WCET-Analyse “Hau den Lukas”

Guilherme Bufolo Martin Oettinger Dirk Pfeiffer

23.06.2009

Inhaltsverzeichnis

Werkzeug / Einstellungen

- verwendetes Werkzeug: ait
- standard EBU Einstellungen

abwaertsbew_bremsen

abwaertsbew_bremsen

- 1 activate_coil
- 1 get_cur_coil

$\longrightarrow$ 99.821 μs

anheben

anheben

- 1 activate_coil
- 1 deactivate_coil
- 2 get_cur_coil

$\rightarrow$ 187.935 μs

aufwaertsbew_bremsen

aufwaertsbew_bremsen

- 1 activate_coil
- 1 deactivate_coil
- 2 get_cur_coil

$\rightarrow$ 187.935 μs

festhalten

festhalten

- 1 activate_coil
- 1 deactivate_coil
- 2 get_cur_coil

$\rightarrow$ 187.935 μs

initial_anheben

initial_anheben

- 2 activate_coil
- 1 deactivate_coil
- 3 get_cur_coil

$\rightarrow$ 287,756 μs

weiterheben

weiterheben

- 1 activate_coil
- 1 deactivate_coil
- 2 get_cur_coil

$\rightarrow$ 187.935 μs

gpio_set_value

- 1504 cycles / 10.027 μs
- user stack usage: 0..16 bytes

gpio_get_value

- 466 cycles / 3.04 μs
- user stack usage: 0..8 bytes

gpio_init

- 34816 cycles / 0.233 ms
- user stack usage: 0..72 bytes

gpio_set_function

- 2635 cycles / 17.587 μs
- user stack usage: 0..64 bytes

deactivate_all_coils

- 12226 cycles / 81.507 μs
- user stack usage: 0..24 bytes

activate_coil

- 13982 cycles / 93.214 μs
- user stack usage: 0..32 bytes

deactivate_coil

- 1594 cycles / 10.627 μs
- user stack usage: 0..24 bytes

get_last_act_coil

- 151 cycles / 1.007 μs
- user stack usage: 0 bytes

get_cur_coil

- 991 cycles / 6.607 μs
- user stack usage: 0..8 bytes

isr

- 892 cycles / 5.947 μs
- user stack usage: 0..16 bytes

gpio_test

gpio_test

- dieser Test überprüft
  - ob die Spulen richtig angesteuert werden
  - ob die Inputpins richtig ausgelesen werden
- Test arbeitet im Abfragebetrieb ledstatus ab

Input:

- alle LEDs leuchten für einige Sekunden auf
- Schieber in Lichtschranke schieben $\longrightarrow$ jeweilige LED soll aufleuchten

Es wird kein anderes Modul benutzt

int_test

int_test

- dieser Test überprüft
  - ob Interrupts generiert und abgefangen werden
- Test benutzt Interrupt, um durchbrochene Lichtschranke abzufangen

Input:

- Schieber wird auf Testboard in eine Lichtschranke geschoben
- und wieder hinaus bewegt

Benutzte Module:

- gpio

coil_control_test

coil_control_test

- dieser Test überprüft ob die eingesetzen Funktionen korrekt funktionieren
- LEDs werden hintereinander eingeschaltet
- beim Einschalten einer LED werden alle anderen abgeschaltet
- $\longrightarrow$ nur eine LED darf gleichzetig aktiv sein
- zwischen Schaltvorgängen 0.5s Pause
- danach: Einschalten aller LEDs, Ausschalten aller LEDs (deactivate_all_coils)
- weiterhin: Einschalten aller LEDs, Ausschalten von LED 0, 2, 4

Kein Input benötigt
Benutzte Module:

- gpio
- coil_control

thread_delay_test

thread_delay_test

- dieser Test überprüft ob sich thread_delay korrekt verhält
- erstellen eines Threads
- Thread terminiert niemals
- eine LED wiederholt 2s an, 2s aus

Kein Input benötigt
Benutzte Module:

- gpio
- coil_control

sched_test

sched_test

- dieser Test überprüft ob der Scheduler funktioniert
- Schieber wird durch Lichtschranke geführt, an welcher eine LED leuchtet
- LED soll erlöschen, nachfolgende LED aufleuchten
- Vorgang wiederholen, bis "Start-LED" wieder erreicht
- Beim passieren der falschen Lichtschranke: alle LEDs leuchten auf $\rightarrow$ Fehlererkennung
- im realen System: Abschalten aller Spulen

Input:

- Schieber wird auf Testboard durch Lichtschranken geschoben

Benutzte Module:

- gpio
- test_thread
- thread_delay

Vielen Dank für Ihre Aufmerksamkeit!
Fragen?