

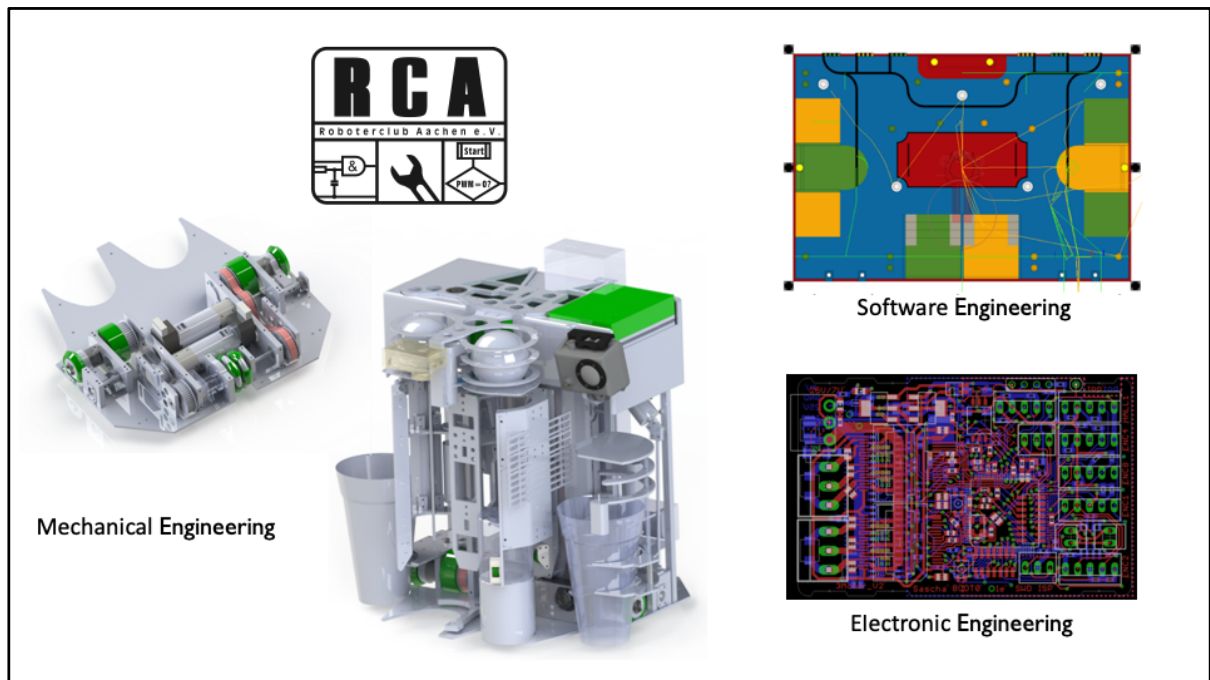
Improving Embedded Software with Data Science

Niklas Hauser

DLR Erfahrungsaustausch 2018

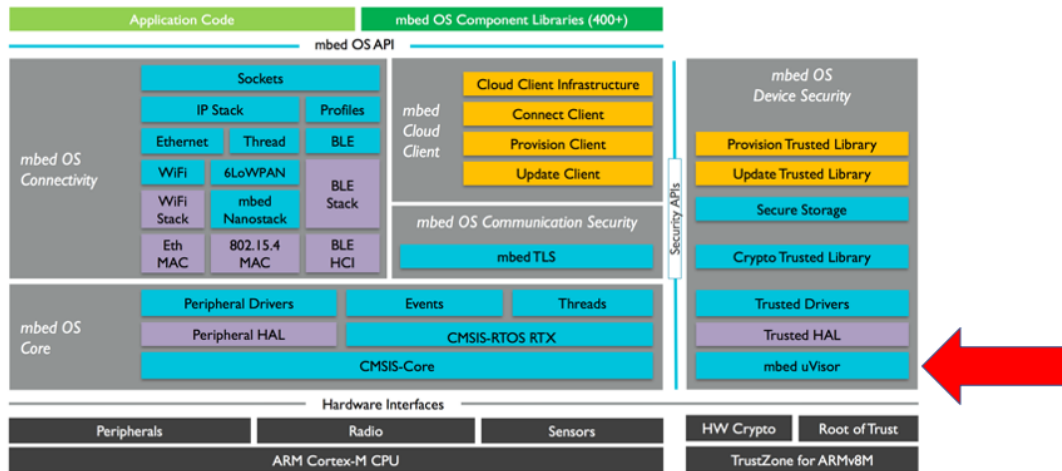


According to my parents I study something with computers.



But I also build autonomous robots in my spare time.

ARM mbed OS: architected platform security



©ARM 2016

<https://github.com/ARMmbed/uvisor/raw/docs/uVisorSecurity-TechCon2016.pdf>

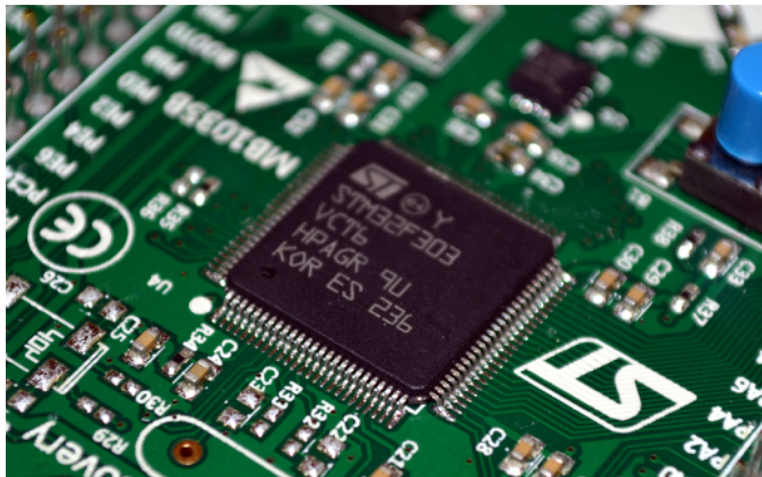
ARM

2.5y ago, I got hired by ARM to work on mbed OS Security, specifically uVisor. I spent almost 2y on the lowest levels of the ARMv7-M and ARMv8-M architectures. I've seen ALL the dirt of Cortex-M and I still (mostly) love it.



Then I quit work to hack on the largest research model railway in Germany.
YOLO.

Automation, Embedded, Data Science



This is an introduction talk, so I'll keep the technical details light and mostly talk about concepts and ideas.

EMBEDDED: For the purpose of this talk I mean MICROCONTROLLERS, specifically AVR and ARM Cortex-M.

Automation is the Key to Productivity



There many types of automation.

It evolved from a pure mechanization of labor to globe-spanning just-in-time industries.

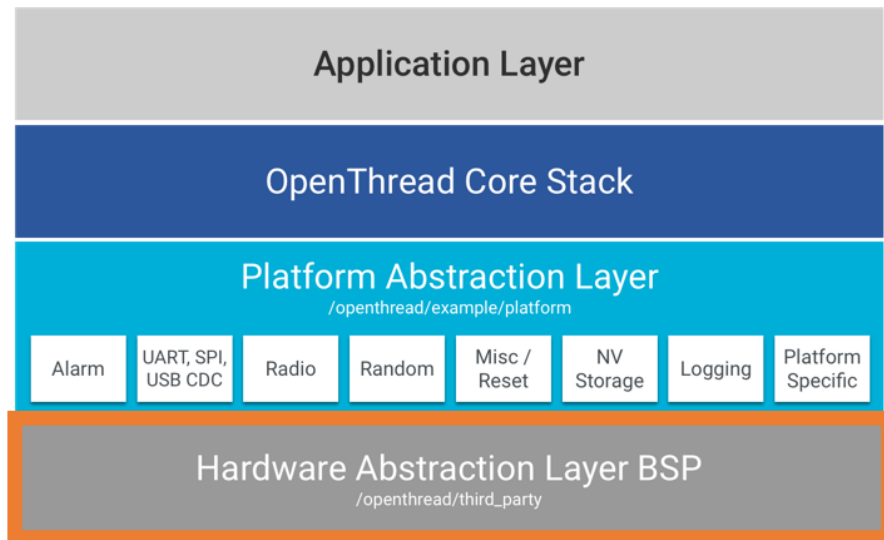
This massive complexity only scaled with the introduction of computers.

The Web is Automation on Steroids

The Web is the largest Automaton we've ever built!
Everything you know about industrial automation, the Web does at >10x scale.

The key here is an asymmetry of effort: The developer configures the automation once, and then it just works™.

Automation for Embedded Software



What similar technology can I use for embedded software to get similar scaling benefits?

What needs to be scaled in embedded software? PORTING

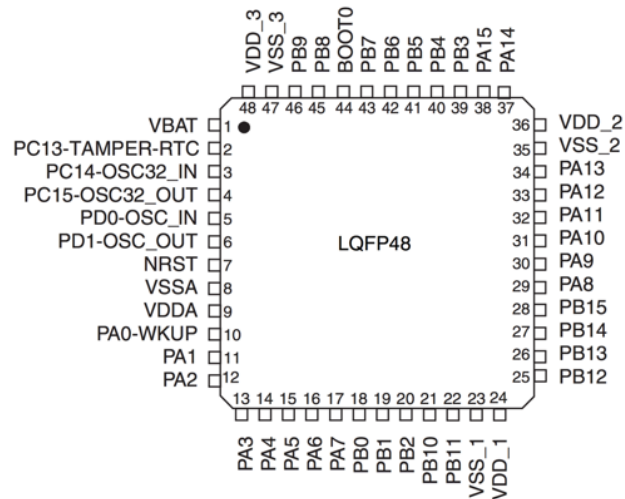
HALs are usually provided by the vendor, to abstract hardware differences between devices.

USUALLY CODED BY HAND!

This is an insane BOTTLENECK.

PAL = Problem Anderer Leute

Device pinout definitions



A typical abstraction of the HAL is to provide functions to access PINS.
A single device can be packaged in different ways, so the pins may differ.

Mbed OS is ported manually (!)

```
86 // TIM4 cannot be used because already used by the us_ticker
87 MBED_WEAK const PinMap PinMap_PWM[] = {
88     {PA_1,  PWM_2,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 2, 0)}, // TIM2_CH2 - Default
89     {PA_2,  PWM_2,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 3, 0)}, // TIM2_CH3 - Default (warning: not connected
90     {PA_3,  PWM_2,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 4, 0)}, // TIM2_CH4 - Default (warning: not connected
91     {PA_6,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 1, 0)}, // TIM3_CH1 - Default
92     {PA_7,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 2, 0)}, // TIM3_CH2 - Default
93     // {PA_7,  PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 6, 1, 1)}, // TIM1_CH1N - GPIO_PartialRemap_TIM1
94     {PA_8,  PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 1, 0)}, // TIM1_CH1 - Default
95     {PA_9,  PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 2, 0)}, // TIM1_CH2 - Default
96     {PA_10, PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 3, 0)}, // TIM1_CH3 - Default
97     {PA_11, PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 4, 0)}, // TIM1_CH4 - Default
98     {PA_15, PWM_2,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 8, 1, 0)}, // TIM2_CH1_ETR - GPIO_FullRemap_TIM2
99
100    {PB_0,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 3, 0)}, // TIM3_CH3 - Default
101    // {PB_0,  PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 6, 2, 1)}, // TIM1_CH2N - GPIO_PartialRemap_TIM1
102    {PB_1,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 4, 0)}, // TIM3_CH4 - Default
103    // {PB_1,  PWM_1,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 6, 3, 1)}, // TIM1_CH3N - GPIO_PartialRemap_TIM1
104    {PB_3,  PWM_2,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 8, 2, 0)}, // TIM2_CH2 - GPIO_FullRemap_TIM2
105    {PB_4,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 7, 1, 0)}, // TIM3_CH1 - GPIO_PartialRemap_TIM3
106    {PB_5,  PWM_3,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 7, 2, 0)}, // TIM3_CH2 - GPIO_PartialRemap_TIM3
107    // {PB_6,  PWM_4,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 1, 0)}, // TIM4_CH1 - Default (used by ticker)
108    // {PB_7,  PWM_4,  STM_PIN_DATA_EXT(STM_MODE_AF_PP, GPIO_PULLUP, 0, 2, 0)}, // TIM4_CH2 - Default (used by ticker)
```

In mbed OS this table describes the pin signal connections to the internal peripherals. You can see some manually commented out lines.

ST has 4-6 full-time engineers just for porting Mbed OS to STM32. They do it completely by hand.

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 8424a | 8424b | 8424c | 8424d | 8424e | 8424f | 8424g | 8424h | 8424i | 8424j | 8424k | 8424l | 8424m | 8424n | 8424o | 8424p | 8424q | 8424r | 8424s | 8424t | 8424u | 8424v | 8424w | 8424x | 8424y | 8424z |
| 8425a | 8425b | 8425c | 8425d | 8425e | 8425f | 8425g | 8425h | 8425i | 8425j | 8425k | 8425l | 8425m | 8425n | 8425o | 8425p | 8425q | 8425r | 8425s | 8425t | 8425u | 8425v | 8425w | 8425x | 8425y | 8425z |
| 8426a | 8426b | 8426c | 8426d | 8426e | 8426f | 8426g | 8426h | 8426i | 8426j | 8426k | 8426l | 8426m | 8426n | 8426o | 8426p | 8426q | 8426r | 8426s | 8426t | 8426u | 8426v | 8426w | 8426x | 8426y | 8426z |
| 8427a | 8427b | 8427c | 8427d | 8427e | 8427f | 8427g | 8427h | 8427i | 8427j | 8427k | 8427l | 8427m | 8427n | 8427o | 8427p | 8427q | 8427r | 8427s | 8427t | 8427u | 8427v | 8427w | 8427x | 8427y | 8427z |
| 8428a | 8428b | 8428c | 8428d | 8428e | 8428f | 8428g | 8428h | 8428i | 8428j | 8428k | 8428l | 8428m | 8428n | 8428o | 8428p | 8428q | 8428r | 8428s | 8428t | 8428u | 8428v | 8428w | 8428x | 8428y | 8428z |
| 8429a | 8429b | 8429c | 8429d | 8429e | 8429f | 8429g | 8429h | 8429i | 8429j | 8429k | 8429l | 8429m | 8429n | 8429o | 8429p | 8429q | 8429r | 8429s | 8429t | 8429u | 8429v | 8429w | 8429x | 8429y | 8429z |
| 8430a | 8430b | 8430c | 8430d | 8430e | 8430f | 8430g | 8430h | 8430i | 8430j | 8430k | 8430l | 8430m | 8430n | 8430o | 8430p | 8430q | 8430r | 8430s | 8430t | 8430u | 8430v | 8430w | 8430x | 8430y | 8430z |
| 8431a | 8431b | 8431c | 8431d | 8431e | 8431f | 8431g | 8431h | 8431i | 8431j | 8431k | 8431l | 8431m | 8431n | 8431o | 8431p | 8431q | 8431r | 8431s | 8431t | 8431u | 8431v | 8431w | 8431x | 8431y | 8431z |
| 8432a | 8432b | 8432c | 8432d | 8432e | 8432f | 8432g | 8432h | 8432i | 8432j | 8432k | 8432l | 8432m | 8432n | 8432o | 8432p | 8432q | 8432r | 8432s | 8432t | 8432u | 8432v | 8432w | 8432x | 8432y | 8432z |
| 8433a | 8433b | 8433c | 8433d | 8433e | 8433f | 8433g | 8433h | 8433i | 8433j | 8433k | 8433l | 8433m | 8433n | 8433o | 8433p | 8433q | 8433r | 8433s | 8433t | 8433u | 8433v | 8433w | 8433x | 8433y | 8433z |
| 8434a | 8434b | 8434c | 8434d | 8434e | 8434f | 8434g | 8434h | 8434i | 8434j | 8434k | 8434l | 8434m | 8434n | 8434o | 8434p | 8434q | 8434r | 8434s | 8434t | 8434u | 8434v | 8434w | 8434x | 8434y | 8434z |
| 8435a | 8435b | 8435c | 8435d | 8435e | 8435f | 8435g | 8435h | 8435i | 8435j | 8435k | 8435l | 8435m | 8435n | 8435o | 8435p | 8435q | 8435r | 8435s | 8435t | 8435u | 8435v | 8435w | 8435x | 8435y | 8435z |
| 8436a | 8436b | 8436c | 8436d | 8436e | 8436f | 8436g | 8436h | 8436i | 8436j | 8436k | 8436l | 8436m | 8436n | 8436o | 8436p | 8436q | 8436r | 8436s | 8436t | 8436u | 8436v | 8436w | 8436x | 8436y | 8436z |
| 8437a | 8437b | 8437c | 8437d | 8437e | 8437f | 8437g | 8437h | 8437i | 8437j | 8437k | 8437l | 8437m | 8437n | 8437o | 8437p | 8437q | 8437r | 8437s | 8437t | 8437u | 8437v | 8437w | 8437x | 8437y | 8437z |
| 8438a | 8438b | 8438c | 8438d | 8438e | 8438f | 8438g | 8438h | 8438i | 8438j | 8438k | 8438l | 8438m | 8438n | 8438o | 8438p | 8438q | 8438r | 8438s | 8438t | 8438u | 8438v | 8438w | 8438x | 8438y | 8438z |
| 8439a | 8439b | 8439c | 8439d | 8439e | 8439f | 8439g | 8439h | 8439i | 8439j | 8439k | 8439l | 8439m | | | | | | | | | | | | | |

12

Mbed OS supports 55 STM32 devices

```
$ find targets/TARGET_STM -name "PeripheralPins.c"
targets/TARGET_STM/TARGET_STM32F0/TARGET_DISCO_F051R8/PeripheralPins.c
target+/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/PeripheralPins.c
target $ find . -name "STM32*.ld"
target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_DISCO_F051R8/device/TOOLCHAIN_GCC_ARM/STM32F0xx.ld
target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/device/TOOLCHAIN_GCC_ARM/STM32F030x8.ld
target ./target $ find . -name "startup_stm32*.S"
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_DISCO_F051R8/device/TOOLCHAIN_ARM_MICRO/startup_stm32f051xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_DISCO_F051R8/device/TOOLCHAIN_ARM_STD/startup_stm32f051xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_DISCO_F051R8/device/TOOLCHAIN_GCC_ARM/startup_stm32f051xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/device/TOOLCHAIN_ARM_MICRO/startup_stm32f030xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/device/TOOLCHAIN_ARM_STD/startup_stm32f030xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/device/TOOLCHAIN_GCC_ARM/startup_stm32f030xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F030R8/device/TOOLCHAIN_ARM_MICRO/startup_stm32f030xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F031K6/device/TOOLCHAIN_ARM_MICRO/startup_stm32f031xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F031K6/device/TOOLCHAIN_ARM_STD/startup_stm32f031xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F031K6/device/TOOLCHAIN_GCC_ARM/startup_stm32f031xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F031K6/device/TOOLCHAIN_ARM_MICRO/startup_stm32f031xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F042K6/device/TOOLCHAIN_ARM_MICRO/startup_stm32f042xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F042K6/device/TOOLCHAIN_ARM_STD/startup_stm32f042xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F042K6/device/TOOLCHAIN_GCC_ARM/startup_stm32f042xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F042K6/device/TOOLCHAIN_ARM_MICRO/startup_stm32f042xx.S
target ./target ./targets/TARGET_STM/TARGET_STM32F0/TARGET_NUCLEO_F070RB/device/TOOLCHAIN_ARM_MICRO/startup_stm32f070xx.S
```

- 55 manually created versions of the pinout data
- 51 Linkerscripts (some manually patched)
- ~200 startup scripts (some manually patched)
- We needed to modify the linker- and startup script for uVisor



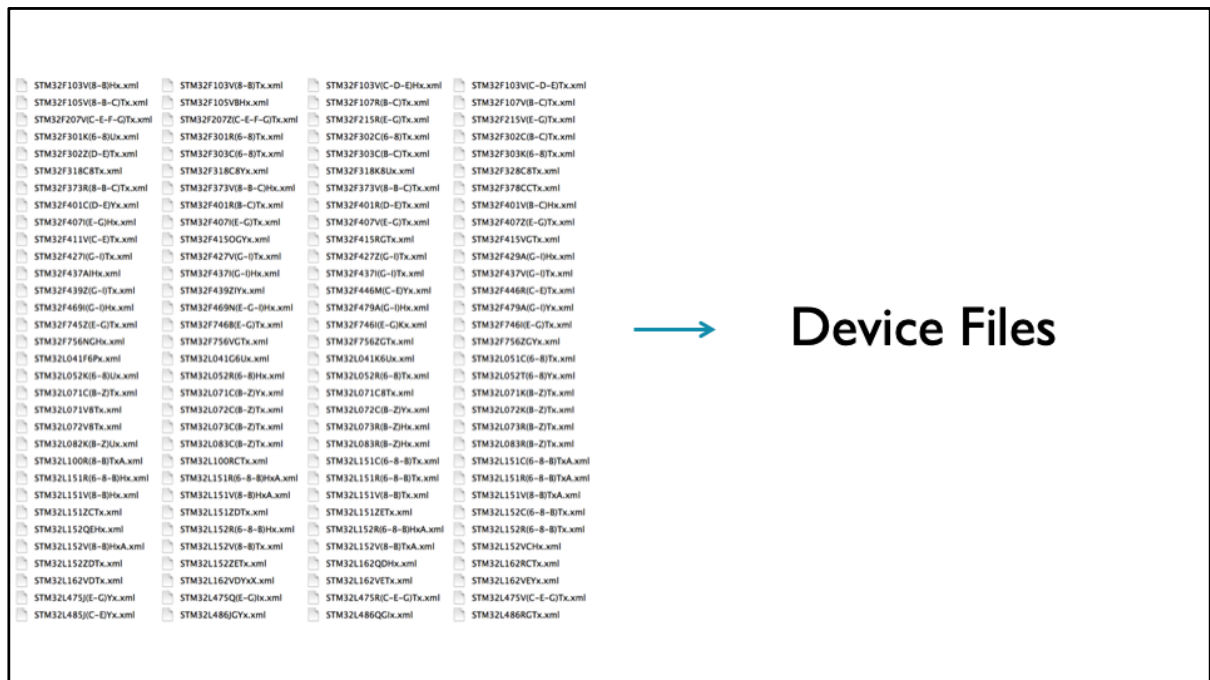
TRIPLE FACEPALM

Not even double facepalm can explain how much you fail

X Billion IoT devices by 2035?
HOW ARE YOU GOING TO MAINTAIN THIS?!?



Let's think about this problem in a calm and structured manner.
Tea?

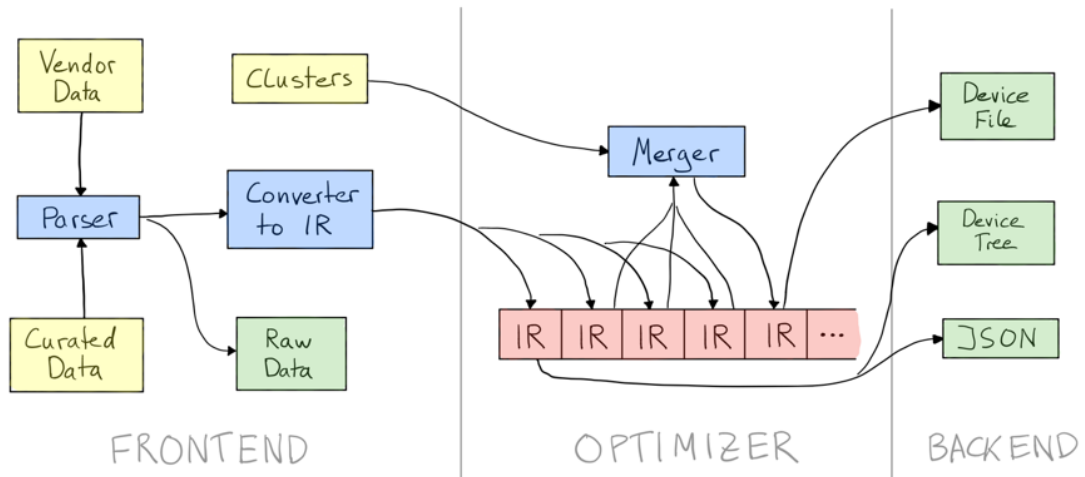


70MB of XML files, with a lot of device details.

modm-devices contains

- CPU: Type and Interrupt Vector Table
- Memories: Flash, RAM, Backup
- Peripherals: Type and Instances
- Gpio: Name and Signals

blog.salkinium.com/modm-devices



The reality is a bit more involved.

~1200 STM32s + ~200 AVR_s

modm-io / modm-devices

Unwatch 5 Unstar 7 Fork 1

Code Issues 0 Pull requests 0 Insights Settings

Curated device data for all AVR and STM32 devices <http://blog.salkinium.com/modm-devices> Edit

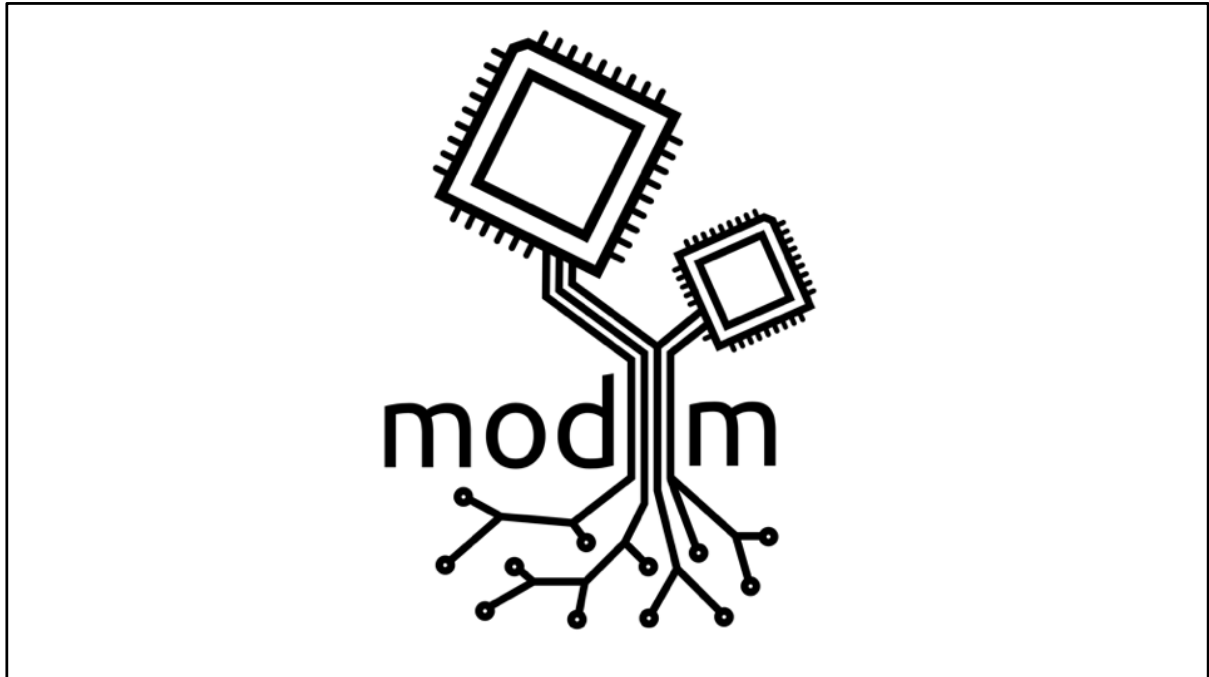
avr stm32 data microcontroller device-tree modm Manage topics

74 commits 3 branches 0 releases 2 contributors MPL-2.0

Branch: develop New pull request Create new file Upload files Find file Clone or download

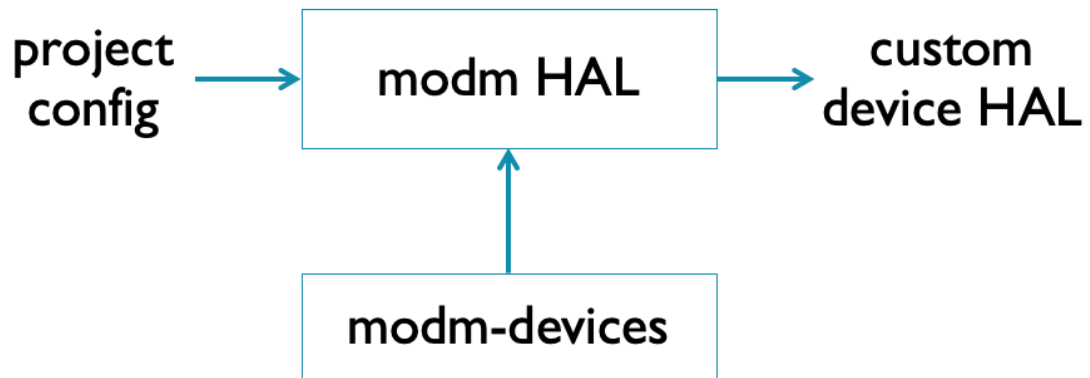
| | | |
|-------------|-------------------------------------|----------------------------------|
| salkinium | Update cmsis-header-stm32 submodule | Latest commit dd631c0 7 days ago |
| devices | Update STM32 and AVR device files. | 3 months ago |
| tools | Update cmsis-header-stm32 submodule | 7 days ago |
| .gitignore | Initial commit | 2 years ago |
| .gitmodules | Add cmsis-header-stm32 submodule. | 3 months ago |
| LICENSE | Initial commit | 2 years ago |

YOU can use this data too, it's on GitHub



Fabian and I worked on modm for the last two years.
It's a C++ HAL for STM32 and AVR.
We use it to build the robot software for the Roboterclub Aachen.

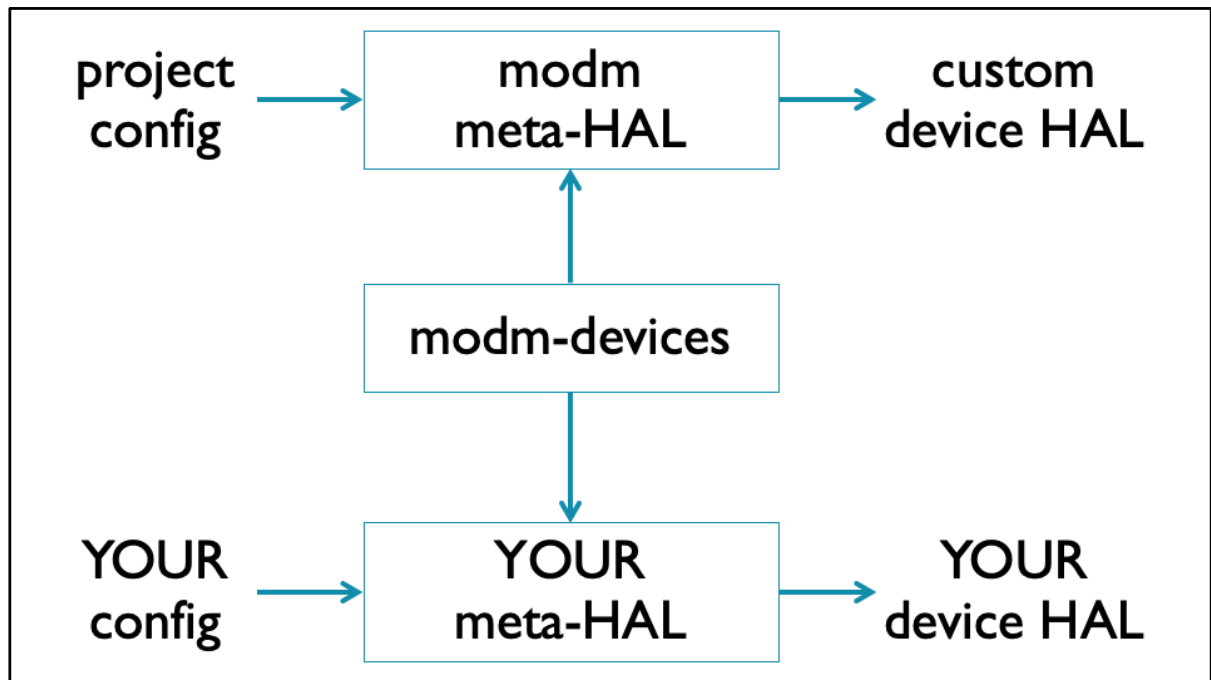
modm is our next generation **toolbox**



modm is our INTERPRETATION of modm-devices.

It's a library GENERATOR, we specify a target id and it generates the HAL for us.

We have ported ~80 AVRs and ~850 STM32 so far.



Share the data, not the HAL.

Easier to customize your HAL to your specific needs.

THIS IS LANGUAGE INDEPENDENT!

You generate C, Go, Rust, TEXT, you can also just generate documentation.

You can generate just ONE file, then gradually expand.

Particularly useful to replace pain-points in existing code bases.

Vertical vs. Horizontal Porting

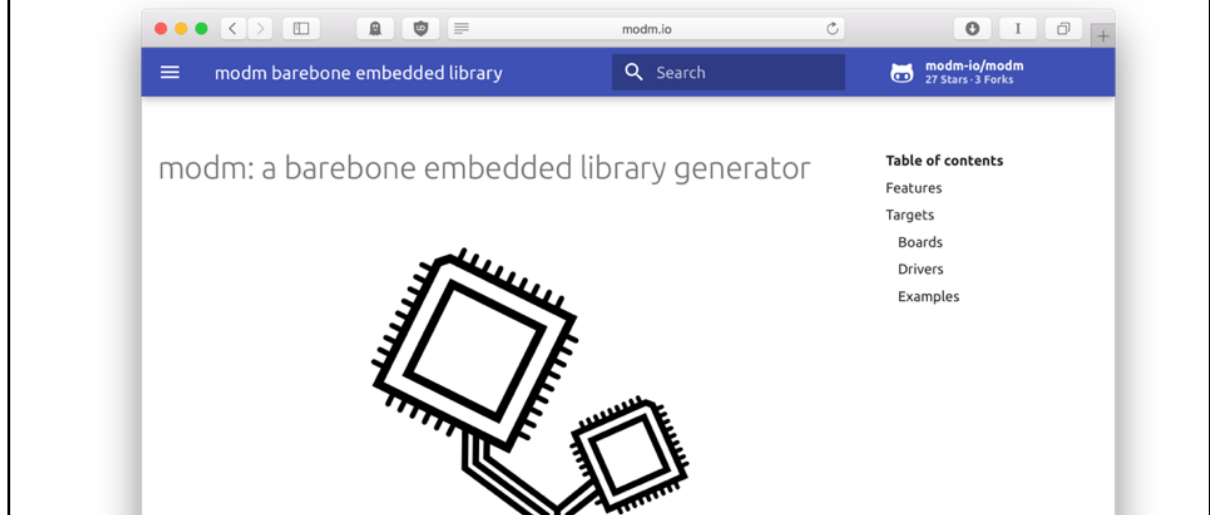
| Feature | STM32F0 | STM32F1 | STM32F3 | STM32F4 | STM32F7 |
|---------|---------|---------|---------|---------|---------|
| Core | ✓ | ✓ | ✓ | ✓ | ✓ |
| GPIO | ✓ | ✓ | ✓ | ✓ | ✓ |
| Clock | ✓ | ✓ | ✓ | ✓ | ✓ |
| UART | ✓ | ✓ | ✓ | ✓ | ✓ |
| SPI | ✓ | ✓ | ✓ | ✓ | ✓ |

There is a change in how you port your targets now:

Before: For each target implement the feature.

Now: for each feature port to all targets

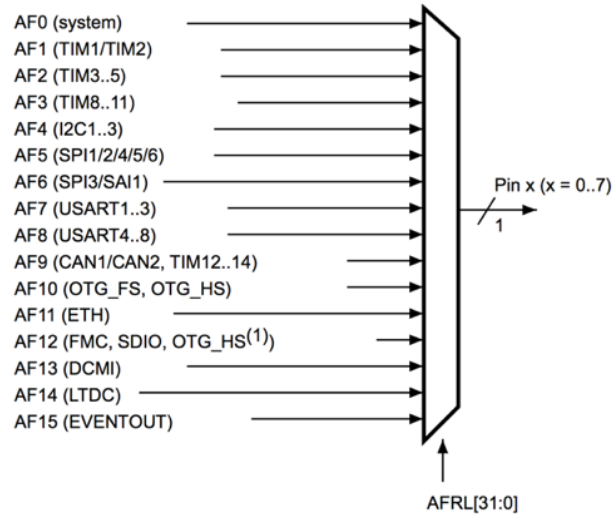
modm.io



I glossed over the details of the actual code generation.
You can read up on this on our website with examples.
Installation guide, getting started guide, explanations how it works.

I'm not here to sell you on our HAL, I want to show you specific problems and how we used our toolbox to solve them.

STM32 GPIO Alternate Functions



I've already touched on signal connections.

Each pin has a number of peripherals that it can be connected to.

| Port | AF0 | AF1 | AF2 | AF3 | AF4 | AF5 | AF6 | AF7 | AF8 | AF9 | AF10 | AF11 | AF12 | AF13 | AF14 | AF15 | |
|--------|------|------------|-------------------|--------------|-----------|----------------|------------------|-------------------|--------------------|--------------------------------|-------------------------|-----------------|---------------------------------|---------------|------------|-----------|-----------|
| | SYS | TIM1/2 | TIM3/4/5 | TIM8/9/10/11 | I2C1/2/3 | SPI1/2/3/4/5/6 | SPI2/3/SAI1 | SPI2/3/USART1/2/3 | USART6/UART4/5/7/8 | CAN1/2/TIM12/13/14/QUADSPI/LCD | QUADSPI/OTG2_HS/OTG1_FS | ETH | FMC/SDIO/OTG2_FS | DCMI/DSI/HOST | LCD | SYS | |
| Port A | PA0 | - | TIM2_CH1/TIM2_ETR | TIM5_CH1 | TIM8_ETR | - | - | - | USART2_CTS | UART4_TX | - | - | ETH_MII_CRS | - | - | - | EVENT OUT |
| | PA1 | - | TIM2_CH2 | TIM5_CH2 | - | - | - | - | USART2_RTS | UART4_RX | QUADSPI_BK1_IO3 | - | ETH_MII_RX_CLK/ETH_RMII_REF_CLK | - | - | LCD_R2 | EVENT OUT |
| | PA2 | - | TIM2_CH3 | TIM5_CH3 | TIM9_CH1 | - | - | - | USART2_TX | - | - | - | ETH_MDIO | - | - | LCD_R1 | EVENT OUT |
| | PA3 | - | TIM2_CH4 | TIM5_CH4 | TIM9_CH2 | - | - | - | USART2_RX | - | LCD_B2 | OTG_HS_ULPI_D0 | ETH_MII_COL | - | - | LCD_B5 | EVENT OUT |
| | PA4 | - | - | - | - | - | SPI1_NSS | SPI3_NSS/I2S3_WS | USART2_CS | - | - | - | - | OTG_HS_SOF | DCMI_HSYNC | LCD_VSYNC | EVENT OUT |
| | PA5 | - | TIM2_CH1/TIM2_ETR | - | TIM8_CH1N | - | SPI1_SCK | - | - | - | - | OTG_HS_ULPI_CLK | - | - | - | LCD_R4 | EVENT OUT |
| | PA6 | - | TIM1_BKIN | TIM3_CH1 | TIM8_BK1N | - | SPI1_MISO | - | - | - | TIM13_CH1 | - | - | - | DCMI_PCLK | LCD_G2 | EVENT OUT |
| | PA7 | - | TIM1_CH1N | TIM3_CH2 | TIM8_CH1N | - | SPI1_MOSI | - | - | - | TIM14_CH1 | QUADSPI_CLK | ETH_MII_RX_DV/ETH_RMII_CRS_DV | FMC_SDNWE | - | - | EVENT OUT |
| | PA8 | MCO1 | TIM1_CH1 | - | - | I2C3_SCL | - | - | USART1_CS | - | - | OTG_FS_SOF | - | - | - | LCD_R6 | EVENT OUT |
| | PA9 | - | TIM1_CH2 | - | - | I2C3_SMB | SPI2_SCK/I2S2_CK | - | USART1_TX | - | - | - | - | - | DCMI_D0 | - | EVENT OUT |
| | PA10 | - | TIM1_CH3 | - | - | - | - | - | USART1_RX | - | - | OTG_FS_ID | - | - | DCMI_D1 | - | EVENT OUT |
| | PA11 | - | TIM1_CH4 | - | - | - | - | - | USART1_CTS | - | CAN1_RX | OTG_FS_DM | - | - | - | LCD_R4 | EVENT OUT |
| | PA12 | - | TIM1_ETR | - | - | - | - | - | USART1_RTS | - | CAN1_TX | OTG_FS_DP | - | - | - | LCD_R5 | EVENT OUT |
| | PA13 | JTMS-SWDIO | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EVENT OUT |
| | PA14 | JTCK-SWCLK | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EVENT OUT |
| | PA15 | JTDI | TIM2_CH1/TIM2_ETR | - | - | - | SPI1_NSS | SPI3_NSS/I2S3_WS | - | - | - | - | - | - | - | - | EVENT OUT |

These connections are hardcoded, and the possibilities are described in a very long table in the datasheet.

Several pages of tables.

You do NOT want to read through that.

Gpio Signal Connection API

```
GpioB7::connect(Uart1);  
GpioB6::connect(Uart1);
```

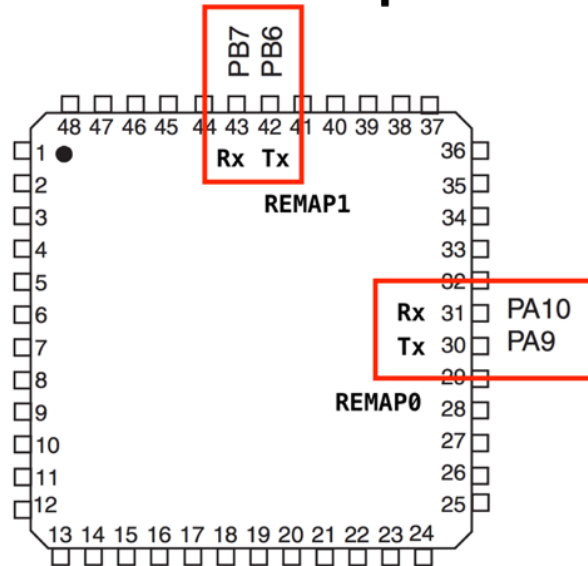
But this code breaks RX on PB7!

```
GpioB7::connect(Uart1);  
GpioA9::connect(Uart1);
```

We came up with this API.
Just connect the pin to the peripheral.

Worked fine for a long time, until we ported to STM32F1!
Things suddenly broke.

STM32F1 Remap Groups



On the STM32F1 the GPIOs can only remap in groups.
NO INDIVIDUAL REMAP POSSIBLE!

Our API has side-effects and the last call to set the group wins.
This is an implicit assumption of our HAL API:
Signal connections are independent of each other!

It's FALSE.

Improved GPIO Connection API

1. Signal connection is **independent** of Pin
2. Map is **unique**: Pin + Peripheral → Signal

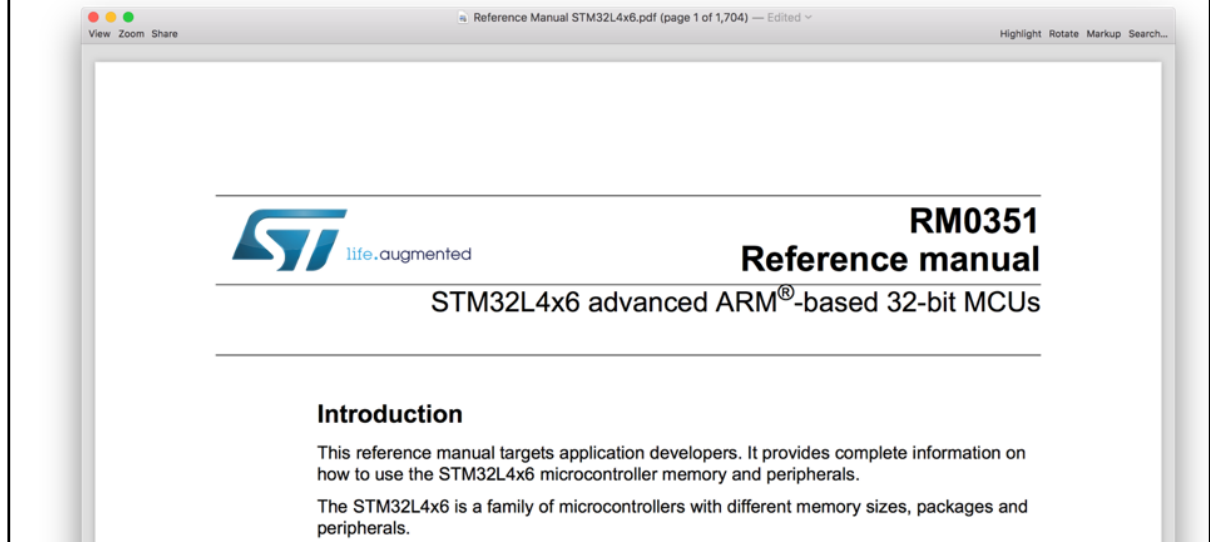
```
Uart1::connect(GpioB7::Rx,  
               GpioB6::Tx);
```

```
Uart1::connect(GpioB7::Rx,  
               ERROR! GpioA9::Tx);
```

So we changed our API. Broke the entire code.
And this API now remaps both in a group.

Compiler checks group remap validity.

We need more data!



The CubeMX data doesn't contain everything we want.

And it also doesn't scale to other vendors.

So let's have a look at a very verbose data source: THE REFERENCE MANUALS.

Let's parse some PDFs

Table 39. Summary of the DMA1 requests for each channel

| Request. number | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7 |
|--------------------|-----------|-----------|-----------|-----------------|-----------------|-----------------|-----------------|
| 0 | ADC1 | ADC2 | ADC3 | DFSDM1_ FLT0 | DFSDM1_ FLT1 | DFSDM1_ FLT2 | DFSDM1_ FLT3 |
| 1 | - | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX | SAI2_A | SAI2_B |
| 2 | - | USART3_TX | USART3_RX | USART1_TX | USART1_RX | USART2_RX | USART2_TX |
| 3 | - | I2C3_TX | I2C3_RX | I2C2_TX | I2C2_RX | I2C1_TX | I2C1_RX |

This is a DMA channel to peripheral mapping.

This is data that's not available in the CubeMX dataset.

We need it anyways to provide a channel connection API.

PDF X-Ray Vision

Table 39. Summary of the DMA1 requests for each channel

| Request number | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7 |
|----------------|-----------|-----------|-----------|-------------|-------------|-------------|-------------|
| 0 | ADC1 | ADC2 | ADC3 | DFSDM1_FLT0 | DFSDM1_FLT1 | DFSDM1_FLT2 | DFSDM1_FLT3 |
| 1 | - | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX | SAI2_A | SAI2_B |
| 2 | - | USART3_TX | USART3_RX | USART1_TX | USART1_RX | USART2_RX | USART2_TX |
| 3 | - | I2C3_TX | I2C3_RX | I2C2_TX | I2C2_RX | I2C1_TX | I2C1_RX |

I wrote a program to give me X-Ray vision of the PDF.
Read the Adobe PDF specification, it's quite fun.

PDF is a print format, it does not contain any semantical information.
This is not a table. It is a bunch of lines and a bunch of text overlaid.

You can recognize tables fairly easily, and then translate their contents.

Table extraction is doable

| Re-quest. num-ber | Chan-nel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7 |
|-------------------|------------|------------|------------|--------------|--------------|--------------|--------------|
| 0 | ADC1 | ADC2 | ADC3 | DFSD-M1_FLT0 | DFSD-M1_FLT1 | DFSD-M1_FLT2 | DFSD-M1_FLT3 |
| 1 | - | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX | SAI2_A | SAI2_B |
| 2 | - | USAR-T3_TX | USAR-T3_RX | USART1_T | USART1_R | USART2_R | USART2_T |
| 3 | - | I2C3_TX | I2C3_RX | I2C2_TX | I2C2_RX | I2C1_TX | I2C1_RX |

Understanding descriptions is very hard

This is a prototype extraction.

Tables are structured information by definition.

So it's somewhat easy to extract and use this data.

However, the textual descriptions of peripherals are very hard to turn into some kind of structured information.

The difficulty is not the table extraction it's the data cleanup and simplification. You need to condense this information into something that easily usable for the developer.

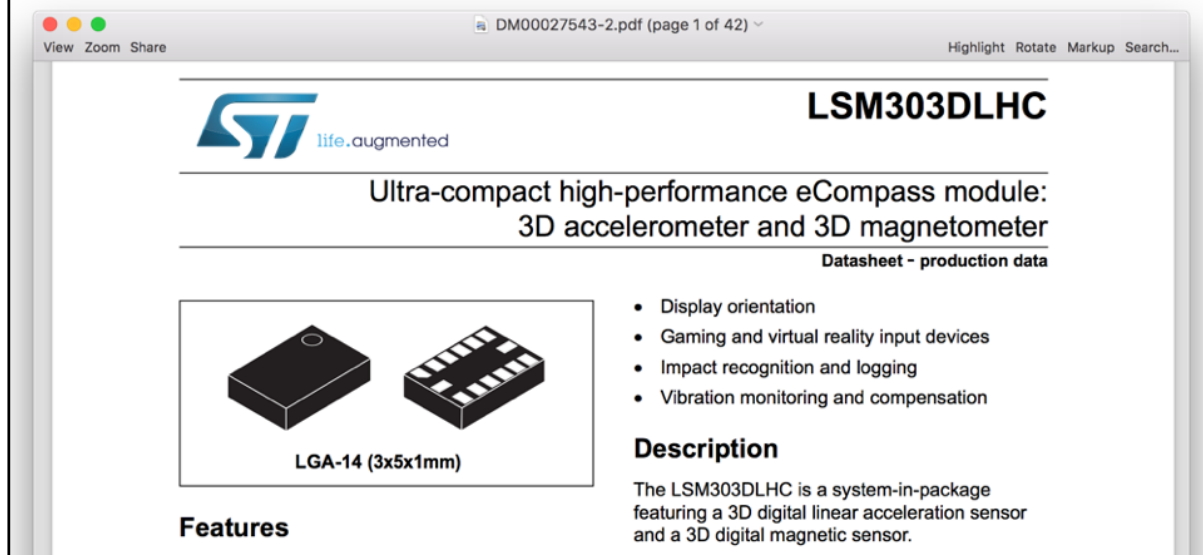
There are hundreds of these datasheets + reference manuals.

This isn't easy.

There are bugs in them, sometimes types in register names, sometimes worse.

It's going to be difficult to get some ground truth out of this.

There are lots of other PDFs



ST also produces SENSORS, which also have Datasheets.

They have the same formatting as the reference manuals, so you can just extract tables in there too.

And so we can build a database of sensors as well.

They are always the same: Memory Mapped IO via SPI/I2C.

They are by definition already platform-independent.

So why aren't there COMMON drivers for all platforms?

Abstract description of the protocol

The Plan

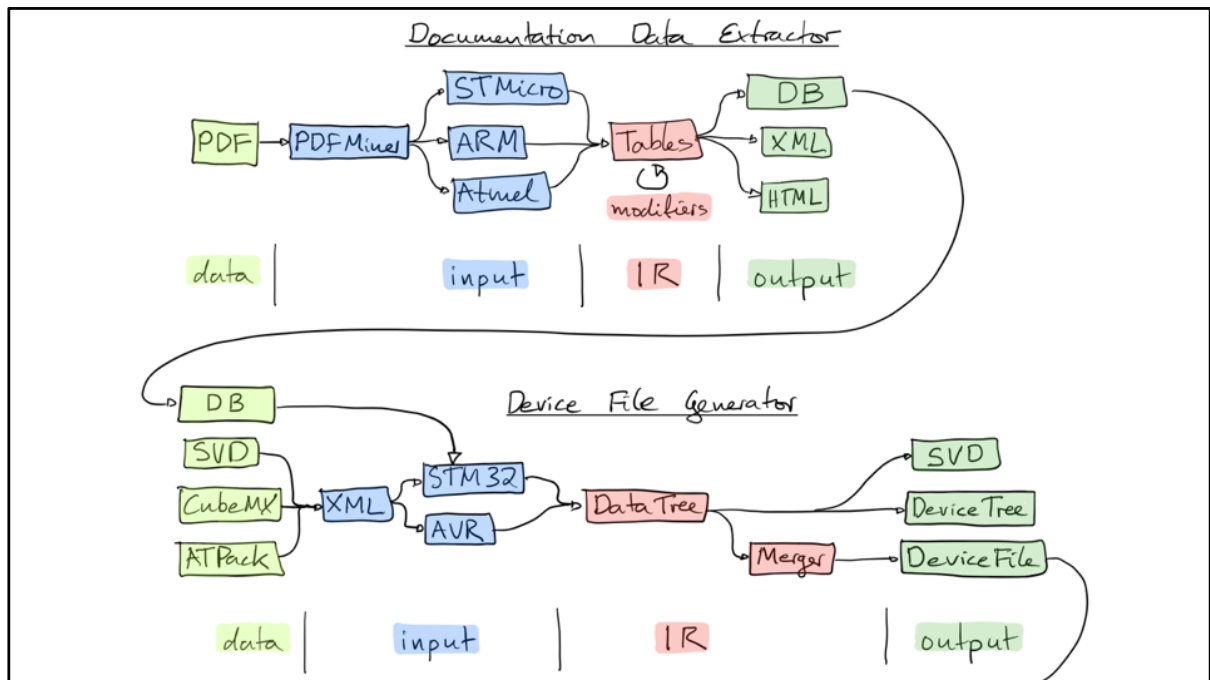
| PHASE 1 | PHASE 2 | PHASE 3 |
|----------------|----------------|----------------|
|----------------|----------------|----------------|

| | | |
|---------------------------------|--|--|
| Parse ALL Datasheets | | |
|---------------------------------|--|--|

| | | |
|--|----------|--|
| | ? | |
|--|----------|--|

| | | |
|--|--|---------------|
| | | Profit |
|--|--|---------------|

This is really self-explanatory.



This is how I envision the future of Porting Embedded Software to hundreds of devices.

We parse ALL of the datasheets, THEN MAGIC, THEN PROFIT.

Thank you for listening!

**Niklas Hauser: @salkinium (GitHub/Twitter)
(blog.)salkinium.com**

**modm.io
github.com/modm-io**

EY MANN, WO IS' MEIN TSCHUNK?