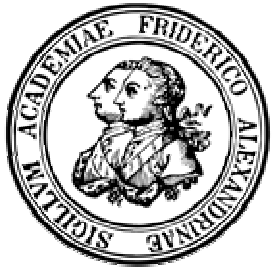


EZS 2
„Hau den Lukas“

Testsystem

Modultest

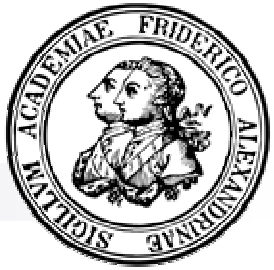
28.06.2006

Patrick Kugler

Inhaltsverzeichnis

- Unser Testsystem
  - Make-Targets
  - Testfall-Verzeichnis
  - Testfall-Beschreibung
- Testfälle: Modultest
  - GPIO
  - COIL
  - PHOTOSENSOR
  - CONFIG
  - CONTROL
  - FSM
- Ausblick

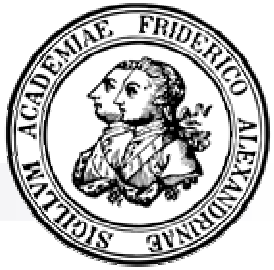
Make-Targets



- Allgemeine Make-Targets
  - ***jtag*** startet den jtag-Server (sollte wegen sudo in eigene Konsole)
  - ***config*** startet proosek
  - ***clean*** löscht alle erzeugten Dateien
  - ***list*** gibt eine Liste aller Testfälle aus

- Make-Targets für Testfälle
  - ***show-%*** zeigt die Testfallspezifikation an
  - ***config-%*** öffnet die OIL-Datei mit proosek
  - ***build-%*** erzeugt den Testfall
  - ***gdb-%*** startet den gdb mit dem Testfall
  - ***run-%*** startet den gdb mit dem Testfall und führt ihn aus
  - ***clean-%*** löscht die erzeugten Dateien

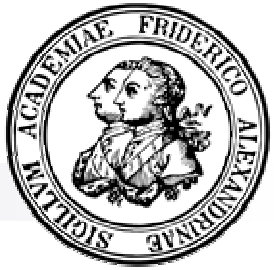
Testfallverzeichnis



- Testfälle liegen unter
  - ***LUKAS/test/module***
    - für Modultests
  - ***LUKAS/test/integration***
    - für Integrationstests
  - ***LUKAS/test/final***
    - für endgültige Anwendungen

- Testfälle bestehen aus
  - Testfallspezifischen Code-Dateien
    - LUKAS/test/final/XXX/*.c
  - Testfallbeschreibung
    - LUKAS/test/final/XXX/testcase.mk

Testfallbeschreibung



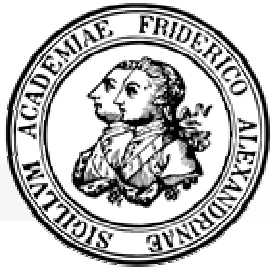
- besteht aus:
  - Testfall-Name (PROJ_NAME)
  - Testfall-Verzeichnis (PROJ_DIR)

  - Testfallbeschreibung (DESCRIPTION)
  - Getestete Komponenten (COMPONENTS)
  - Getestete / Nötige Anforderungen (REQUIREMENTS)

  - Testablauf, d.h. was muss der Benutzer tun um den Test durchzuführen? (TEST_PROCEDURE)
  - Woran erkennt man ob der Test OK war, bzw. wie werden Fehler angezeigt? (TEST_OK u. evtl. TEST_FAIL)

  - Von welchen Dateien ist der Testfall abhängig? (SRC_FILES)
  - Was muss kompiliert und gelinkt werden? (OBJ_FILES)
  - Wo liegt die OIL-Datei? (OIL_FILE)

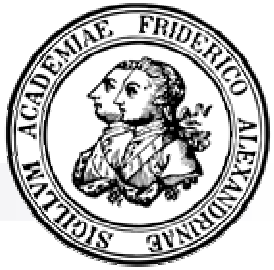
Testfallbeschreibung



- Beispiel: LUKAS/test/final/counter/testcase.mk:

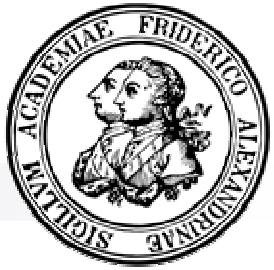
```
PROJ_NAME := counter
PROJ_DIR := test/final/counter

DESCRIPTION := Ein binaerer Counter mit Start, Stop, Reset und +100 Funktion
COMPONENTS := GPIO
REQUIREMENTS := GPIO_WRITE GPIO_READ

TEST_PROCEDURE := Mit den Tasten 0-3 kann der Counter gestartet, gestoppt, auf 0
gesetzt und um 100 erhoeht werden, der Counterwert wird binaer angezeigt.
TEST_OK := Counterwert erheoht sich in 500ms Abstaenden, die Tasten funktionieren
wie beschrieben.

SRC_FILES := src/device/gpio.c src/device/gpio.h src/device/coil.c
src/device/coil.h src/device/photosensor.c src/device/photosensor.h
test/final/counter.c
OBJ_FILES := gpio.o counter.o
OIL_FILE := test/final/counter/counter.oil

include common.mk
```

Testfälle: GPIO



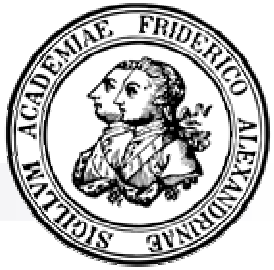
- Was muss getestet werden?
  - Input und Output
  - Auslesen von Ausgängen
  - Grenzen der Funktions-Parameter
- Testfälle:
  - gpio1 – gpio3

Testfälle: GPIO



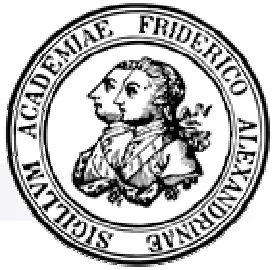

```
================================================================================================
Testcase    : gpio1
Diretory    : test/module/gpio1

Description : Testen der Input und Output-Funktion des GPIO-Moduls
Components  : GPIO
Requirements: GPIO_WRITE GPIO_READ

Procedure   : Es muessen die Tasten 0-3 betaetigt werden.
Test is OK  : Die zur Taste gehoerende Lampe leuchtet auf.
Test is NOK :
================================================================================================

================================================================================================
Testcase    : gpio2
Diretory    : test/module/gpio2

Description : Testen der Read-Funktion auf Ausgaengen
Components  : GPIO
Requirements: GPIO_WRITE GPIO_READ GPIO_READWRITTEN

Procedure   :
Test is OK  : LED 0.0 leuchtet
Test is NOK :
================================================================================================

================================================================================================
Testcase    : gpio3
Diretory    : test/module/gpio3

Description : Testen der maximal und minimal-Werte der Parameter von GPIO
Components  : GPIO
Requirements: GPIO_WRITE GPIO_READ

Procedure   :
Test is OK  : LED 0.0 leuchtet
Test is NOK :
================================================================================================
```

Testfälle: COIL



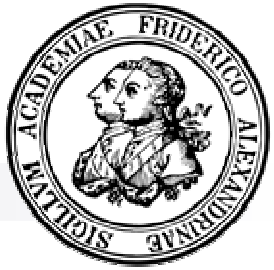
- Was muss getestet werden?
  - Ein und Ausschalten von Spulen
  - Abfrage welche Spule aktiv ist
  - Abfrage ob eine bestimmte Spule aktiv ist
  - Grenzen der Funktions-Parameter
- Testfälle:
  - coil1-3

Testfälle: COIL




```
================================================================================================
Testcase    : coil1
Diretory    : test/module/coil1

Description : Testen der ON und OFF Funktionen, sowie das max eine Spule aktiv ist
Components  : GPIO COIL
Requirements: GPIO_WRITE GPIO_READ COIL_ON COIL_OFF COIL_MAXONE

Procedure   :
Test is OK  : Alle Spulen werden 2x der Reihe nach aktiviert, am Ende ist alles wieder aus
Test is NOK : Mehr als eine Spule ist gleichzeitig an, am Ende bleibt eine Spule an
================================================================================================

================================================================================================
Testcase    : coil2
Diretory    : test/module/coil2

Description : Testen der Abfragemoeglichkeiten des COIL-Moduls
Components  : GPIO COIL
Requirements: GPIO_WRITE GPIO_READ COIL_ON COIL_MAXONE COIL_STATUS

Procedure   :
Test is OK  : Die Spulen werden der Reihe nach aktiviert, am Ende leuchtet Spule 0 (= alle Abfragen OK)
Test is NOK : Mehr als eine Spule ist gleichzeitig an, am Ende leuchtet Spule 1
================================================================================================

================================================================================================
Testcase    : coil3
Diretory    : test/module/coil3

Description : Testen der Grenzwerte der Funktionsparameter von COIL
Components  : GPIO COIL
Requirements: GPIO_WRITE GPIO_READ COIL_ON COIL_MAXONE

Procedure   :
Test is OK  : Die Spulen werden der Reihe nach aktiviert, am Ende leuchtet Spule 0 (= alle Abfragen OK)
Test is NOK : Mehr als eine Spule ist gleichzeitig an, am Ende leuchtet Spule 1 (= eine Abfrage NOK)
================================================================================================
```

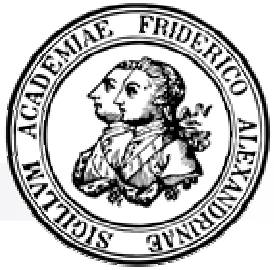
Testfälle: PHOTOSENSOR



- Was muss getestet werden?
  - Abfrage ob eine bestimmte LS aktiv ist
  - Interrupt-Auslösung (none, rising, falling, both)
  - Interrupt-Maskierung (nur bei bestimmter LS)

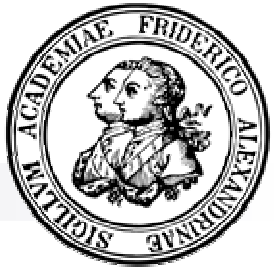
- Testfälle:
  - photosensor1 – photosensor4

Testfälle: PHOTOSENSOR



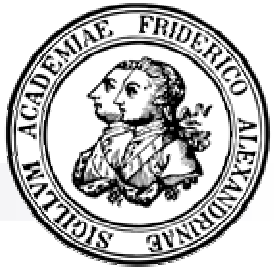

```
================================================================================================
Testcase    : photosensor1
Diretory    : test/module/photosensor1

Description : Abfrage der Lichtschranken (polling)
Components  : GPIO PHOTOSENSOR
Requirements: GPIO_WRITE GPIO_READ PHOTOSENSOR_STATUS

Procedure   : Jede Lichtschranke sollte der Reihe nach aktiviert werden
Test is OK  : Die zur Lichtschranke gehoerende LED leuchtet solange die Lichtschranke unterbrochen ist
Test is NOK :
================================================================================================

================================================================================================
Testcase    : photosensor2
Diretory    : test/module/photosensor2

Description : Testen der Interrupt-Funktion der Lichtschranken
Components  : GPIO PHOTOSENSOR
Requirements: GPIO_WRITE GPIO_READ PHOTOSENSOR_STATUS PHOTOSENSOR_INTERRUPT

Procedure   : Alle Lichtschranken sollten der Reihe nach unterbrochen und wieder freigegeben werden.
Test is OK  : Die zur Lichtschranke gehoerende LED leuchtet solange die Lichtschranke unterbrochen ist
Test is NOK :
================================================================================================

================================================================================================
Testcase    : photosensor3
Diretory    : test/module/photosensor3

Description : Testen der speziellen Interrupt-Ausloesung (nur rising, nur falling, none) der Lichtschranken
Components  : GPIO PHOTOSENSOR
Requirements: GPIO_WRITE GPIO_READ PHOTOSENSOR_STATUS PHOTOSENSOR_INTERRUPT PHOTOSENSOR_INTERRUPTMODE

Procedure   : Alle Lichtschranken muessen der Reihe nach unterbrochen und wieder freigegeben werden.
Test is OK  : LED0 leuchtet bei Unterbrechnung von PS0, LED1 bei Freigabe von PS1, LED2 gar nicht, LED3 geht bei PS3
              an und wieder aus.
Test is NOK :
================================================================================================
```

Testfälle: PHOTOSENSOR



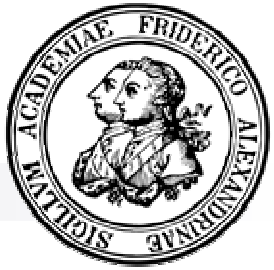

```
================================================================================================
Testcase    : photosensor4
Diretory    : test/module/photosensor4

Description : Testen der Interrupt-Maskierung (nur bestimmte LS) der Lichtschranken
Components  : GPIO PHOTOSENSOR
Requirements: GPIO_WRITE GPIO_READ PHOTOSENSOR_STATUS PHOTOSENSOR_INTERRUPT PHOTOSENSOR_INTERRUPTMASK

Procedure   : Alle Lichtschranken sollen der Reihe nach unterbrochen und wieder freigegeben werden.
Test is OK  : nur LED0 leuchtet bei Unterbrechung von LS0 und geht bei Freigabe wieder aus.
Test is NOK :
================================================================================================
```

Testfälle: CONFIG



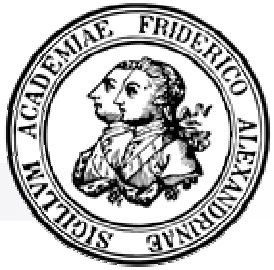
- Was muss getestet werden?
  - Eingabe und Abruf eines Programms
  - Mehrmaliger Aufruf von NewProgram
  - Maximale Anzahl von Befehlen
- Wie kann man dies testen?
  - Aufruf von AddXXX() und danach GetCommand()
- Testfälle:
  - config1 – config2

Testfälle: CONFIG



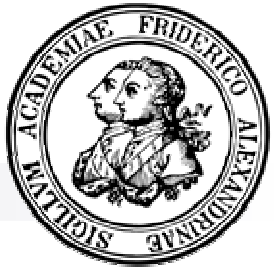

```
================================================================================================
Testcase    : config1
Diretory    : test/module/config1

Description : Es werden verschiedene Befehle zum CONFIG-Modul hinzugefuegt und danach wieder ausgelesen.
Components  : GPIO CONFIG
Requirements: GPIO_WRITE CONFIG_NEW CONFIG_ADD CONFIG_GET

Procedure   :
Test is OK  : LED0 leuchtet.
Test is NOK :
================================================================================================

================================================================================================
Testcase    : config2
Diretory    : test/module/config2

Description : Es werden viele und zu viele Befehle zum CONFIG-Modul hinzugefuegt und danach wieder ausgelesen.
Components  : GPIO CONFIG
Requirements: GPIO_WRITE CONFIG_NEW CONFIG_ADD CONFIG_GET

Procedure   :
Test is OK  : LED0 leuchtet.
Test is NOK :
================================================================================================
```

Testfälle: CONTROL



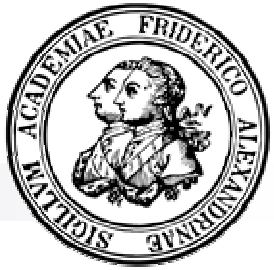
- Was muss getestet werden?
  - stimmt die Übersetzung in Aufrufe von CONFIG?
  - Continous UP / DOWN
  - Stepwise UP / DOWN
  - Oscilation
- Wie kann man dies testen?
  - Statt CONFIG wird ein Stub eingefügt der die Aufrufe abfängt und checkt
- Testfälle:
  - control1 – control3

Testfälle: CONTROL



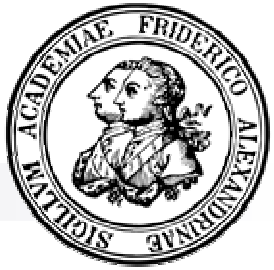

```
================================================================================================
Testcase    : control1
Diretory    : test/module/control1

Description : Es werden CONTINOUS-Befehle abgesetzt und gecheckt ob die Funktionen von CONFIG aufgerufen werden.
Components  : GPIO CONTROL
Requirements: GPIO_WRITE CONTROL_CONTINOUS

Procedure   :
Test is OK  : LED0 leuchtet.
Test is NOK :
================================================================================================

================================================================================================
Testcase    : control2
Diretory    : test/module/control2

Description : Es werden STEPWISE-Befehle abgesetzt und gecheckt ob die Funktionen von CONFIG aufgerufen werden.
Components  : GPIO CONTROL
Requirements: GPIO_WRITE CONTROL_STEPWISE

Procedure   :
Test is OK  : LED0 leuchtet.
Test is NOK :
================================================================================================

================================================================================================
Testcase    : control3
Diretory    : test/module/control3

Description : Es werden OSCILLATE-Befehle abgesetzt und gecheckt ob die Funktionen von CONFIG aufgerufen werden.
Components  : GPIO CONTROL
Requirements: GPIO_WRITE CONTROL_OSCILATE

Procedure   :
Test is OK  : LED0 leuchtet.
Test is NOK :
================================================================================================
```

Testfälle: FSM



- Wie kann man die FSM testen?
  - eigentlich nur die „RUN“ Funktion vorhanden, dies wäre aber schon der Integrationstest
- zwei Möglichkeiten:
  - FSM zerlegen und einzelne Funktionen testen
  - viele Stubs einfügen für COIL, CONFIG und PS – außerdem Timer emulieren
- Noch keine Testfälle da noch nicht implementiert

Ausblick



- Was muss noch getestet werden?
  - Performance (WCET-Messung)
  - Timer-Modul falls keine Alarme verwendet werden
- Was kann man noch verbessern?
  - Automatisch ablaufende Testfälle mit Debugger-Skripten => Problematisch wegen Benutzerinteraktion
- Was fehlt noch?
  - FSM-Testfälle