Real-Time Systems

Exercise #1

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

Laboratory assignment

Organization:

The schedule

- The laboratory sessions run weekly, during the period from study week 2 to study week 7
- Each project group is guaranteed one session per week
  (extra sessions may be requested in later half of study period)

The parts

- The assignment consists of Part 0, Part 1 and Part 2
- Part 0 is expected to take 1 session to complete*
- Part 1 is expected to take 1−2 sessions to complete*
- Part 2 is expected to take 3−4 sessions to complete*

(*Assuming you have prerequisite knowledge for the course)

Laboratory assignment

Basic prerequisites for approval:

Respecting the Rules of Conduct (see separate document)

- Contribute to your group (be present, be active)
- Respect the lab sessions (be on time, follow instructions)
- Respect the deadlines (submit report in time)
- Refrain from cheating (no use of other groups' code)

Sufficient lab session attendance

- To guarantee sufficient interaction each project group is expected to attend at least four laboratory sessions, the first attendance being in study week 2.
- To allow for a fair grading both project group members must be present at each session the group attends.

Laboratory assignment

Grading of the 'Laboratory' course element:

The grade (U, 3, 4, 5) will reflect your practical skills at the laboratory sessions as well as your presentation skills.

The grade is determined by the following:

- The quality of laboratory performance
  - sub-score is awarded* based on a set of four criteria
  - sub-score sets a preliminary grade
- The quality of project report
  - sub-score is awarded* based on a set of three criteria
  - sub-score can potentially adjust the final grade

(* See corresponding modules in Canvas for details)

Laboratory assignment

Laboratory performance sub-score criteria:

Implementation

- How many of the coding challenges in Part 2 that you can successfully implement and demonstrate.

Design

- How well you know the design and behavior of your code.

Debugging

- How well you identify, and solve, problems with your code.

Paradigm

- How well you understand, and make use of, the concurrent, object-oriented, reactive, timing-aware programming method.

Laboratory assignment

Project report sub-score criteria:

English language

- How well you handle vocabulary, spelling and grammar.

Document structure

- To what degree you include requested document sections.
- How well you handle the use of headings, sub-headings, paragraphing and sentence structure.

Technical content

- To what degree you include descriptive text and illustrations for your implementation of selected challenges in Part 2.
- To what degree you include argumentation for, and reflection on, your implementation of selected challenges in Part 2.

Laboratory assignment – Part 0

Getting started:

- Compile the template code using the cross compiler
- Upload the machine code to the target computer

Interacting with the target computer:

- Take input from the workstation's keyboard
- Generate output to the workstation's console window

Preparatory work for Part 1 and Part 2:

- Pre-compute periods for all tones that will be played
- Prepare data structures to allow a melody to be transposed to different keys

Laboratory assignment – Part 1

Tone generator:

- Generate a 1 kHz tone (square wave signal) and output it to the audio jack on the target computer

Background load:

- Add a background task with a scalable load
- Experiment: disturb tone generator by increasing the load
- Repeat the experiment with deadline scheduling enabled

Worst-case execution times:

- Measure the execution times of the program code in the tone generator task and the background load task

Laboratory assignment – Part 2

Single instrument: [each group individually]

- Capable to play tones in a 12-tone scale in different keys
- Capable to play the melody “Brother John”, and be able to change key and/or tempo while playing

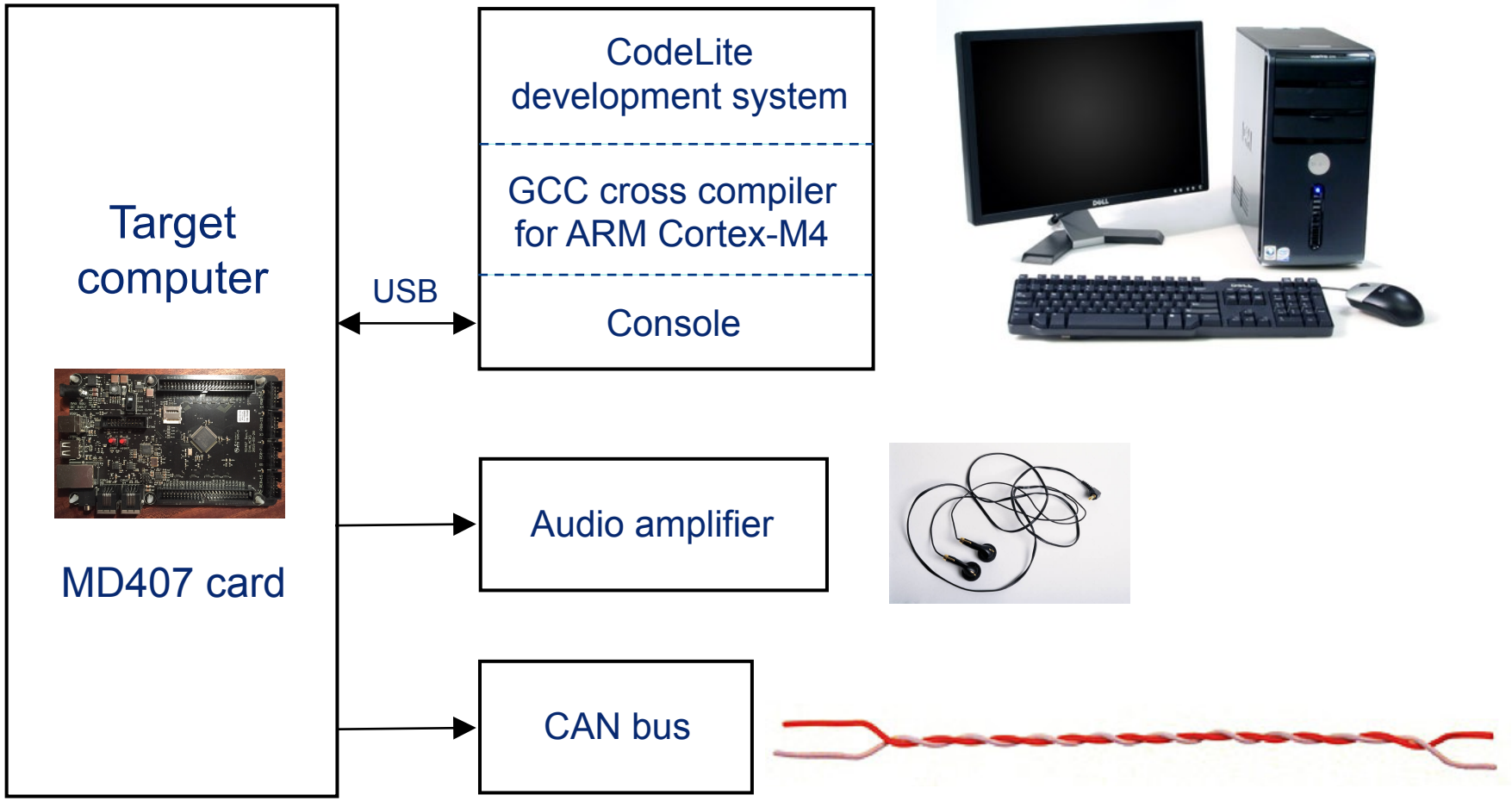
Basic orchestra: [collectively, with two or more computers]

- Capable to play “Brother John” in chorus form (unison), with one target computer being the orchestra leader (conductor)
- Conductor should be able to set initial key and tempo

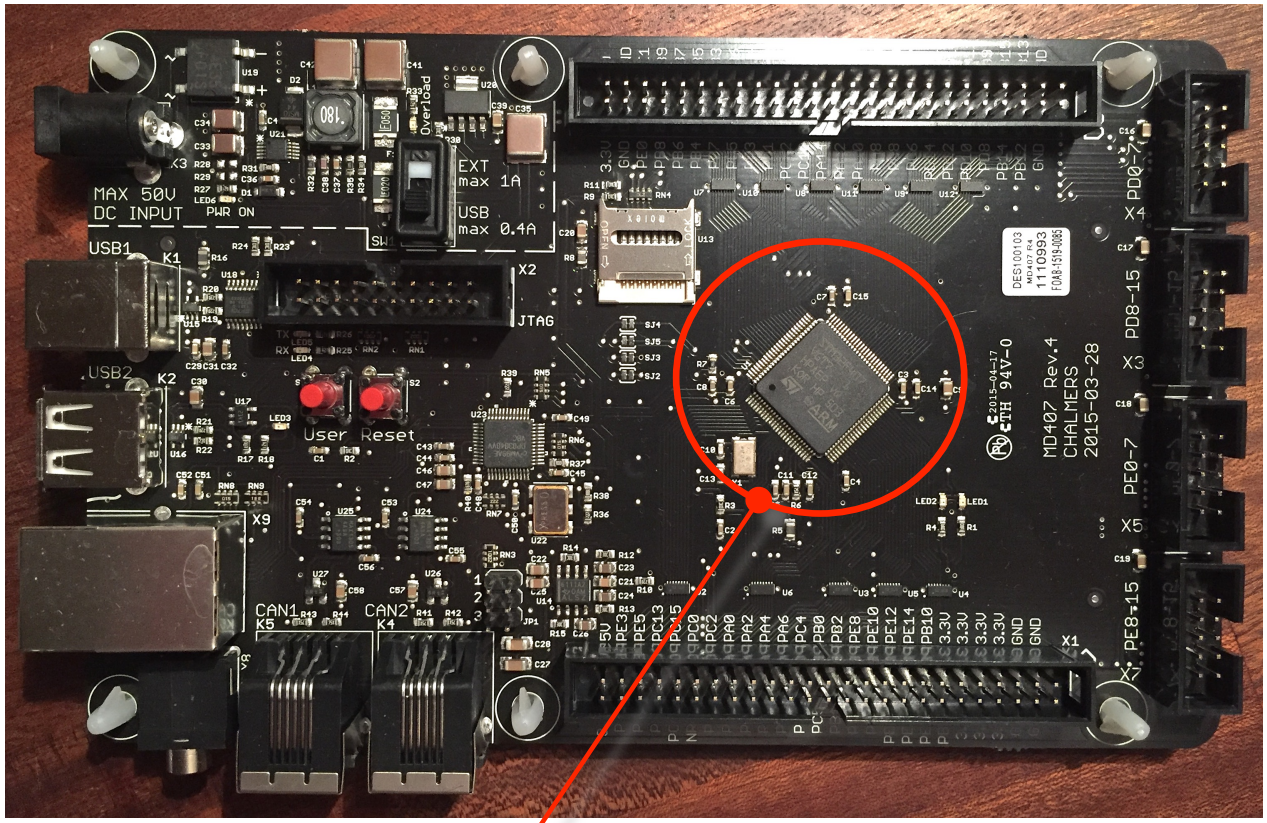
Advanced orchestra: [collectively]

- Play in canon form, with conductor role as above
- Conductor should be able to change tempo dynamically

Laboratory assignment – Setup

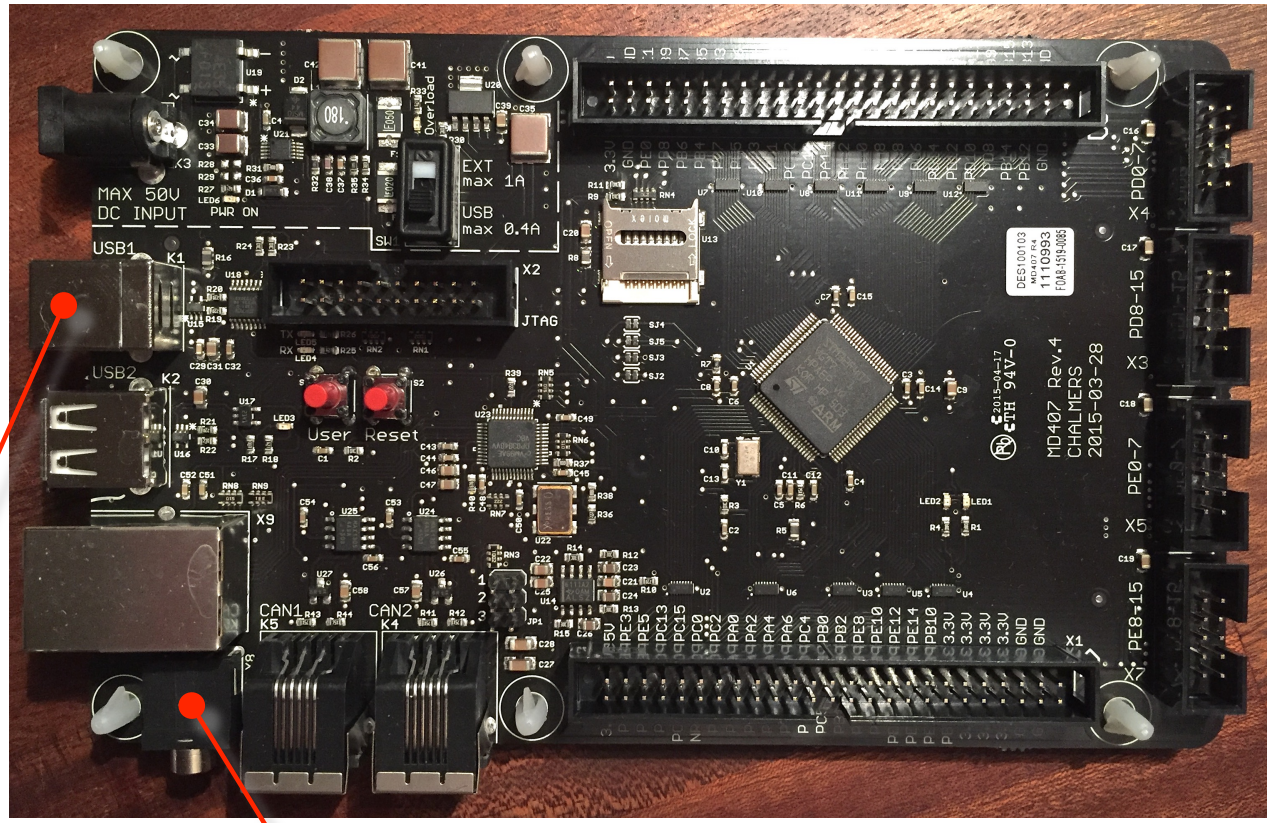


The MD407 computer card



STMicroelectronics' STM32F407 microcontroller /w ARM Cortex-M4 core

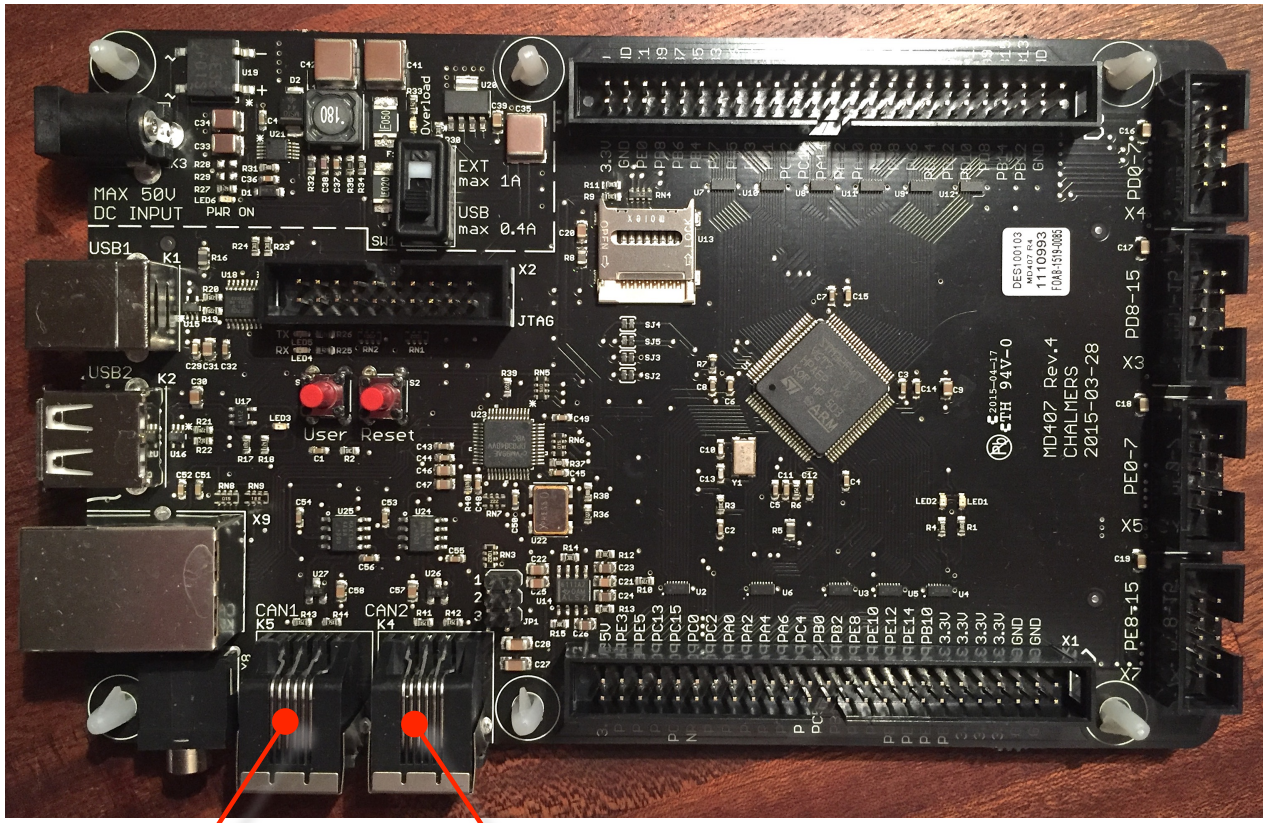
The MD407 computer card



USB debug port

Audio output jack

The MD407 computer card



Main CAN bus

Loopback CAN bus

The MD407 computer card

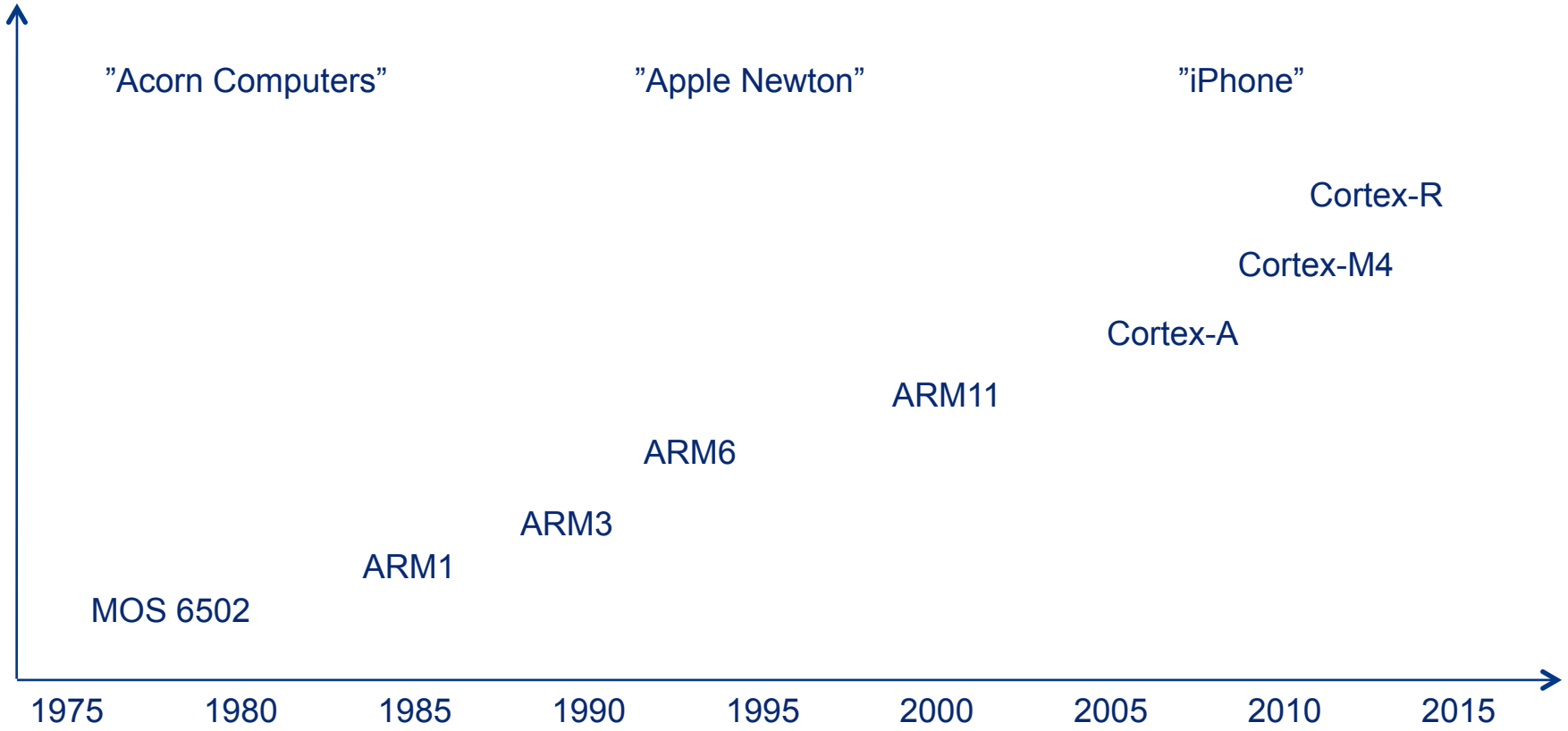
The STM32F407 microcontroller:

Based on the ARM Cortex-M4 processor core

- 168 MHz processor clock
- 32-bit registers (data and address)
- 16-bit instructions (Thumb)
- 1 MB of Flash memory (for resident monitor/debugger)
- 128 kB of RAM (for user programs)
- On-chip floating-point unit
- On-chip CAN modules, serial communications interfaces, parallel ports, digital-to-analog converters, high-resolution timers, ...

The MD407 computer card

The ARM processor family tree:



STM32F407 block diagram:

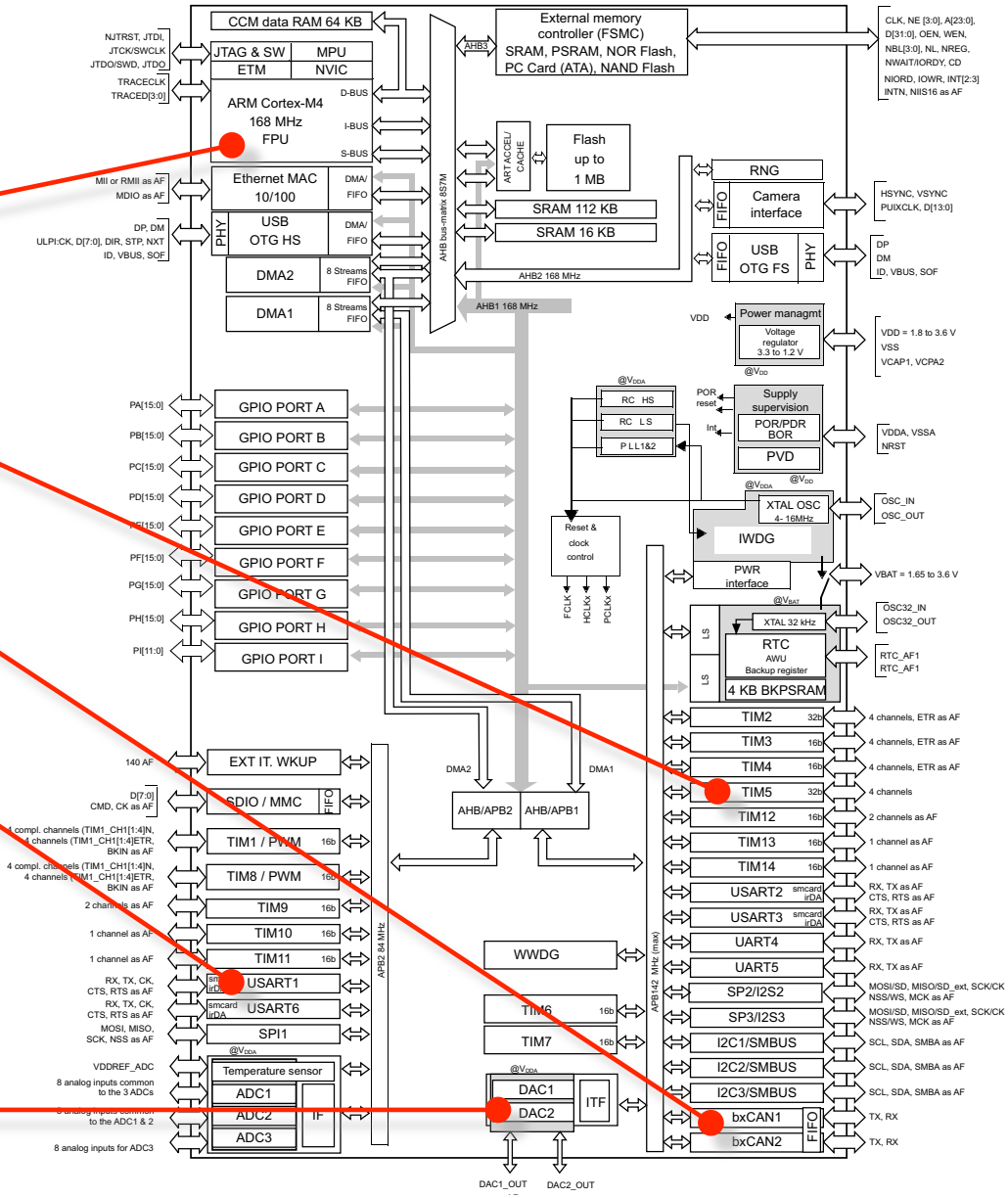
CPU – Cortex-M4 core

TIM5 – used by TinyTimber for high-resolution clock

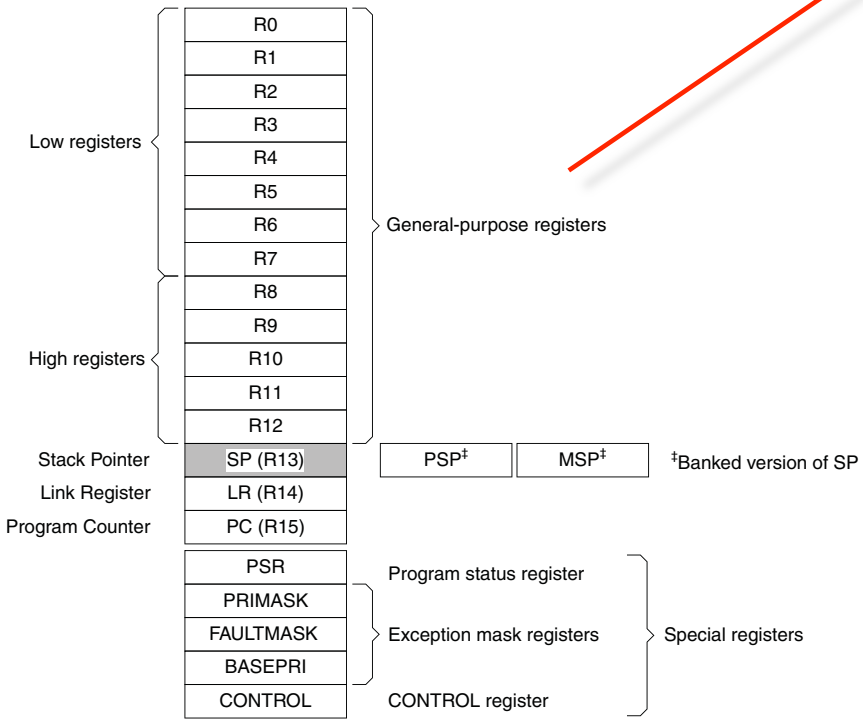
bxCAN1 – available for your connection to other cards

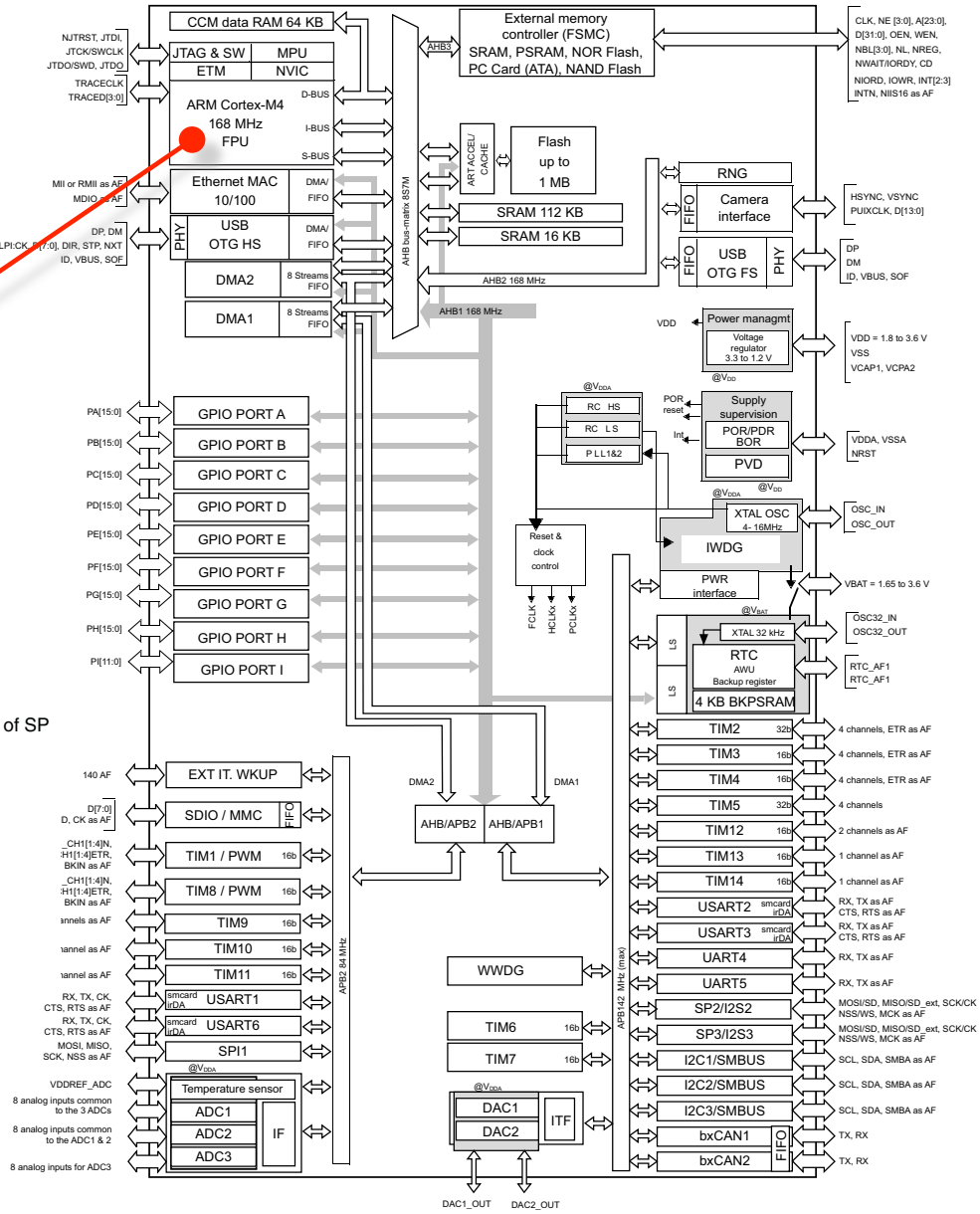
USART1– connects to console via USB debug port

DAC2 – available for your sound generation

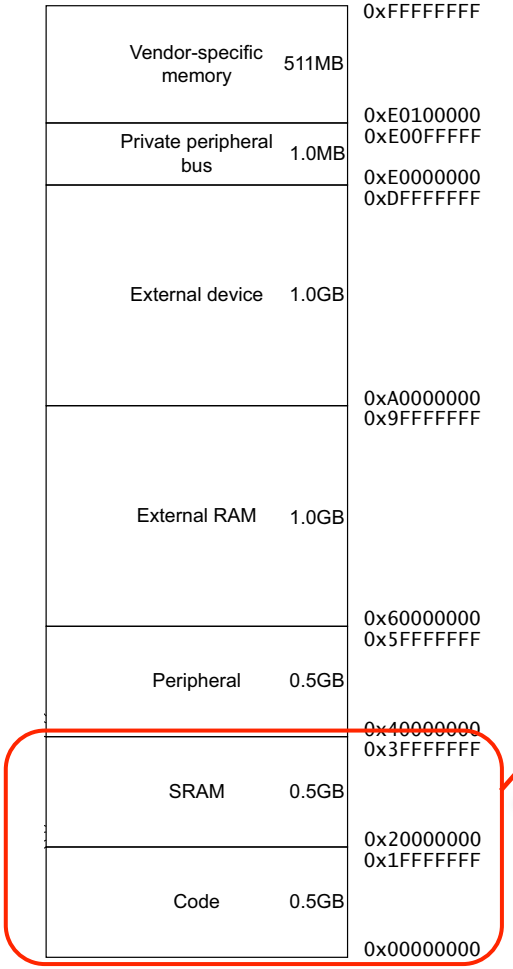


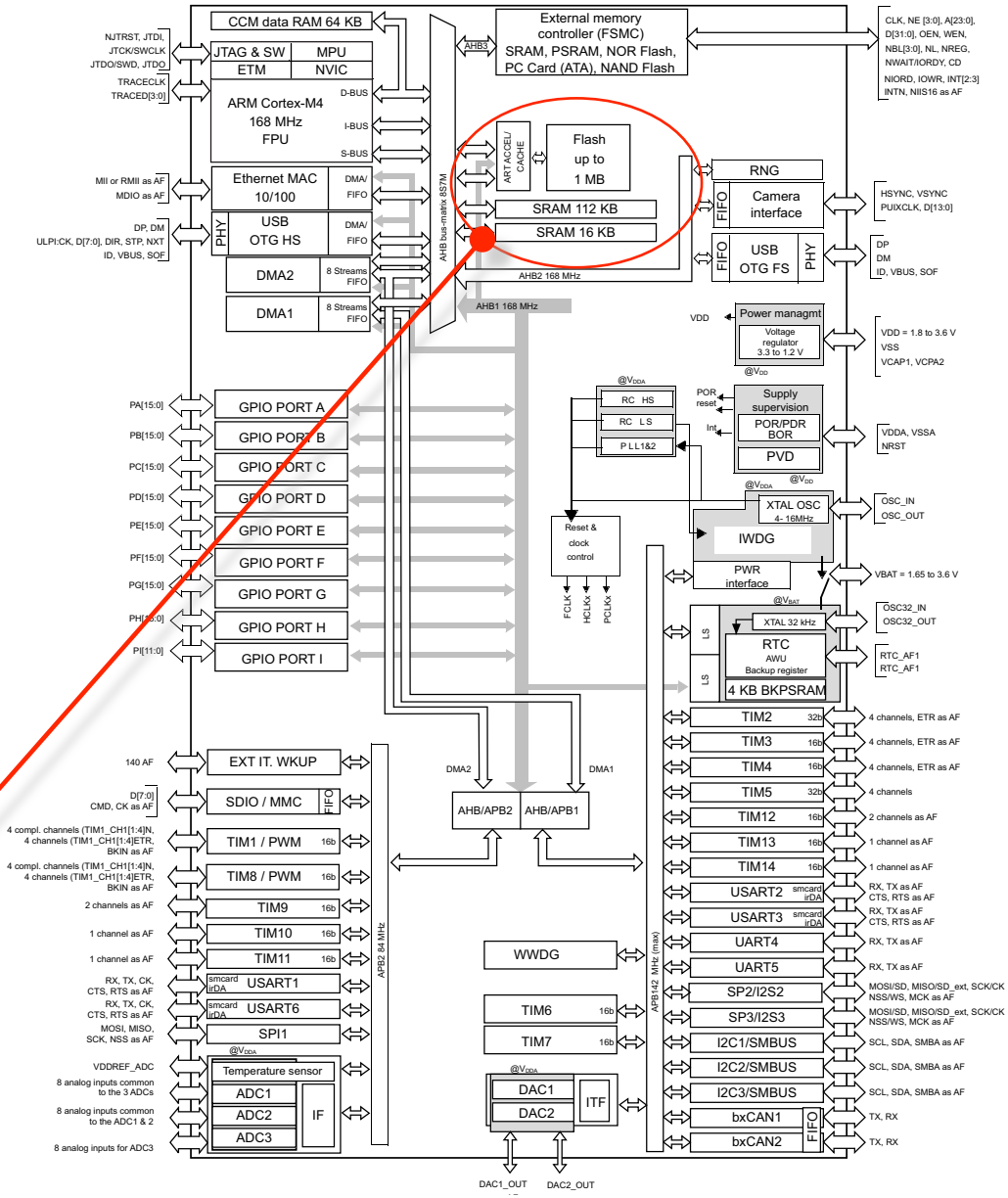
STM32F407 processor registers:





STM32F407 address space:





STM32F407 address space: