Tools for Chips

Daniel Maslowski aka CyReVolt

Hello, I am Daniel aka CyReVolt :-)

Work and education

- IT security and computer science
- software engineering
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

Hello, I am Daniel aka CyReVolt :-)

Work and education

- IT security and computer science
- software engineering
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

I created Fiedka the firmware editor (https://fiedka.app) and started the Platform System Interface project: https://github.com/platform-system-interface/

Agenda

- Systems and Chips
- Mask ROM Protocols
- Implementations

Systems and Chips

What is a System?

[1] https://en.wikipedia.org/wiki/System (adapted)

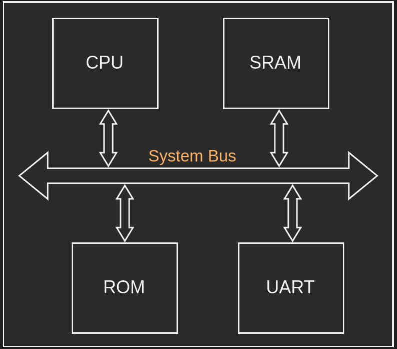
What is a System?

System[1] a set of components following rules and acting as a whole

[1] https://en.wikipedia.org/wiki/System (adapted)

What is a System?

System[1] a set of components following rules and acting as a whole



- modern chips are designed as systems
  - aka *System on a Chip* (SoC)
- systems may as well be virtual
  - e.g., *operating system*

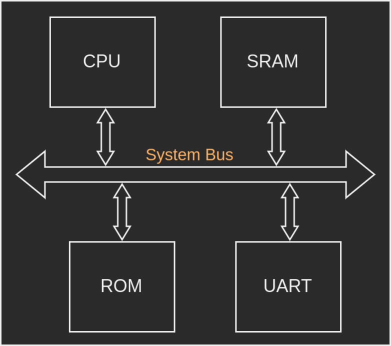
https://github.com/platform-system-interface/psi-spec/issues/24

[1] https://en.wikipedia.org/wiki/System (adapted)

[2] https://en.wikipedia.org/wiki/Computing_platform (adapted)

What is a System?

System[1] a set of components following rules and acting as a whole



- modern chips are designed as systems
  - aka *System on a Chip* (SoC)
- systems may as well be virtual
  - e.g., *operating system*

https://github.com/platform-system-interface/psi-spec/issues/24

Platform[2] a *system* with stable *interfaces*, providing an environment

Note: *stable* here means being only extended or changing slowly/rarely.

[1] https://en.wikipedia.org/wiki/System (adapted)

[2] https://en.wikipedia.org/wiki/Computing_platform (adapted)

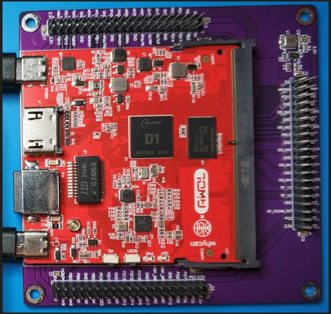
SoCs and SoMs

SoCs and SoMs

System on Chip

- contained in a chip package
  - often with many pins
- multiple form factors[a]
  - BGA (ball grid array)
  - QFP (quad flat package)

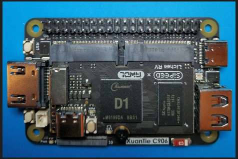
[a] https://electrical-information.com/package-types/



System on Module

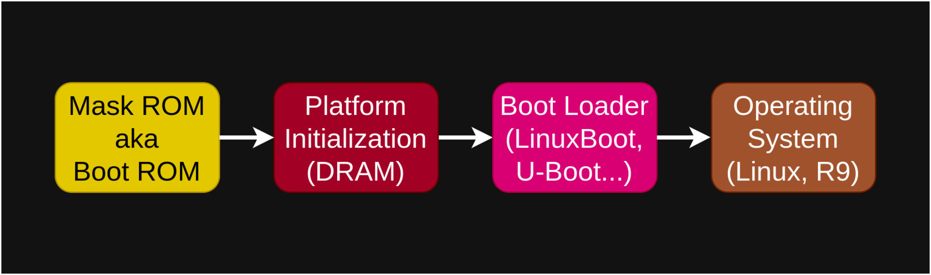
- a PCB to integrate in a product
- many form factors, few standards[a]
  - “stamp”, a rectangle with contacts at the edges
  - “gold finger” connectors
  - CM (Compute Module)

[a] https://www.compulab.com/blog/how-to-choose-the-right-system-on-module-som-selection-guide

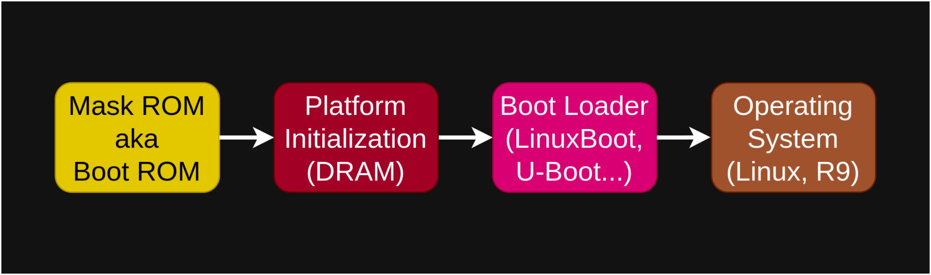


From ROM to OS

From ROM to OS

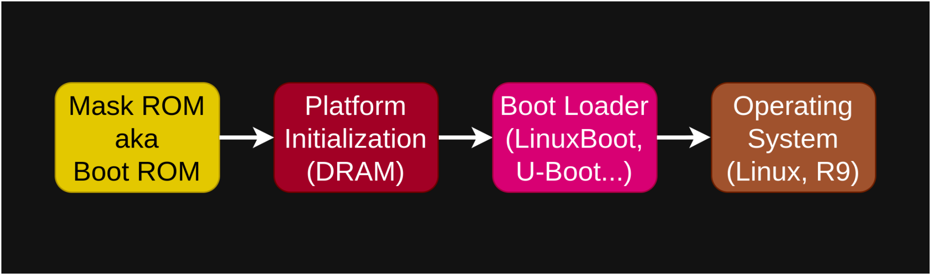

Mask ROM aka Boot ROM
Platform Initialization (DRAM)
Boot Loader (LinuxBoot, U-Boot...)
Operating System (Linux, R9)

From ROM to OS


Mask ROM aka Boot ROM
Platform Initialization (DRAM)
Boot Loader (LinuxBoot, U-Boot...)
Operating System (Linux, R9)

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

From ROM to OS



Mask ROM aka Boot ROM → Platform Initialization (DRAM) → Boot Loader (LinuxBoot, U-Boot...) → Operating System (Linux, R9)

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over serial or USB ports, which is great for development and *ownership*.

Mask ROM Protocols

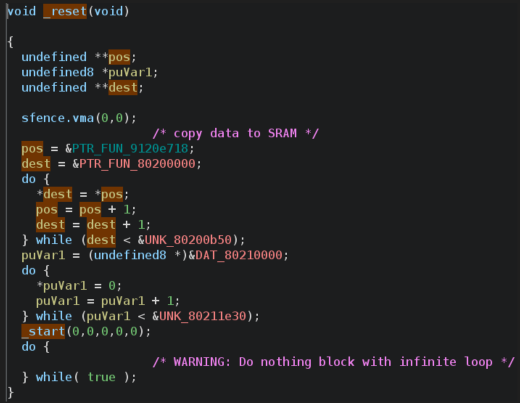
Mask ROM

Mask ROM

- baked into a chip
- initial code run by a processor/SoC
  - hence aka *boot ROM*
- often mapped to memory
- dump to file and load into Ghidra to study
- find strings, figure out flow
- flow may depend on settings
  - OTP
  - GPIOs

Mask ROM

- baked into a chip
- initial code run by a processor/SoC
  - hence aka *boot ROM*
- often mapped to memory
- dump to file and load into Ghidra to study
- find strings, figure out flow
- flow may depend on settings
  - OTP
  - GPIOs



```
void _reset(void)

{
  undefined **pos;
  undefined8 *puVar1;
  undefined **dest;

  sfence.vma(0,0);
                    /* copy data to SRAM */
  pos = &PTR_FUN_9120e718;
  dest = &PTR_FUN_80200000;
  do {
    *dest = *pos;
    pos = pos + 1;
    dest = dest + 1;
  } while (dest < &UNK_80200b50);
  puVar1 = (undefined8 *)&DAT_80210000;
  do {
    *puVar1 = 0;
    puVar1 = puVar1 + 1;
  } while (puVar1 < &UNK_80211e30);
  _start(0,0,0,0,0);
  do {
                    /* WARNING: Do nothing block with infinite loop */
  } while( true );
}
```

Why bother?

Why bother?

Many SBCs and consumer products are based on SoCs.

Why bother?

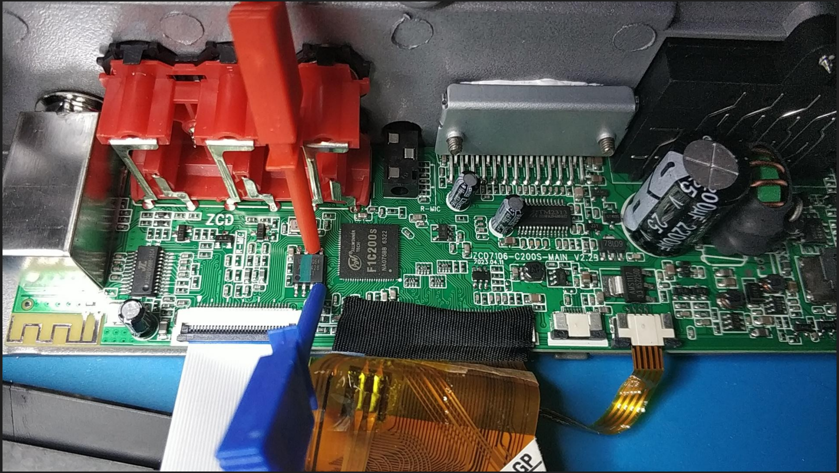
Many SBCs and consumer products are based on SoCs.

You may want to develop or run custom software…

Why bother?

Many SBCs and consumer products are based on SoCs.

You may want to develop or run custom software…



General notes and issues

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

We can look at vendor tools if they provide source code, or reverse engineer.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

We can look at vendor tools if they provide source code, or reverse engineer.

Serial output can tell us something about the machine state.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

We can look at vendor tools if they provide source code, or reverse engineer.

Serial output can tell us something about the machine state.

Additional knowledge can be gained from mask ROM analysis.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

We can look at vendor tools if they provide source code, or reverse engineer.

Serial output can tell us something about the machine state.

Additional knowledge can be gained from mask ROM analysis.

Sometimes there are hidden commands or details that are not documented.

General notes and issues

Many vendors have proprietary procotols and/or custom commands.

There is usually no or little documentation on the protocol.

We can look at vendor tools if they provide source code, or reverse engineer.

Serial output can tell us something about the machine state.

Additional knowledge can be gained from mask ROM analysis.

Sometimes there are hidden commands or details that are not documented.

Dealing with *OTP* (one-time programmable) configuration can be complex.

Implementations

StarFive JH71x0

https://github.com/platform-system-interface/jh_boot

Interface	UART
Protocol	Xmodem, with quirks

The JH71x0 mask ROM either loads code from a storage part (recommended: SPI flash) or via serial, which is slow. No other functionality appears to be available.

https://www.youtube.com/watch?v=SWrjYX8ZSb8&list=PLenOHeTI_A9MJlYIOAVC0JDpKKXX9mZgK&pp=gAQB

Amlogic

https://github.com/platform-system-interface/aml_boot

Interface	USB
Protocol	proprietary, later fastboot

Different SoCs offer different functionality, sometimes restricted, possibly due to OTP fuses.

https://mastodon.social/@CyReVolt/111194596957100647

Canaan Kendryte

https://github.com/platform-system-interface/kendryte_boot

Interface	USB
Protocol	proprietary

The protocol has simple commands to load and run code. The client supplies the address to load to. Jumping back into the mask ROM to load additional code is possible.

https://www.youtube.com/watch?v=hfz8QBB4M3g&list=PLenOHeTI_A9N0hj5wNEezqirGm7JaLgDP&pp=gAQB

DEMO

Bouffalo Lab

https://github.com/platform-system-interface/bl_boot

Interface	UART
Protocol	proprietary

The BL808 SoC offers a lot of functionality. It can read from and write to flash, read out and program OTP fuses, and load and execute code. It can run at high baud rates, so big payloads are not much of an issue. There need to be large and complex headers to run code though.

We gained a lot of knowledge thanks to earlier work from the community:

- https://openbouffalo.github.io/chips/bl808/efuse/
- https://openbouffalo.org/index.php/BL808

https://www.youtube.com/watch?v=ARyhNbjE0VM&list=PLenOHeTI_A9MwA0HlNogiJVvU5RtsDSz9&pp=gAQB

DEMO

Sophgo

https://github.com/platform-system-interface/sg_boot

Interface	UART
Protocol	proprietary

SG200x/CVITEK SoCs are very sensitive. Some serial adapters would mostly error, and the software running on the SoC has to define the load address.

More vendors and tools[3]

More vendors and tools[3]

Proprietary

- Allwinner: `sunxi-fel`, `xfel`, `aw-fel-cli` (we forked it)
- Rockchip: `rkflashtool`, `rkdeveloptool`
- Amlogic: `pyamlboot` (starting point for `aml_boot`)
- NXP: `uuu`, `imx_usb_loader`
- Qualcomm: `qtools`, `qbootcl`, `qdl`, ...
- ... keep your eyes open :-)

[3] https://platform-system-interface.github.io/psi-spec/mask-roms-loaders.html

More vendors and tools[3]

Proprietary

- Allwinner: `sunxi-fel`, `xfel`, `aw-fel-cli` (we forked it)
- Rockchip: `rkflashtool`, `rkdeveloptool`
- Amlogic: `pyamlboot` (starting point for `aml_boot`)
- NXP: `uuu`, `imx_usb_loader`
- Qualcomm: `qtools`, `qbootcl`, `qdl`, …
- … keep your eyes open :-)

General

- Android: `fastboot` (details vary per vendor)
  - some chips support it in their mask ROM
  - we forked a Rust client implementation: https://github.com/platform-system-interface/fastboot
  - also available in U-Boot
  - Qualcomm ported it to LK (little kernel)
- `snagboot` (multitool)

[3] https://platform-system-interface.github.io/psi-spec/mask-roms-loaders.html

Conclusion

Many different chips and protocols exist.

With the right tools, we can leverage their capabilities.

The lowest common denominator is to *load and run code*.

Our goal is to run our code as early as possible.

We can provide our own interfaces again for portability.

Thanks! :)

Follow Me

Daniel Maslowski

https://github.com/orangecms
https://twitter.com/orangecms
https://mastodon.social/@cyrevolt
https://twitch.tv/cyrevolt
https://youtube.com/@cyrevolt

https://metaspora.org/tools-for-chips.pdf

https://pretalx.installfest.cz/installfest-2024/speaker/HDFYXV/

https://metaspora.org/before-linux.pdf

https://platform-system-interface.github.io/psi-spec