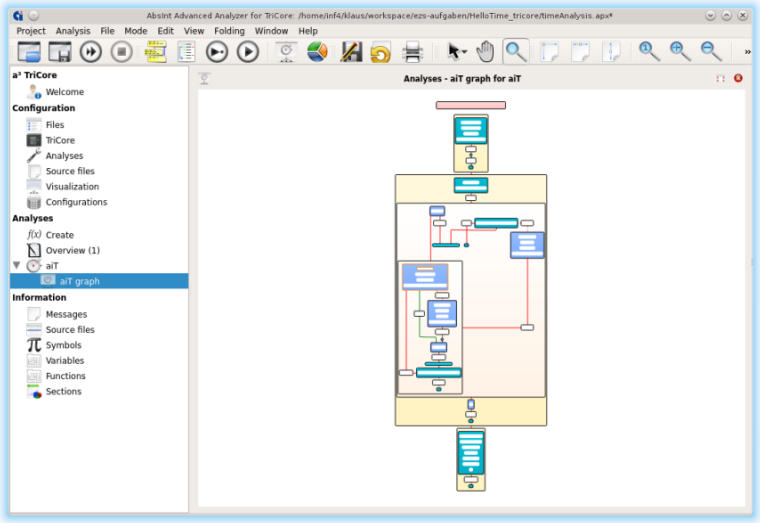
Analyse Parameter



Analyse starten



Analyse untersuchen



Fragen?