UEFI Compliance Testing

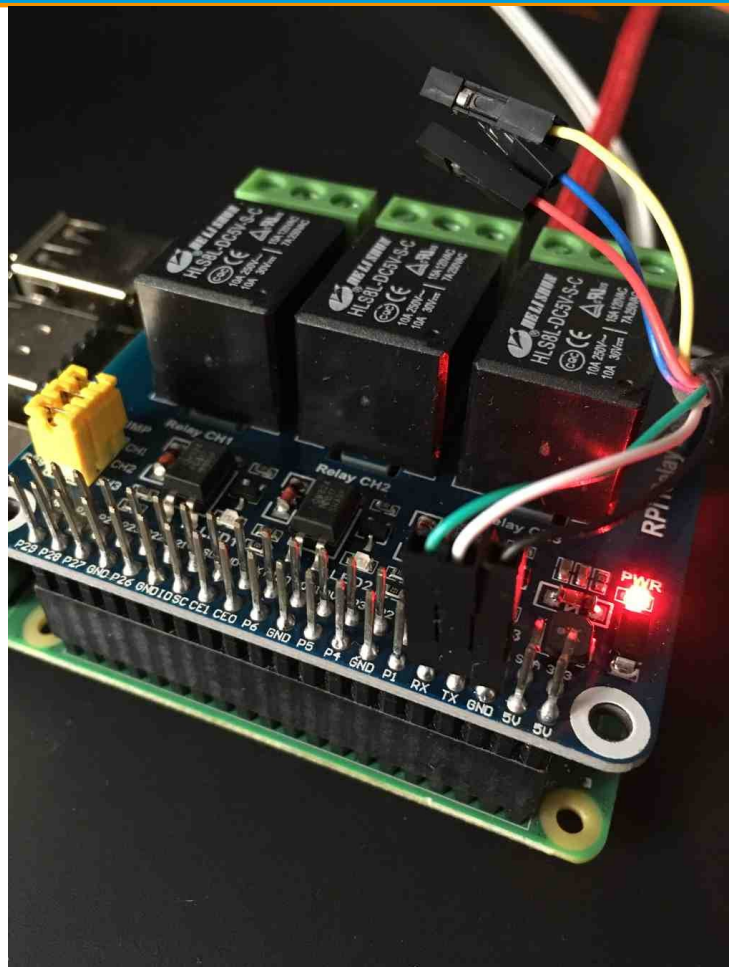
Heinrich Schuchardt

Agenda

- Motivation
- Open Source UEFI implementations
- UEFI test tools: SCT, FWTS
- Gaps
- Device tree validation
- Test environment in U-Boot
- Summary

About Me

- U-Boot UEFI Maintainer
- Contributions to
  - Linux
  - GNU Linear Programming Kit (GLPK)
  - ...



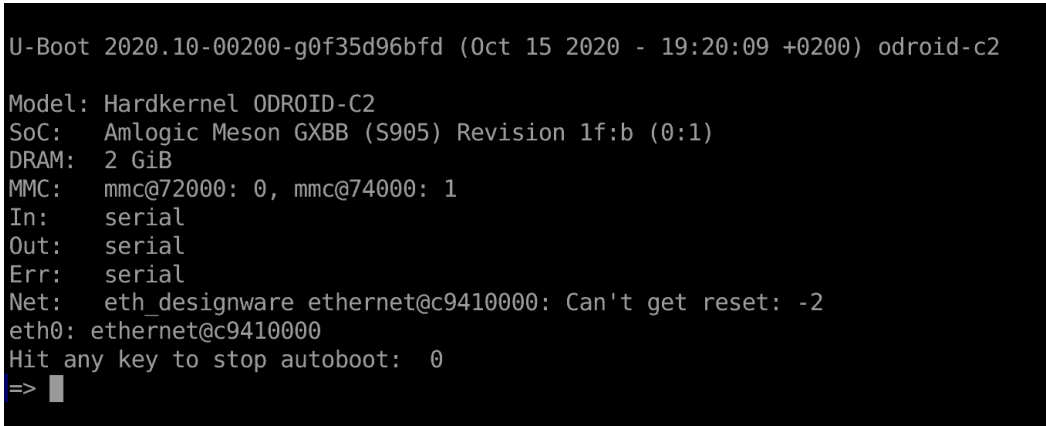
Open Source UEFI Implementations

Tianocore EDK II

- Reference implementation
- Full UEFI scope
- SBBR compliance
- 36 configurations

U-Boot

- UEFI for embedded systems
- Reduced UEFI scope
- EBBR compliance
- 786 of 1312 configurations



Embedded Base Boot Requirements (EBBR) Specification

- Defines subset of UEFI specification
  - Targets embedded systems
  - Enough to boot operating systems
- EBBR v1.0 released in 2019
  https://github.com/arm-software/ebbr
- Applicable to all UEFI architectures: ARM, RISC-V, x86

UEFI in U-Boot

- Scope
  - Embedded Base Boot Requirements (EBBR) Specification
- Boot and Runtime Services, Secure Boot
- Protocols
  - Simple Text Input (Ex), Simple Text Output, Graphics Output
  - Block IO, Simple File System, File
  - Simple Network
  - Device Path To Text, Unicode Collation, RNG
  - Only partially: HII (enough to run SCT)

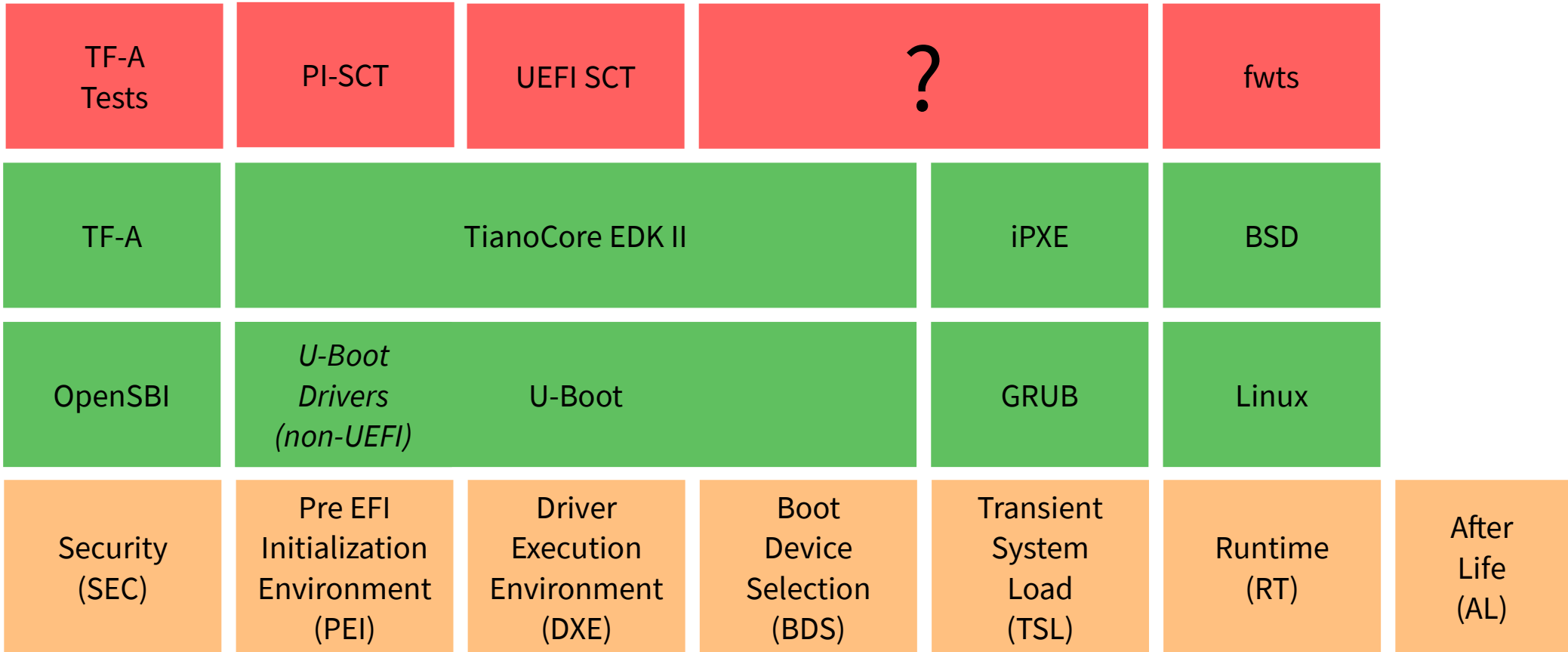
U-Boot UEFI Usage

- Used by default for booting on embedded boards by
  - Fedora
  - Suse
  - FreeBSD
  - OpenBSD

Motivation for Complicance Testing

- Implementation perspective
  - UEFI 2.8 Errata B specification has 2484 pages
  - Requirements are highly complex and interrelated
- Application perspective
  - Provide a reliable basis for UEFI boot flow

UEFI Lifetime



UEFI Self-Certification Test

- The UEFI SCT is an EFI shell application.
- Coverage
  - boot and runtime services
  - protocols
- Test types
  - Conformance tests for handling of invalid parameters
  - Functionals tests
- 606 EFI test cases with 7142 asssertion
- 1.2 million lines of code

SCT Test Modes

- Native mode
  - Testing API
  - Runs on single system
- Passive mode
  - Testing network protocols (DHCP, IPv4, IPv6, TCP, UDP, HTTP, …)
  - Test controlled from Windows system
  - Requires Managed Network Protocol (MNP)

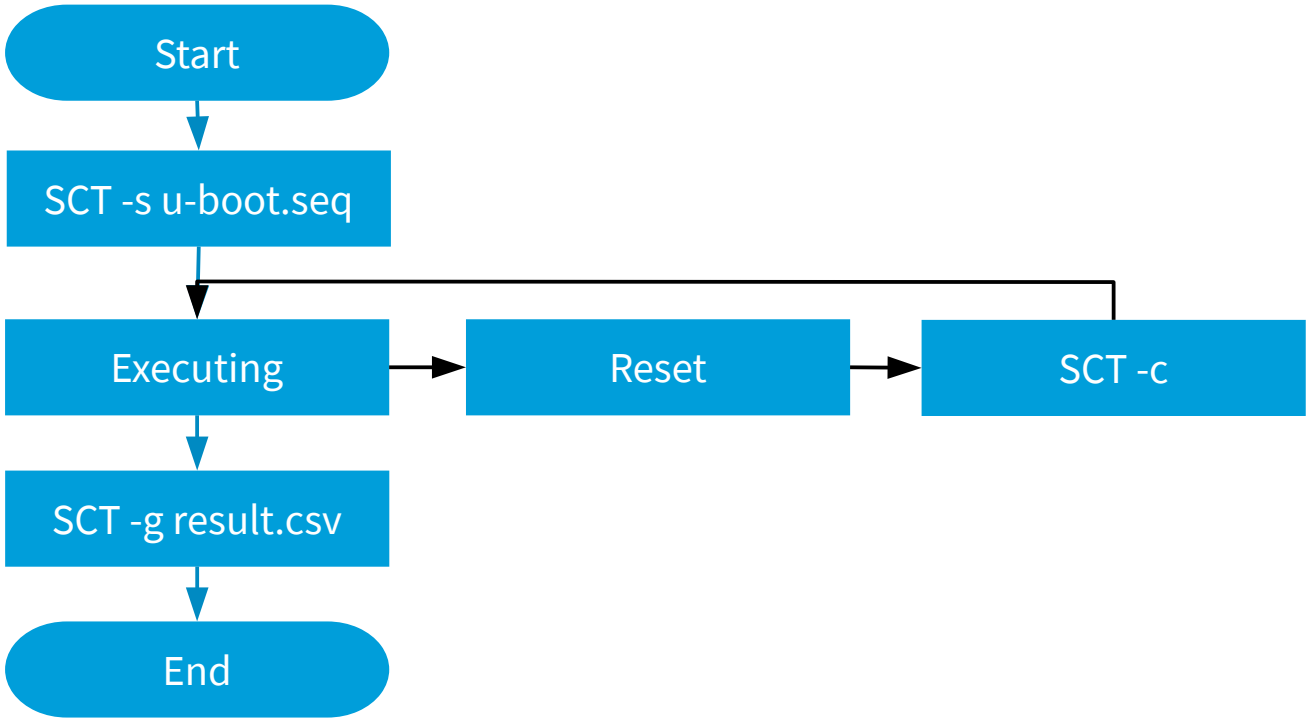
Running the SCT

- The U-Boot Sandbox can run UEFI binaries natively
- Requires no KVM/QEMU emulation
- Fastest method of running the SCT on build system

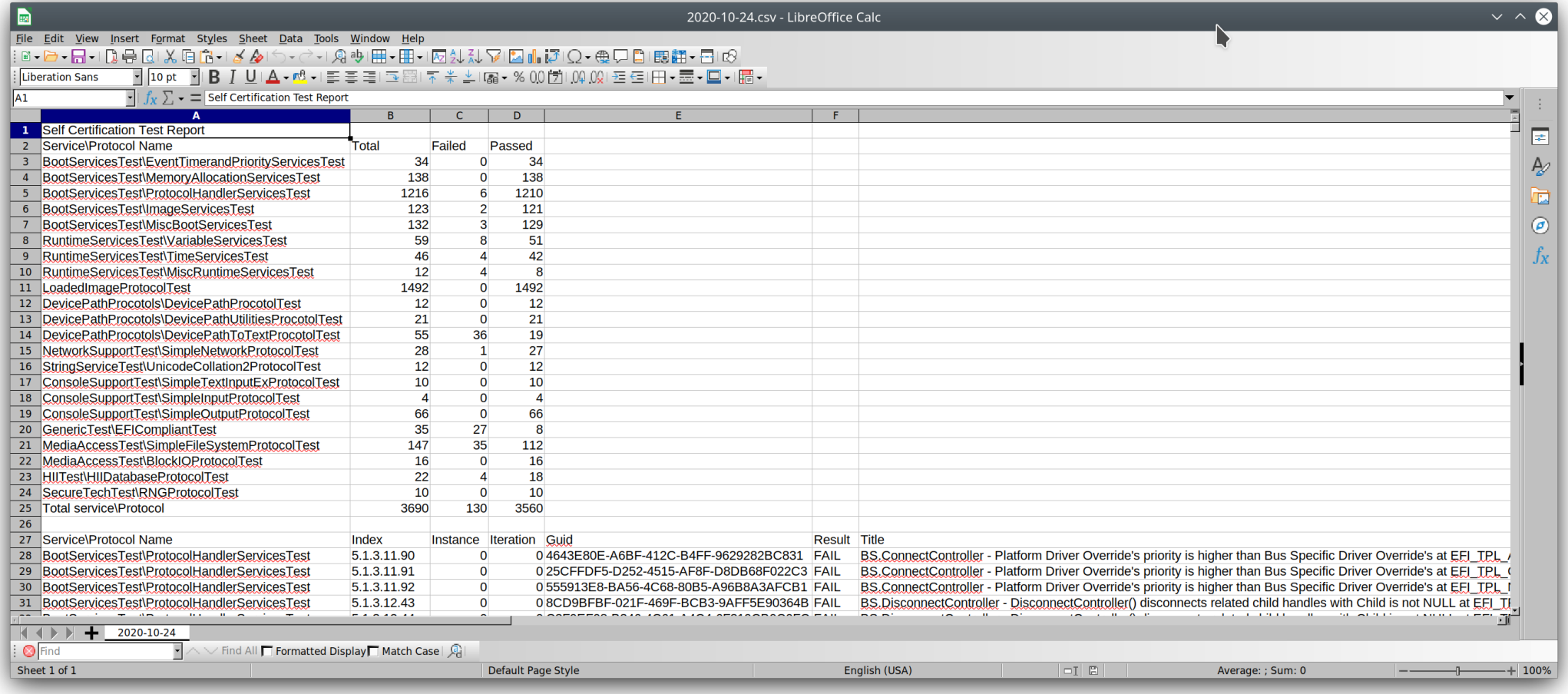
UEFI Self-Certification Test

Shell Script - start.nsh

```
FS0:
if exist run then
  rm run
  SCT -s uboot.seq
else
  SCT -c
endif
SCT -g result.csv
reset -s
```

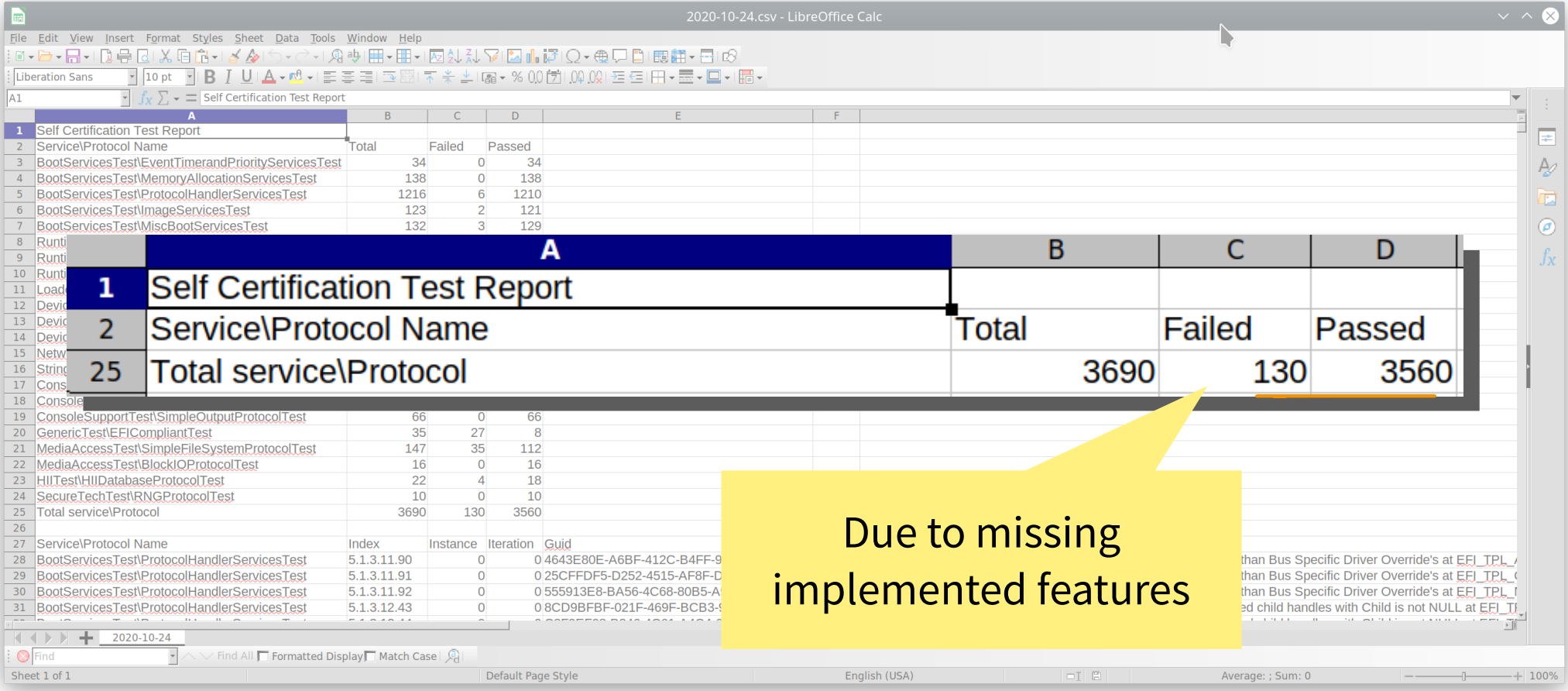


Report



	A	B	C	D	E	F	
1	Self Certification Test Report						
2	Service\Protocol Name	Total	Failed	Passed			
3	BootServicesTest\EventTimerandPriorityServicesTest	34	0	34			
4	BootServicesTest\MemoryAllocationServicesTest	138	0	138			
5	BootServicesTest\ProtocolHandlerServicesTest	1216	6	1210			
6	BootServicesTest\ImageServicesTest	123	2	121			
7	BootServicesTest\MiscBootServicesTest	132	3	129			
8	RuntimeServicesTest\VariableServicesTest	59	8	51			
9	RuntimeServicesTest\TimeServicesTest	46	4	42			
10	RuntimeServicesTest\MiscRuntimeServicesTest	12	4	8			
11	LoadedImageProtocolTest	1492	0	1492			
12	DevicePathProcotols\DevicePathProcotolTest	12	0	12			
13	DevicePathProcotols\DevicePathUtilitiesProcotolTest	21	0	21			
14	DevicePathProcotols\DevicePathToTextProcotolTest	55	36	19			
15	NetworkSupportTest\SimpleNetworkProtocolTest	28	1	27			
16	StringServiceTest\UnicodeCollation2ProtocolTest	12	0	12			
17	ConsoleSupportTest\SimpleTextInputExProtocolTest	10	0	10			
18	ConsoleSupportTest\SimpleInputProtocolTest	4	0	4			
19	ConsoleSupportTest\SimpleOutputProtocolTest	66	0	66			
20	GenericTest\EFICompliantTest	35	27	8			
21	MediaAccessTest\SimpleFileSystemProtocolTest	147	35	112			
22	MediaAccessTest\BlockIOProtocolTest	16	0	16			
23	HIITest\HIIDatabaseProtocolTest	22	4	18			
24	SecureTechTest\RNGProtocolTest	10	0	10			
25	Total service\Protocol	3690	130	3560			
26							
27	Service\Protocol Name	Index	Instance	Iteration	Guid	Result	Title
28	BootServicesTest\ProtocolHandlerServicesTest	5.1.3.11.90	0	0	4643E80E-A6BF-412C-B4FF-9629282BC831	FAIL	BS.ConnectController - Platform Driver Override's priority is higher than Bus Specific Driver Override's at EFI_TPL_
29	BootServicesTest\ProtocolHandlerServicesTest	5.1.3.11.91	0	0	25CFFDF5-D252-4515-AF8F-D8DB68F022C3	FAIL	BS.ConnectController - Platform Driver Override's priority is higher than Bus Specific Driver Override's at EFI_TPL_
30	BootServicesTest\ProtocolHandlerServicesTest	5.1.3.11.92	0	0	555913E8-BA56-4C68-80B5-A96B8A3AFCB1	FAIL	BS.ConnectController - Platform Driver Override's priority is higher than Bus Specific Driver Override's at EFI_TPL_
31	BootServicesTest\ProtocolHandlerServicesTest	5.1.3.12.43	0	0	8CD9BFBF-021F-469F-BCB3-9AFF5E90364B	FAIL	BS.DisconnectController - DisconnectController() disconnects related child handles with Child is not NULL at EFI_TI

SCT Results for U-Boot



	A	B	C	D
1	Self Certification Test Report			
2	Service\Protocol Name	Total	Failed	Passed
25	Total service\Protocol	3690	130	3560

Pros & Cons

- Conformance well covered
- Functionality of individual API services well tested

- Long runtime (hours)
- Gaps in coverage
  - Test for image authentication missing (Bug #2230)
- RISC-V support missing
- Missing clean separation between SCT and EDK II

Example
UnicodeCollationProtocol2 StrUpr()

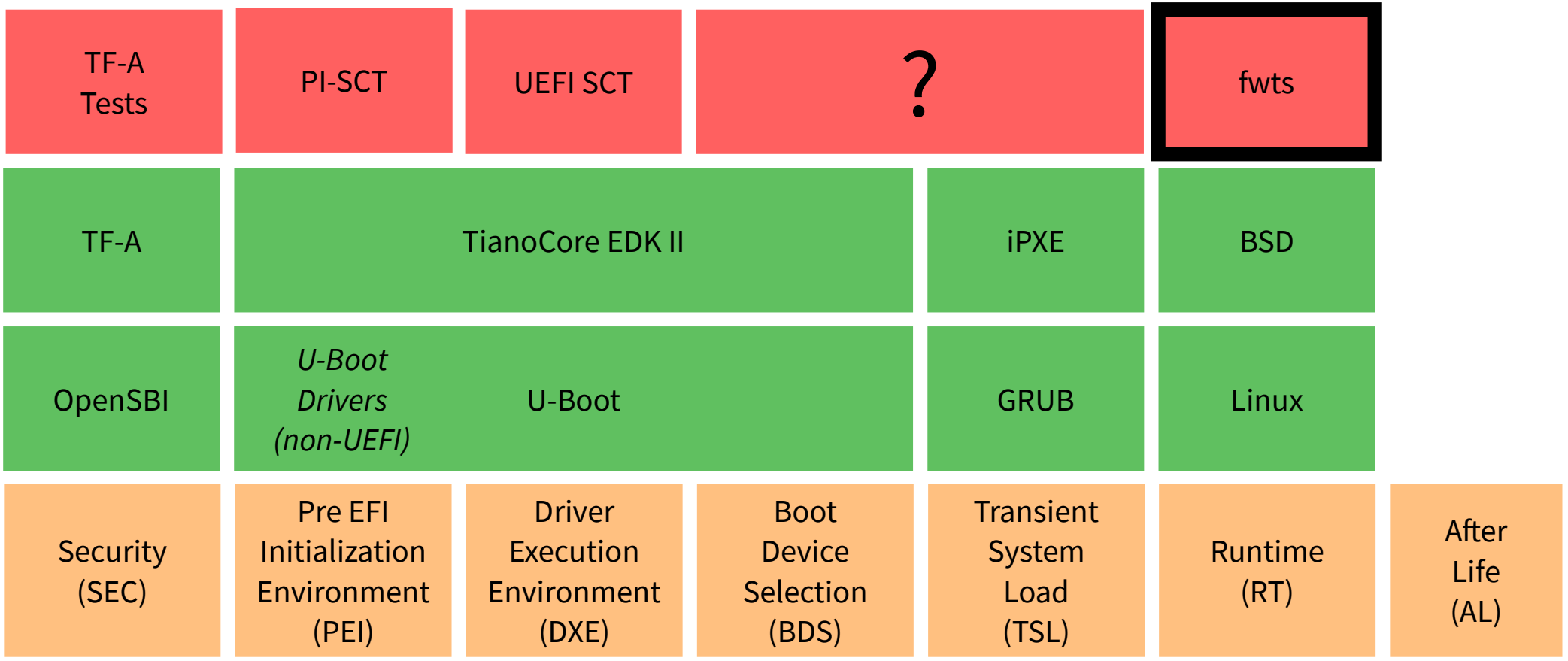
- U-Boot's conversion
  "0P[A-D]@ 0ʊ@ " –> "0P[A-D]@ 0Ʊ@ "

- SCT claimed this was an error
  - EDK does not support uppercasing Unicode
  - SCT used the EDK library (sic!).

UEFI SCT Status

- Development has nearly stopped
- Stuck on path to UEFI 2.7 (current spec is 2.8 B)
- Building for X86_64 fails with GCC and VS2019

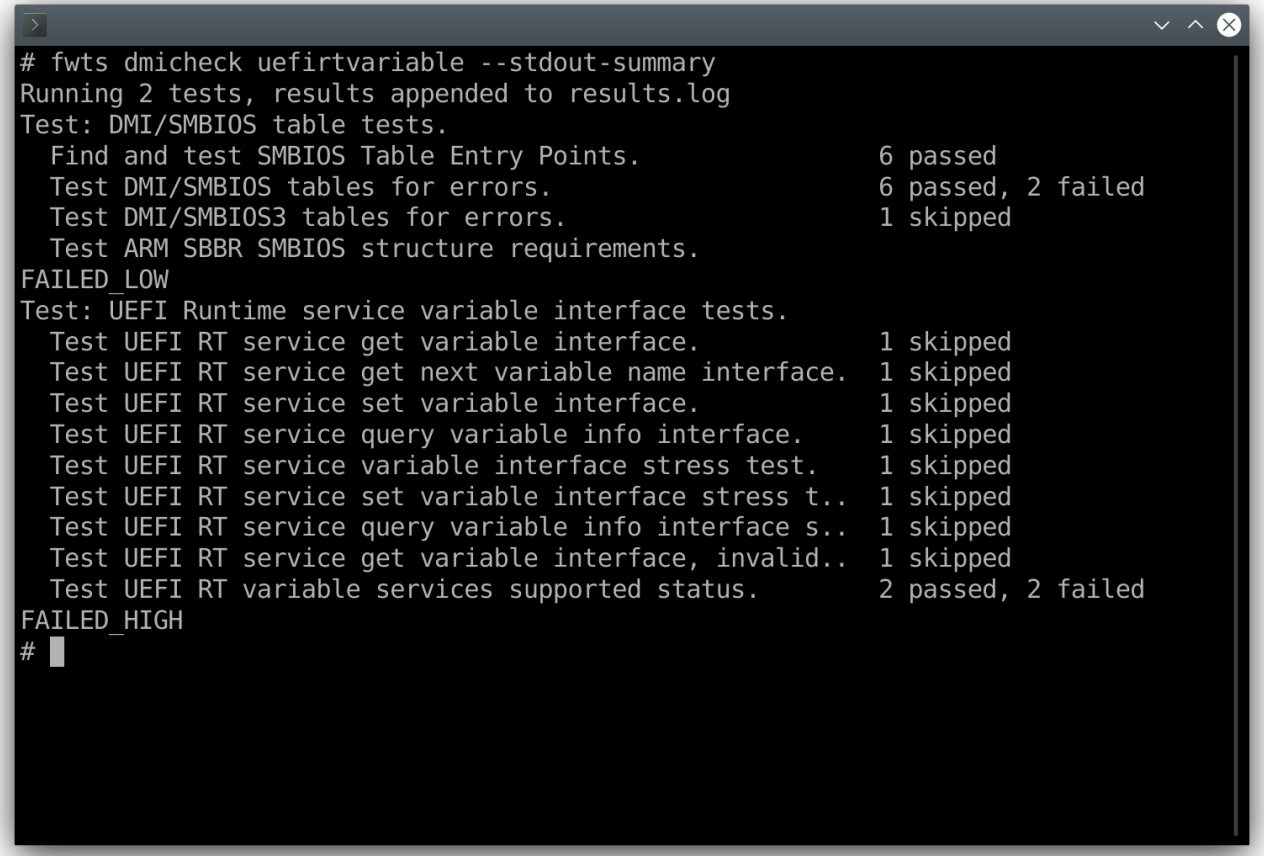
UEFI Lifetime



Firmware Test Suite

- Check firmware from Linux
- Scope
  - ACPI tables
  - SMBIOS table
  - UEFI runtime services

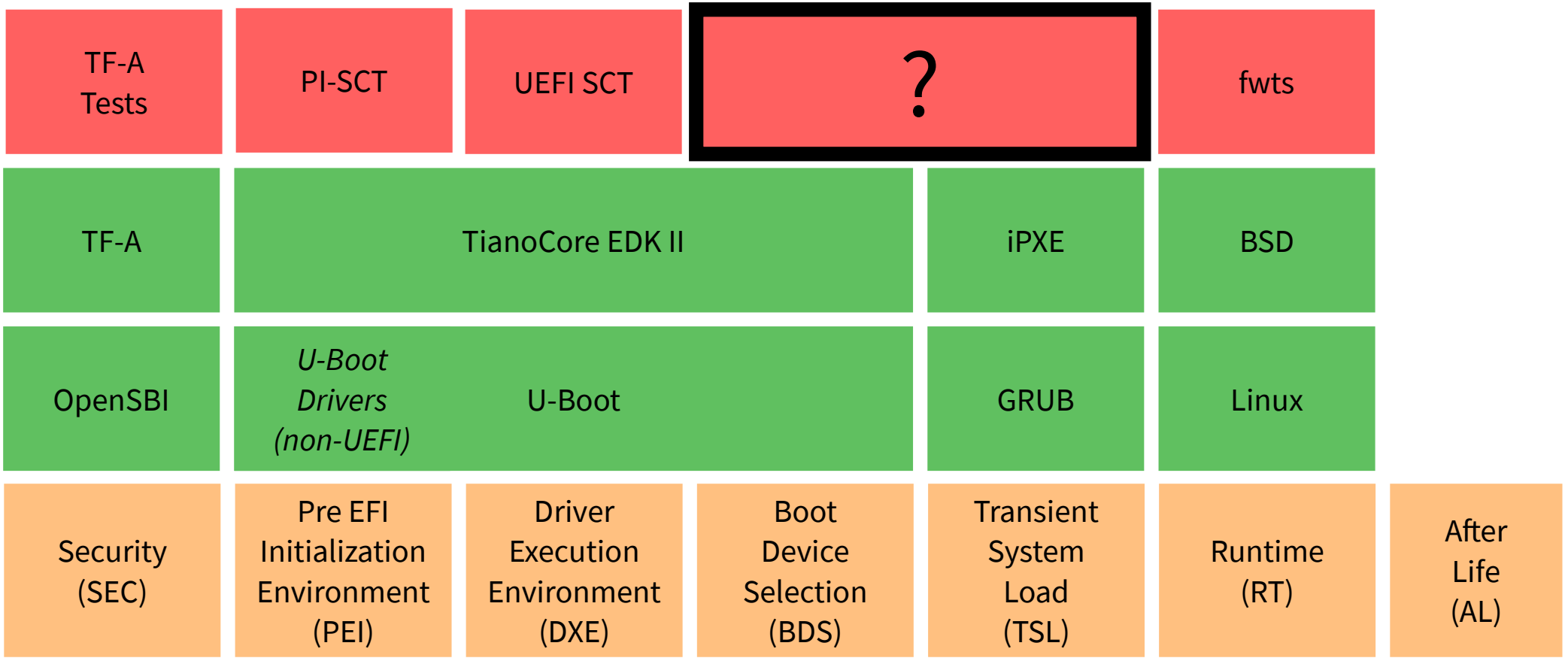
Firmware Test Suite



FWTS: results.log

```
High failures: 2
 uefirtvariable: Get the Setvariable runtime service supported via RuntimeServicesSupported variable. But
actually is not supported by firmware.
 uefirtvariable: Get the QueryVarInfo runtime service supported via RuntimeServicesSupported variable. But
actually is not supported by firmware.

Medium failures: NONE

Low failures: 2
 dmicheck: String index 0x01 in table entry 'Processor Information (Type 4)' @ 0x7aee5114, field 'Processor
Manufacturer', offset 0x07 has a default value 'Unknown' and probably has not been updated by the BIOS
vendor.
 dmicheck: String index 0x01 in table entry 'Processor Information (Type 4)' @ 0x7aee5114, field 'Processor
Version', offset 0x10 has a default value 'Unknown' and probably has not been updated by the BIOS vendor.

Other failures: NONE

Test           |Pass |Fail |Abort|Warn |Skip |Info |
---------------+-----+-----+-----+-----+-----+-----+
dmicheck       |   12|    2|     |     |    1|     |
uefirtvariable |    2|    2|     |     |    8|     |
---------------+-----+-----+-----+-----+-----+-----+
Total:         |   14|    4|    0|    0|    9|    0|
---------------+-----+-----+-----+-----+-----+-----+
```

UEFI Lifetime



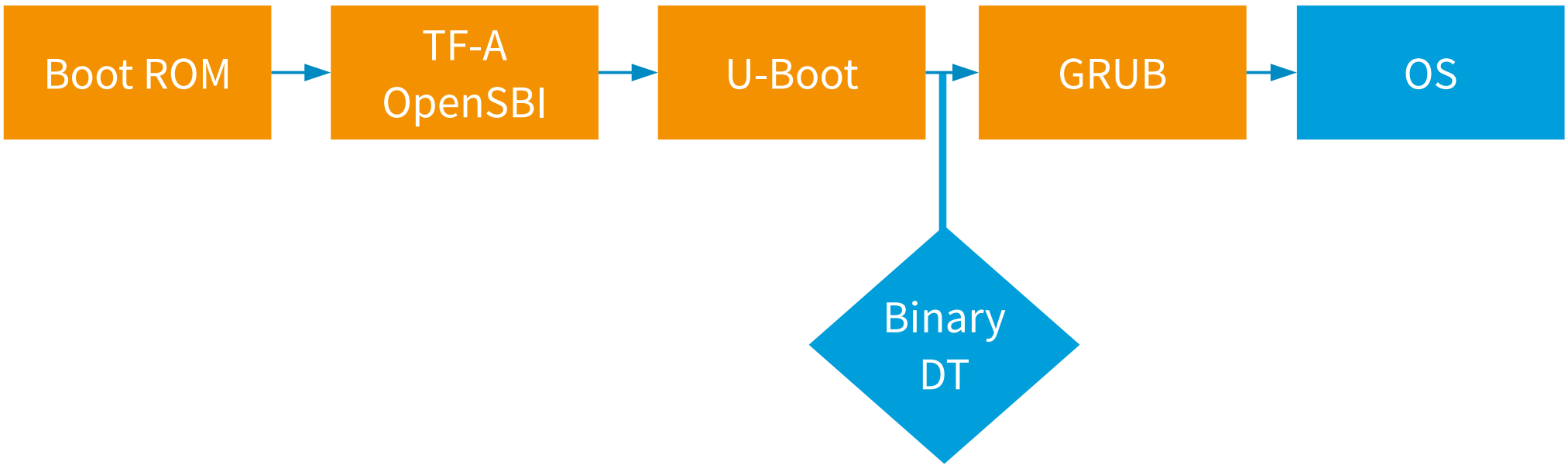
Not Covered by SCT and FWTS

- Boot device selection
  - Boot manager, boot variables
- Transient system load
  - ExitBootServices()
    - Release of non-runtime memory
    - MMU, cache, and interrupts state
  - SetVirtualAddressMap(), ConvertPointer()
  - Device tree

Sources of Device Tree

- Device trees can be be provided on different stages.
- Device trees are fixed up on all stages.
- Device trees can be generated on the fly (e.g. QEMU)

Where to Validate Device Tree

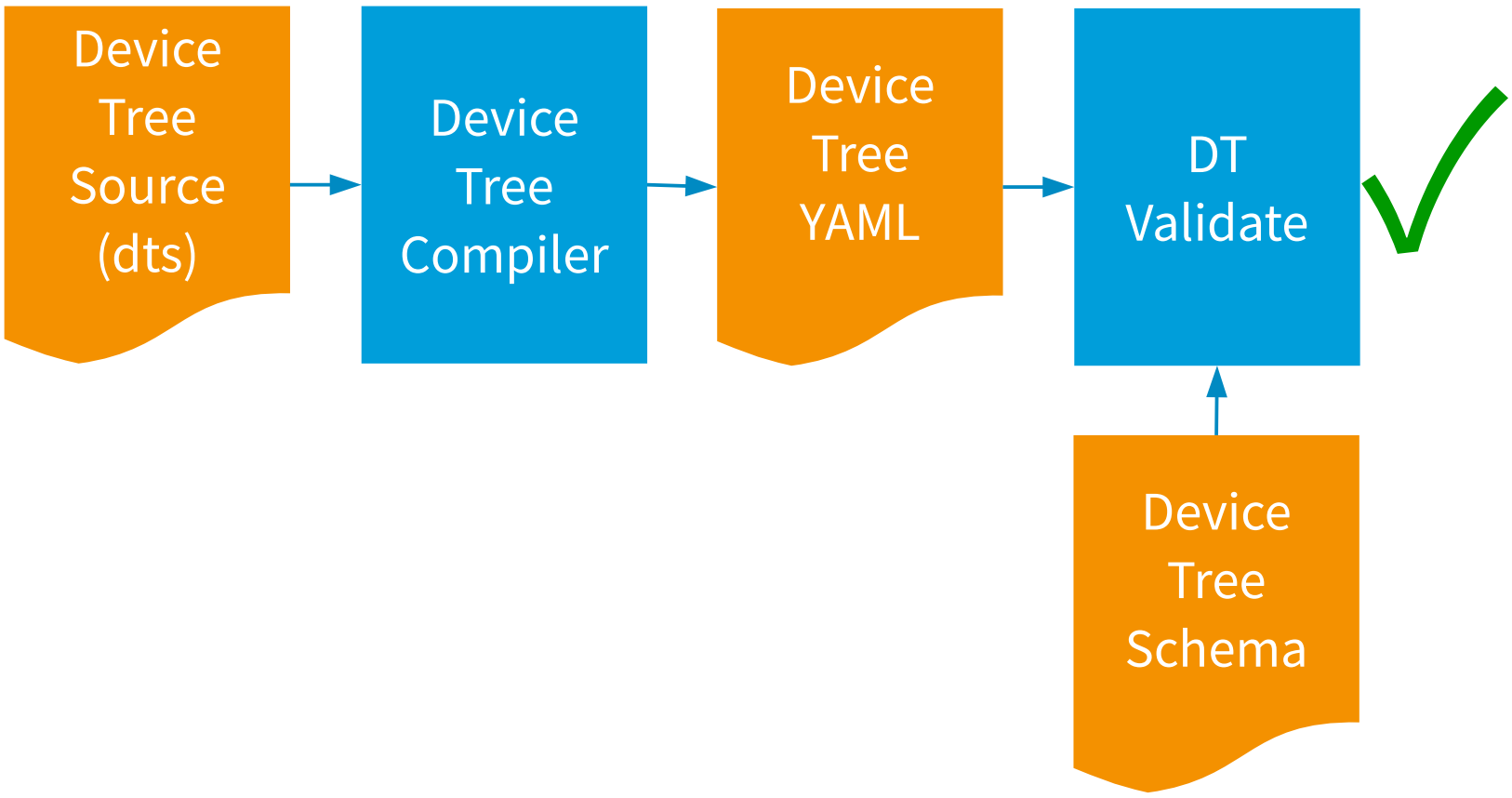


Device Tree Bindings

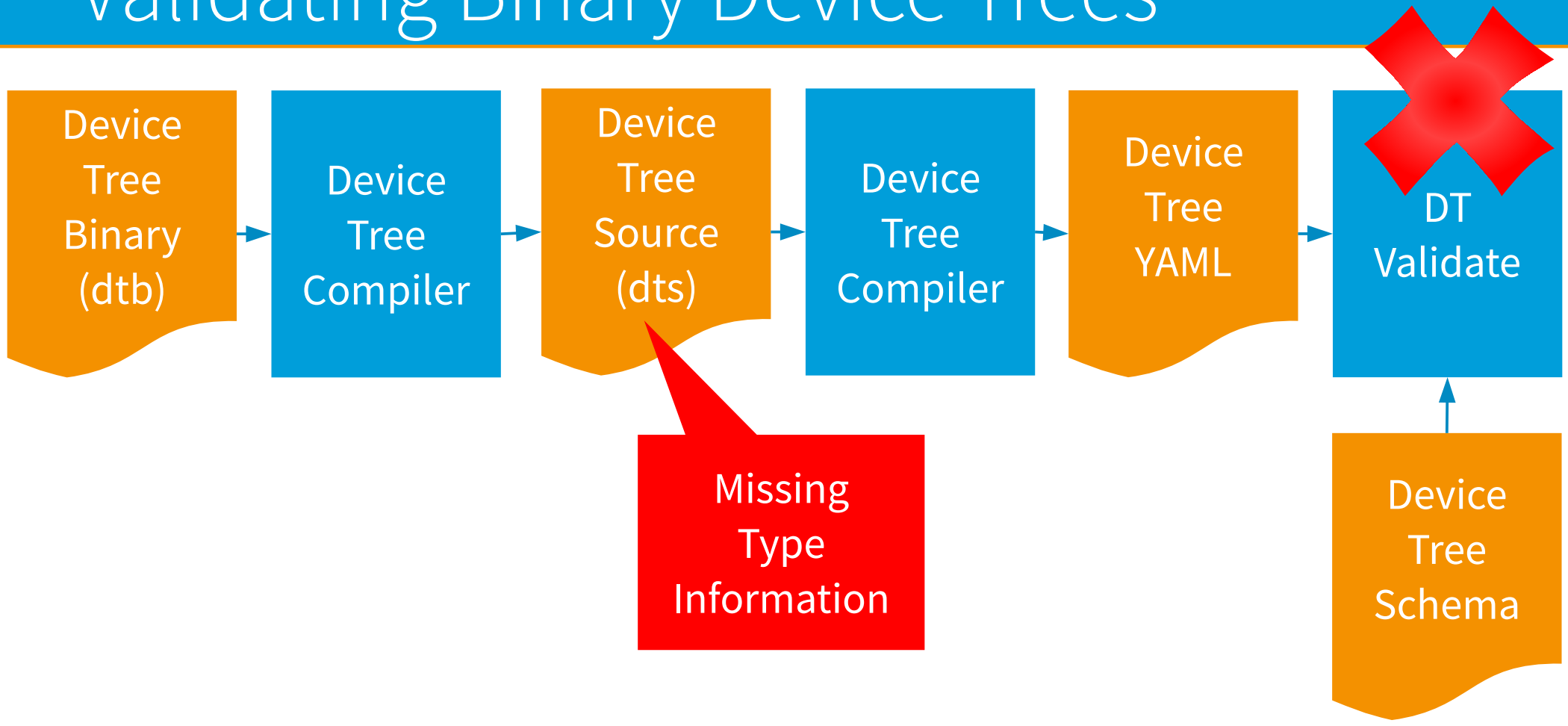
- On the move from text to YAML schema
- Definitions can be validated
- Device trees can be validated

For details see
https://github.com/devicetree-org/dt-schema
Maintainer: Rob Herring

Device Tree Validation



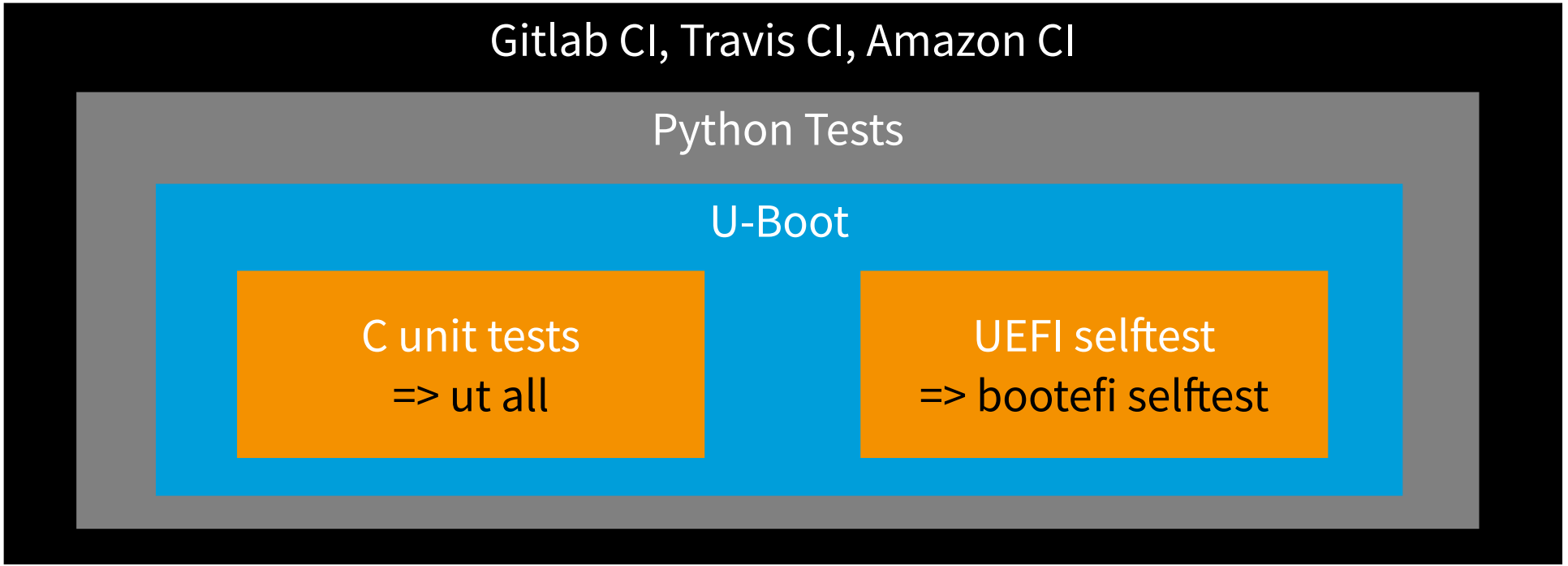
Validating Binary Device Trees



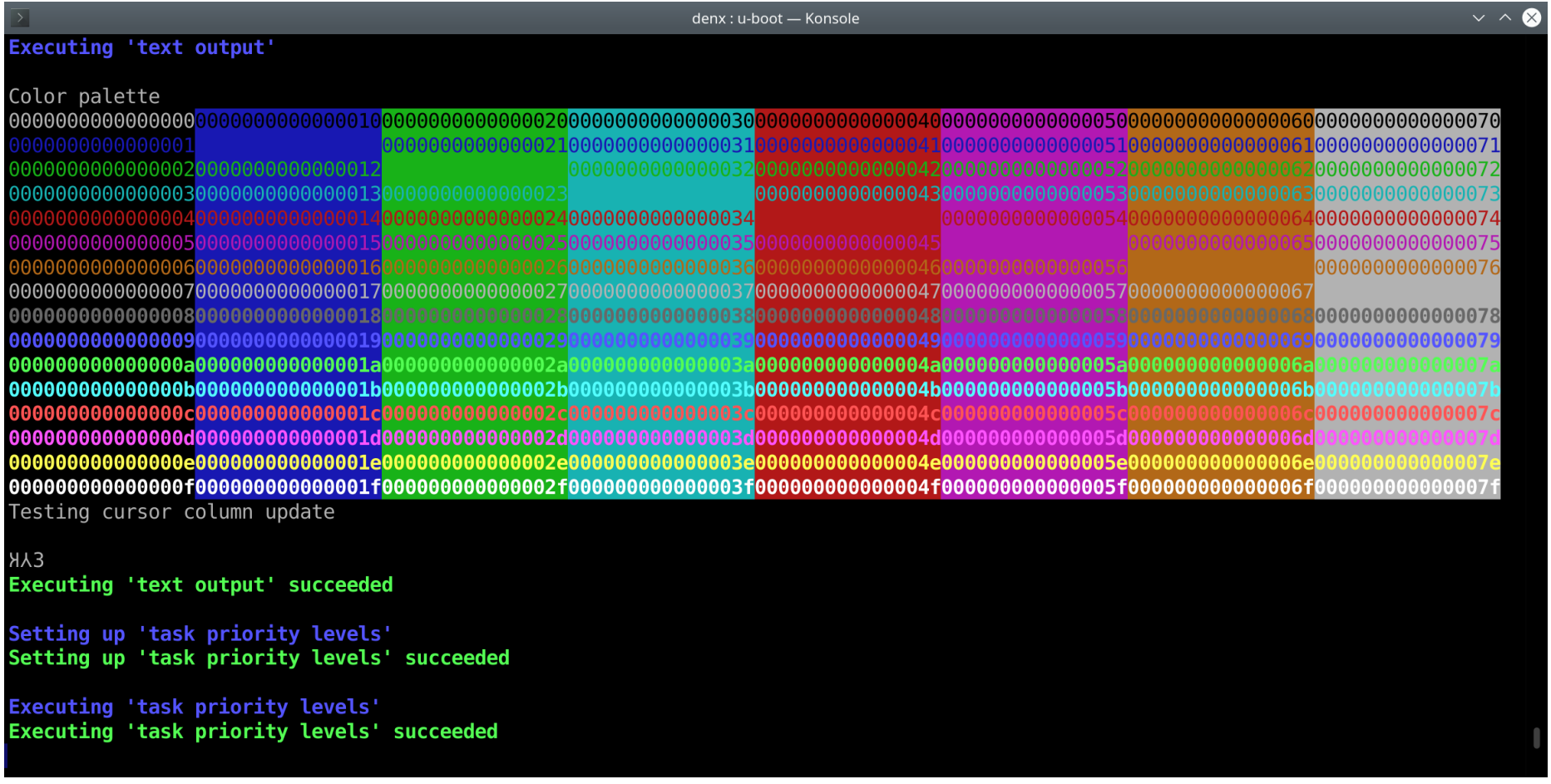
U-Boot UEFI Selftests

- Originally needed as U-Boot could not run SCT
- Covers Transient System Load
  - SetVirtualAddressMap(), ConvertPointer()
  - Device Tree (present, well formed)
  - Boot Hart ID
- Completely separated code base (except headers)
  - Only EFI API used to call into U-Boot
- Compiled into U-Boot for easy debugging
- 43 test case, 597 assertions, 10576 lines

Testing U-Boot



U-Boot UEFI Selftests



Take Aways

- UEFI Self-Certification Tests needs more developers
- Missing test coverage for boot device selection and transient system load
- Compliance testing for binary device trees missing

Q&A