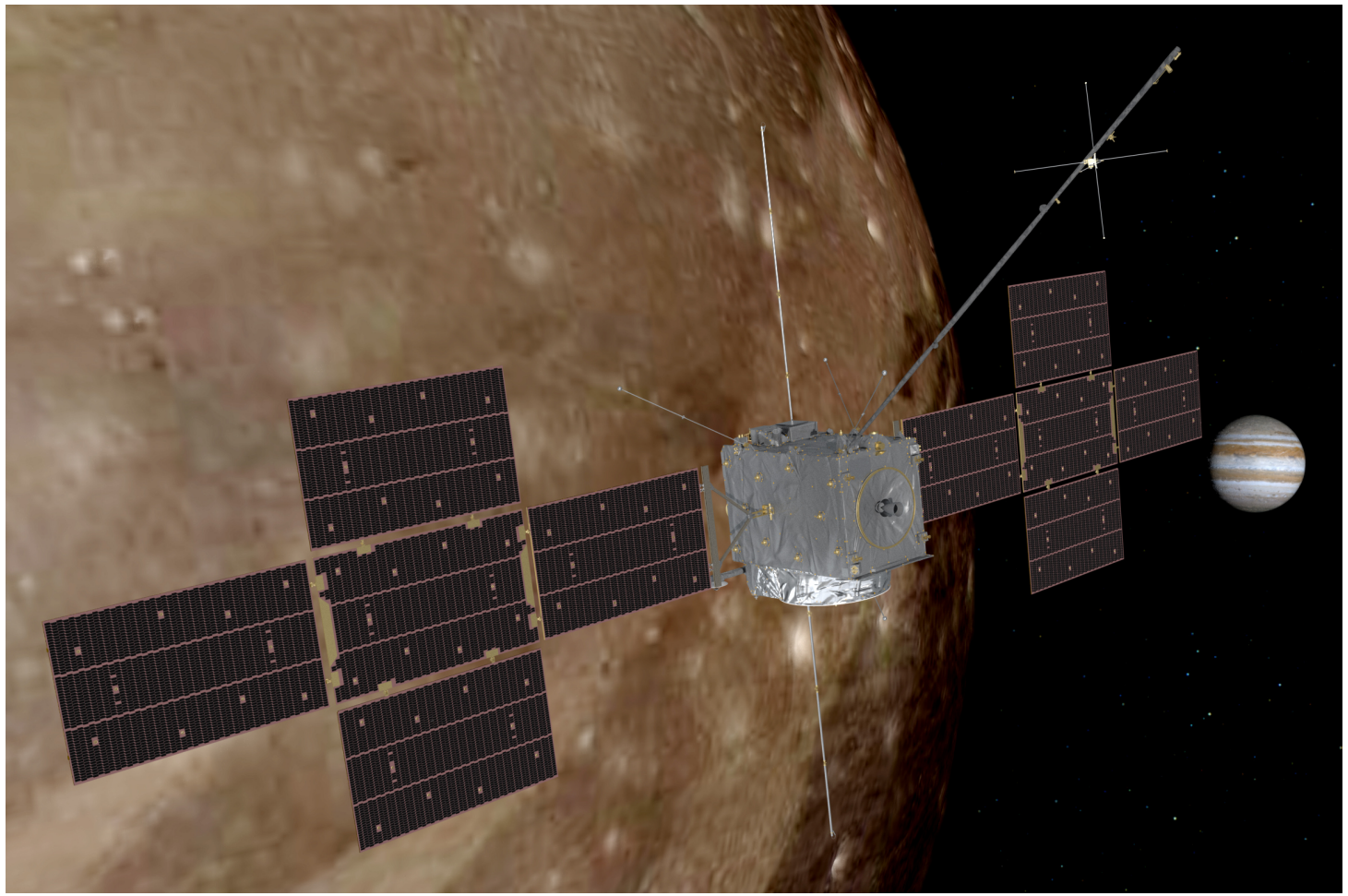
Kontrollsoftware für eine Jupitermission der ESA
Ein Erfahrungsbericht

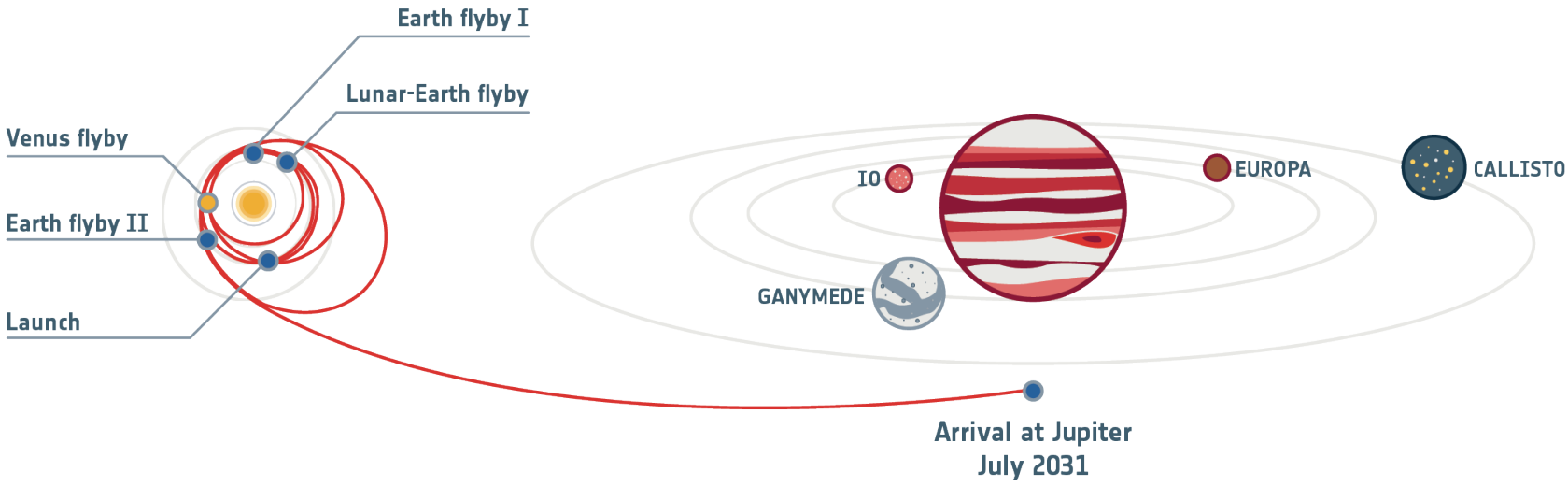
Oskar Schirmer, Felix Winkelmann

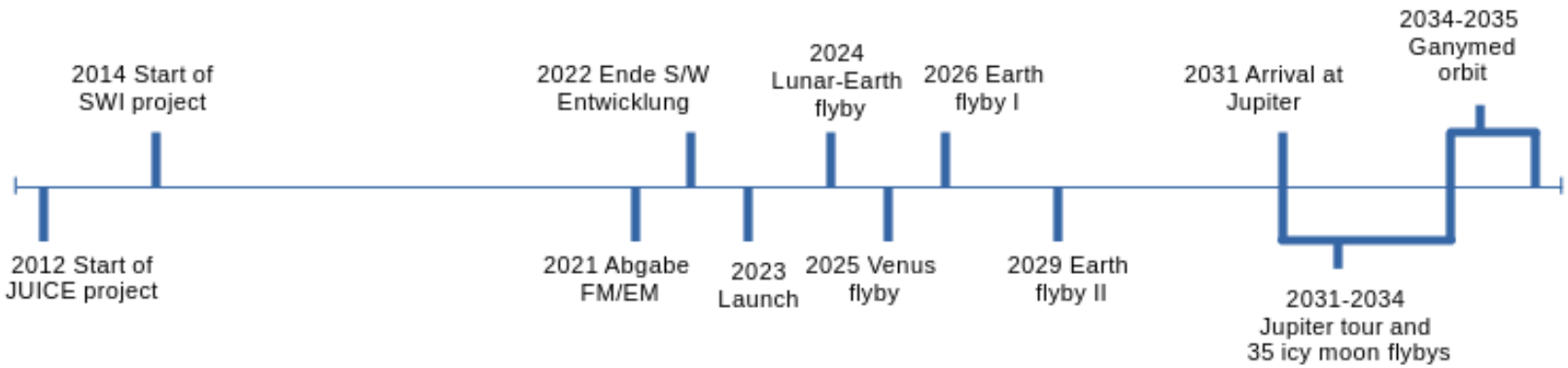
Göttingen, 15.3.2024

JUICE
European Jupiter Space Mission



JUICE

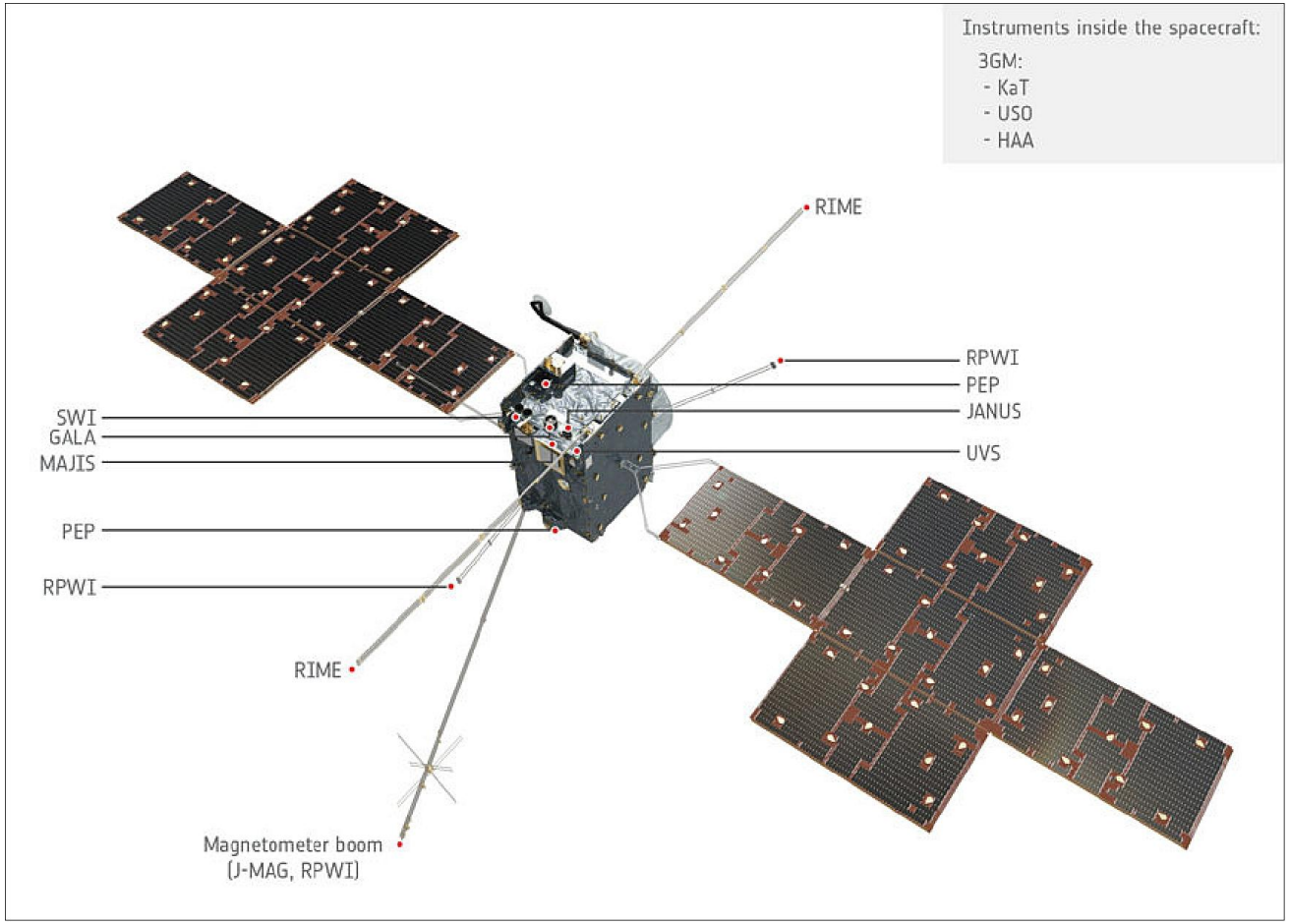

Earth flyby I
Lunar-Earth flyby
Venus flyby
Earth flyby II
Launch
IO
EUROPA
CALLISTO
GANYMEDE
Arrival at Jupiter
July 2031


2014 Start of SWI project
2012 Start of JUICE project
2022 Ende S/W Entwicklung
2021 Abgabe FM/EM
2023 Launch
2024 Lunar-Earth flyby
2025 Venus flyby
2026 Earth flyby I
2029 Earth flyby II
2031 Arrival at Jupiter
2031-2034 Jupiter tour and 35 icy moon flybys
2034-2035 Ganymed orbit

JUICE

Temperaturbereich: +250°C (Venus) / -230°C (Jupiter)
Elektromagnetische Strahlung: $>$ 100 GW (Erde: 0.1 GW)

JUICE



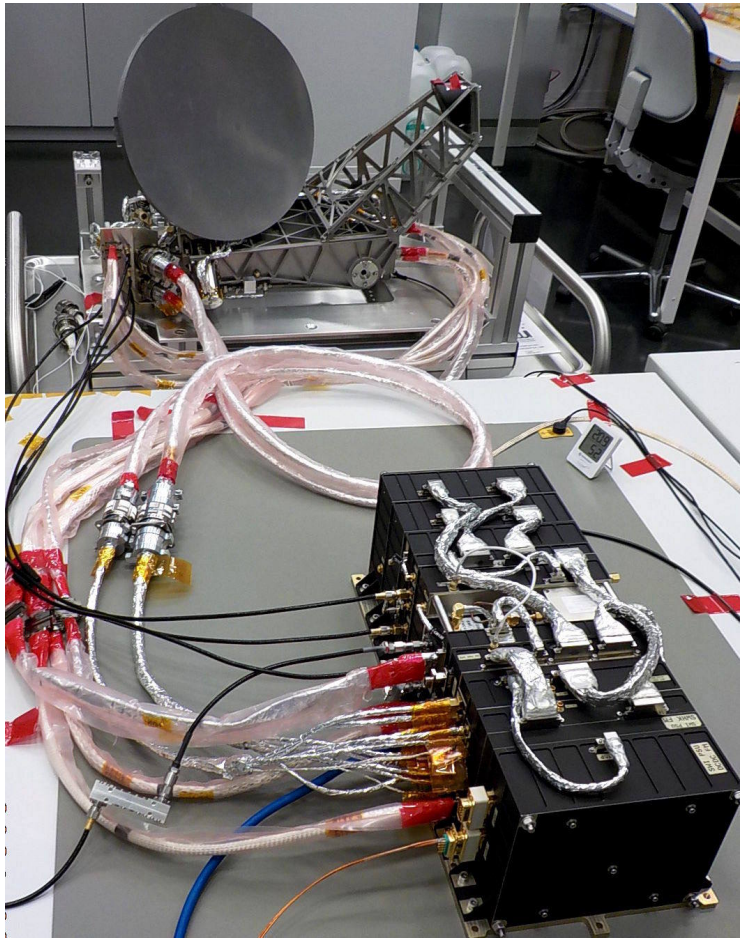
Solarpanel: 850 W, SWI: 60 W, DPU: 3 W

Sendelatenz Erde – Jupiter: 45 Minuten

Limitierte Bandbreite Uplink: 100 TC/Tag (je 226 Byte max.)

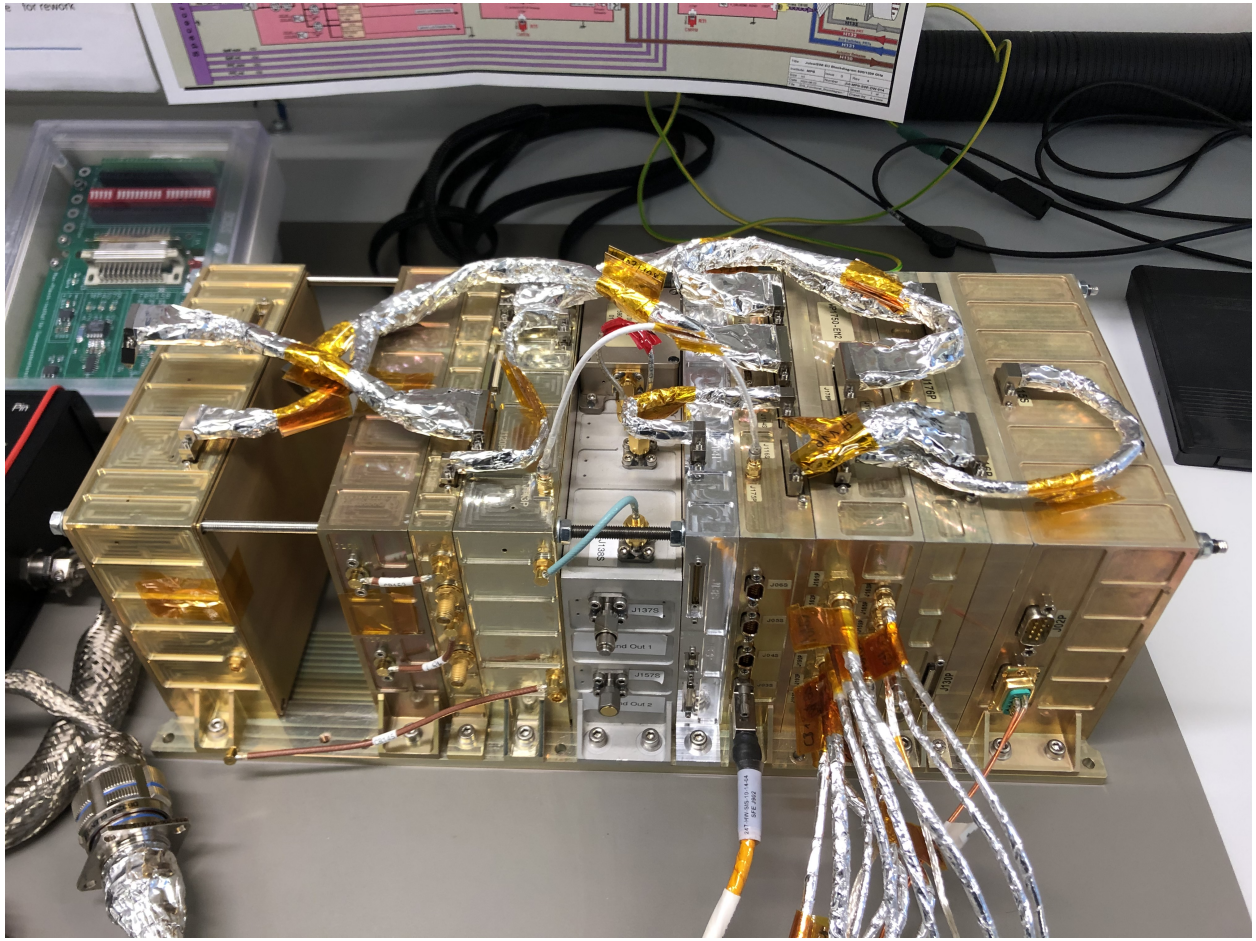
Bandbreite Downlink: 1.4 GB/Tag (total)

SWI



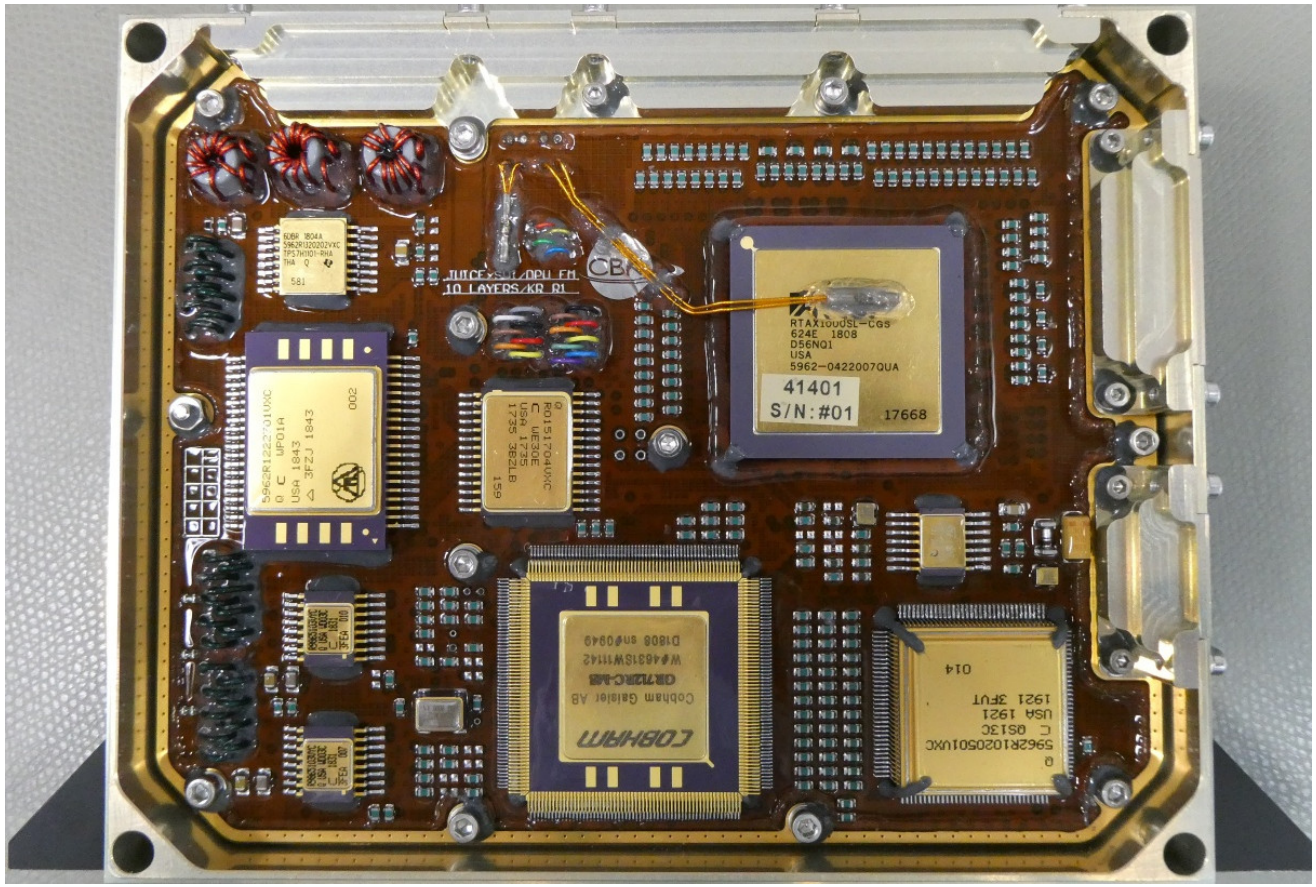
2 Spektrometer: 600 GHz / 1200 GHz
Schwebung gegen Chirp: 6 GHz $\rightarrow$ ASIC
40 kB pro Messung (typisch 60 sec, min 1.5 sec)

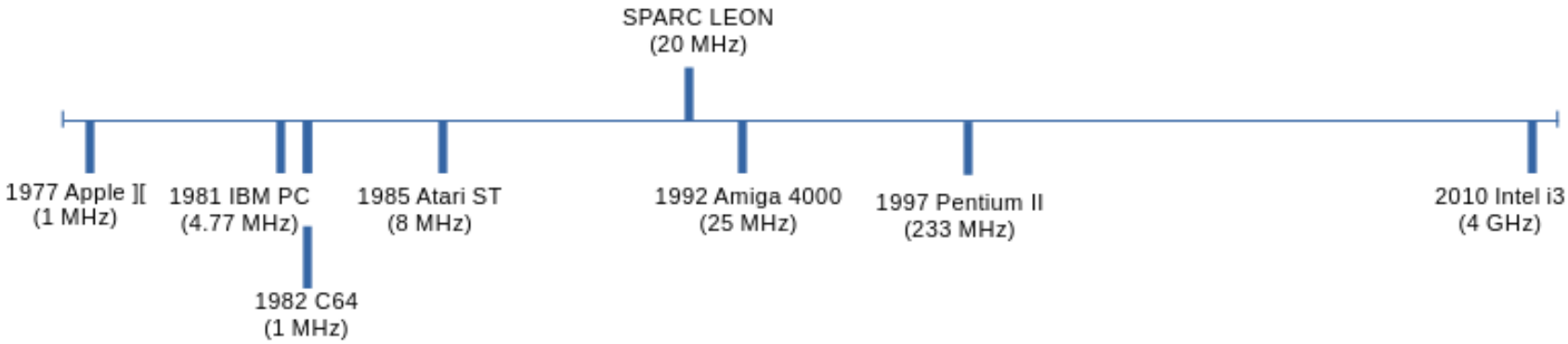
SWI



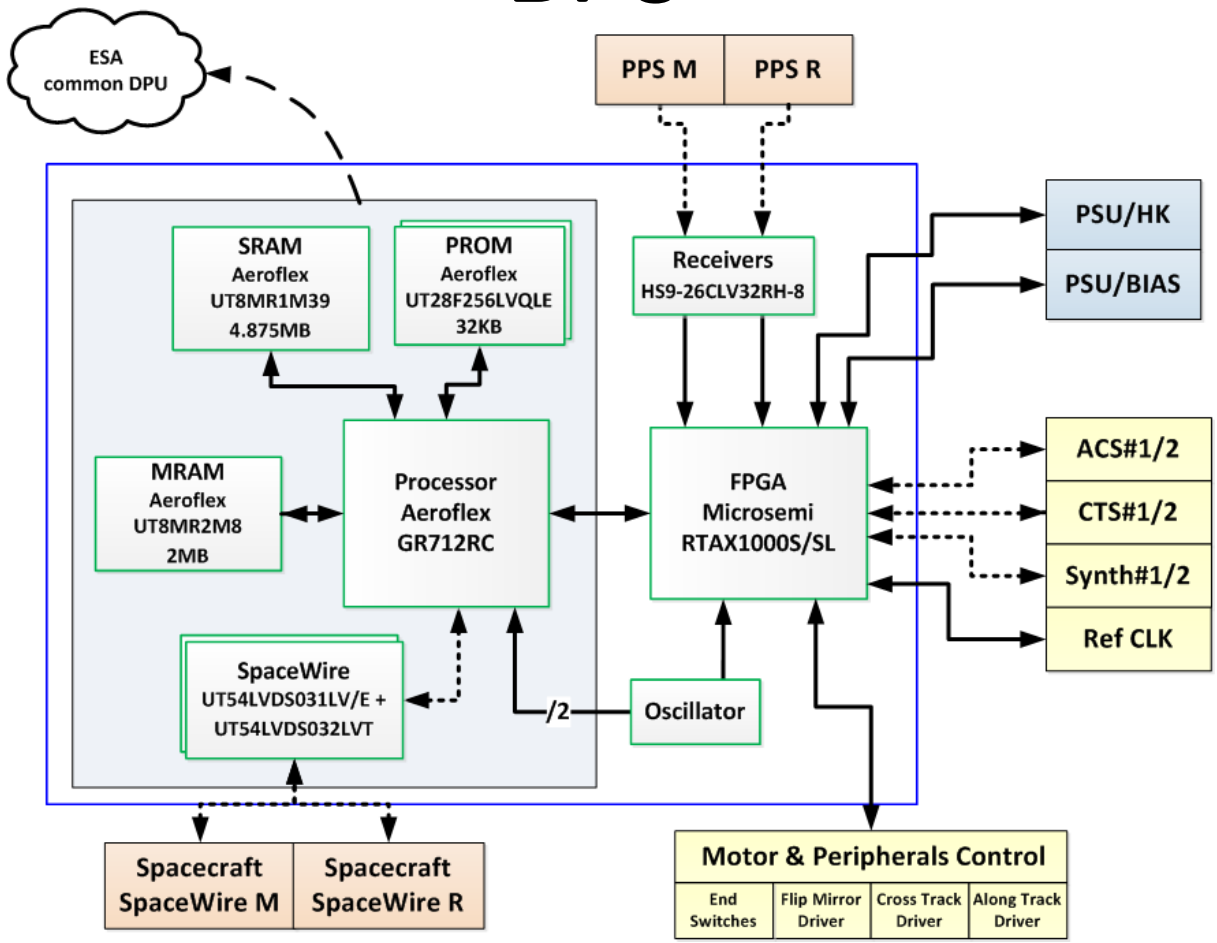
Electronic Unit: 10 Baugruppen
Bis 2021 nicht vollständig verfügbar

DPU


RTAX1000SL-CQ8
624E 1808
D56NQ1
USA
5962-0422007QUA
41401
S/N:#01
17668
COBHAM


SPARC LEON
(20 MHz)
1977 Apple][
(1 MHz)
1981 IBM PC
(4.77 MHz)
1982 C64
(1 MHz)
1985 Atari ST
(8 MHz)
1992 Amiga 4000
(25 MHz)
1997 Pentium II
(233 MHz)
2010 Intel i3
(4 GHz)

DPU



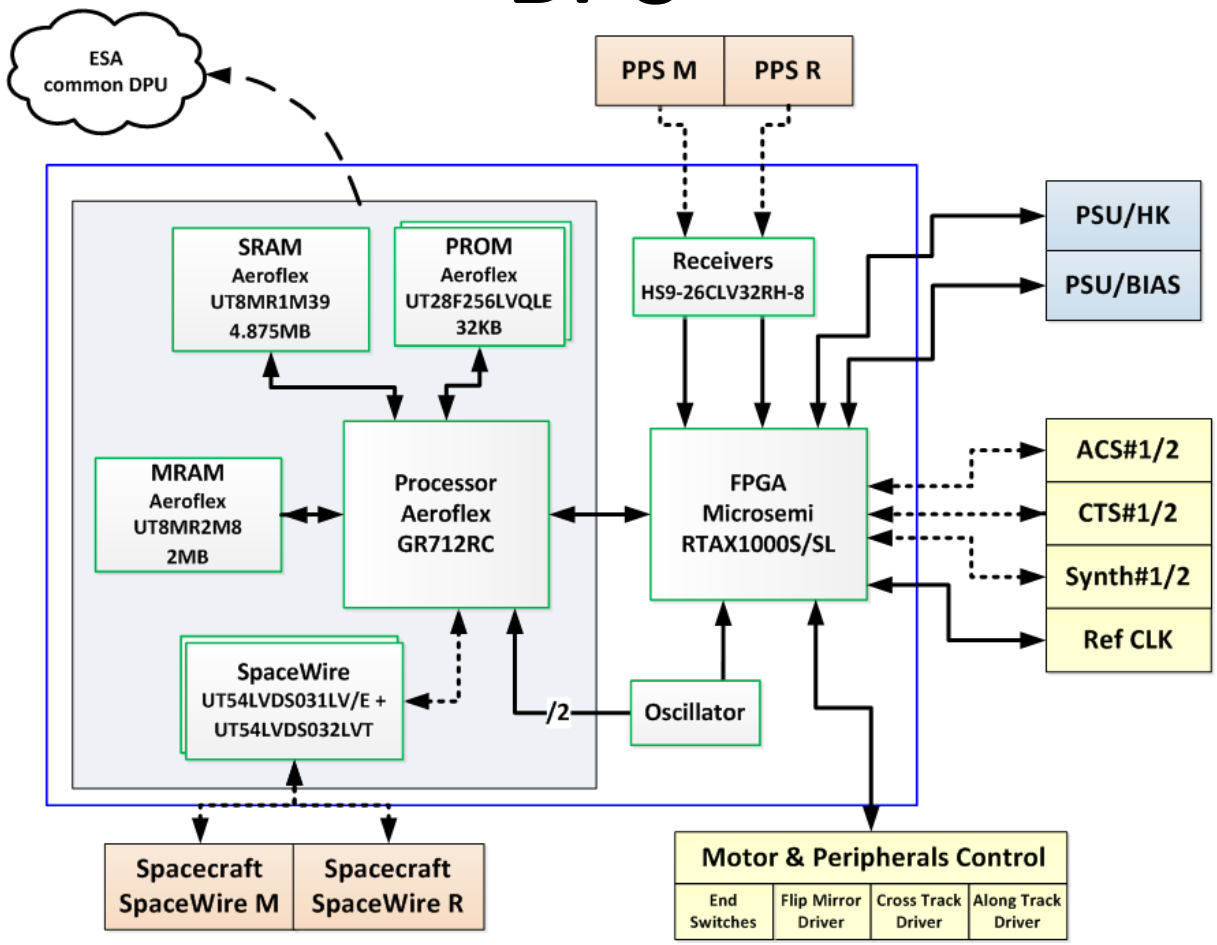
SpaceWire: LVDS basiert, Paket-orientiert, 40 Mbit/sec

PUS: Telecommand/Telemetrie (TC/TM), ESA Protokoll

PROM: bootloader “BSW” von ESA/CBK

GR712RC: LEON3 20 MHz 32 bit Sparc CPU

DPU

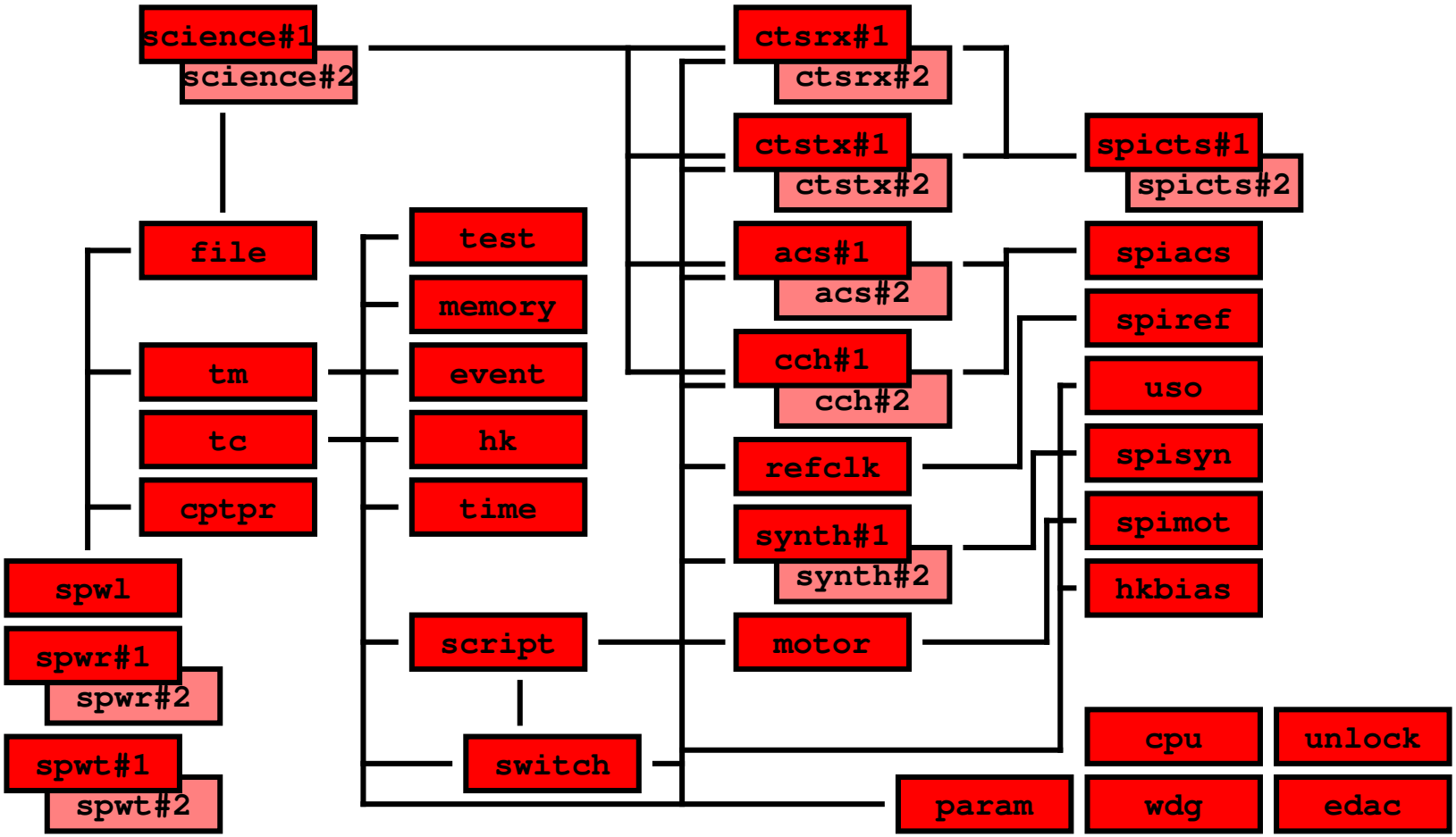


Sparc CPU: GNU/GCC toolchain, C

DM: JTAG dongle, UART $\rightarrow$ EM/FM: kein debugging

Software Update per SpaceWire ins MRAM

Software Architektur



34 Module, davon 9 “dual”, also 43 runtime Module

Jedes Modul implementiert eine state machine

Aufruf round robin $\rightarrow$ “cyclic executive”

Software Architektur

Erster Prototyp in C:
$\rightarrow$ zu aufwendig, komplex, fehleranfällig

Ursprüngliches Budget:
$\rightarrow$ 5 Jahre, 1 FTE Software Team komplett

Zwei halbe Stellen:
$\rightarrow$ Kontroll-Software
$\rightarrow$ Toolchain, Test-Setup, Datenbank

Spezifikation: laufende Änderungen, auch
im nicht-spezifischen Bereich unvollständig bis Anfang 2022

Software Architektur

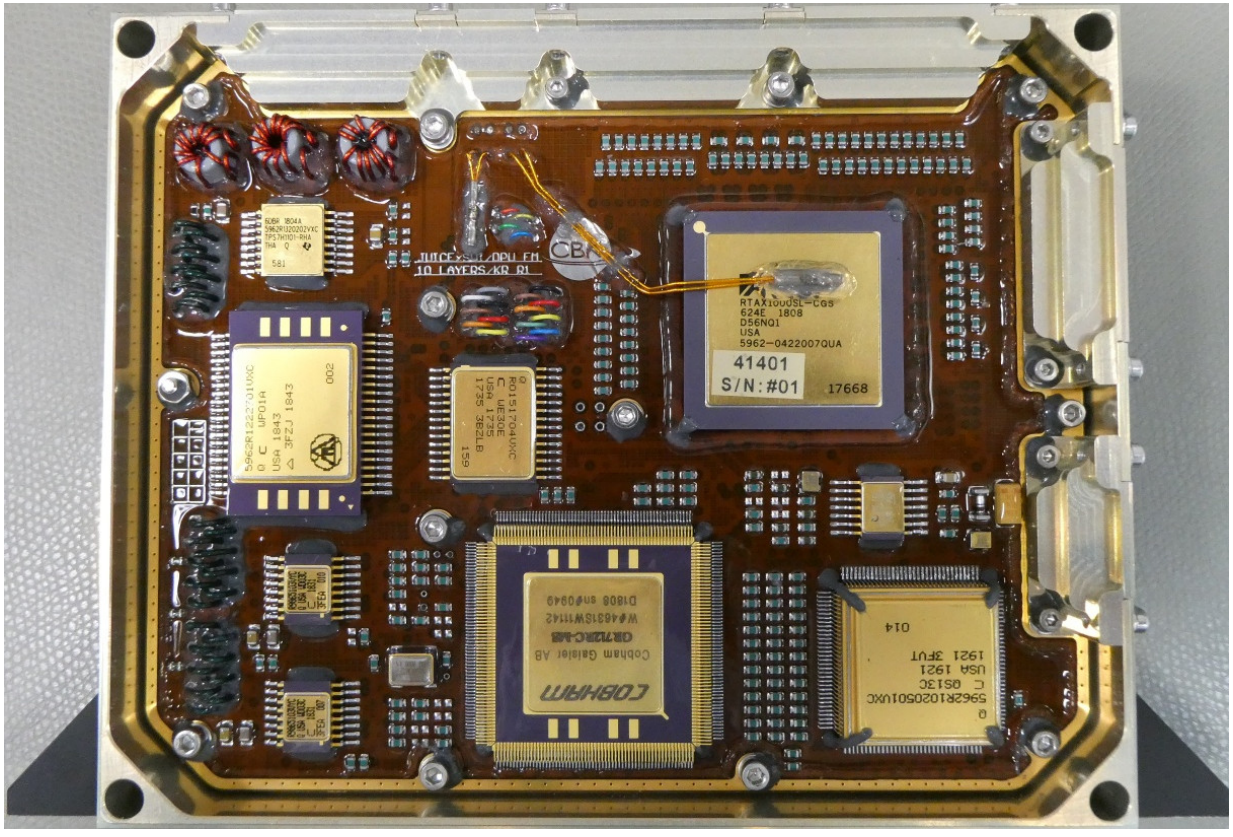
MISRA-C 2004: 142 rules for C
$\rightarrow$ nur 26 davon treffen zu auf Oberon (†Wirth)
$\rightarrow$ viele potentielle Fehler per se ausgeschlossen

Anpassung von Oberon: “hO”
$\rightarrow$ “cyclic executive”
$\rightarrow$ message passing (blockierend)
$\rightarrow$ state machines, module-local memory
$\rightarrow$ konstante maximale loop counts
Implementierung: keine Interrupts, Determinismus

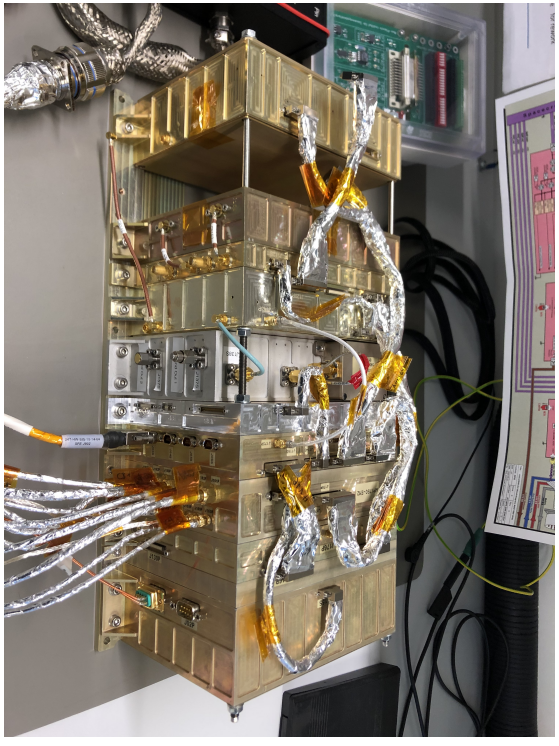
Compiler “hO to C”: Chicken / Scheme

Baugruppen: Elektronik

DPU

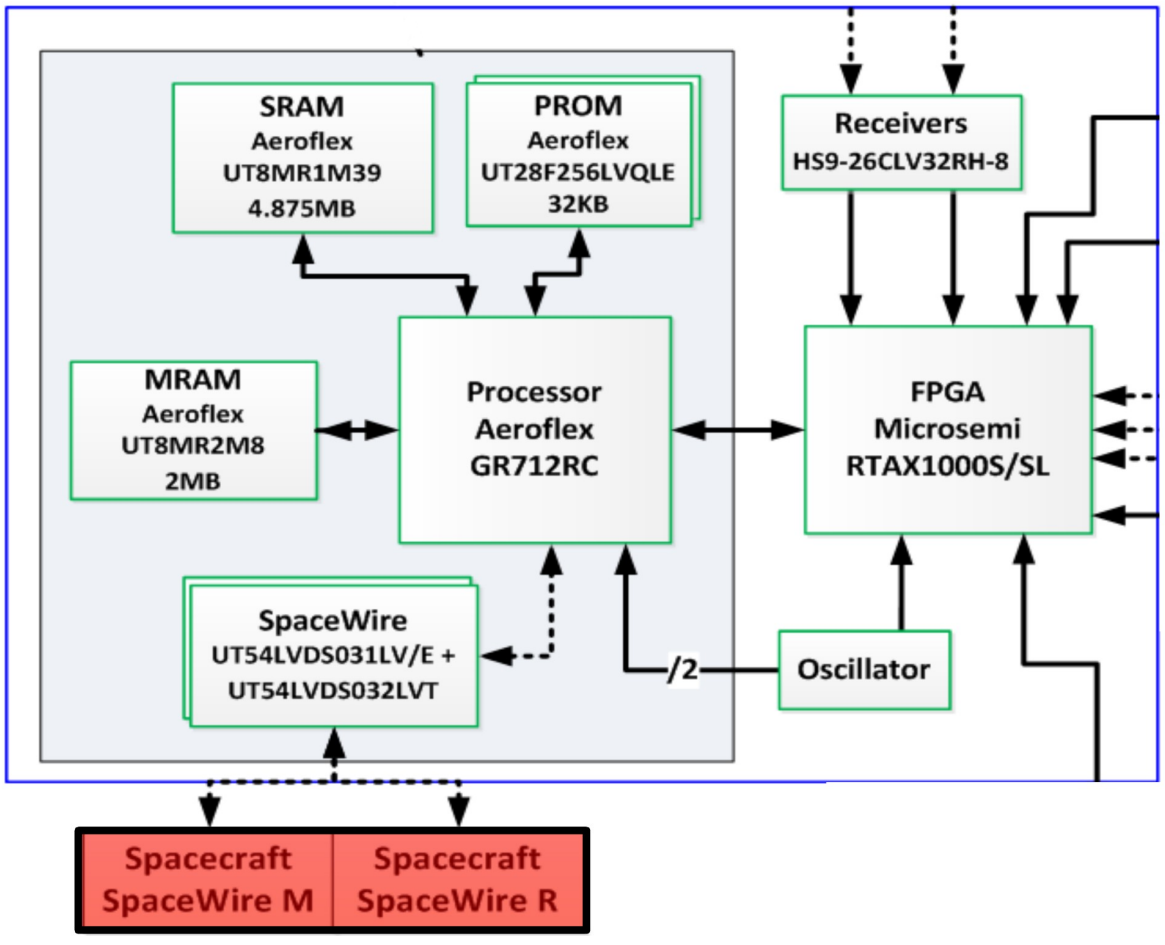


EU

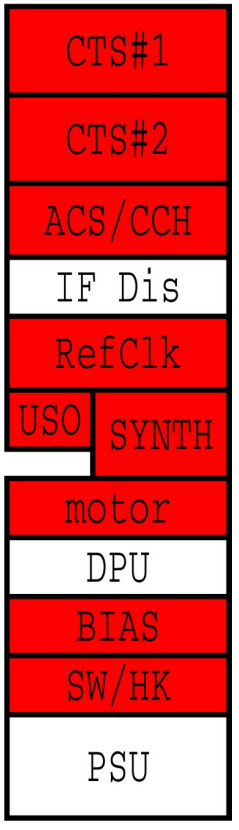


Baugruppen: schematisch

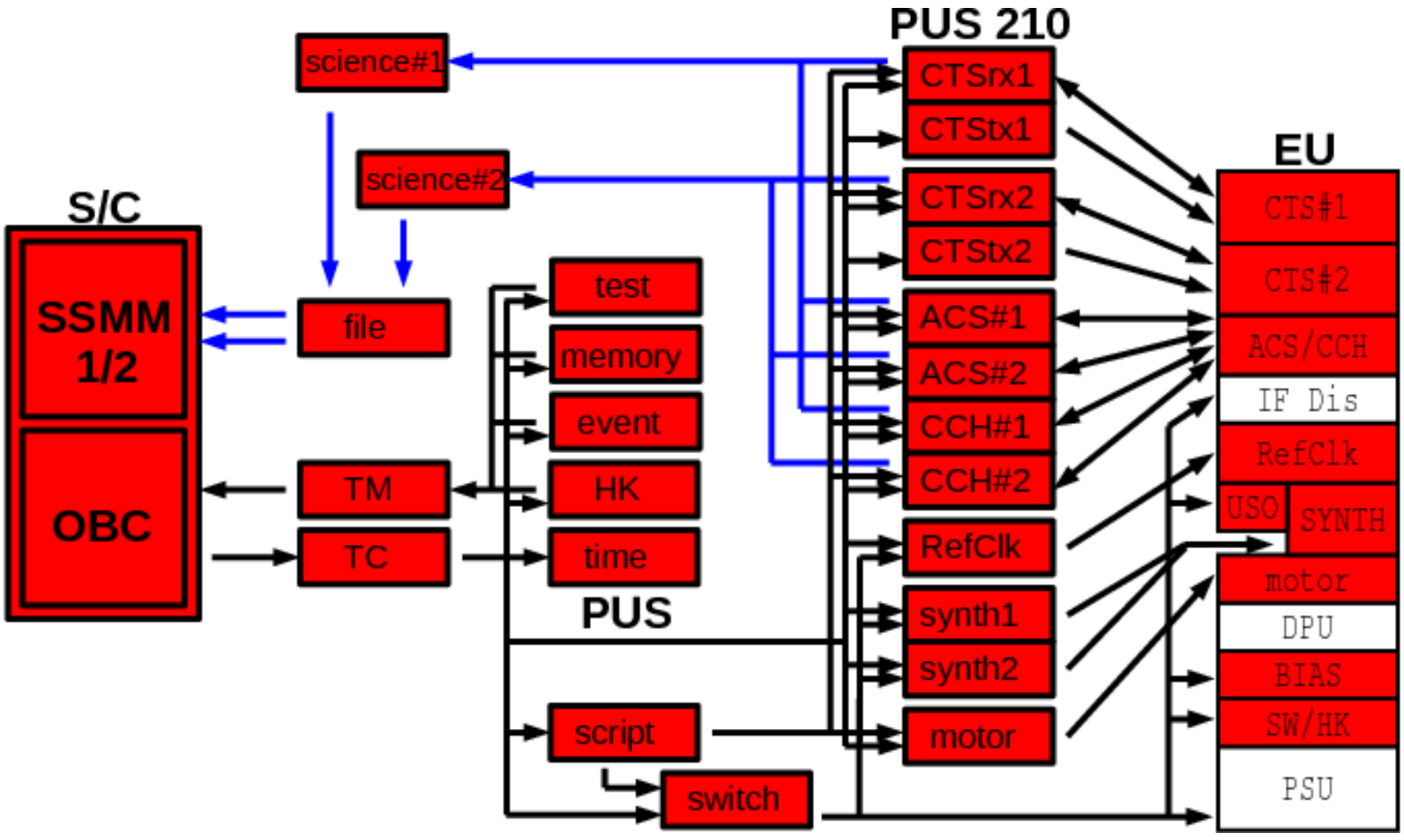
DPU



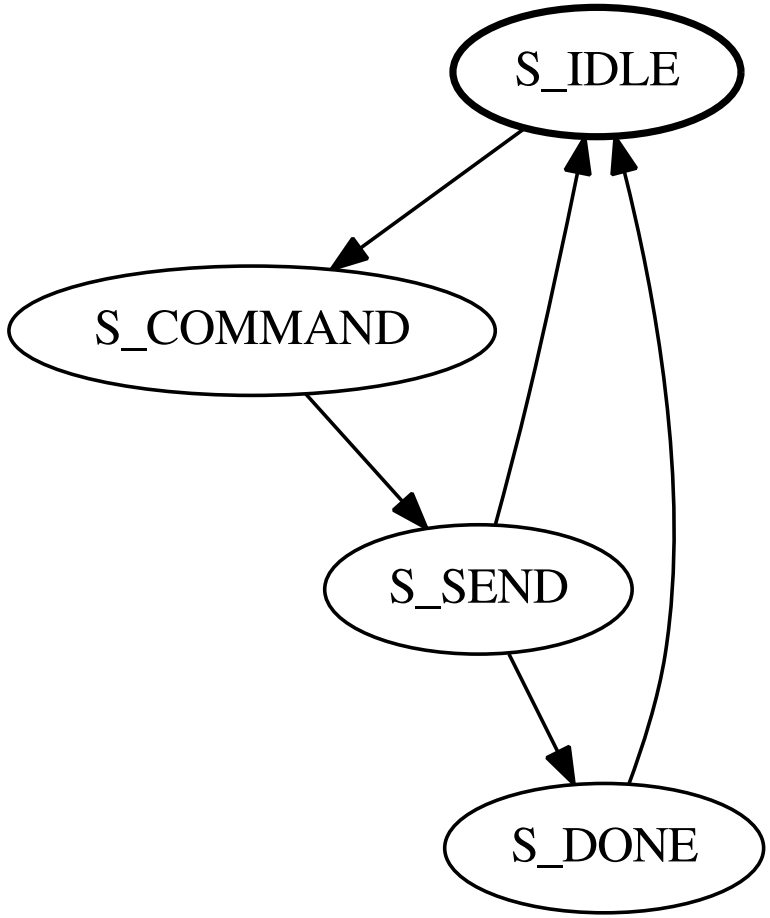
EU



Baugruppen: Software-Module



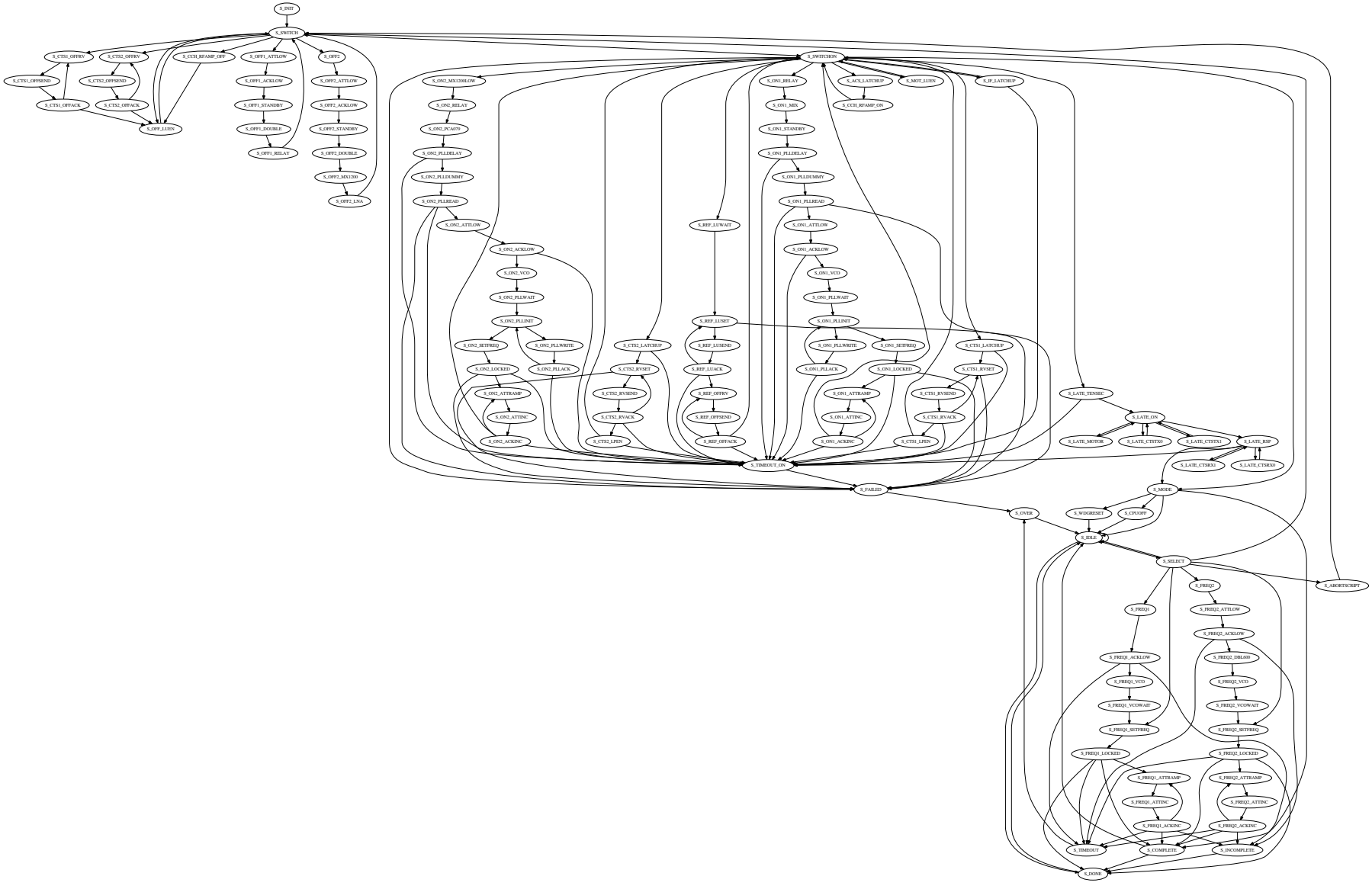
State Machine pro Modul



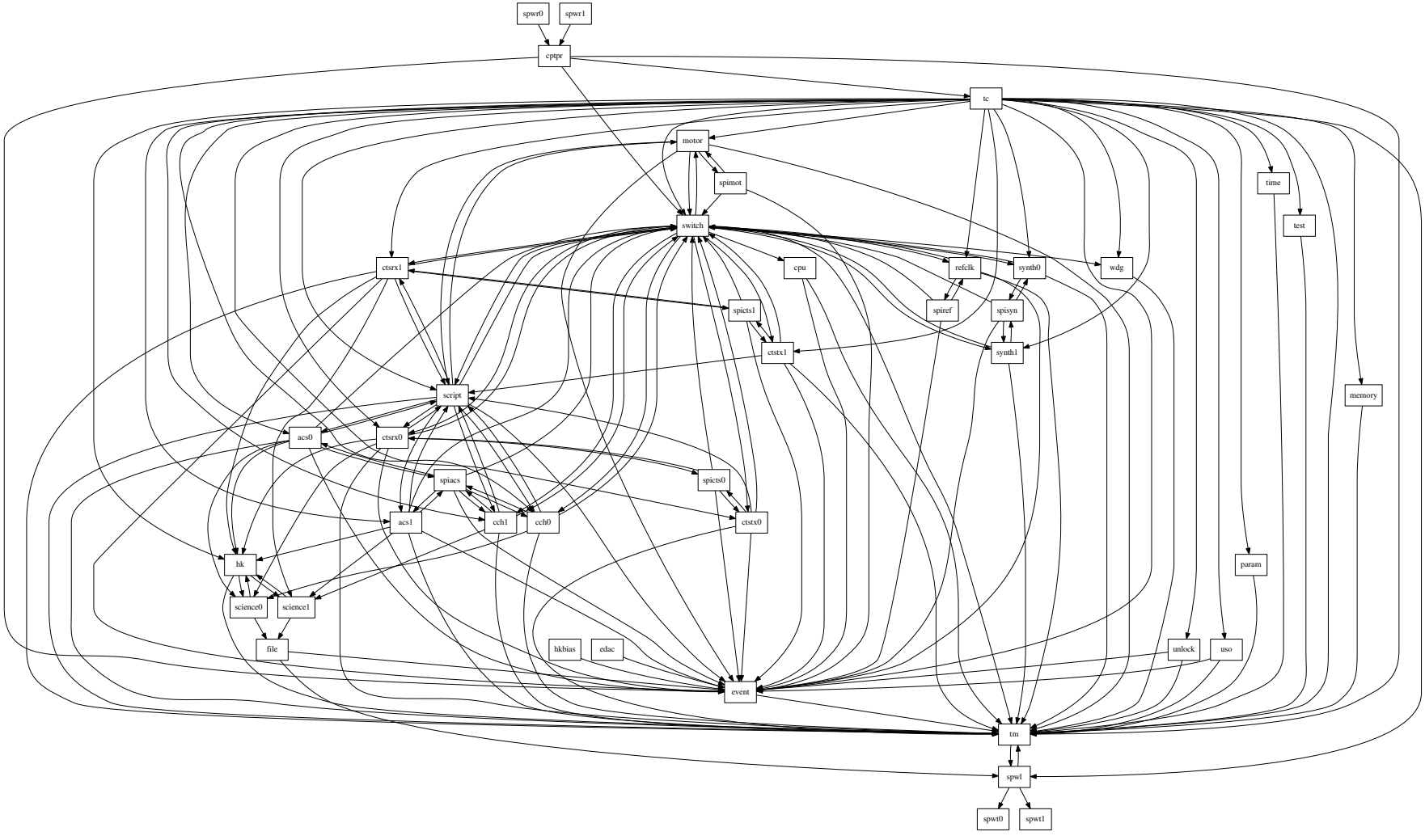
State Machine pro Modul

```
module test;

var
  rx*, tx: port;
  st: u32

begin
  state S_IDLE, S_COMMAND, S_SEND, S_DONE;
  import tm;
  select st of
    S_IDLE:
      if pending(rx) then
        case tc_dfh_get_subtype(rx) of
          TC_Ping:
            next S_COMMAND
          else
            dispose(rx)
        end
      end
  | S_COMMAND:
      new_fixed(tx, TM_Pong_size());
      tm_dfh_set_service(tx, SRV_TEST);
      tm_dfh_set_subtype(tx, TM_Pong);
      tm_dfh_set_destinationid(tx, tc_dfh_get_sourceid(rx));
      next S_SEND
  | S_SEND:
      if send(tx, tm.tx) then
        local f := ack_completed(tx, rx);
        dispose(rx);
        if f then
          next S_DONE
        else
          next S_IDLE
        end
      end
  | S_DONE:
      if send(tx, tm.tx) then
        next S_IDLE
      end
  end
end test.
```

State Machine pro Modul



Message Routes



Keep It Simple

kein RTEMS → keine RTEMS Treiber bugs
→ SpaceWire Treiber ca. 500 Zeilen; SPI ähnlich
→ Scheduler, Trap handler, Paket Management, CPU init.
Alle Module statisch gelinkt an feste Adressen → SW Update

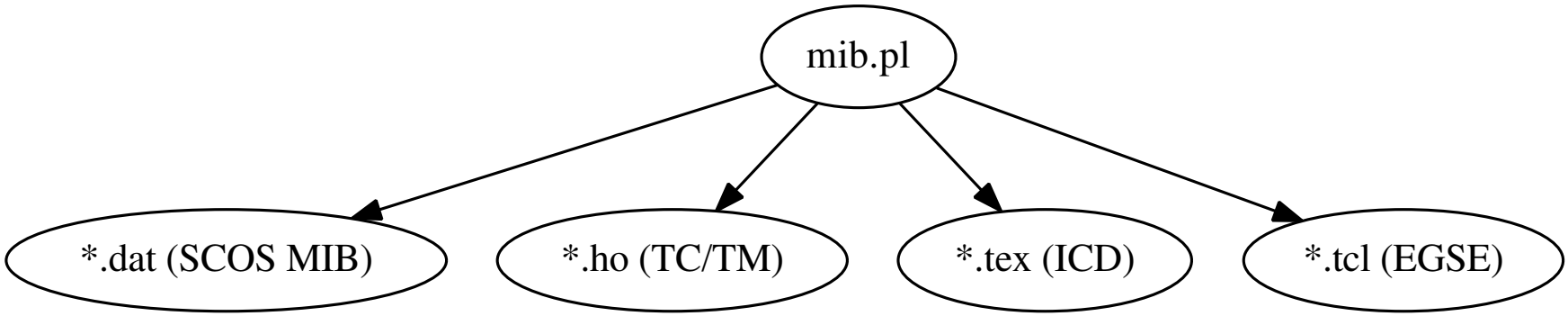
Alles Text-basiert, Command-Line driven, dokumentiert in GIT:
vollständig reproduzierbar inkl. 3rd party tools, Doku, Tests...

Automatisch generieren aus source code:
→ Diagramme (message routes, module states)
→ Tabellen ("requirement matrix")

Keep It Simple

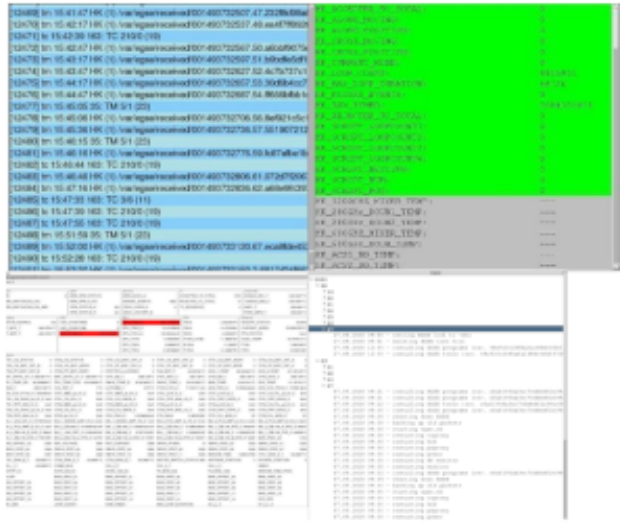
"Mission Information Base" (MIB)

$\rightarrow$ formale Spezifikation der TC/TM Schnittstelle

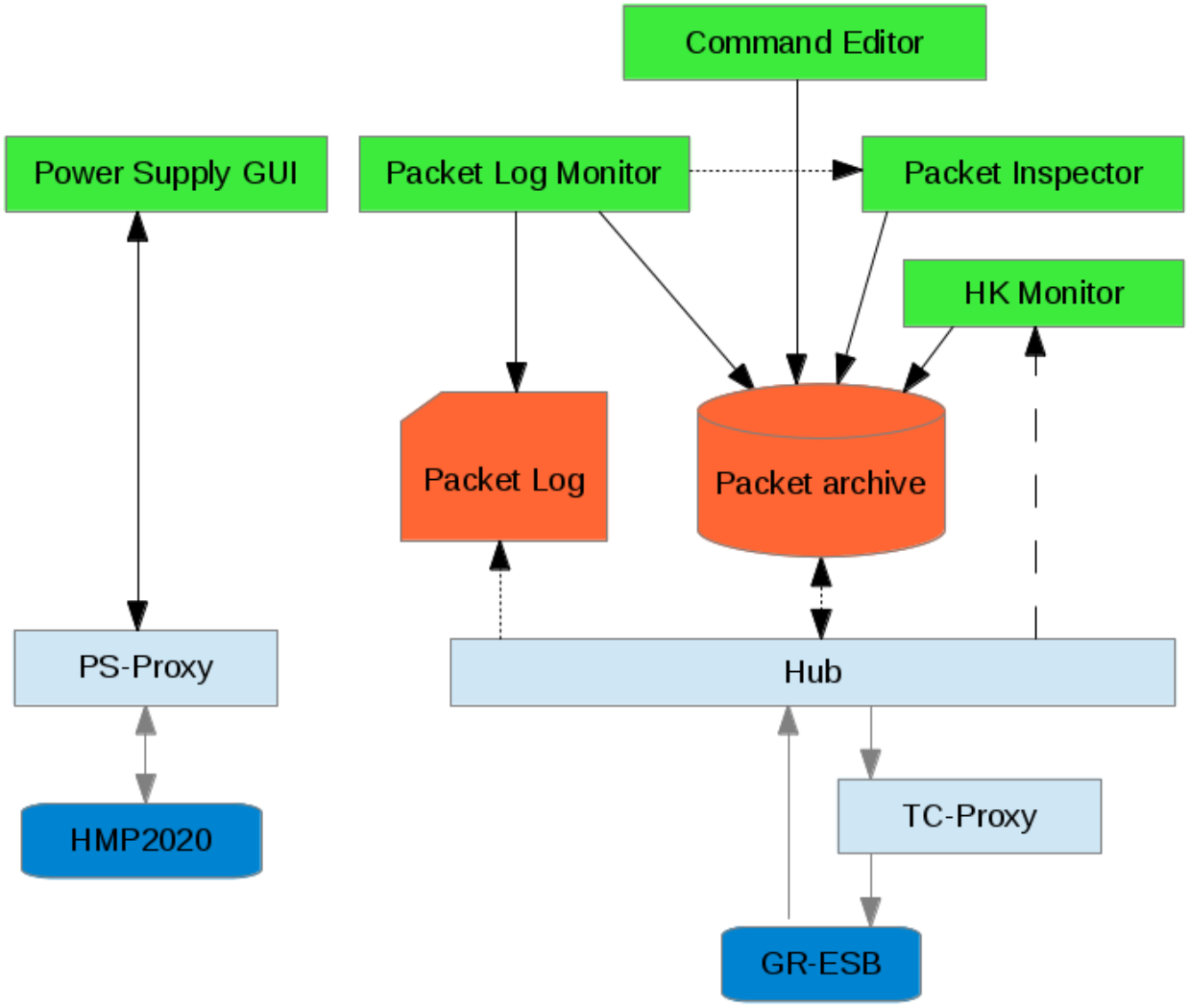



```
tc(3, 3/136, 10, 'ReqHKReport', asw,
  'Request HK Parameter Report',
  ['TC_SID'],
  ['FID_HKD_INVALID_SID']).
...
type(43, lna_temp, 3/12, u16, 'Temperature',  'DegC',
        [-256.35780, 0.0484452], range(-160, -155, 25, 30)).
```

EGSE OBC/ground Simulator



EGSE OBC/ground Simulator



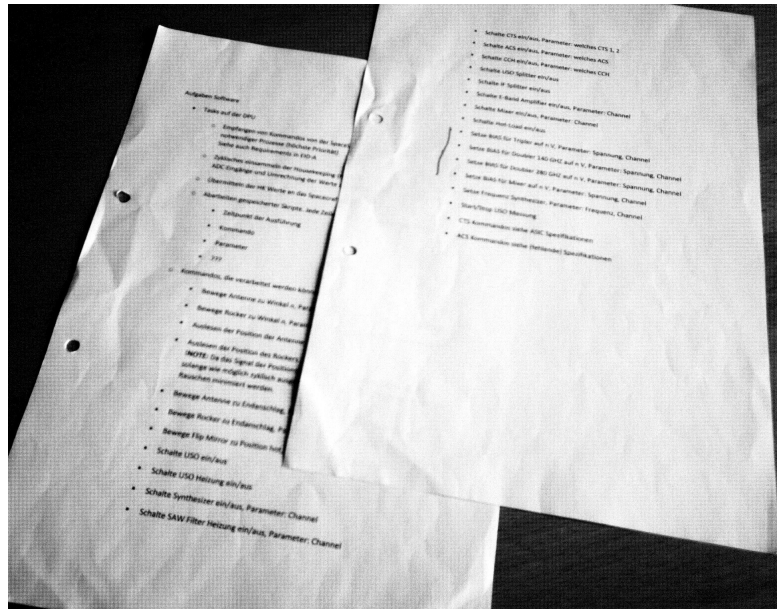
EGSE / Testsuite

$\rightarrow$ bash Skripte
$\rightarrow$ pro Service, pro Use Case
$\rightarrow$ Failure Detection, Regression Tests, Validation

Dokumentation

→ Software Development Plan (SDP)
→ Software Requirements Specification (SRS) (*)
→ Software Design Document (SDD) (*)
→ Interface Control Document (ICD) (*)
→ Software Product Assurance Plan (SPAP)
→ Software Scripting Manual
→ hO Language Reference Manual
→ Software User Manual
→ Validation, Verification, Test Plan, etc.
→ verschiedene Technical Notes, Reports, etc.
→ Instrument Software Configuration File (ISC) (*)

Evolvierende Requirements

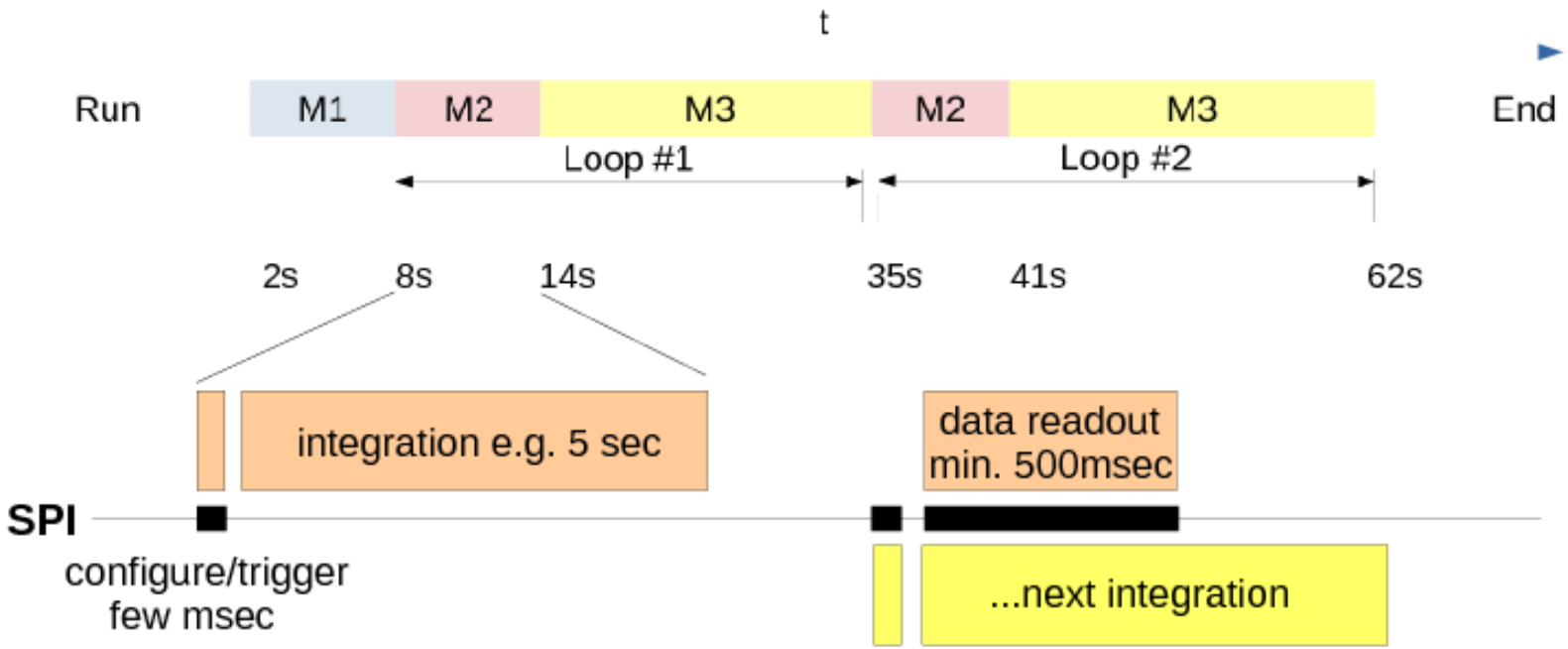


→ ursprünglich 2 Seiten Stichpunkte
→ 500 Requirements allein für die Software
→ projektweit aus PDFs nach Trac SQL
→ nach TeX, und generierte Tabellen

```
new_fixed(tx, TM_Pong_size());  (* EIDA-R010602 *)
tm_dfh_set_service(tx, SRV_TEST);  (* EIDA-R010011 *)
tm_dfh_set_subtype(tx, TM_Pong);  (* PUS-8806 *)
```

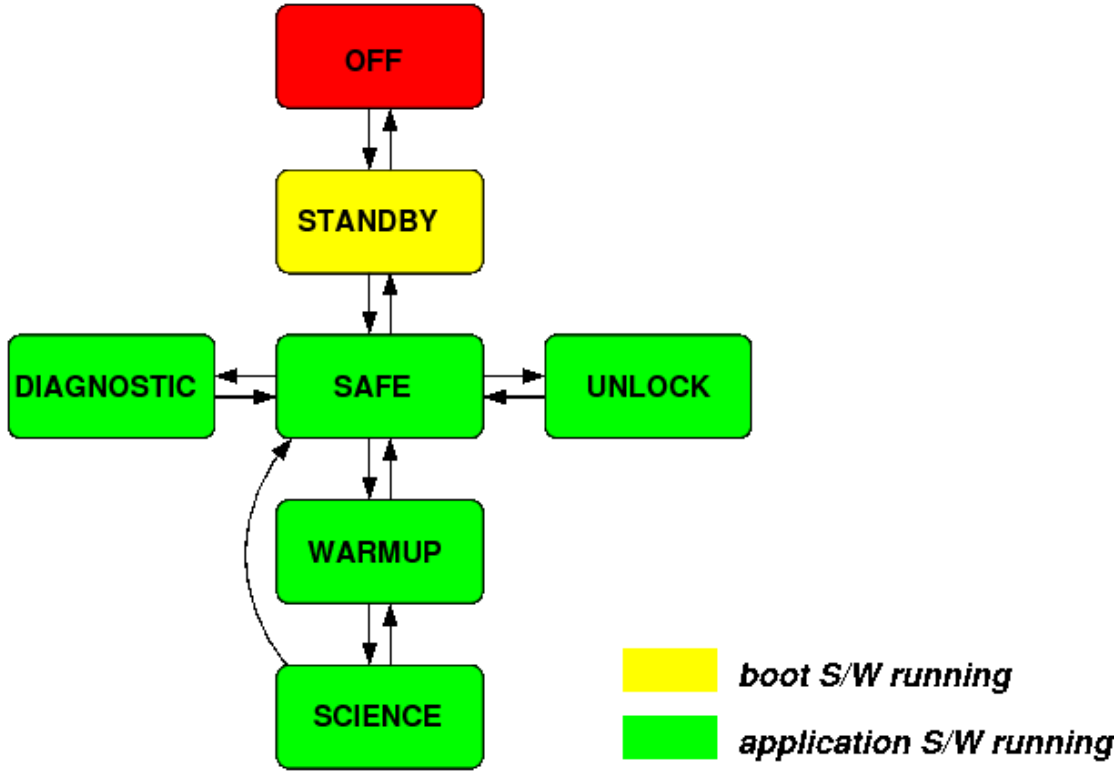
Science Scripts

```
+1s:    measurects {1, 1} 5i {0, 10000} {0, 10000}
-15s:   loop 2
+21s:   measurects {1, 1} 5i {0, 10000} {0, 10000}
+6s:    measurects {1, 1} 20i {0, 10000} {0, 10000}
        endloop
        end
```



100 TC/Tag (je 226 Byte max) → tausende Messungen
Byte Code, TC Parameter → deterministisch, nicht Turing-komplett

“Software Modes”



ESA: Stromverbrauch (Solar-Budget, Überwachung)
ADS: Auswahl TC/TM, Interaktion OBC
ESOC: Umfang Messergebnisse (SSMM, Datenrate Downlink)
Science Team: Observation Modes

Solid State Mass Memory

"SSMM": 125 GB Festspeicher (total)
Speichern: 2 Datenströme ("files") pro Instrument
Wechsel: 10 Sekunden Delay
Raw Data: Verlustkontrolle durch instrumentenspezifisches Format

Oskar Schirmer [oskar@scara.com]
Felix Winkelmann [felix.winkelmann@bevuta.com]

Alle Abbildungen: Credit ESA/ADS, außer Seite 6/7/8: M. Yedla, Seite 9/10: CBK, Seite 29: C. Jarchow