Neues vom STRIP Forth-Prozessor

Tagung der Forth-Gesellschaft

April 2011 in Goslar

Willi Stricker

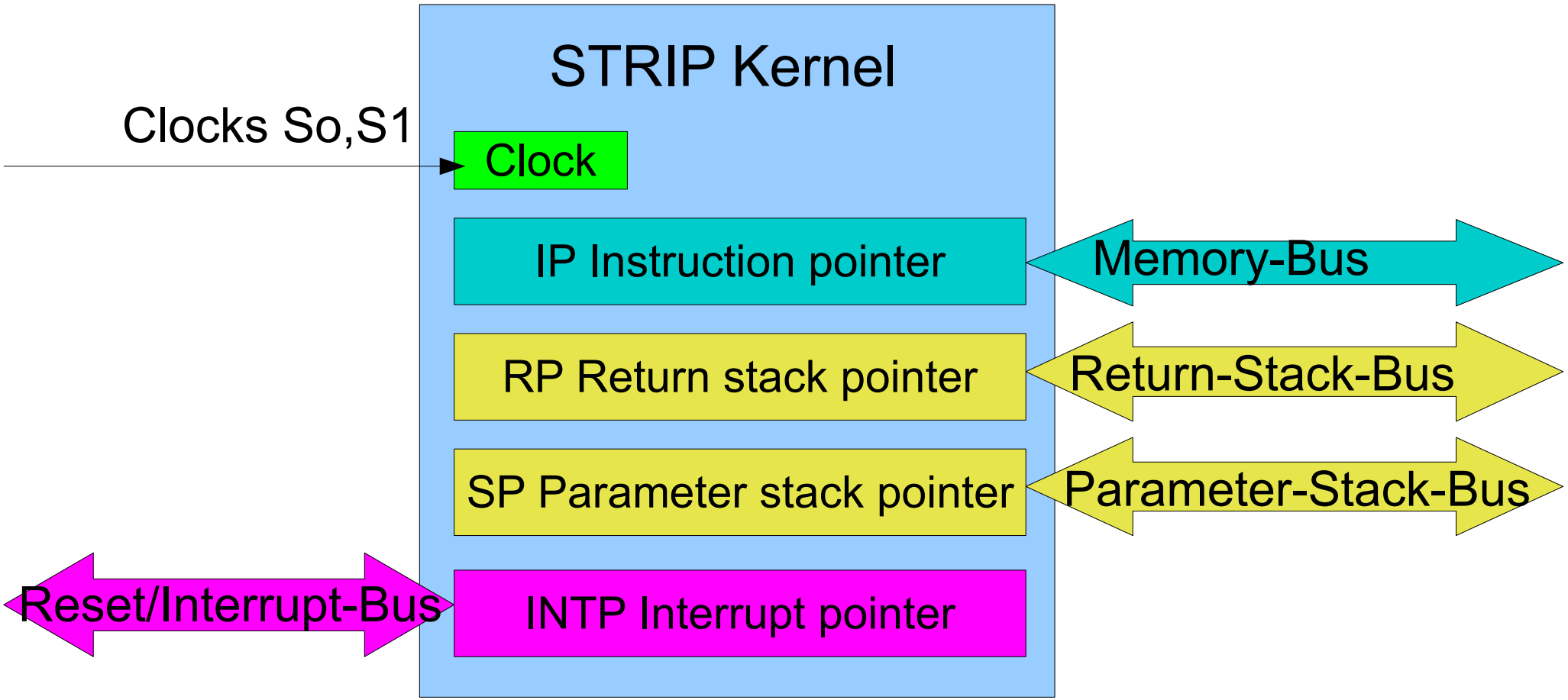
STRIP Forth-System

Praxisdemonstration

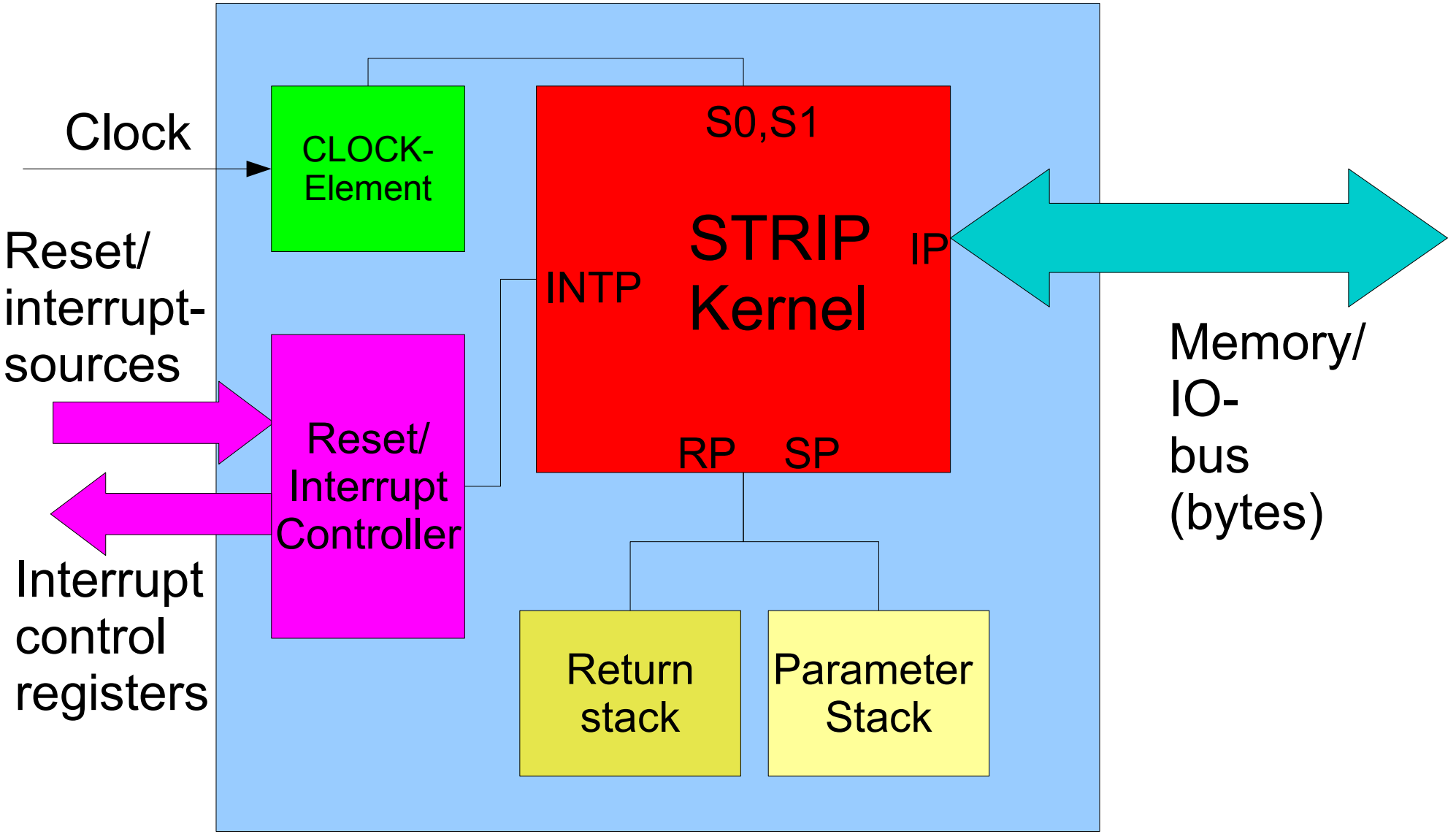
Aufbau Hardware

Aufbau Software

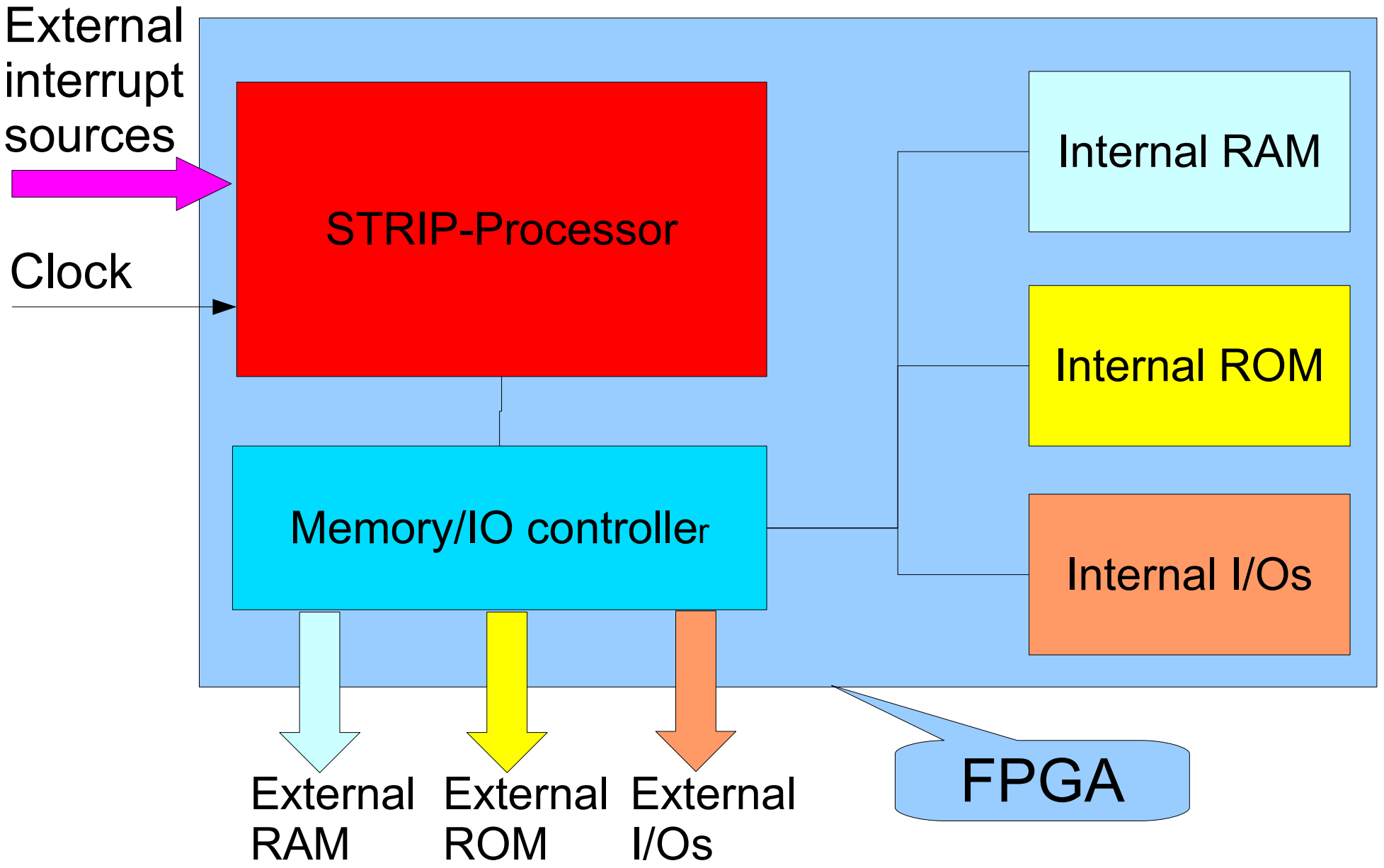
STRIP-Kernel



STRIP-Prozessor



STRIP-Controller



STRIP-Controller
interne Interrupt Quelle

Stack Interrupt

Nicht sperrbarer Interrupt, erzeugt im Kernel

wird ausgelöst bei
- Return-Stack-Überlauf = ROV
- Return-Stack Unterlauf = RUN
- Parameter-Stack Überlauf = ROV
- Paraeter-Stack-Unterelauf = RUN

Control-Byte: Jede Störquelle hat ein Control-Bit

7	6	5	4	3	2	1	0
XXX	XXX	XXX	XXX	ROV	RUN	POV	PUN

STRIP-Controller
interne Interrupt Quelle

Real time Interrupt

Sperrbarer interrupt,
wird von einem programmierbaren Timer ausgelöst

Funktion:
16-Bit-Zählregister, getaktet durch internen Takt.
Das Laden mit einer Zahlt startet den Timer, er zählt rückwärts. Bei null wird der Interrupt ausgelöst.

STRIP-Controller
Interne I/Os

Eingebaute I/O-Elemente:

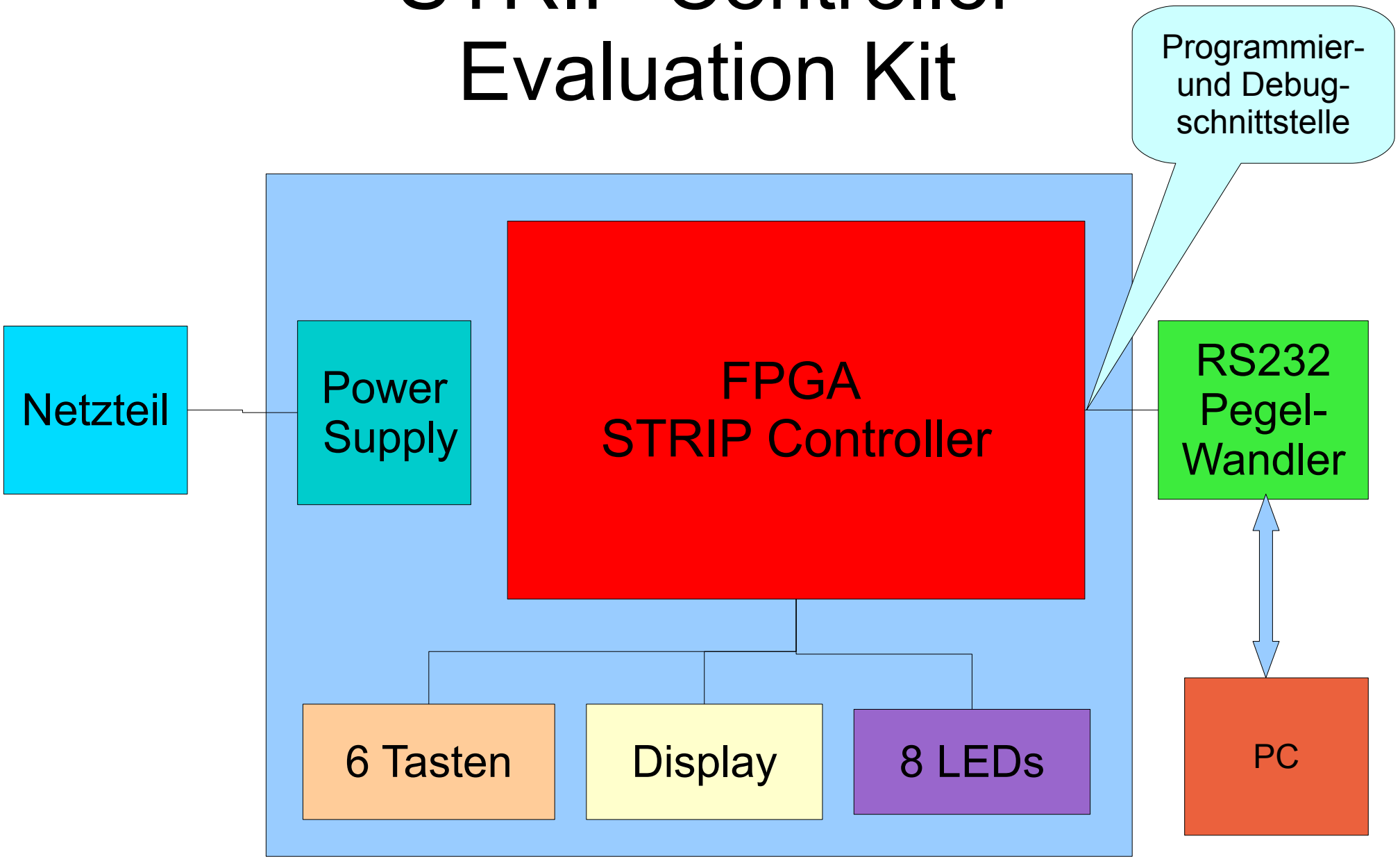
- Stack-Interrupt
- Timer für Real-Time-Interrupt
- Output Register (Parallelports)
- Input Register (Parallelports)
- **Programmier- und Debug-Schnittstelle**

STRIP-Controller Evaluation Kit

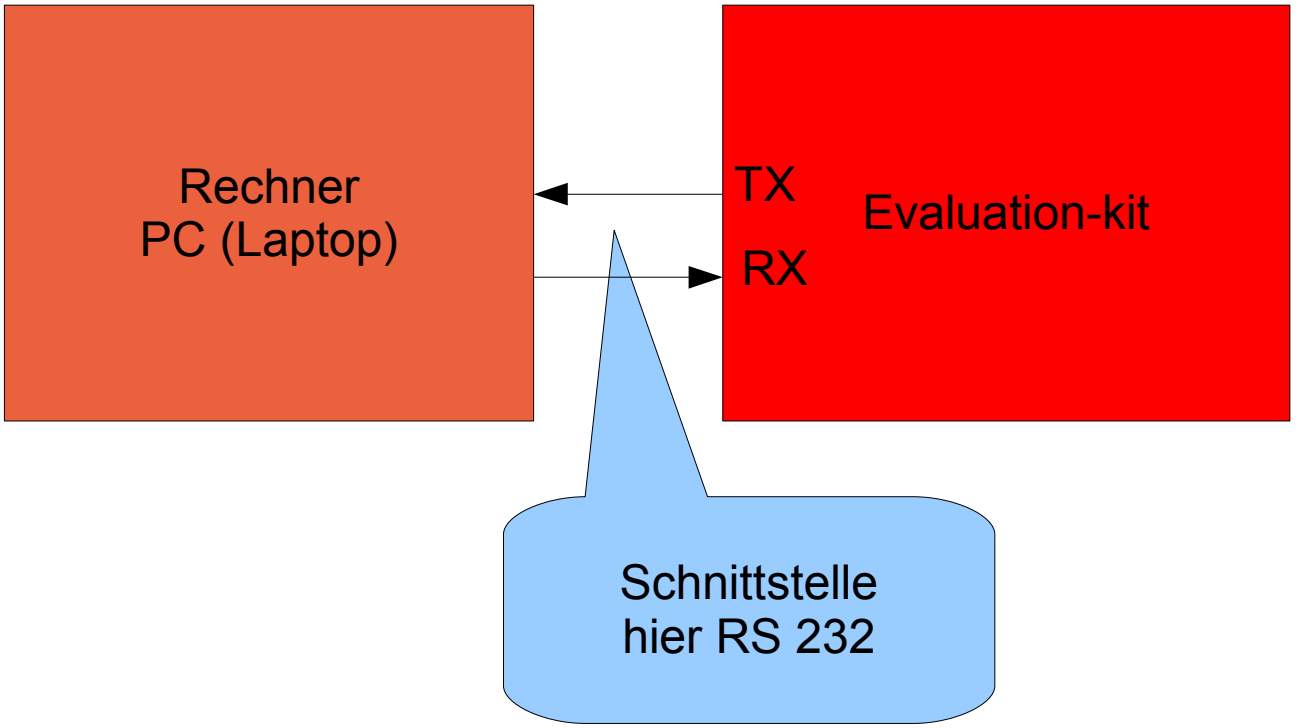
Enthält:

- FPGA: Es enthält den STRIP-Controller
- In/Out-pins des FPGA, davon werden 2 für die Programmier- und Debug-Schnittstelle verwendet;
- 6 Tasten: Boot-pin, Reset-Pin 2 Interrupt-Pins, 2 freie Tasten;
- 8 Leuchtdioden;
- 2 Hex-Schalter;
- ein kleines LCD-Display.

STRIP-Controller Evaluation Kit



Programmier- und Debug-Schnittstelle



Die Schnittstelle arbeitet im Halbduplex-Betrieb:
Sende- und Empfangs-Betrieb zeitlich getrennt,
nicht gleichzeitig

Programmier- und Debug-Schnittstelle

Software

Eigenschaften:

Die Schnittstellen-Programme für den Host (Rechner, PC) und den Target (Eval-Kit, FPGA) sind (nahezu) identisch.
Der Daten-Transfer erfolgt Byte-weise im Software-Handshake
Der Host ist immer Master, der Target immer Slave.
Der Datentransfer erfolgt asynchron im UART-Format.

Programmier- und Debug-Schnittstelle

Arbeitsablauf bei der Ausführung eines Target-Befehls durch den Host

Host		Target
• arbeitet		• wartet
• sendet Parameter	=>	• empfängt Parameter
• sendet Befehl (cfa)	=>	• empfängt Befehl (cfa)
• wartet		• führt Befehl aus
• empfängt Parameter	<=	• sendet Parameter
• arbeitet weiter		• wartet

STRIP-Controller
Praxis-Demonstration

Die Demonstration erfolgt auf einem Eval-Board der Firma Actel (ProASIC3/E Starter Kit) mit dem FPGA Typ A3P250

Der FPGA-Baustein enhält:

- 6144 „Tiles“ (Logikeinheiten mit 3 Eingängen und einem Ausgang zur Realisierung von Flipflops oder Logikgattern),
- dazu 4 kB RAM.

Benutzt werden davon:

- für den STRIP-Kernel: ca. 1700 Tiles,
- für den STRIP-Controller insgesamt: ca. 2500 Tiles,
- für die Stacks 1,5 kB RAM
- für den Daten-Speicher: 2,5 kB RAM

STRIP-Controller
Praxis-Demonstration

Das Bootprogramm für die Programmier- und Debug-Schnittstelle wird beim (Hardware-) Reset aufgerufen, wenn gleichzeitig der Boot-Pin aktiviert wird (Logik-Signal „high“)

Mit Hilfe eines korrespondierendem Schnittstellen-Service-Programms im Host-Rechner ist eine Korrespondenz zwischen beiden möglich

STRIP-Controller
Praxis-Demonstration

Über die Rechner-Schnittstelle werden auf den Target folgende Programmteile ins interne RAM geladen:

- Der Forth-Kernel,
- ein Real Time Interrupt-Programm (Taktzeit 100 ms mit Blinker),
- ein kleines Echtzeit Multitasking Betriebssystem,
- ein einfaches Blinkprogramm mit einer LED,
- ein manipulierbares Blinkprogramm mit 6 LEDs.

STRIP-Controller
Praxis-Demonstration

Programm-Ablauf:
Drei Programme werden als einzelne Tasks definiert:

- Zerotask: Schnittstellen-Programm
- Task1: Festes Blinkprogramm
- Task2: Variierbares Blinkprogramm

Sie werden mit Hilfe des Betriebssystems quasi-parallel abgearbeitet.
(Zeitscheiben-Verfahren im Round Robin Prinzip)

Forth Real Time Operating System FORTOS

Ziel:

Aufbau eines einfachen aber vielseitig einsetzbaren Echtzeit-Multitasking-Betriebssystems

FORTOS

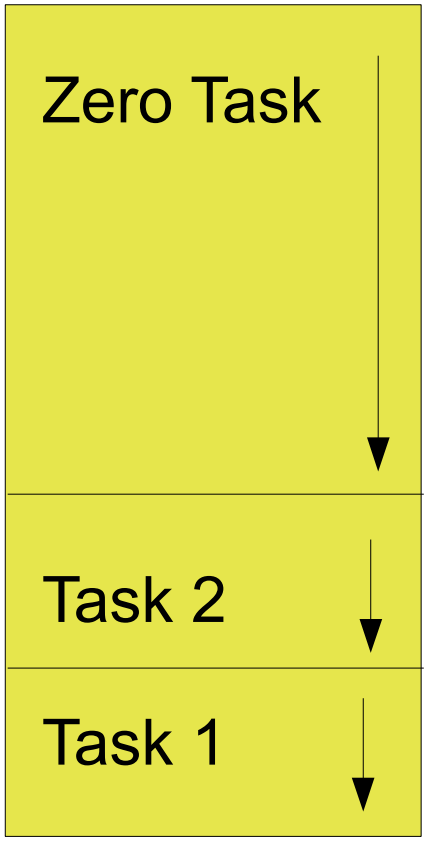
Aufbau

Jede Task erhält einen separaten Return-Stack und einen separaten Parameter-Stack

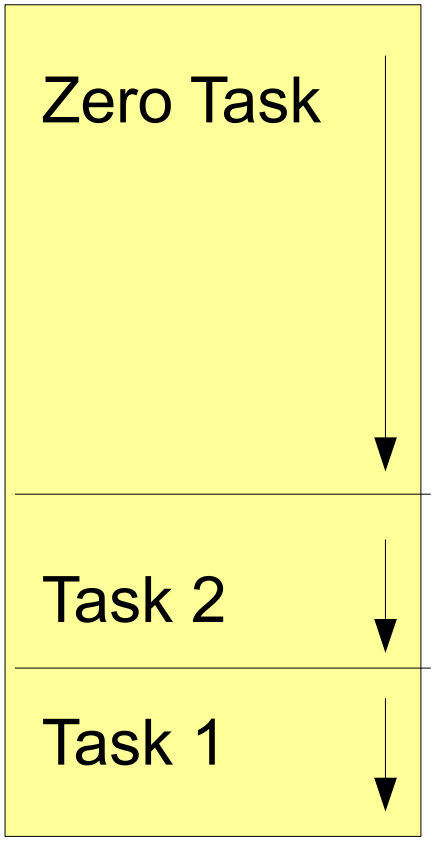
Die vorhandenen Stacks werden auf die einzelnen Tasks aufgeteilt, jede Task erhält einen Anteil des jeweiligen Gesamtstacks

Anmerkung:
Dann müssen beim Task-Wechsel keine Register- oder RAM-Inhalte umgeladen bzw. auf dem Stack gesichert werden

FORTOS
Stack-Aufteilung



Return stack



Parameter Stack

FORTOS
Task-Allocation Block TAB
(Task-Verwaltungs-Block)

Jede Task erhält im ROM (Programmspeicher) einen Task-Verwaltungs-Block. Die Blöcke der Tasks sind im Programm über eine verkettete Liste verbunden.

Der TAB enthält:

- RAM-Adresse für veränderbare Parameter
- Link-Adresse zur vorangegangenen Task
- Anfangs-Priorität
- Start-Adresse im Return-Stack
- Start-Adresse im Parameter-Stack
- Adresse des zugehörigen Task-Programms (Forth-Word)

Der Aufruf des Task-Namens ergibt die Adresse des TAB

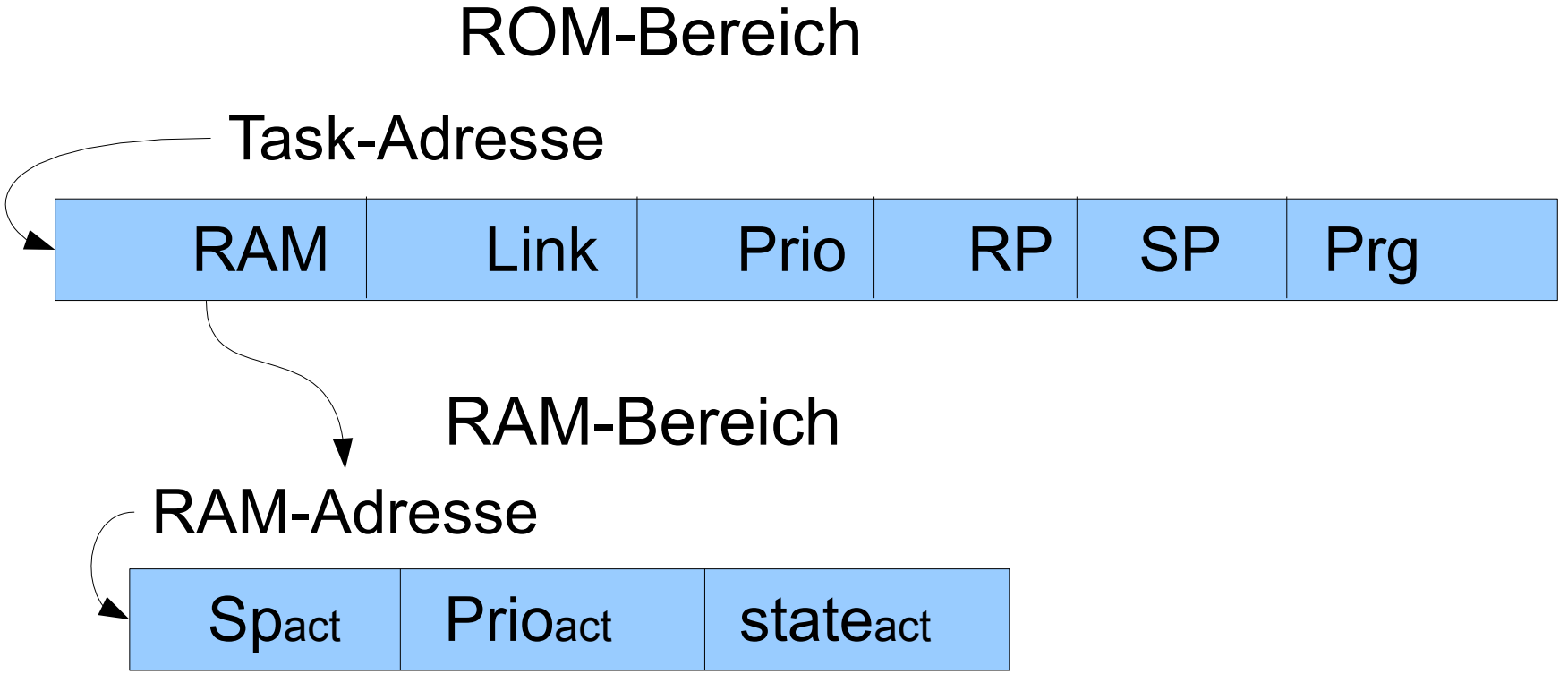
FORTOS
Task-Allocation Block TAB
(Task-Verwaltungs-Block)

Jede Task erhält im RAM (Datenspeicher) einen zusätzlichen Task-Verwaltungs-Bereich für veränderliche Daten. Er wird über die im TAB gespeicherte Adresse aufgerufen.

Er enthält:

- Aktueller SP (Parameter Stack-Pointer)
- Aktuelle Priorität
- Aktueller Task-Zustand

FORTOS
Task-Allocation Block TAB
(Task-Verwaltungs-Block)



FORTOS

Task-Umschaltung

Eingangsparameter: laufende Task, neue Task

3 Task-Parameter müssen gewechselt werden: IP, RP, SP

Die laufende Task muss gespeichert werden:
- aktuelle Programmadresse => Return Stack
- aktuelle Return-Stack-Position => Parameter Stack
- aktuelle Parameter-Stack-Position => Task-RAM

Die neue Task muss aktiviert werden:
- neue Parameter-Stack-Position <= Task-RAM
- neue Return-Stack-Position <= Parameter Stack
- neue Programmadresse aktivieren <= Return-Stack

FORTOS

Der Task-Schalter (Forth-Programm)

```
: TSK-SWTCH  ( srce-task dest-task - )
\ ------------------------------------
   >R              \ dest task >R
   @ SP@ SWAP ! \ srce stack pointer => store
   R<              \ dest task R<
   RP@             \ source return ptr fetch
   SWAP @ @ SP! \ dest stack ptr => store
   RP!             \ dest return pointer => store
;                 \ return to destination
```

Anmerkung: Wenn der Task-Schalter aufgerufen wird liegt die aktuelle Programmadresse (Instruction Pointer) auf dem Return-Stack!

FORTOS

Task-Dispatcher

```
: TSK-DISPATCH-RR ( - )
   TSK-RUN @ DUP
   BEGIN
      2+ @
      DUP -1 = IF DROP TSK-LAST @ THEN
      DUP @ 2+ C@ READY =
   UNTIL
   DUP TSK-RUN ! TSK-SWTCH
;
```

Anmerkungen: Variable
TSK-RUN enthält die laufende Task
TSK-LAST letzte Task in der verketteten Liste