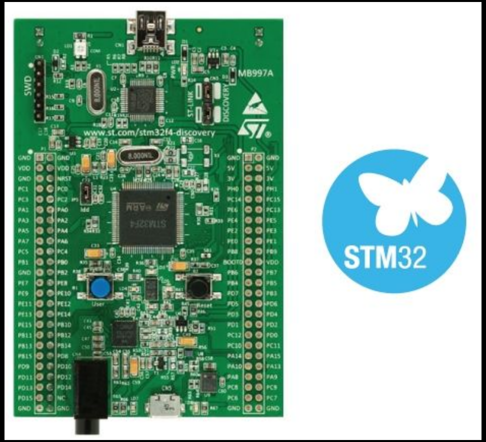
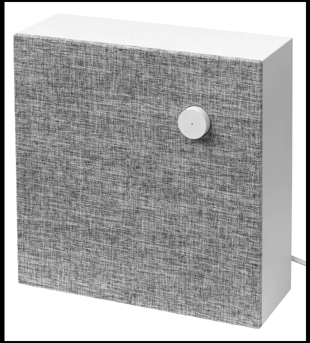
Introduction to reverse engineering deeply embedded devices

Benjamin Kollenda

r2con 2020, 03.09.2020

Deeply embedded devices





Embedded essentials

- No dynamic loader, no environment setup
- Architecture/MCU determines start instruction
- Very different memory types in same address space:
  - Flash/ROM
  - RAM
  - EEPROM
  - Peripherals
  - Co-processors
  - External memory
- Interrupts for async events

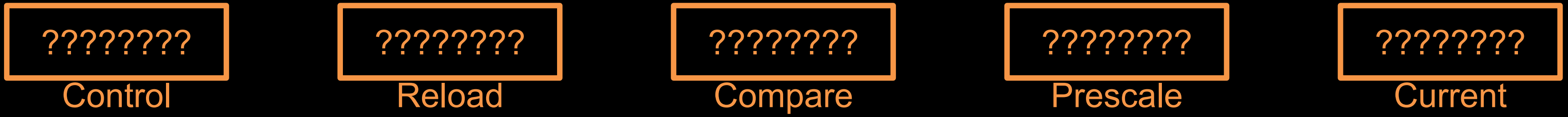
Embedded essentials

Peripherals as API

- Interaction with outside world (bus, network, …)
- Interaction with chip services (timer, DMA, accelerator, …)
- Represented as read/write memory address
- Write command to address, read back response

Embedded essentials

Peripherals as API - Timer example



Embedded essentials

Peripherals as API - Timer example

00000000	????????	????????	????????	????????
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0

Embedded essentials

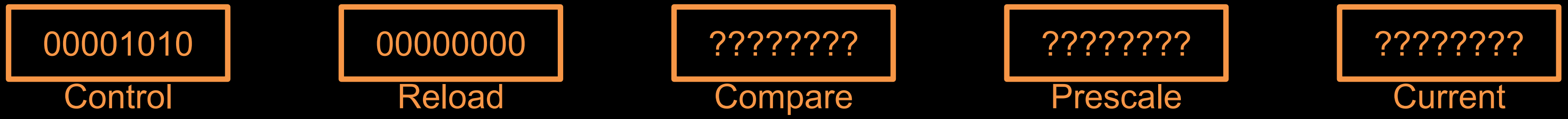
Peripherals as API - Timer example

00001010	????????	????????	????????	????????
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control |= 0x1010

Embedded essentials

Peripherals as API - Timer example



00001010	00000000	????????	????????	????????
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control \|= 0x1010
3. Set Reload: Reload = 0x0

Embedded essentials

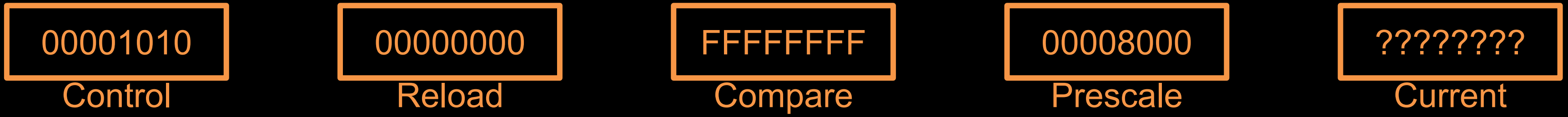
Peripherals as API - Timer example

00001010	00000000	FFFFFFFF	????????	????????
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control \|= 0x1010
3. Set Reload: Reload = 0x0
4. Set Compare to max: Compare = 0xFFFFFFFF

Embedded essentials

Peripherals as API - Timer example



00001010
Control

00000000
Reload

FFFFFFFF
Compare

00008000
Prescale

????????
Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control |= 0x1010
3. Set Reload: Reload = 0x0
4. Set Compare to max: Compare = 0xFFFFFFFF
5. Set Prescale: Prescale = 0x8000

Embedded essentials

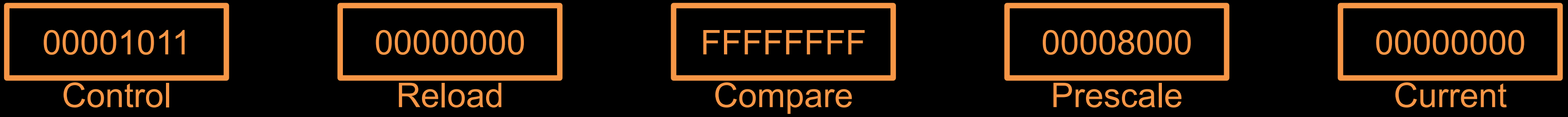
Peripherals as API - Timer example

00001010	00000000	FFFFFFFF	00008000	00000000
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control |= 0x1010
3. Set Reload: Reload = 0x0
4. Set Compare to max: Compare = 0xFFFFFFFF
5. Set Prescale: Prescale = 0x8000
6. Reset current value: Current = 0x0

Embedded essentials

Peripherals as API - Timer example

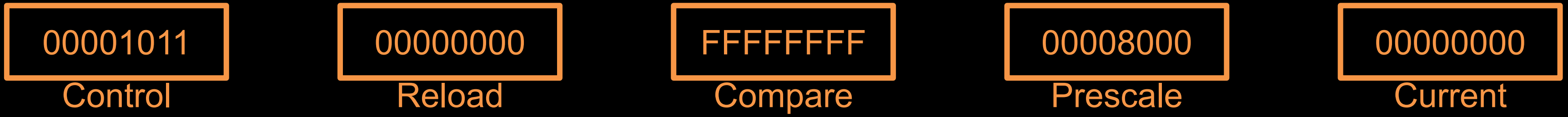


00001011	00000000	FFFFFFFF	00008000	00000000
Control	Reload	Compare	Prescale	Current

1. Stop timer: Control = 0x0
2. Select direction and source: Control |= 0x1010
3. Set Reload: Reload = 0x0
4. Set Compare to max: Compare = 0xFFFFFFFF
5. Set Prescale: Prescale = 0x8000
6. Reset current value: Current = 0x0
7. Start timer: Control |= 0x1

Embedded essentials

Peripherals as API - Timer example



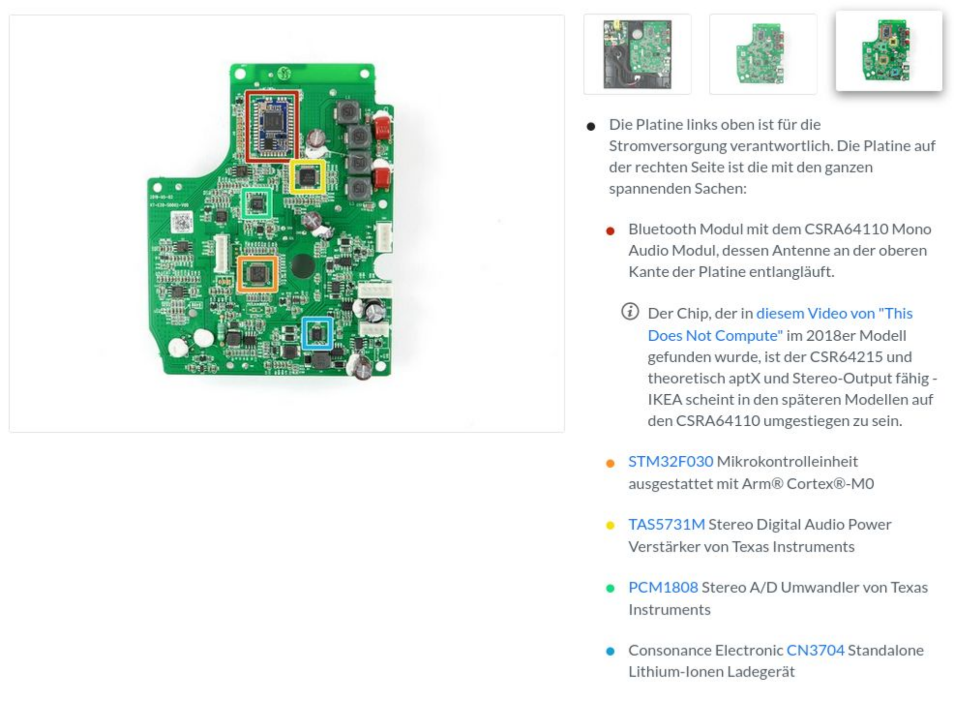
00001011	00000000	FFFFFFFF	00008000	00000000
Control	Reload	Compare	Prescale	Current

- Source: 32768 Hz clock
- Prescale: 1/32768
- -> Increment once every second
- -> Current contains the number of seconds since start

Step 0: Information Gathering

Hardware

- MCU model
- Teardowns



- Die Platine links oben ist für die Stromversorgung verantwortlich. Die Platine auf der rechten Seite ist die mit den ganzen spannenden Sachen:
  - 🔴 Bluetooth Modul mit dem CSRA64110 Mono Audio Modul, dessen Antenne an der oberen Kante der Platine entlangläuft.

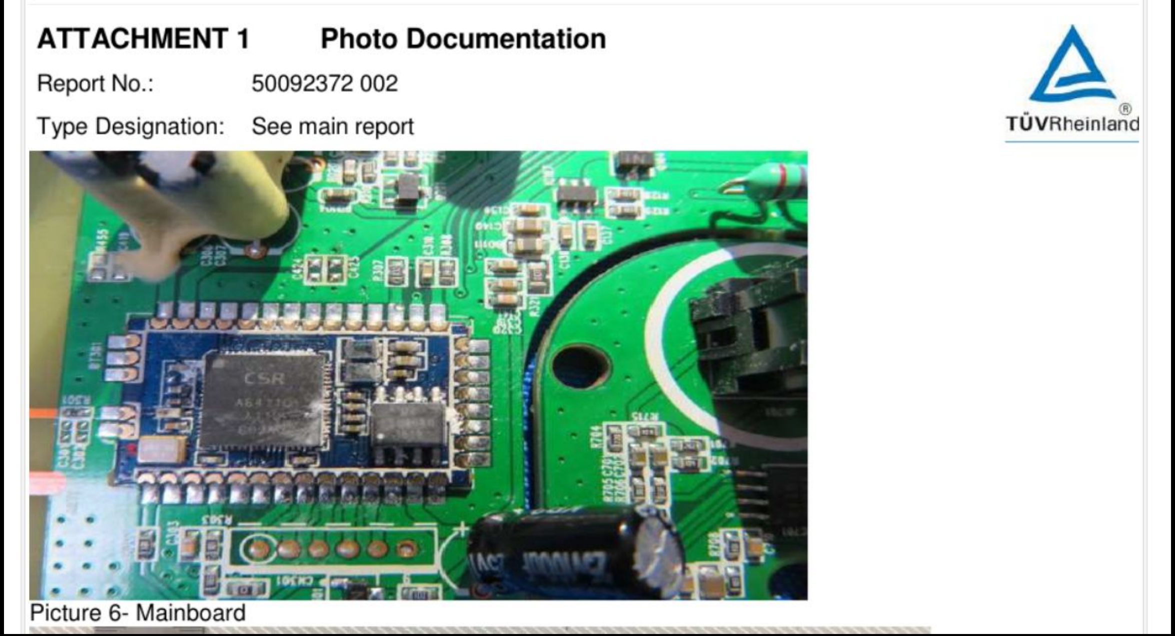
    ⓘ Der Chip, der in diesem Video von "This Does Not Compute" im 2018er Modell gefunden wurde, ist der CSR64215 und theoretisch aptX und Stereo-Output fähig - IKEA scheint in den späteren Modellen auf den CSRA64110 umgestiegen zu sein.
  - 🟠 STM32F030 Mikrokontrolleinheit ausgestattet mit Arm® Cortex®-M0
  - 🟡 TAS5731M Stereo Digital Audio Power Verstärker von Texas Instruments
  - 🟢 PCM1808 Stereo A/D Umwandler von Texas Instruments
  - 🔵 Consonance Electronic CN3704 Standalone Lithium-Ionen Ladegerät

https://de.ifixit.com/Teardown/IKEA+Lautsprecher+Teardown/130948

Step 0: Information Gathering

Hardware

- MCU model
- Teardowns
- FCC for RF devices



ATTACHMENT 1 **Photo Documentation**

Report No.: 50092372 002

Type Designation: See main report

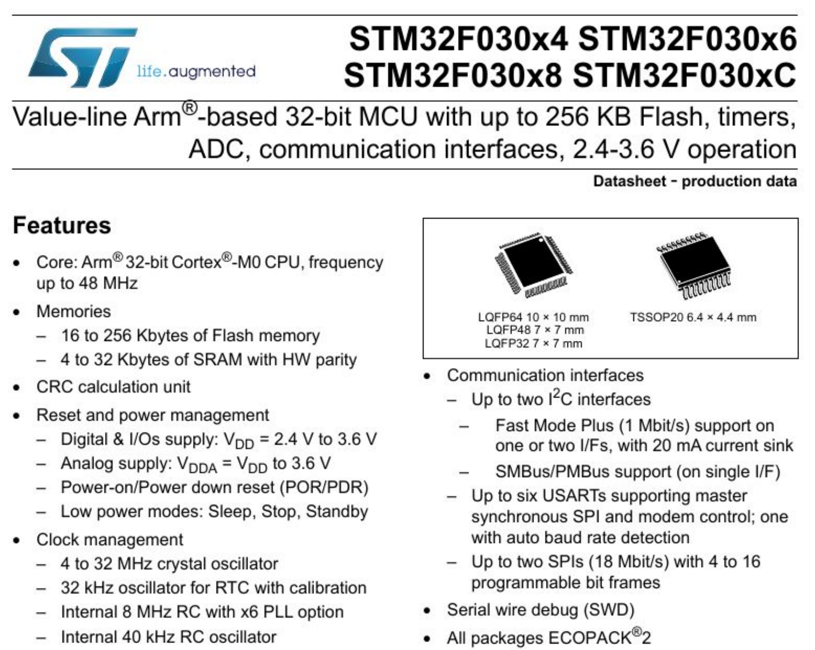
Picture 6- Mainboard

https://fccid.io/FHO-E1730/Internal-Photos/Internal-Photos-3627246

Step 0: Information Gathering

Documentation

- Datasheet



life.augmented

STM32F030x4 STM32F030x6
STM32F030x8 STM32F030xC

Value-line Arm®-based 32-bit MCU with up to 256 KB Flash, timers, ADC, communication interfaces, 2.4-3.6 V operation

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M0 CPU, frequency up to 48 MHz
- Memories
  - 16 to 256 Kbytes of Flash memory
  - 4 to 32 Kbytes of SRAM with HW parity
- CRC calculation unit
- Reset and power management
  - Digital & I/Os supply: V_{DD} = 2.4 V to 3.6 V
  - Analog supply: V_{DDA} = V_{DD} to 3.6 V
  - Power-on/Power down reset (POR/PDR)
  - Low power modes: Sleep, Stop, Standby
- Clock management
  - 4 to 32 MHz crystal oscillator
  - 32 kHz oscillator for RTC with calibration
  - Internal 8 MHz RC with x6 PLL option
  - Internal 40 kHz RC oscillator

LQFP64 10 × 10 mm
LQFP48 7 × 7 mm
LQFP32 7 × 7 mm

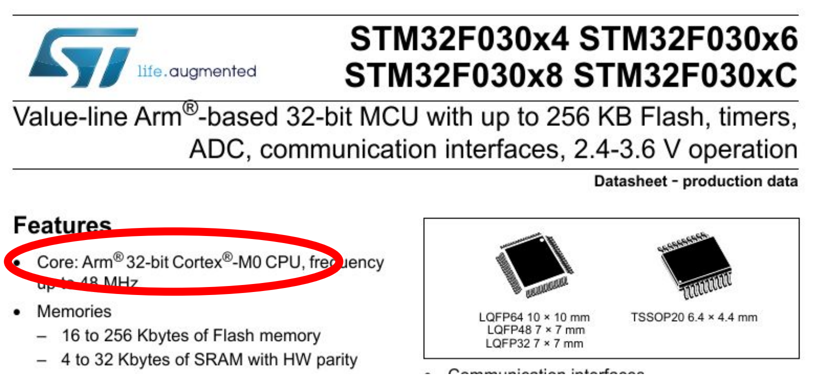
TSSOP20 6.4 × 4.4 mm

- Communication interfaces
  - Up to two I^2C interfaces
    - Fast Mode Plus (1 Mbit/s) support on one or two I/Fs, with 20 mA current sink
    - SMBus/PMBus support (on single I/F)
  - Up to six USARTs supporting master synchronous SPI and modem control; one with auto baud rate detection
  - Up to two SPIs (18 Mbit/s) with 4 to 16 programmable bit frames
- Serial wire debug (SWD)
- All packages ECOPACK®2

Step 0: Information Gathering

Documentation

- Datasheet
- Architecture



STM32F030x4 STM32F030x6
STM32F030x8 STM32F030xC

Value-line Arm®-based 32-bit MCU with up to 256 KB Flash, timers, ADC, communication interfaces, 2.4-3.6 V operation

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M0 CPU, frequency up to 48 MHz
- Memories
  - 16 to 256 Kbytes of Flash memory
  - 4 to 32 Kbytes of SRAM with HW parity

life.augmented
LQFP64 10 × 10 mm
LQFP48 7 × 7 mm
LQFP32 7 × 7 mm
TSSOP20 6.4 × 4.4 mm

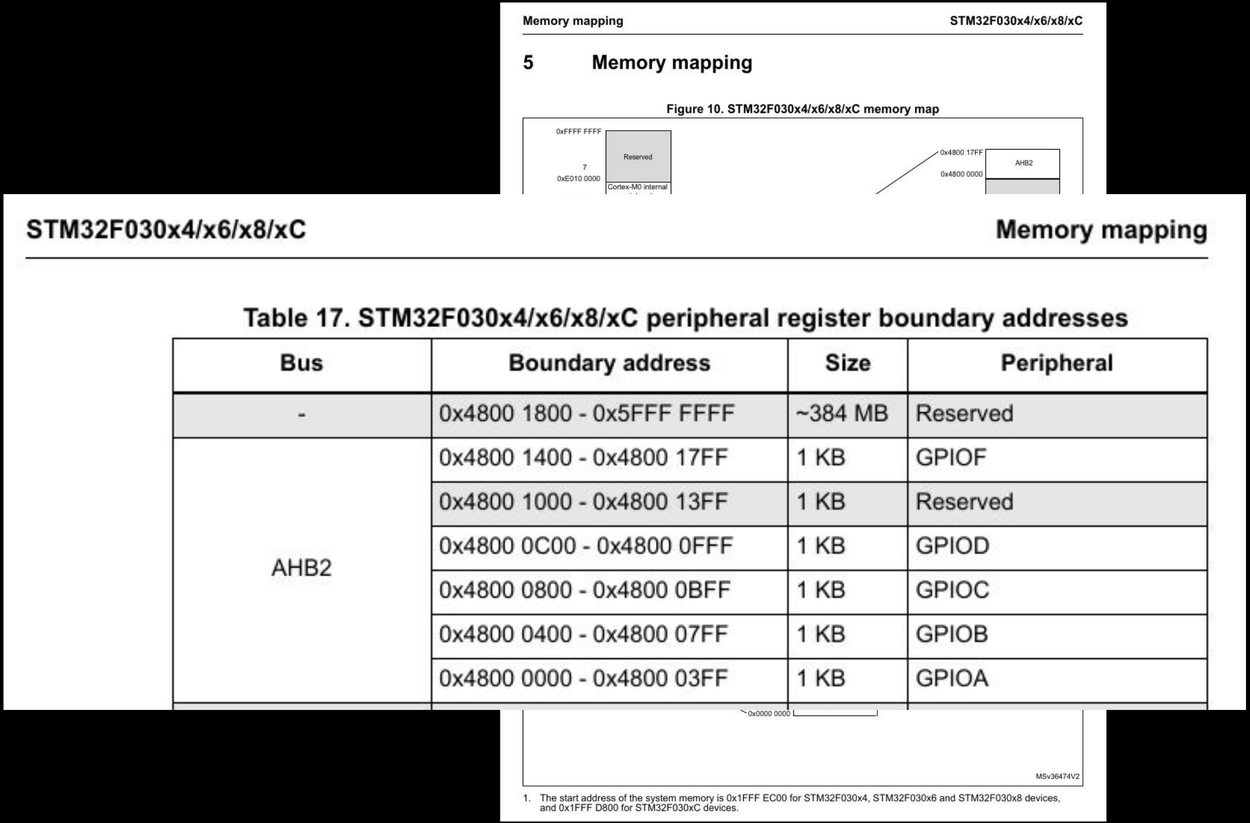
- Communication interfaces

ARM® v6-M Architecture
Reference Manual

Step 0: Information Gathering

Documentation

- Datasheet
- Architecture
- Memory Map



Memory mapping STM32F030x4/x6/x8/xC

5 Memory mapping

Figure 10. STM32F030x4/x6/x8/xC memory map

1. The start address of the system memory is 0x1FFF EC00 for STM32F030x4, STM32F030x6 and STM32F030x8 devices, and 0x1FFF D800 for STM32F030xC devices.

STM32F030x4/x6/x8/xC Memory mapping

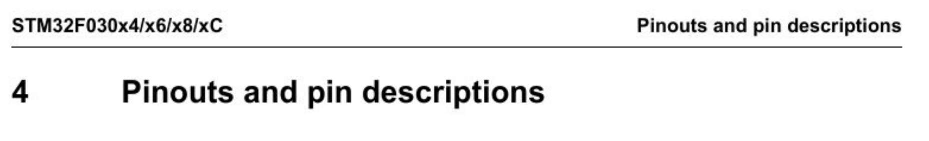
Table 17. STM32F030x4/x6/x8/xC peripheral register boundary addresses

Bus	Boundary address	Size	Peripheral
-	0x4800 1800 - 0x5FFF FFFF	~384 MB	Reserved
AHB2	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF
	0x4800 1000 - 0x4800 13FF	1 KB	Reserved
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB
	0x4800 0000 - 0x4800 03FF	1 KB	GPIOA

Step 0: Information Gathering

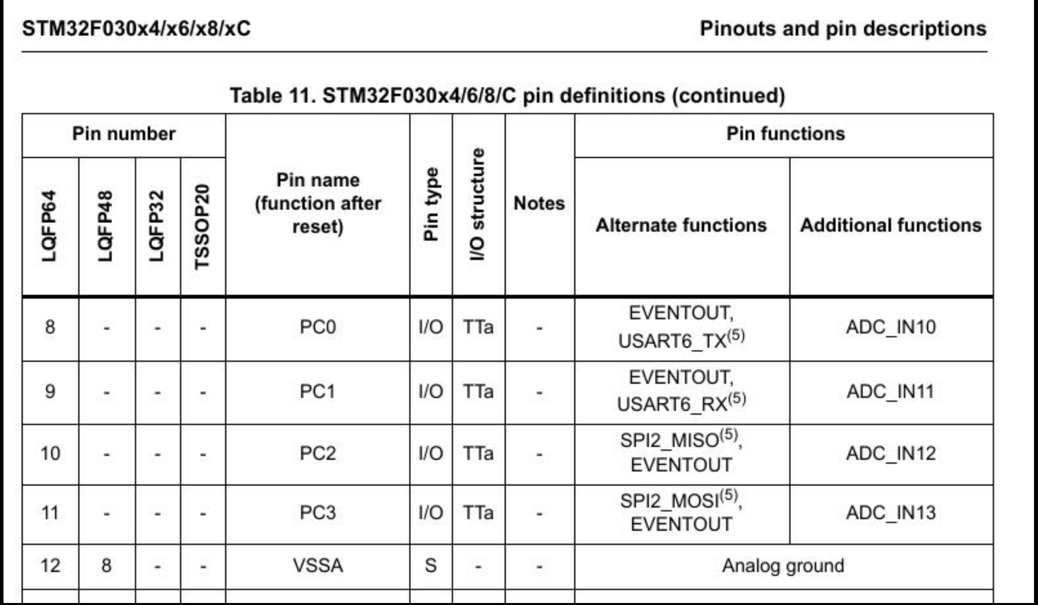
Documentation

- Datasheet
- Architecture
- Memory Map
- Pinout



STM32F030x4/x6/x8/xC — Pinouts and pin descriptions

4 Pinouts and pin descriptions



STM32F030x4/x6/x8/xC — Pinouts and pin descriptions

Table 11. STM32F030x4/6/8/C pin definitions (continued)

Pin number				Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
LQFP64	LQFP48	LQFP32	TSSOP20					Alternate functions	Additional functions
8	-	-	-	PC0	I/O	TTa	-	EVENTOUT, USART6_TX[5]	ADC_IN10
9	-	-	-	PC1	I/O	TTa	-	EVENTOUT, USART6_RX[5]	ADC_IN11
10	-	-	-	PC2	I/O	TTa	-	SPI2_MISO[5], EVENTOUT	ADC_IN12
11	-	-	-	PC3	I/O	TTa	-	SPI2_MOSI[5], EVENTOUT	ADC_IN13
12	8	-	-	VSSA	S	-	-	Analog ground	

Step 0: Information Gathering

Documentation

- Datasheet
- Architecture
- Memory Map
- Pinout
- SDK

Step 1: Firmware Extraction

- OTA updates
- Firmware dumps
- External storage
- Serial
- Debugger
- Paid services

Step 2: Analysis

Entry Point

- Determine address (check your manuals)
- Entry point usually a short assembly stub
- Last branch usually goes to main()
- Check the SDK for the assembly stub

Demo

Step 2: Analysis

Main()

- Common pattern: main loop
- Setup followed by endless loop
- Polls peripherals, processes data, writes to peripherals

Demo

Step 2: Analysis

Peripherals

- Interaction with the real world
- Know your address regions
- Provide important context
- Check the init code

Demo

Step 2: Analysis

Interrupts

- Async events (e.g. timer)
- Registered in vector table
- Commonly some stub entries with same address
- Real ISRs are implemented in C, define as function

Demo

Step 2: Analysis

Debugging

- Helps with complex dataflow
- Check peripheral state after init
- Track access to peripheral
- Well supported on ARM devices:
    - J-Link/ST-Link
    - Openocd
    - GDB

Demo

Conclusion

- Different setting, but core skills transfer to embedded devices
- Information collection is vital and saves you a lot of time
- Take apart some devices and write about what you find :)

- Also currently looking for new opportunities

benjamin.kollenda@rub.de