Lecture 15
Wrapup + Microprocessors

CE346 – Microcontroller System Design

Branden Ghena – Spring 2026

Some slides borrowed from:
Josiah Hester (Northwestern), Prabal Dutta (UC Berkeley)

Administrivia

- Project check-ins this Friday

- I can do a last round of orders this weekend
  - After that, anything but Amazon will take too long to arrive
  - Part of the check-ins: is there any remaining hardware you need?

- Project Demo Scheduling
  - Form link sent out through Ed
  - Half of the class from 11:00-1:30 and half from 2:00-4:30

  - Demos will be continuous during that period and open to the public!

More administrivia

- Last quiz today!
  - Tell your friends

Less administrivia

- You’re using pin numbers wrong in your project
  - Yes you
  - (maybe)

- Several groups forgot to switch to I2C_QWIIC_SDA (and SCL) when configuring their I2C

- Bigger issue: several groups are just using numbers as if they mean output pins
  - EDGE_P19 is NOT the same as 19
  - EDGE_P8 is NOT the same as 8
  - Use definitions in microbit_v2.h instead!!

Today’s Goals

- Discuss remaining parts of the Microbit and nRF52833
  - Realize that we’ve covered almost everything on the system!!

- Compare and contrast other microcontrollers and other similar hardware platforms

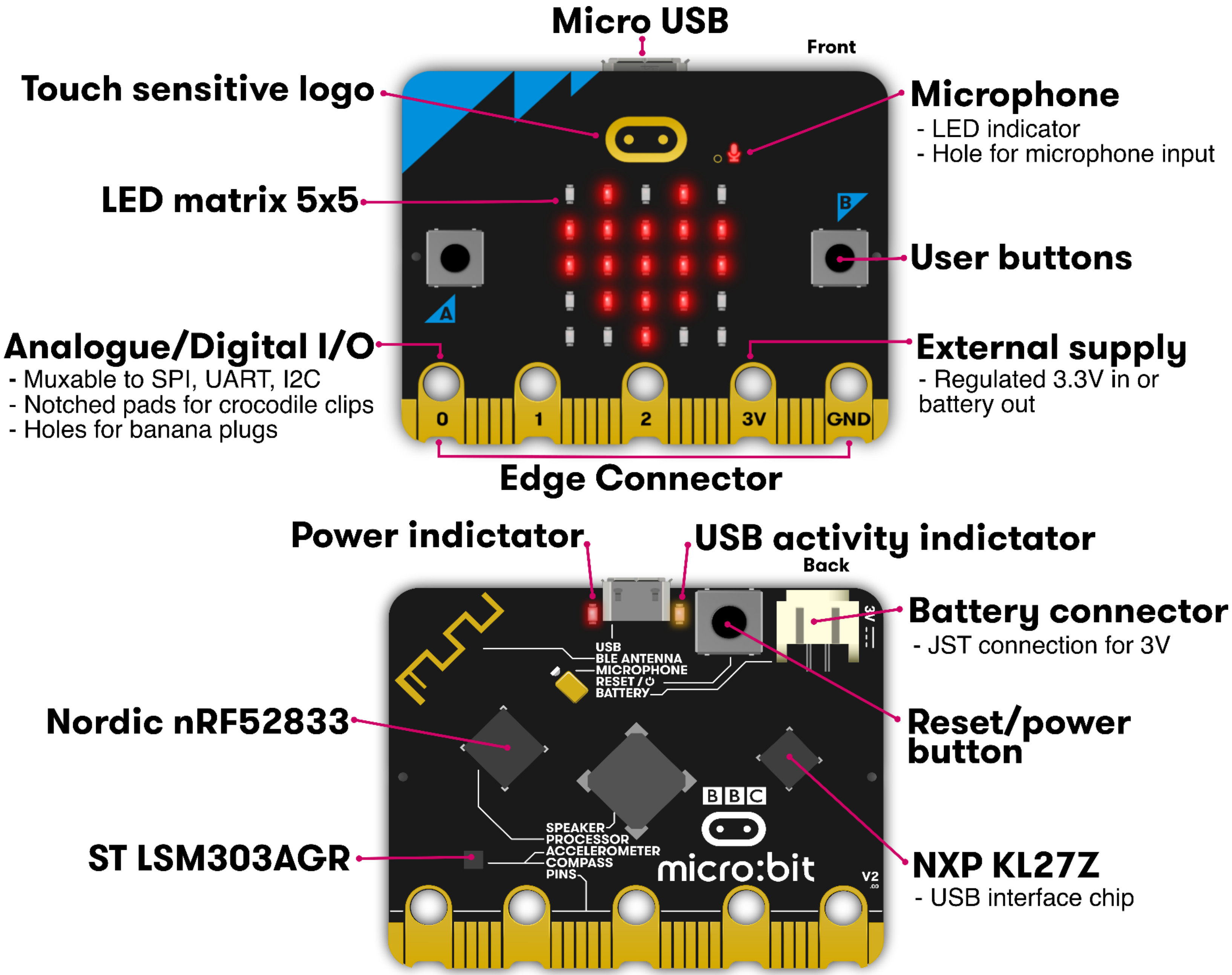
Outline

- **What haven't we talked about?**
  - **Microbit**
  - nRF52833
- Other hardware systems
  - Other microcontroller peripherals
  - Microprocessors

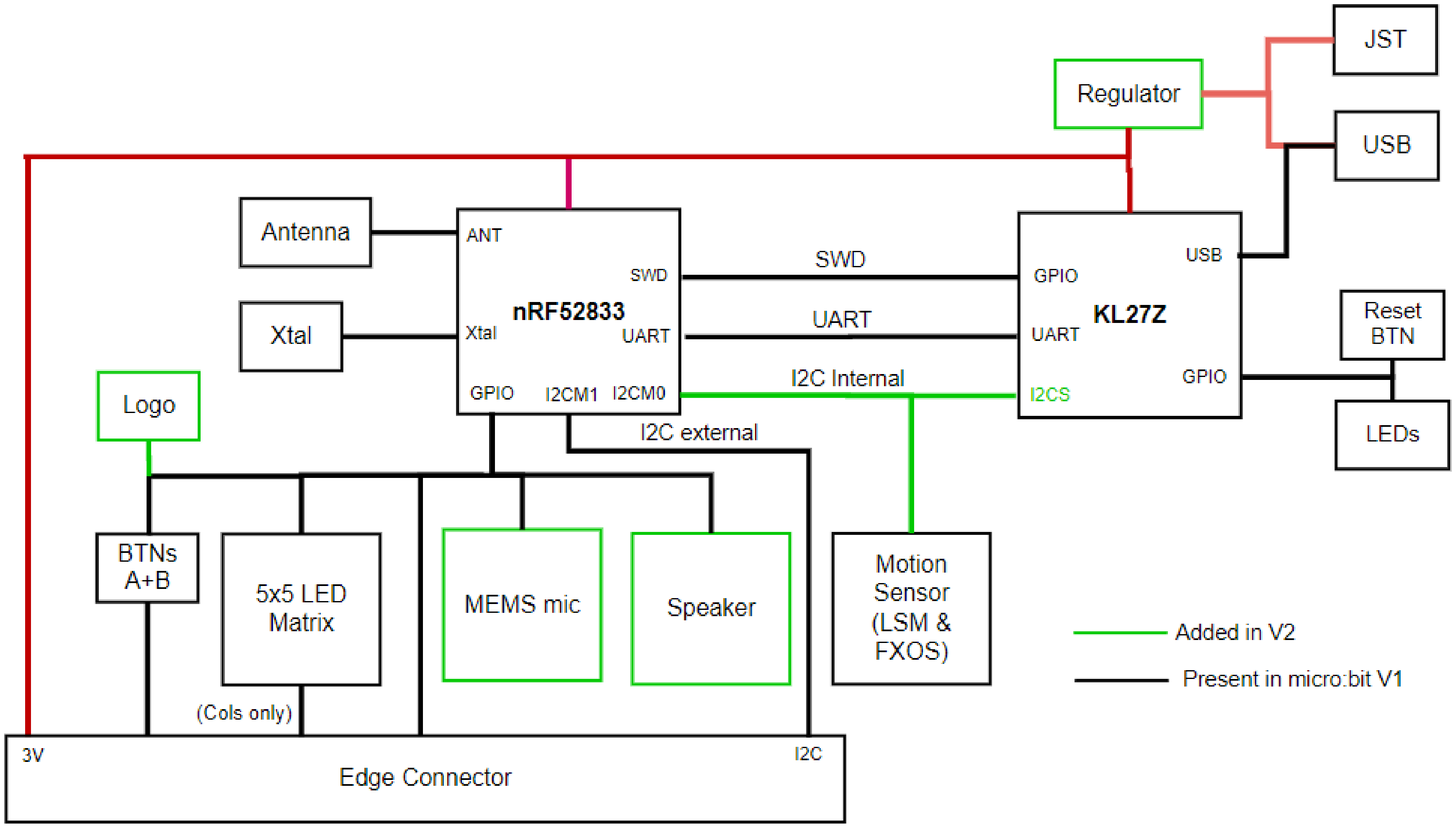
Microbit

- Used almost all of this!

- Remaining:
  - Batteries
  - Wireless
  - KL27Z I2C



Internal Microbit connections



KL27 I2C Interface

- Device information
  - Version of board and JTAG firmware
  - Power state of board
    - USB, Battery, both
    - Voltage values for battery and VIN
  - USB connection state
  - Disable the power LED!!

- Flash Storage
  - 128 kB of the KL27's Flash is readable/writable over I2C

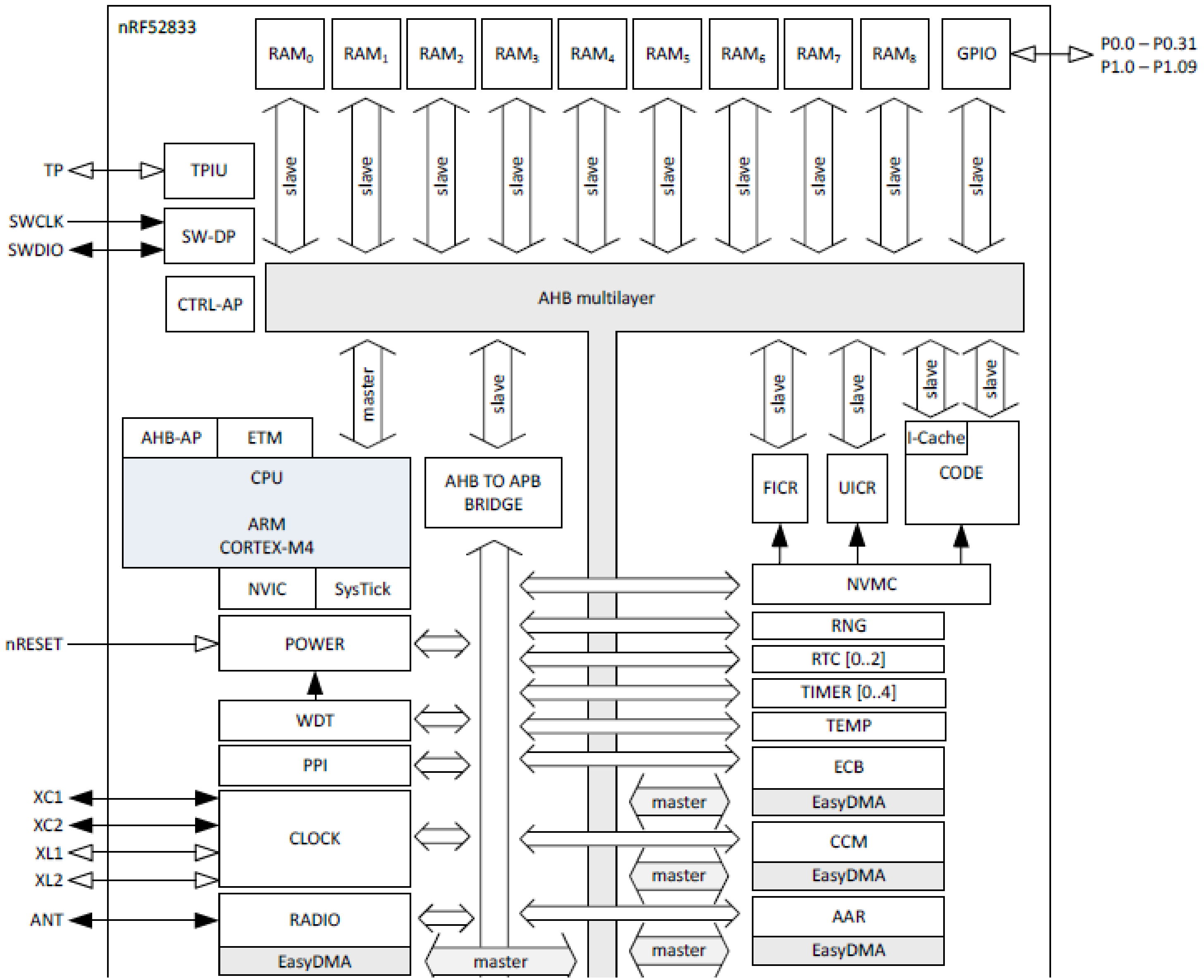
https://github.com/microbit-foundation/spec-i2c-protocol/blob/main/spec/index.md

Outline

- **What haven't we talked about?**
  - Microbit
  - **nRF52833**
- Other hardware systems
  - Other microcontroller peripherals
  - Microprocessors

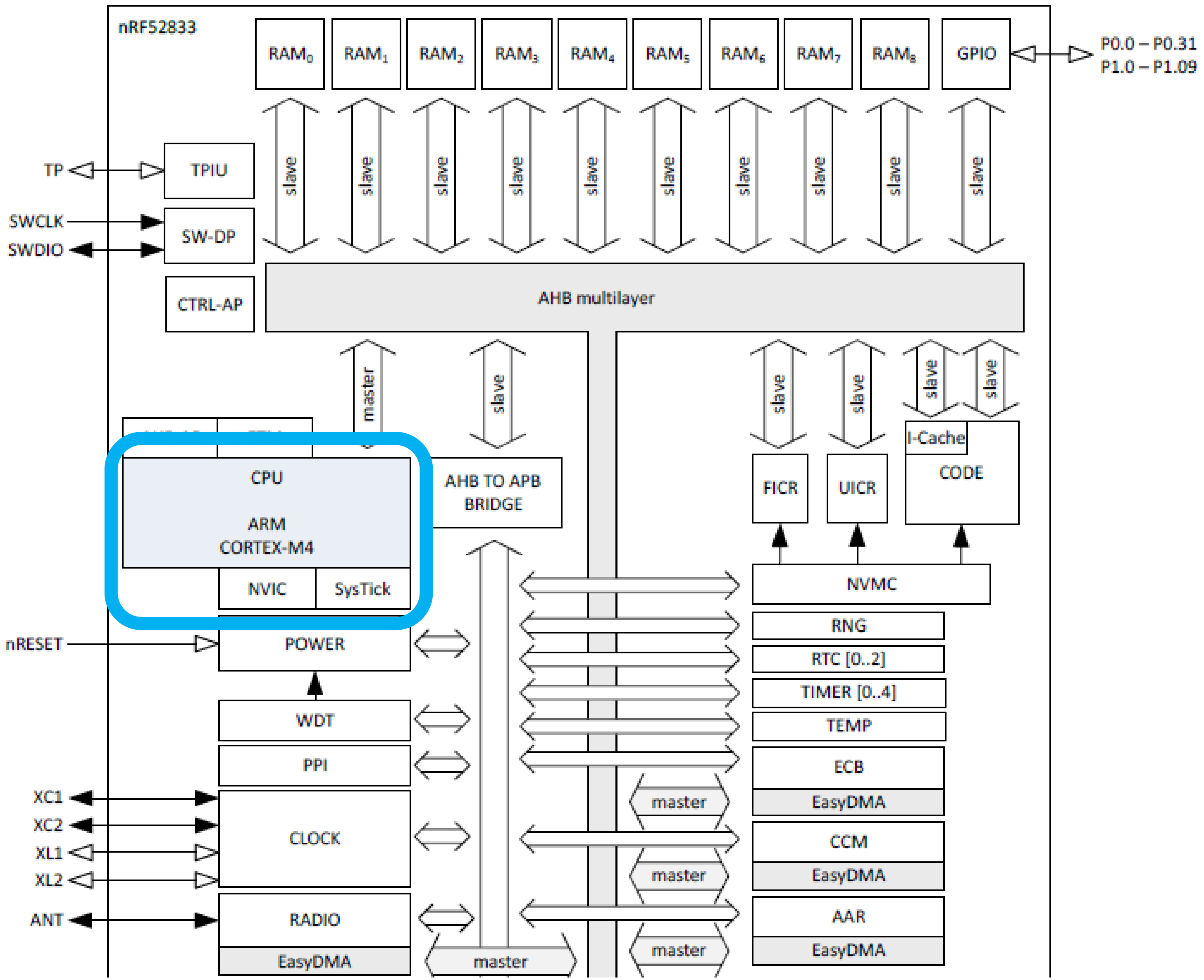
nRF52833 Peripherals

- Tour of the nRF52833 peripherals
- With some details on the ones we haven't talked about
  - Wireless
  - Crypto
  - Audio



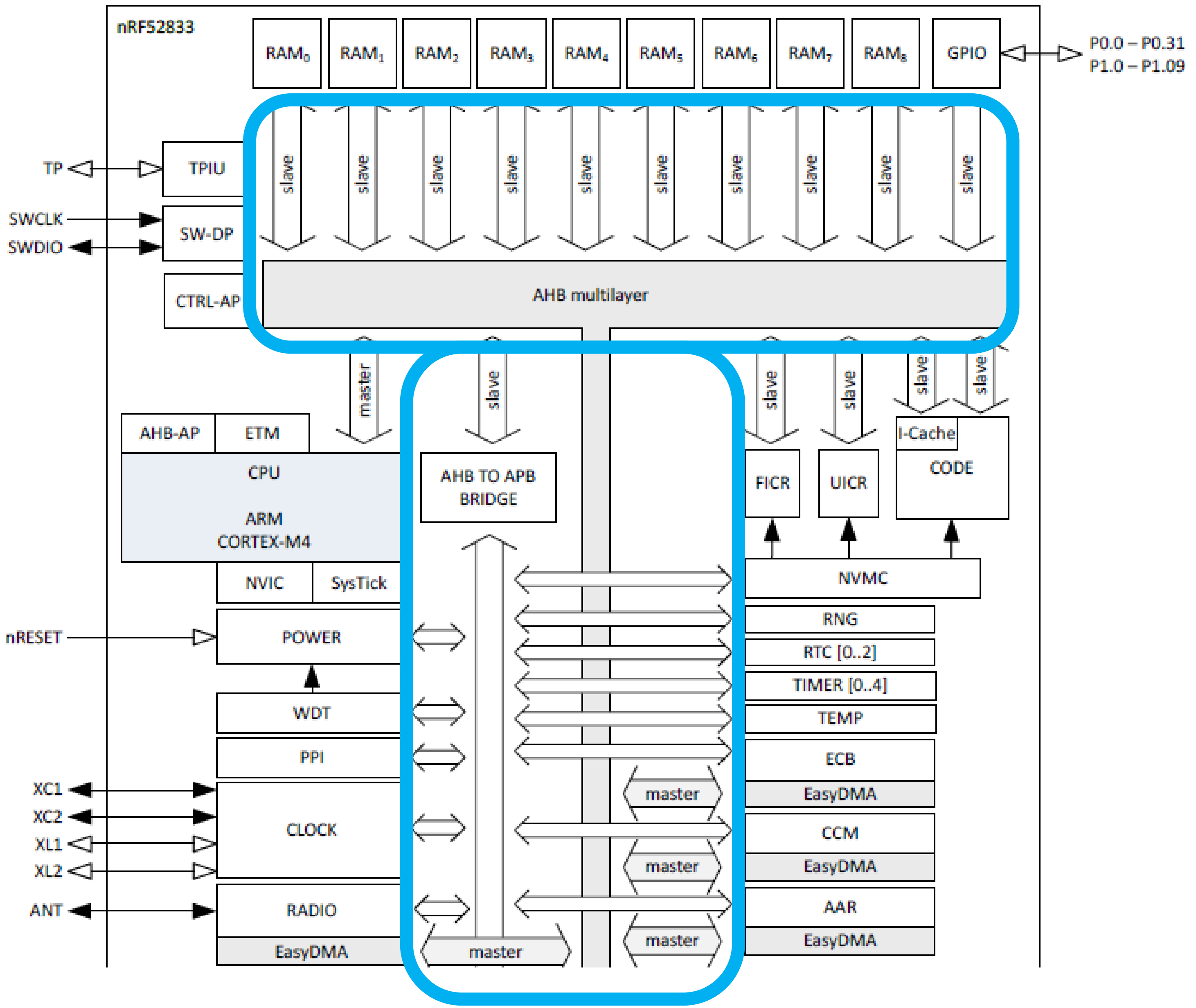
nRF52833 Peripherals

- Cortex-M4F processor
- 32-bit ARM core
- Floating point
- Includes Interrupt control and SysTick (an extra timer)



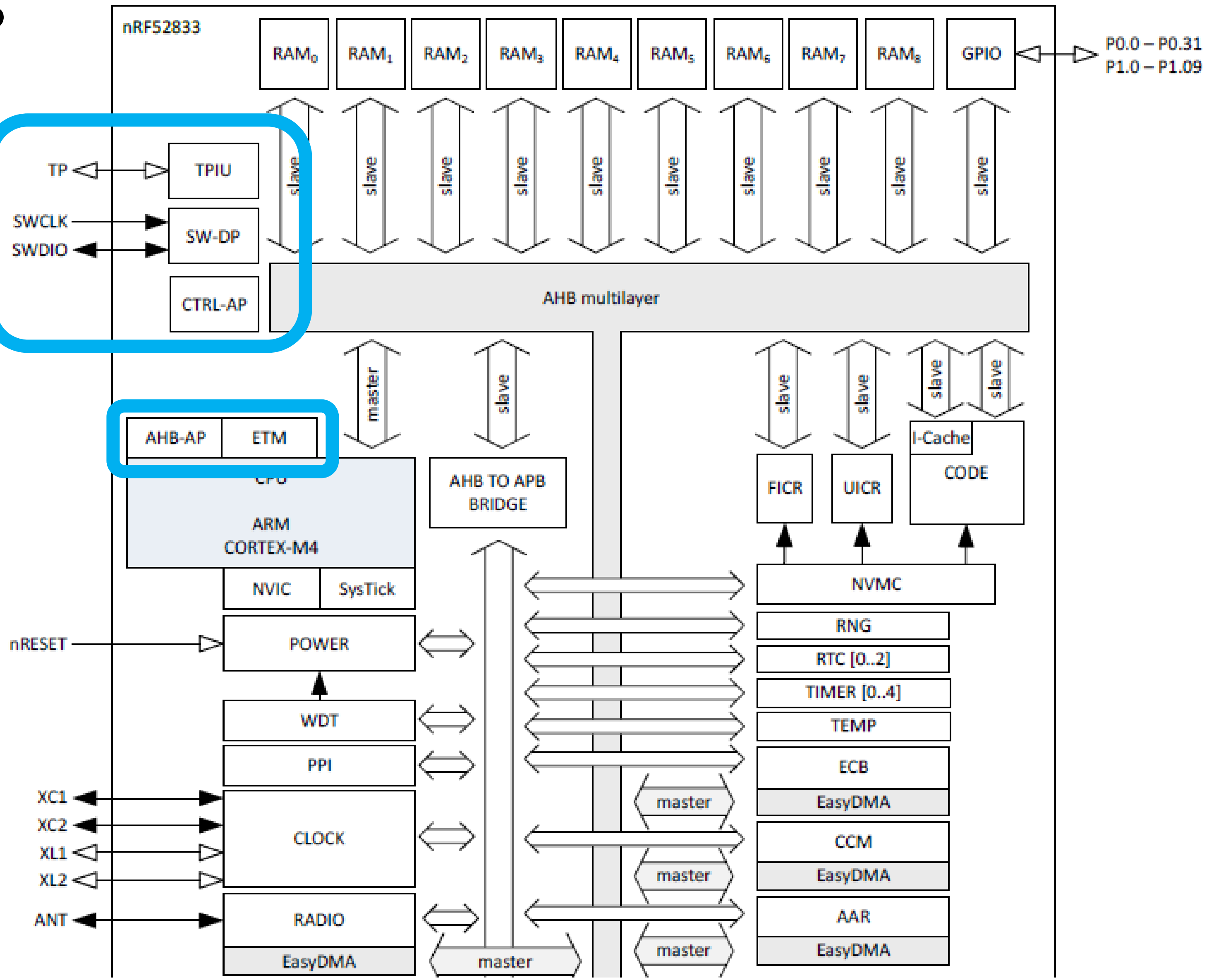
nRF52833 Peripherals

- Memory buses



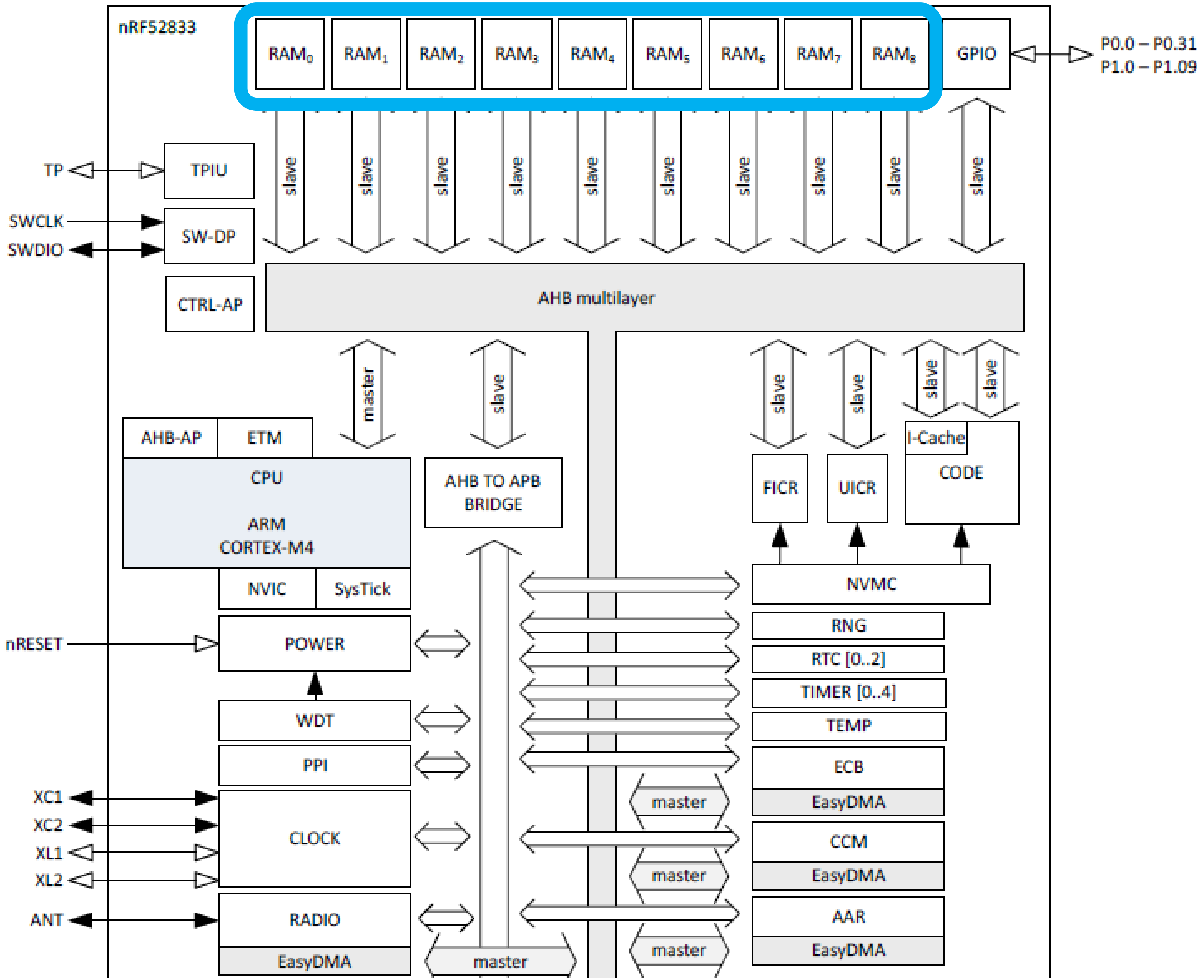
nRF52833 Peripherals

- JTAG and Debugging
- Allows code updates
- Allows GDB to step through code



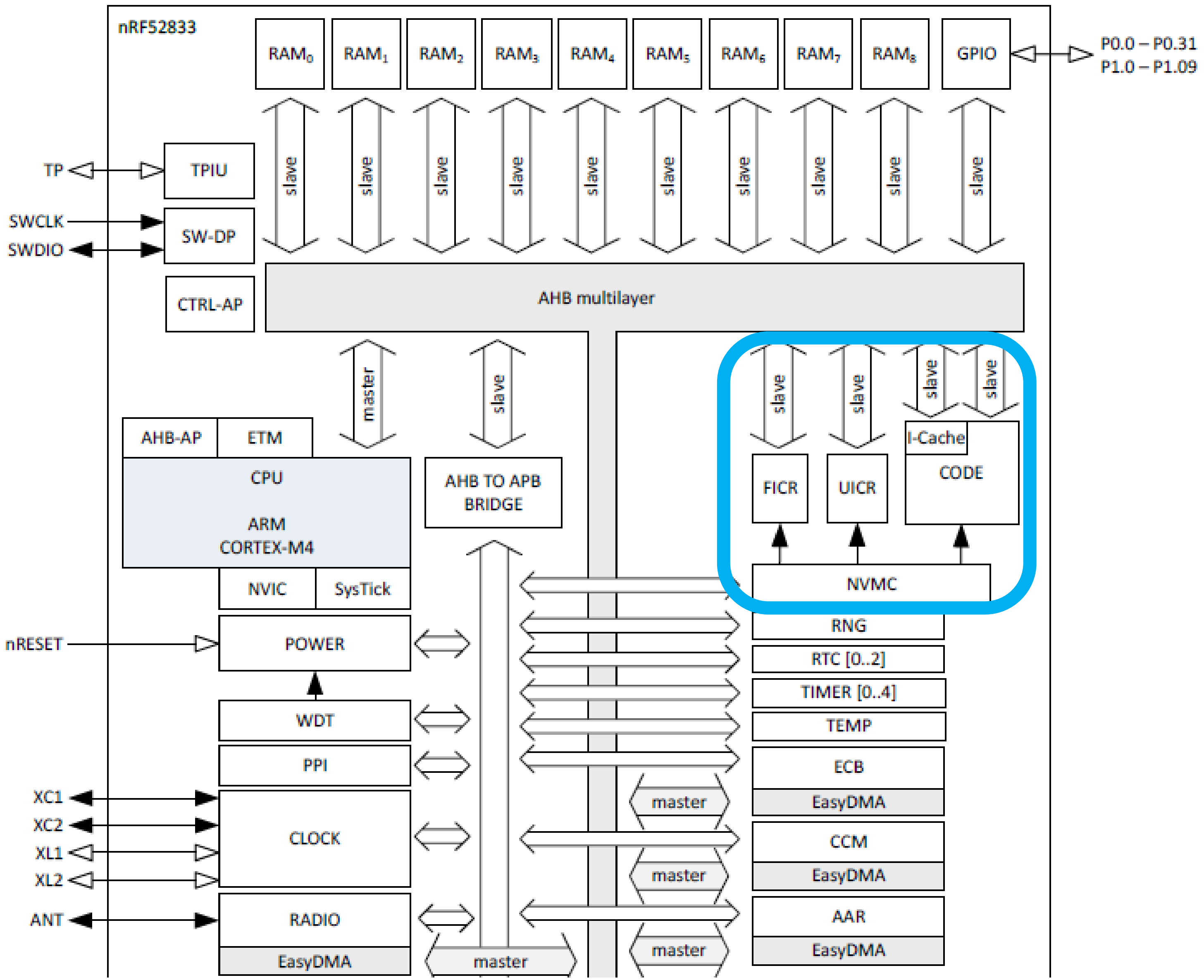
nRF52833 Peripherals

- Volatile memory
- SRAM, 128 kB



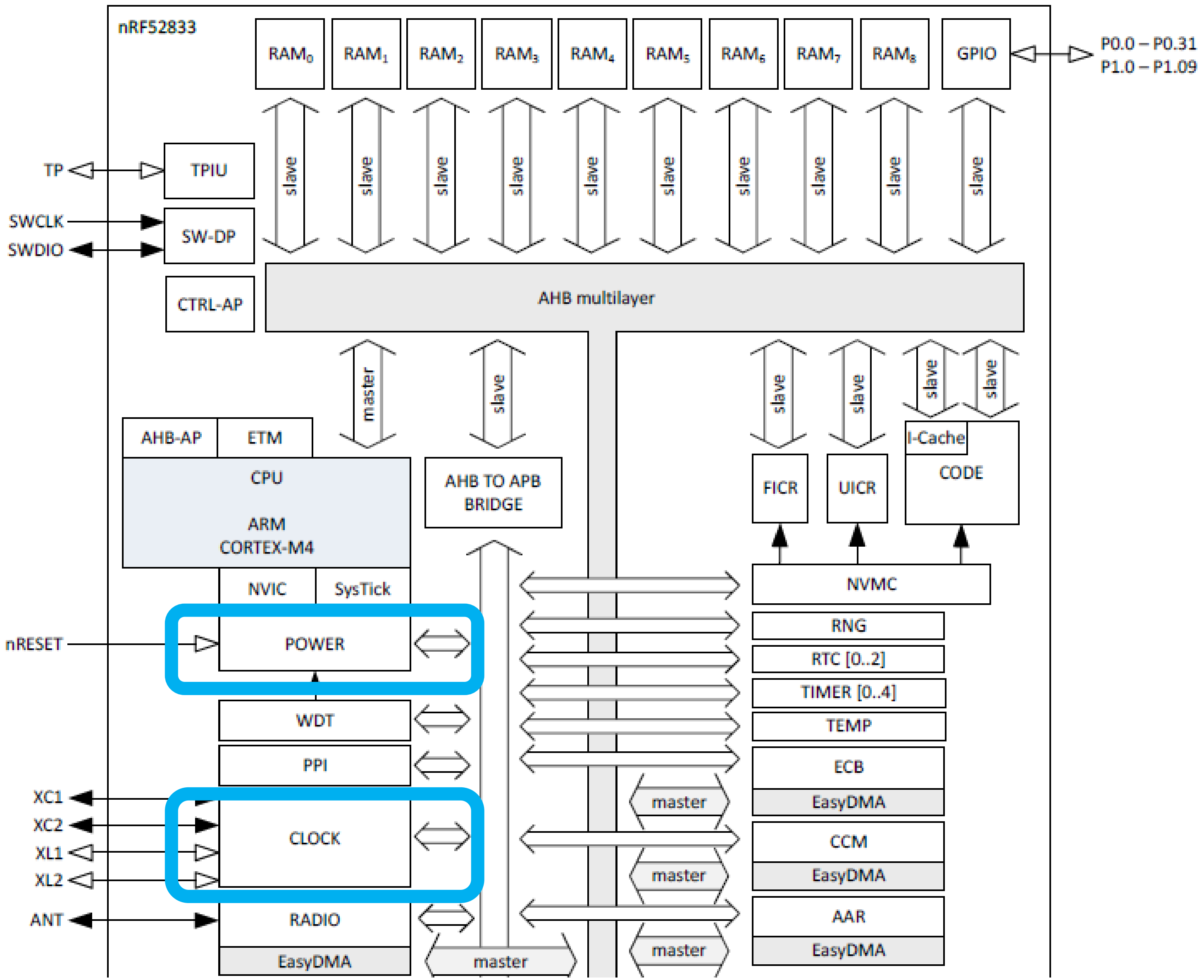
nRF52833 Peripherals

- Nonvolatile memory
- Flash, 512 kB
- Non-Volatile Memory Controller



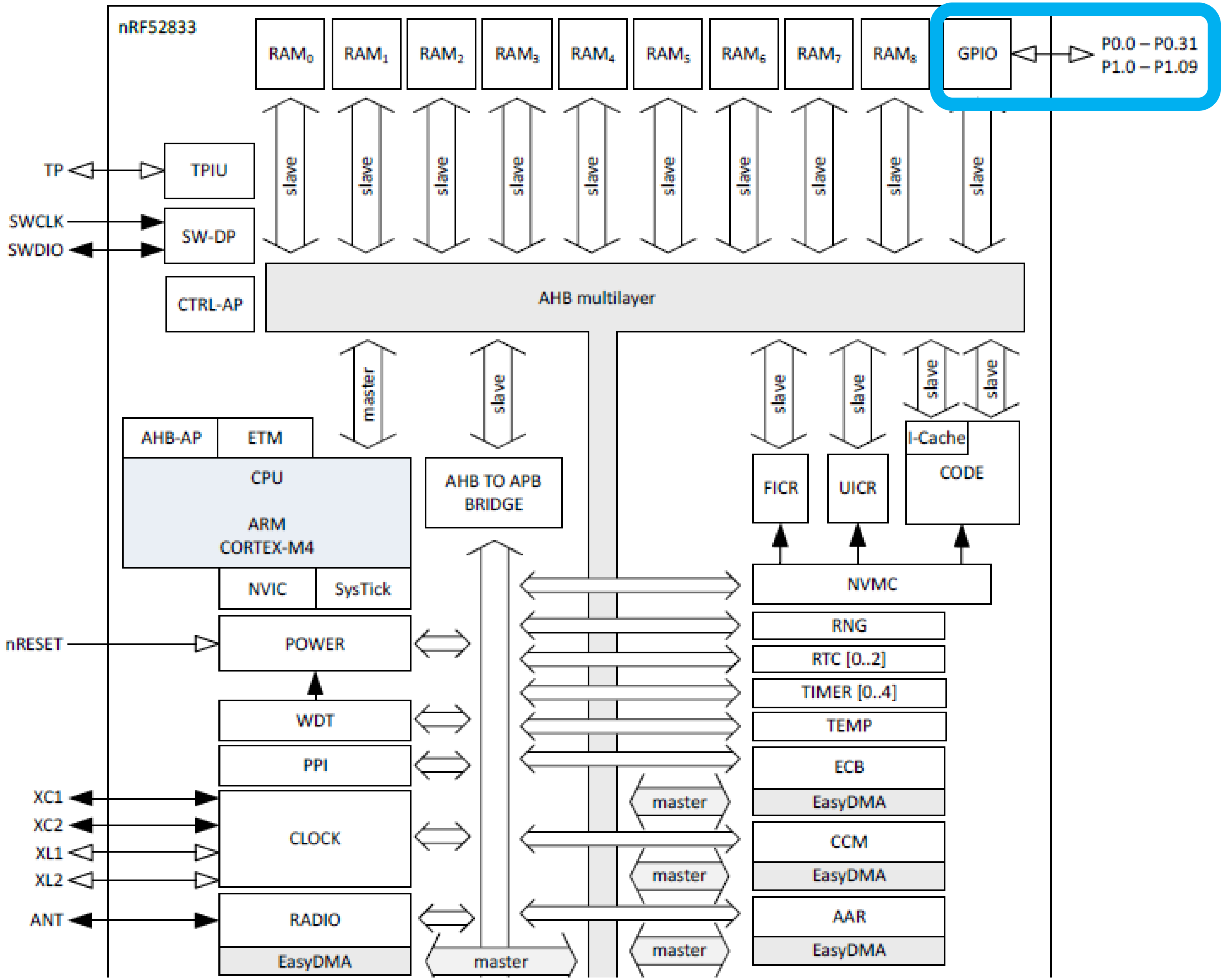
nRF52833 Peripherals

- Power and Clock management



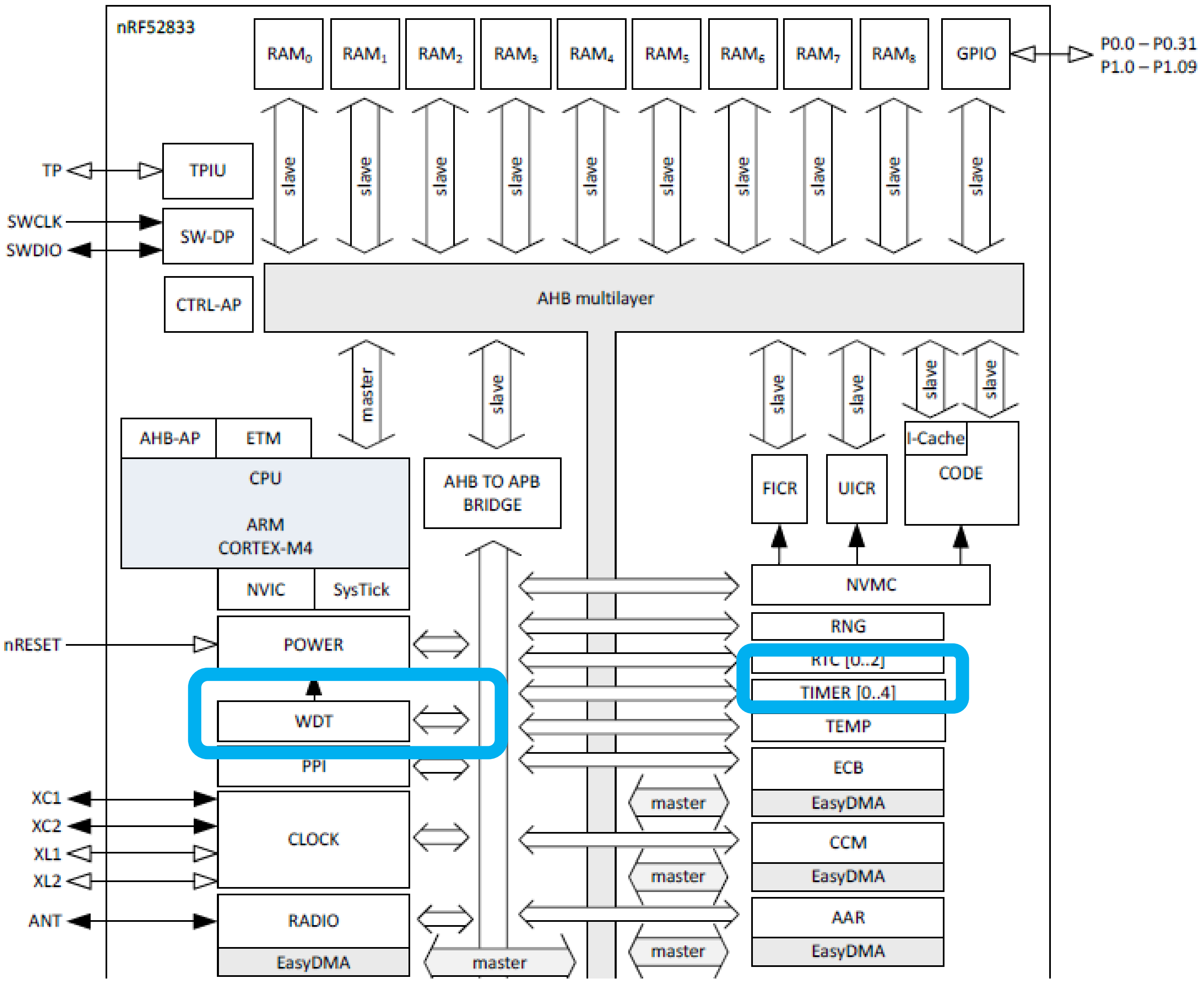
nRF52833 Peripherals

- GPIO pins



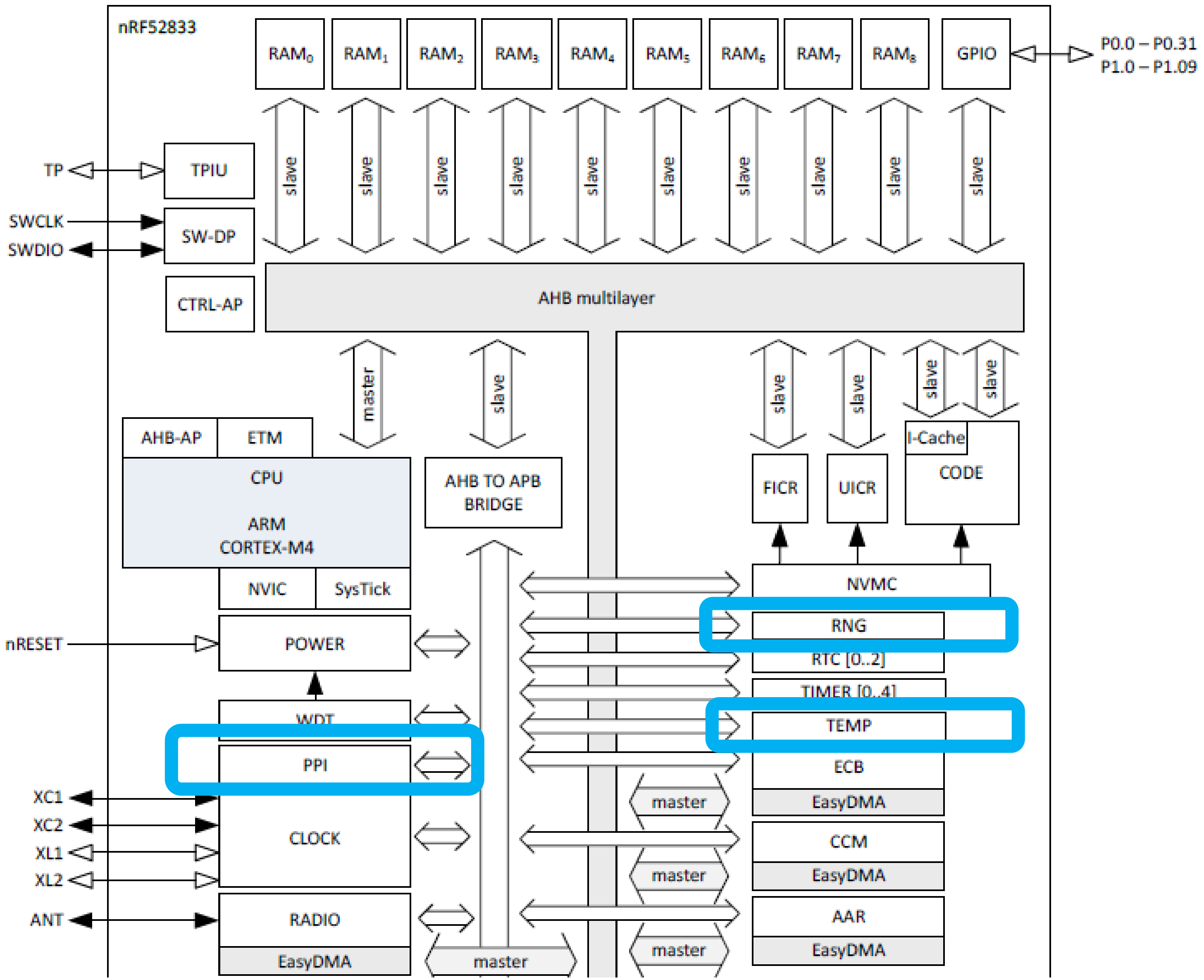
nRF52833 Peripherals

- Various timers
- Watchdog Timer
- Real-Time Counter
- Timer peripheral



nRF52833 Peripherals

- Programmable Peripheral Interconnect
  - Event->Task chaining
- Random Number Generator
- Temperature sensor

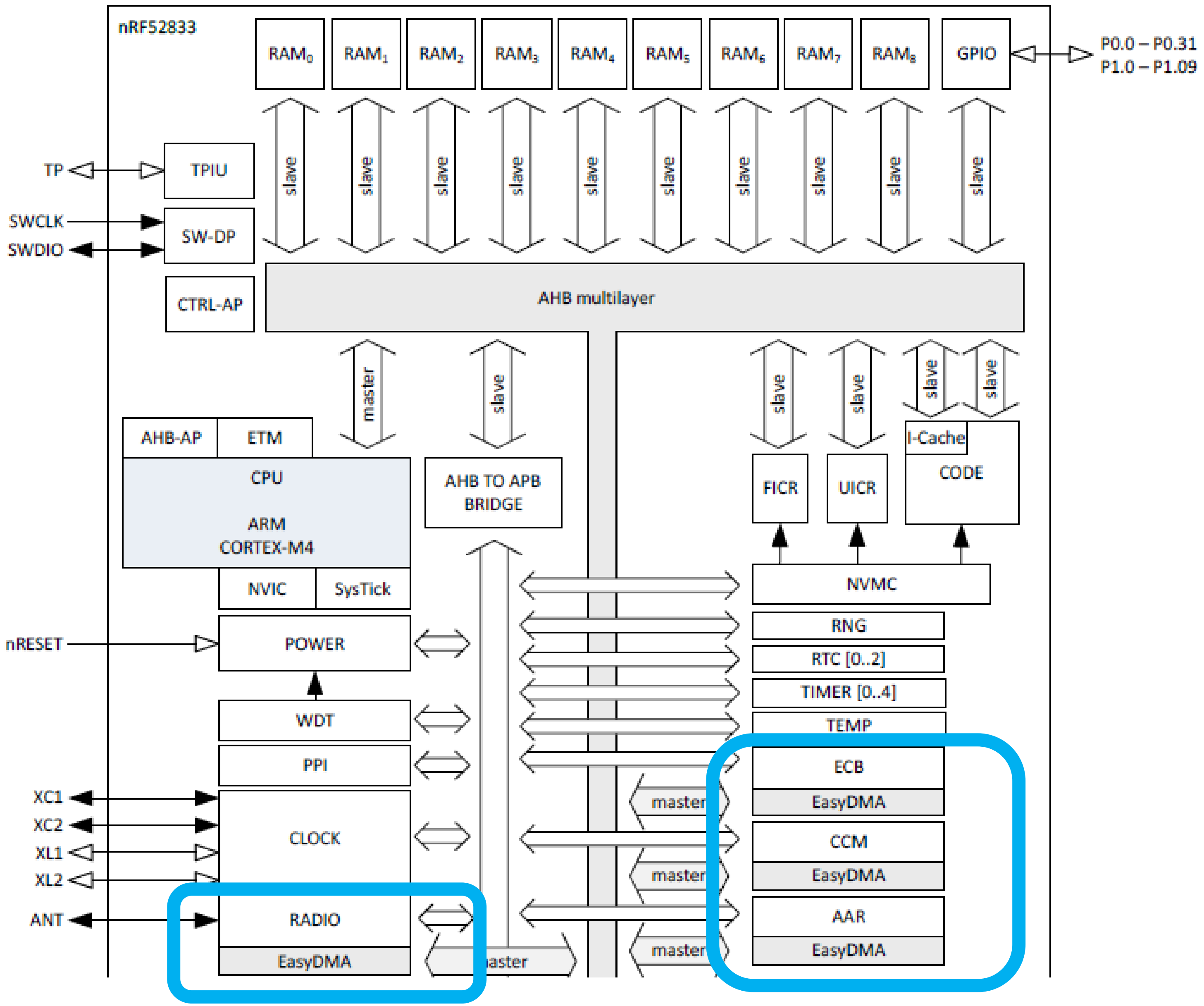


nRF52833 Peripherals

- Wireless radio
  - Bluetooth Low Energy
  - 802.15.4
    (Zigbee or Thread)

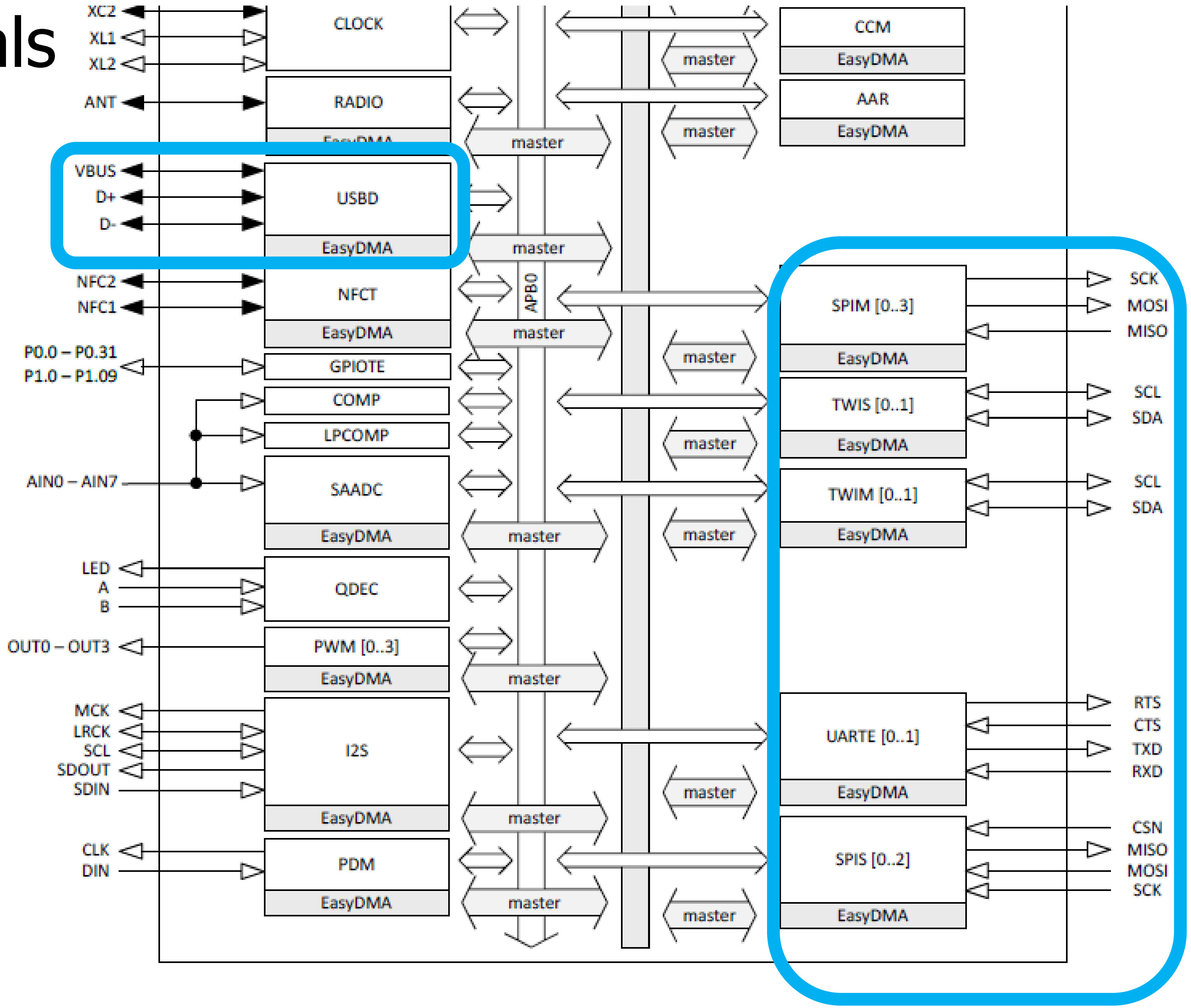
- Cryptography
  - ECB (AES mode)
  - CCM (AES mode)

  - AAR (Accelerated Address Resolver)
    - For BLE random addresses



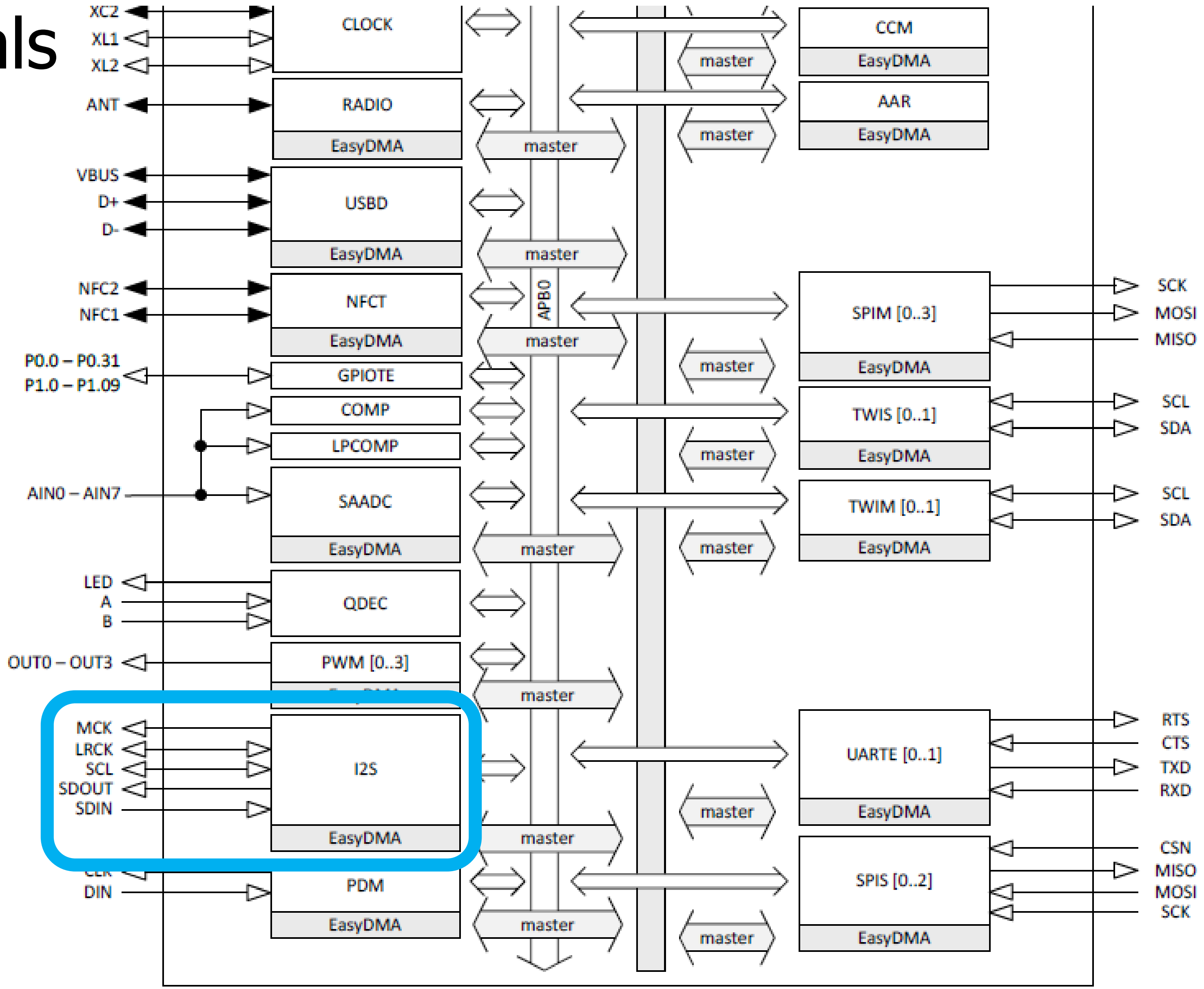
nRF52833 Peripherals

- Wired communication protocols
- USB Device
- SPI
  - Controller/Peripheral
- TWI (I2C)
  - Controller/Peripheral
- UART



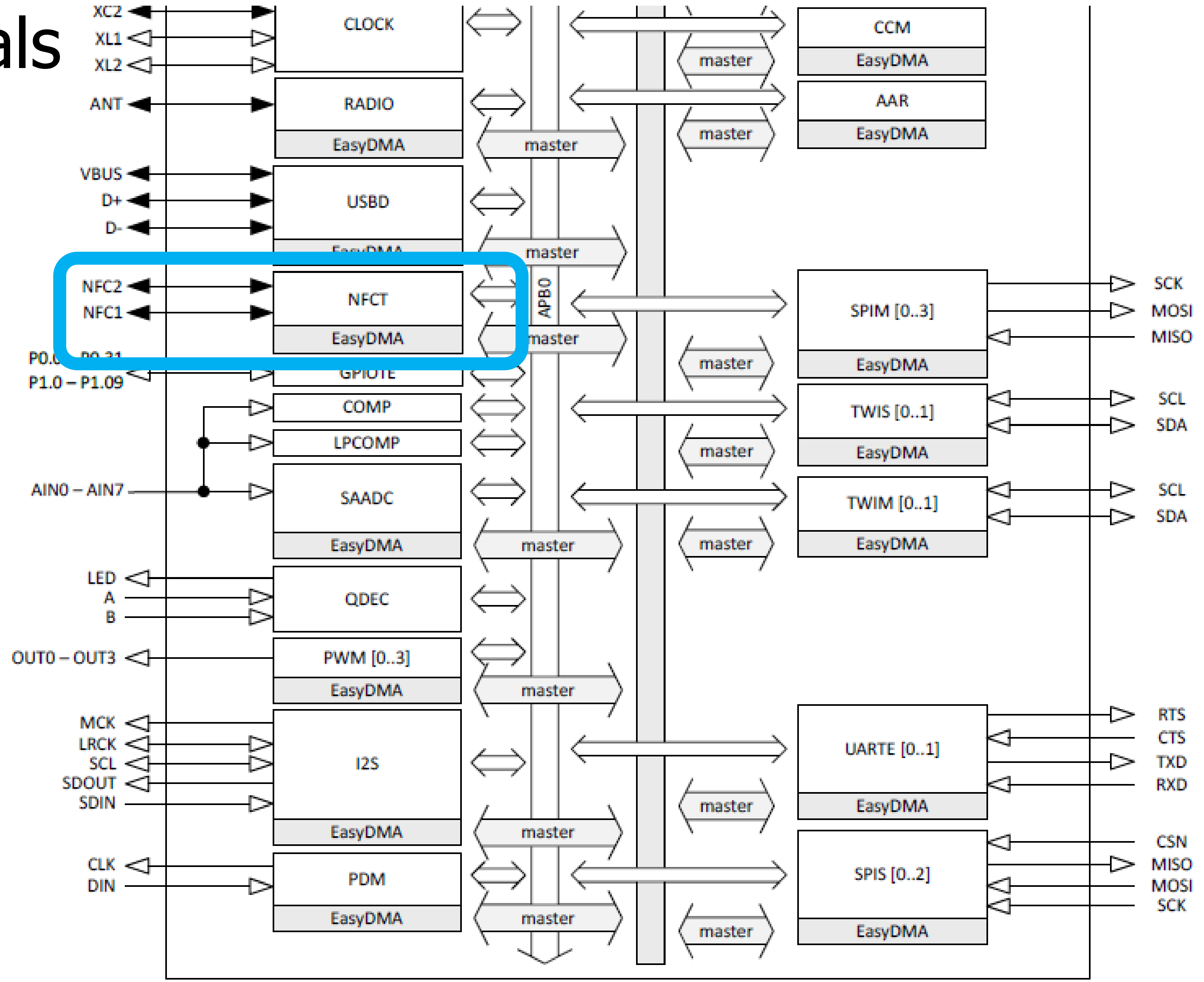
nRF52833 Peripherals

- Inter-IC Sound (I2S)
  - Wired communication bus explicitly for audio data
  - Unrelated to I2C
- Like a synchronous UART
  - Clock, data in, data out
- Additional signals
  - MCK – synchronization
  - LRCK – left/right channel select



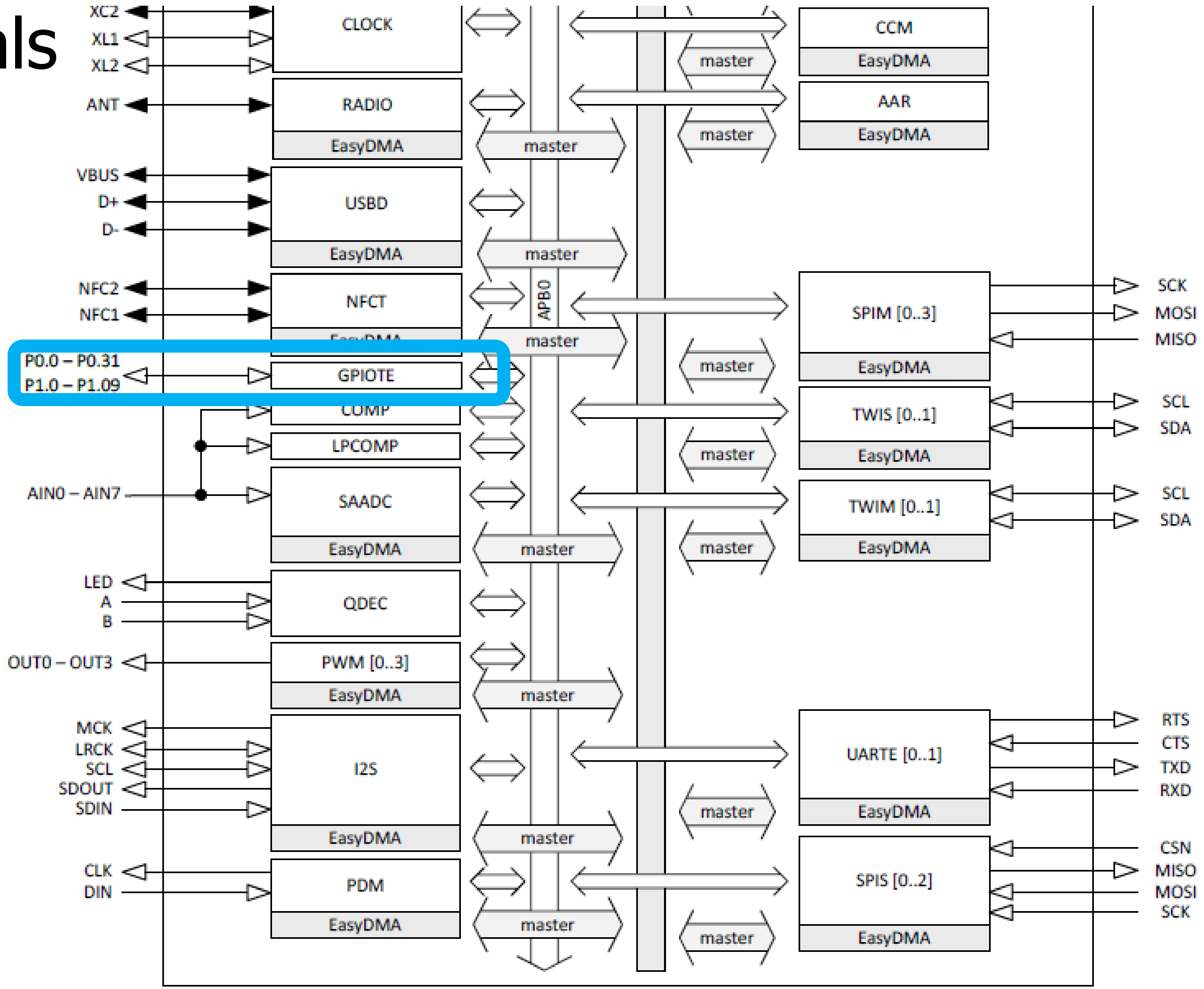
nRF52833 Peripherals

- NFC
  - Near-Field Communication
- Close-range wireless communication protocol
- "Tap-to-pay" systems



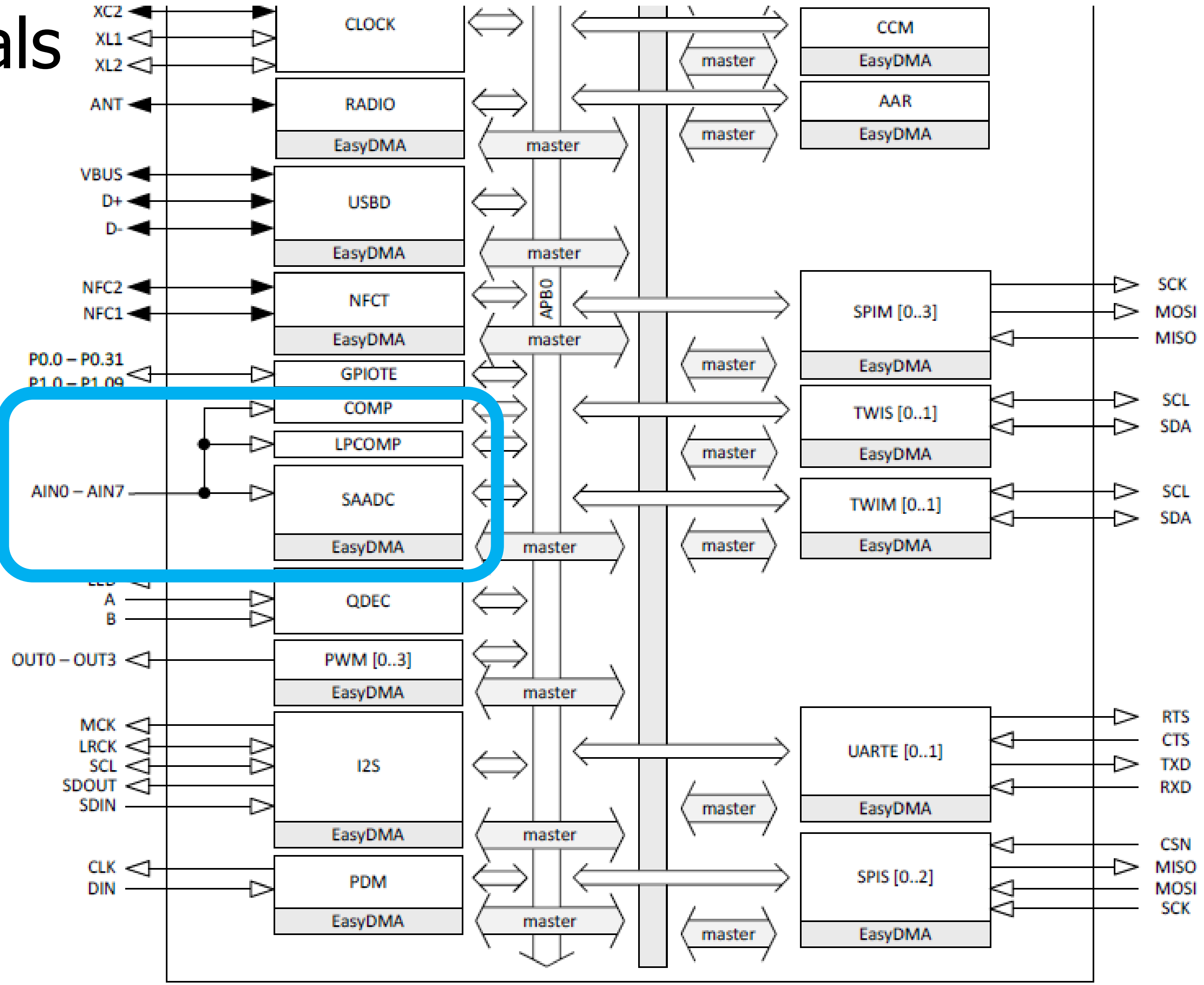
nRF52833 Peripherals

- GPIOTE
  - GPIO interrupts



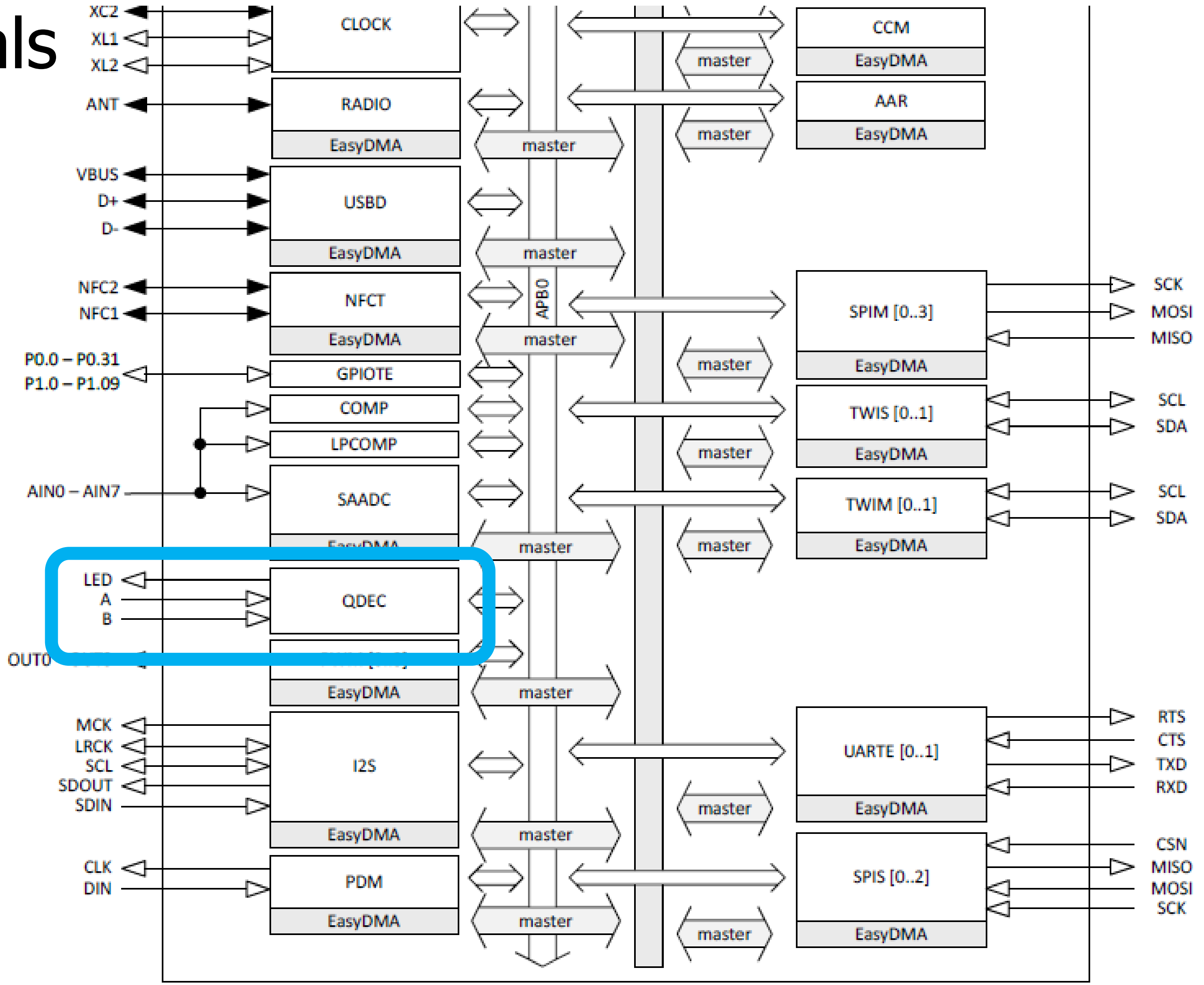
nRF52833 Peripherals

- Analog inputs
- Comparator
- Low-Power Comparator
- Successive Approximation Analog-to-Digital Converter



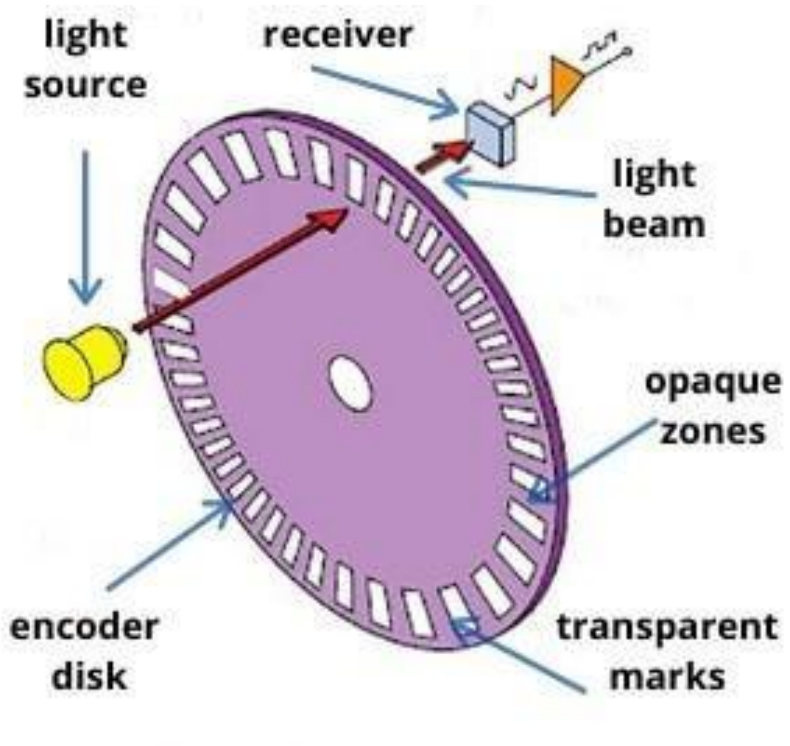
nRF52833 Peripherals

- Quadrature Decoder peripheral
- Detects rotation speeds and direction
  - Usually for motors

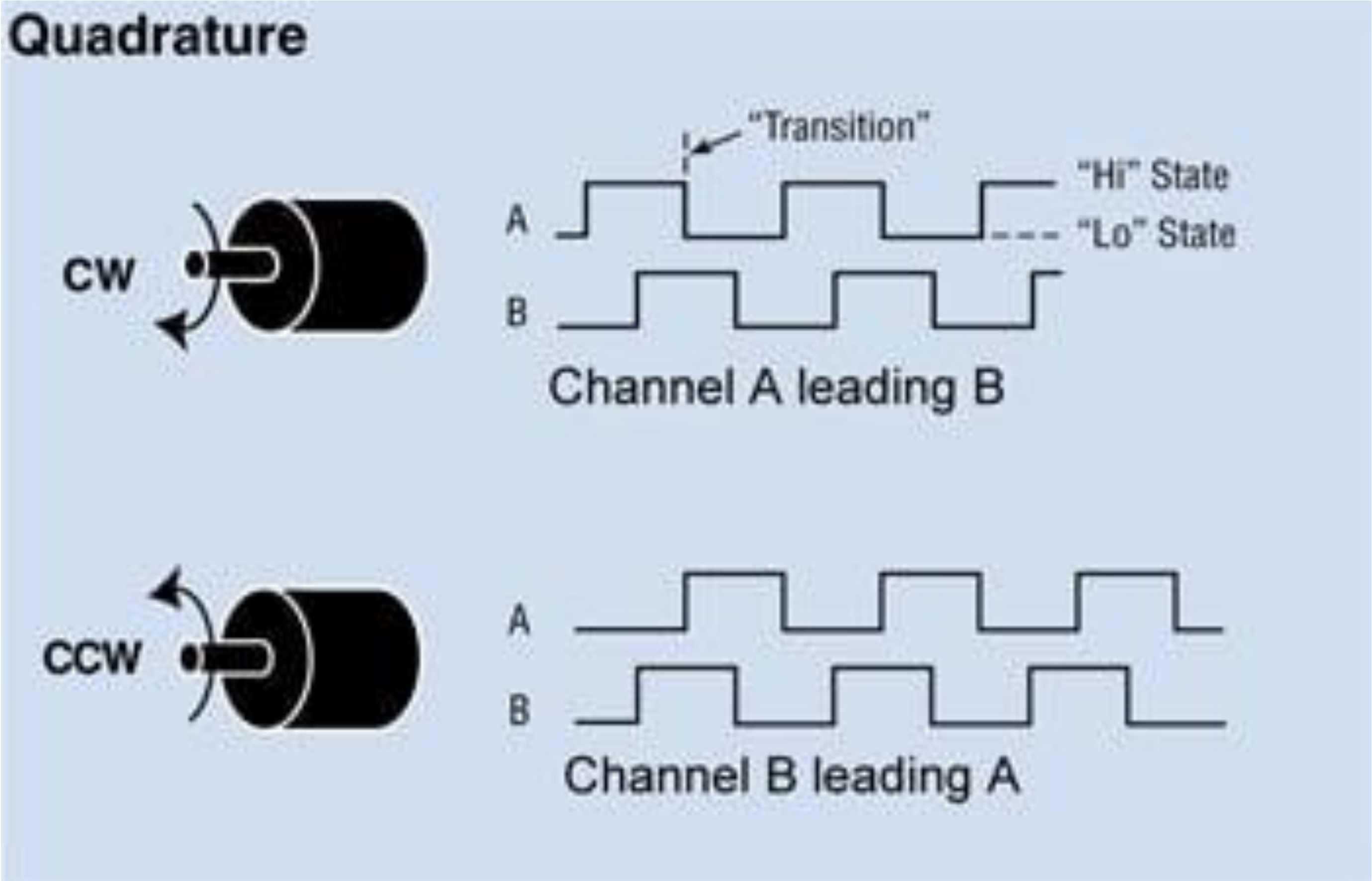


Quadrature Encoding

Usually two receivers and disks: A & B
That gets you direction and speed

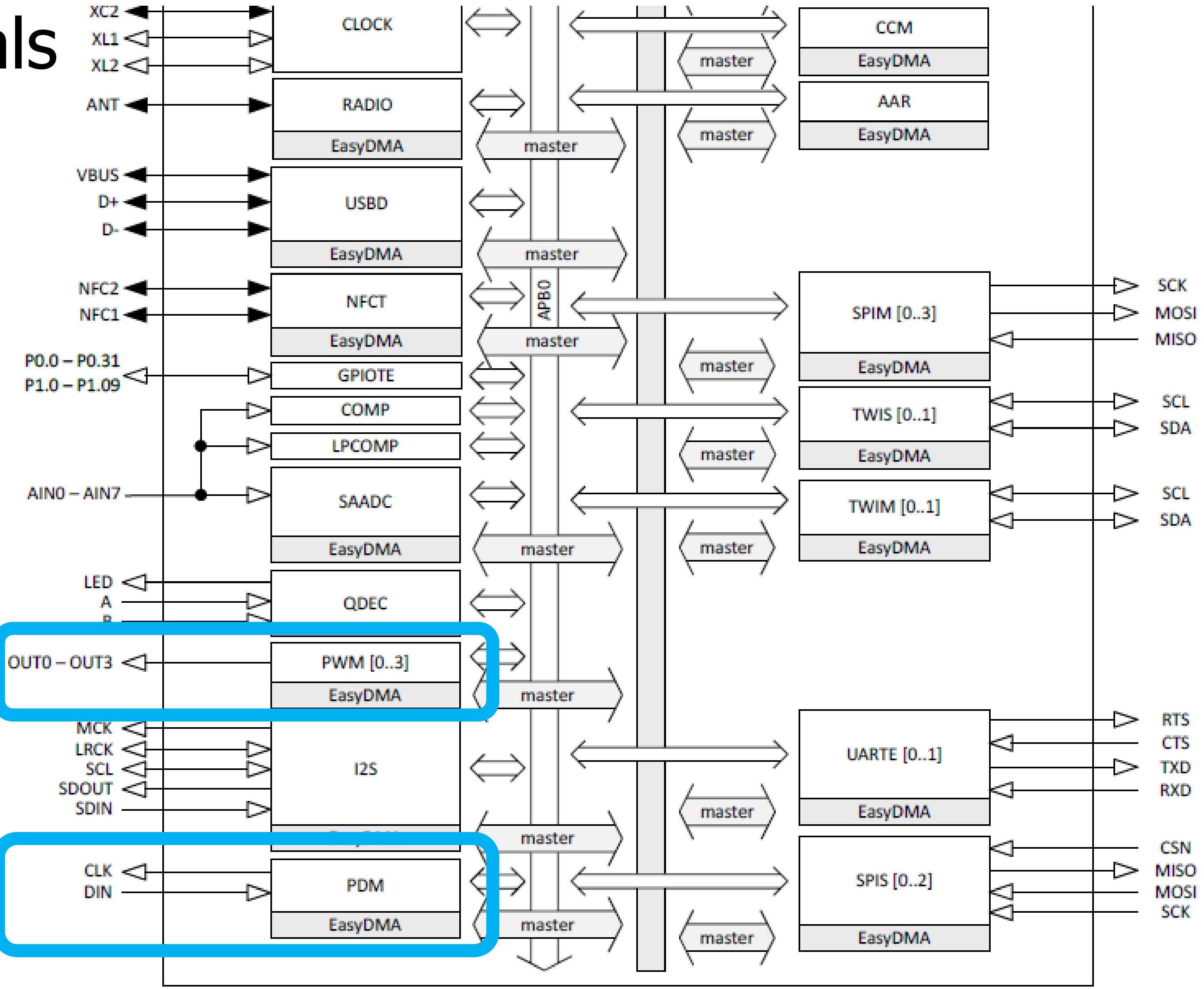


Optical encoder working principle



nRF52833 Peripherals

- Pulse Width Modulation
- Pulse Density Modulation
  - Similar idea to PWM
  - Input-only peripheral
  - Targets microphones



nRF52833 is complete

- That’s just about everything!

- First 550 out of 600 pages of nRF52833 datasheet
  - Remaining 50 are hardware details
    - Pinout for different packages
    - Recommended circuit layout
    - Soldering details

Break + Open Question

- What **doesn't** the nRF52833 have a peripheral for?
  (i.e., what could you imagine a peripheral for that it doesn't have)

Outline

- What haven’t we talked about?
  - Microbit
  - nRF52833
- **Other hardware systems**
  - **Other microcontroller peripherals**
  - Microprocessors

Microcontroller knowledge is transferrable

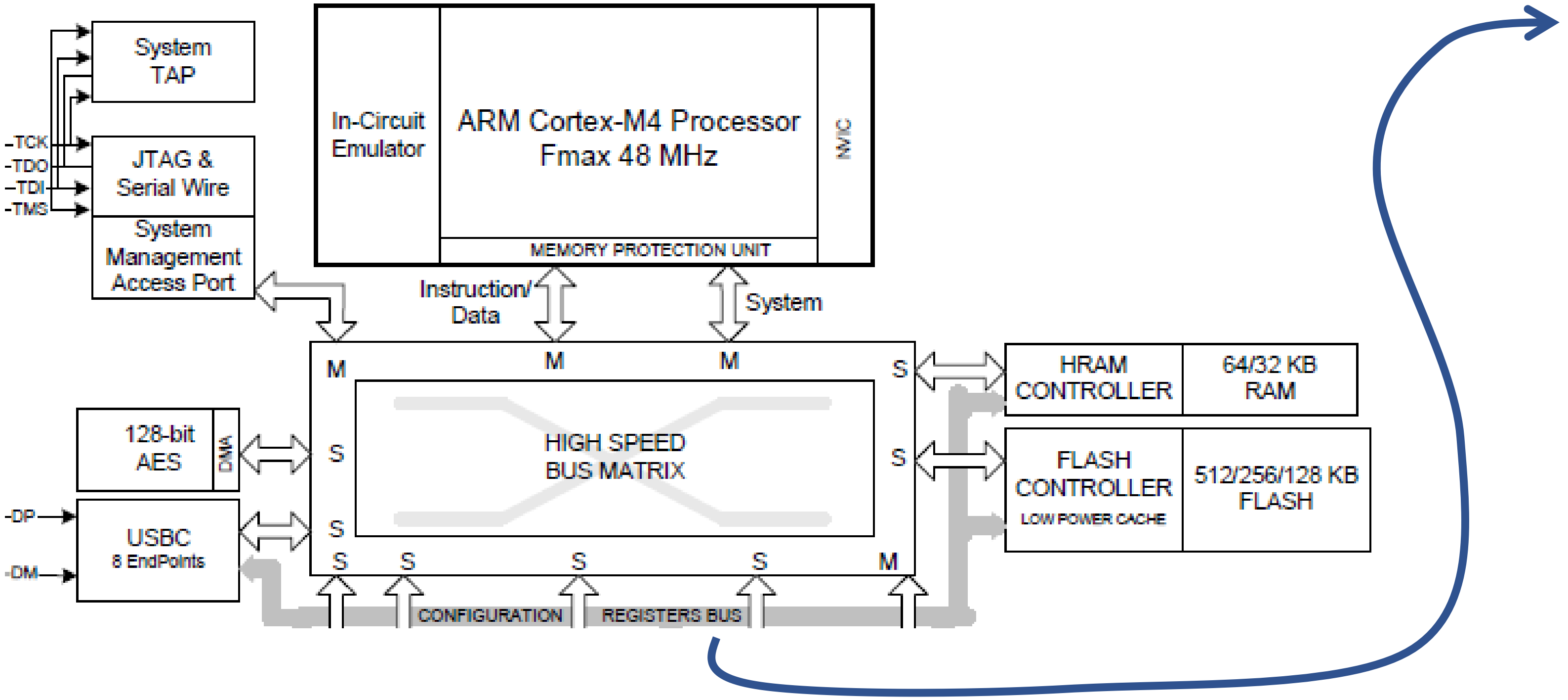
- Your knowledge of microcontrollers applies to almost all of them
  - They have similar peripherals that work in similar ways

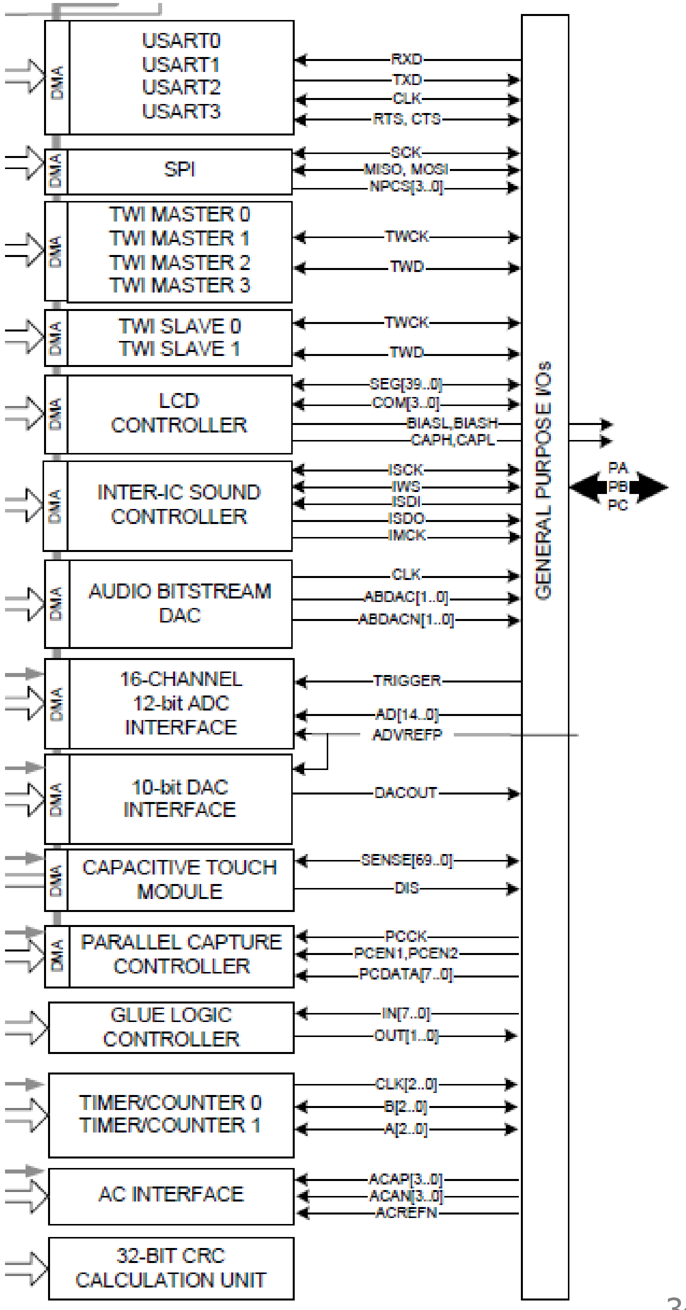
- The exact names and configurations will definitely be different
  - But the fundamentals stay the same

- Let’s prove this with a quick view of two microcontrollers
  1. Atmel SAM4L
  2. Texas Instruments MSP430

Atmel SAM4L Microcontroller

- ARM Cortex M4F (same processor!)
- Lots of very configurable peripherals
  - USART, SPI, TWI, I2S, DAC, ADC, Timer, …
  - Kind of a “do-everything” microcontroller





SAM4L GPIO

- GPIO register map (heavily abbreviated)

Table 23-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x000	GPIO Enable Register	Read/Write	GPER	Read/Write	-[1]	Y	N
0x040	Output Driver Enable Register	Read/Write	ODER	Read/Write	-[1]	Y	N
0x050	Output Value Register	Read/Write	OVR	Read/Write	-[1]	N	N
0x060	Pin Value Register	Read	PVR	Read-only	Depending on pin states	N	N
0x070	Pull-up Enable Register	Read/Write	PUER	Read/Write	-[1]	Y	N
0x080	Pull-down Enable Register	Read/Write	PDER	Read/Write	(1)	Y	N
0x090	Interrupt Enable Register	Read/Write	IER	Read/Write	-[1]	N	N

Setting or clearing individual bits

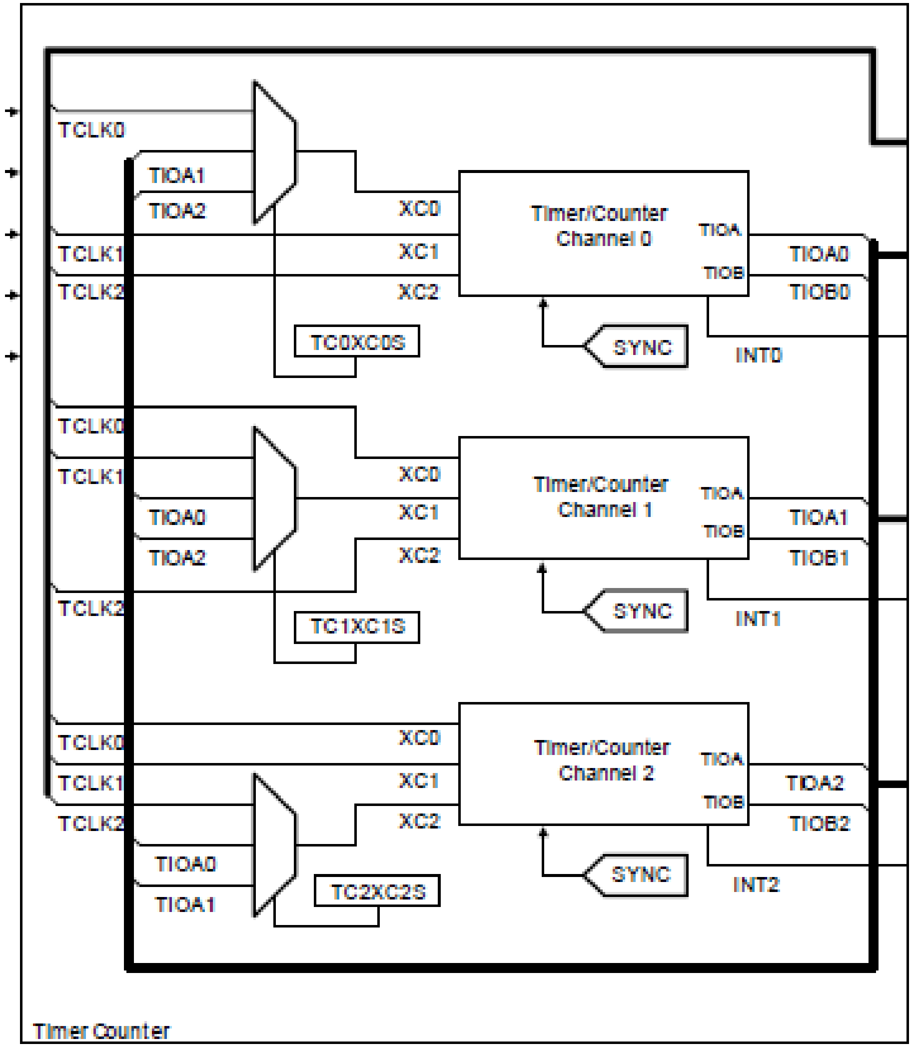
- Actually, each register appears four times in a row
  - Read/write version
  - Set version (like nRF *SET)
  - Clear version (like nRF *CLR)
  - Toggle version

Table 23-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x000	GPIO Enable Register	Read/Write	GPER	Read/Write	-[1]	Y	N
0x004	GPIO Enable Register	Set	GPERS	Write-only		Y	N
0x008	GPIO Enable Register	Clear	GPERC	Write-only		Y	N
0x00C	GPIO Enable Register	Toggle	GPERT	Write-only		Y	N

SAM4L Timers

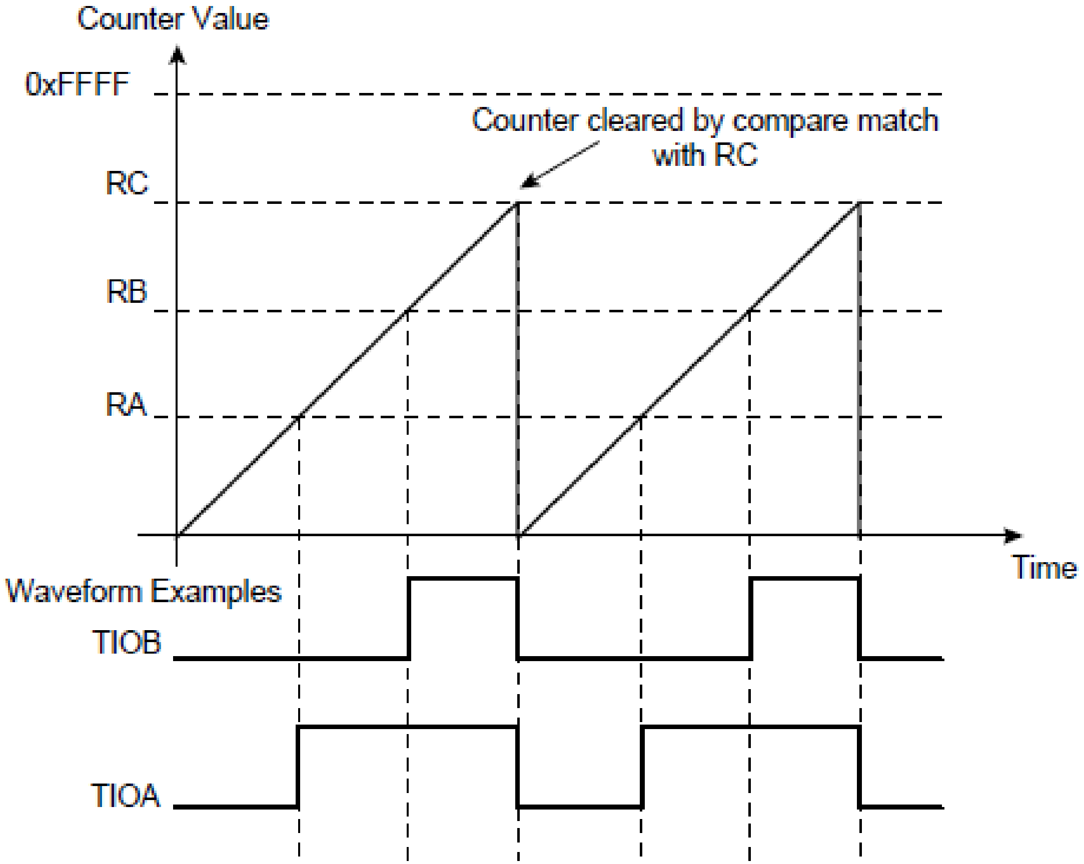
- Three 16-bit timers
  - Each of which can have multiple inputs clock signals with prescaler values
  - Timers can be chained together to make up to a 48-bit timer
- One register for reading counter
- Three registers for “compare interrupts”



0x10	Channel 0 Counter Value	CV0	Read-only
0x14	Channel 0 Register A	RA0	Read/Write[1]
0x18	Channel 0 Register B	RB0	Read/Write[1]
0x1C	Channel 0 Register C	RC0	Read/Write

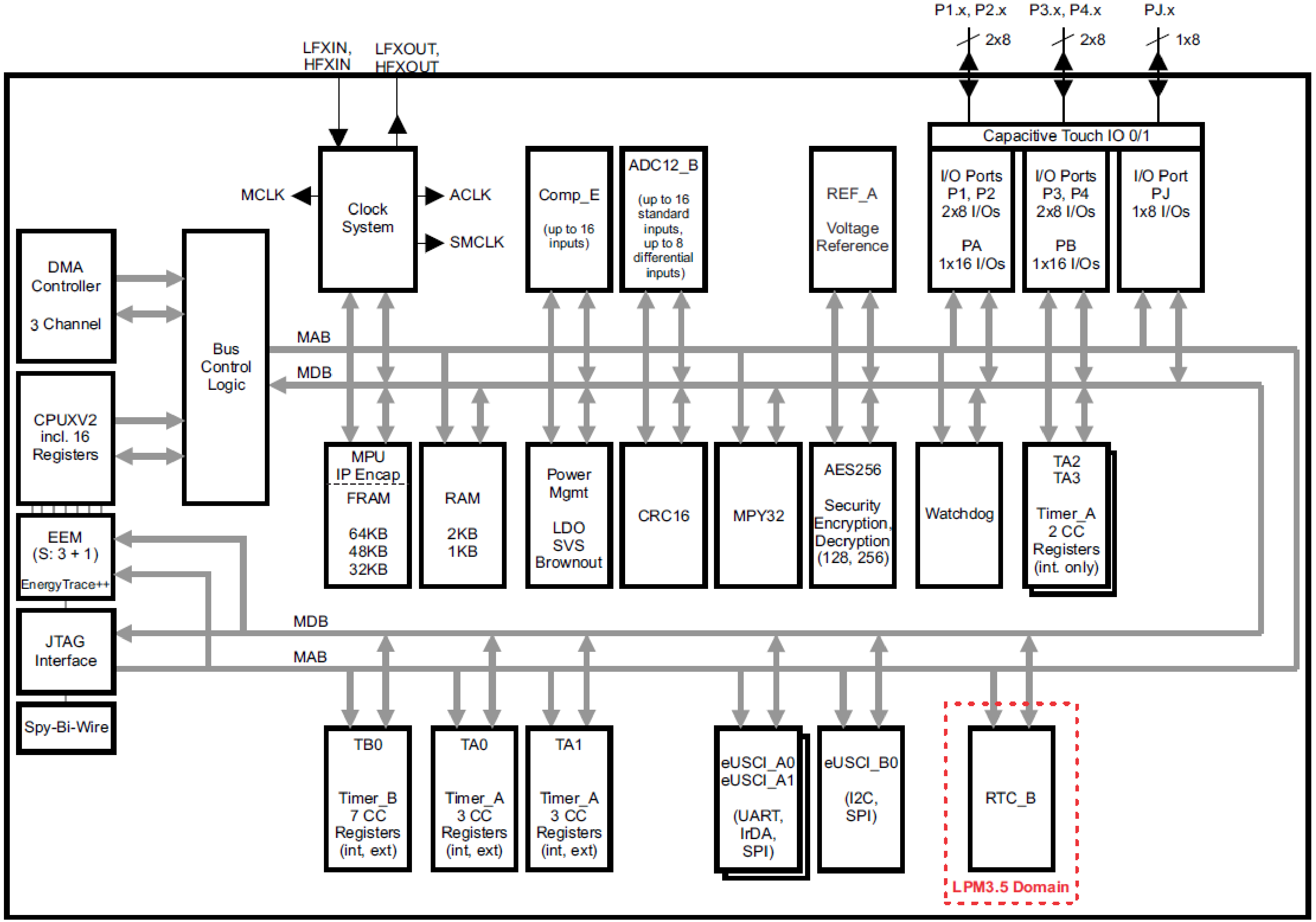
SAM4L PWM

- PWM peripheral is essentially built into Timer peripheral
- Timer peripheral also supports a “Waveform” mode where it generates an output GPIO signal, i.e. PWM
- One comparison point is low-to-high transition, another is high-to-low transition



Texas Instruments MSP430 (MSP430FR59xx)

- Example of a *very* different microcontroller
- 16-bit custom architecture processor
- 2 kB RAM
- 64 kB FRAM
  - Non-volatile, byte-writable



MSP430 GPIO register map

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P2IN or PAIN_H	Port 2 Input	Read only	Byte	undefined	Section 12.4.2
03h	P2OUT or PAOUT_H	Port 2 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P2DIR or PADIR_H	Port 2 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P2REN or PAREN_H	Port 2 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P2SEL0 or PASEL0_H	Port 2 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P2SEL1 or PASEL1_H	Port 2 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P2SELC or PASELC_L	Port 2 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P2IES or PAIES_H	Port 2 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P2IE or PAIE_H	Port 2 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P2IFG or PAIFG_H	Port 2 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

- Registers are a single byte in size
- Oddly, some have an “undefined” reset state 😱

Example MSP430 GPIO register: OUT

- Example 8-bit register. Controls 8 pins in a GPIO "port"

12.4.3 *PxOUT Register*

Port x Output Register

Figure 12-3. PxOUT Register

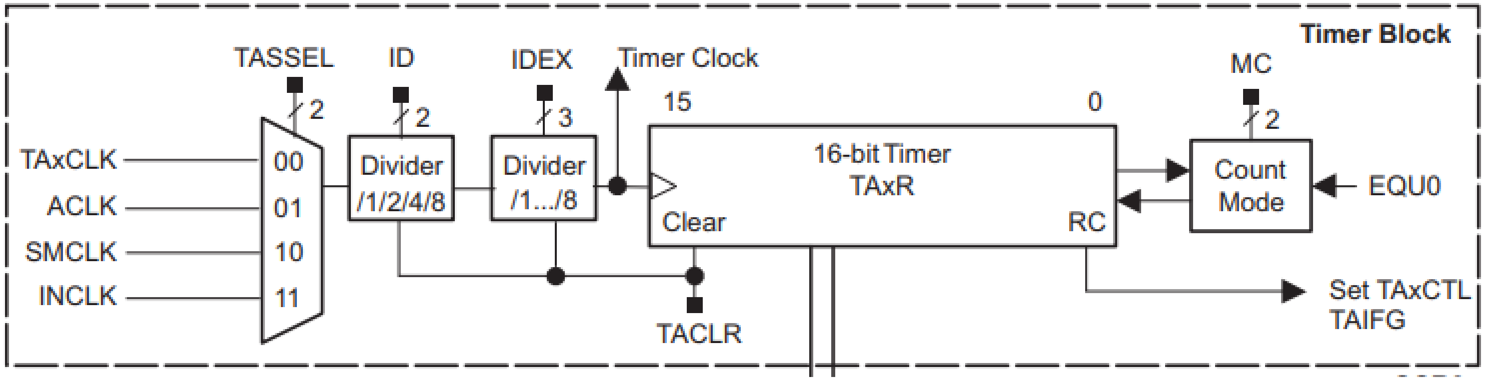
7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-6. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	RW	Undefined	Port x output When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

MSP430 Timer peripheral

- A choice of input clocks goes through dividers to run a 16-bit timer
  - Five total timers available on the system
  - Some can trigger external output pins, some are only internal



MSP430 PWM

- The MSP430 Timer peripheral handles PWM as well!

25.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TAxCCR0, which defines the period (see Figure 25-2). The number of timer counts in the period is TAxCCR0 + 1. When the timer value equals TAxCCR0, the timer restarts counting from zero. If up mode is selected when the timer value is greater than TAxCCR0, the timer immediately restarts counting from zero.

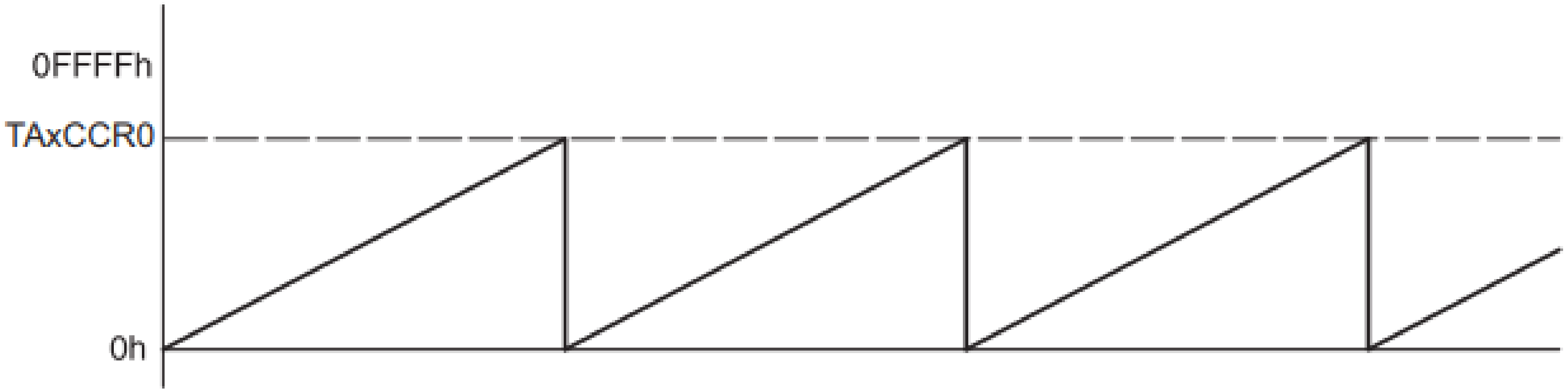
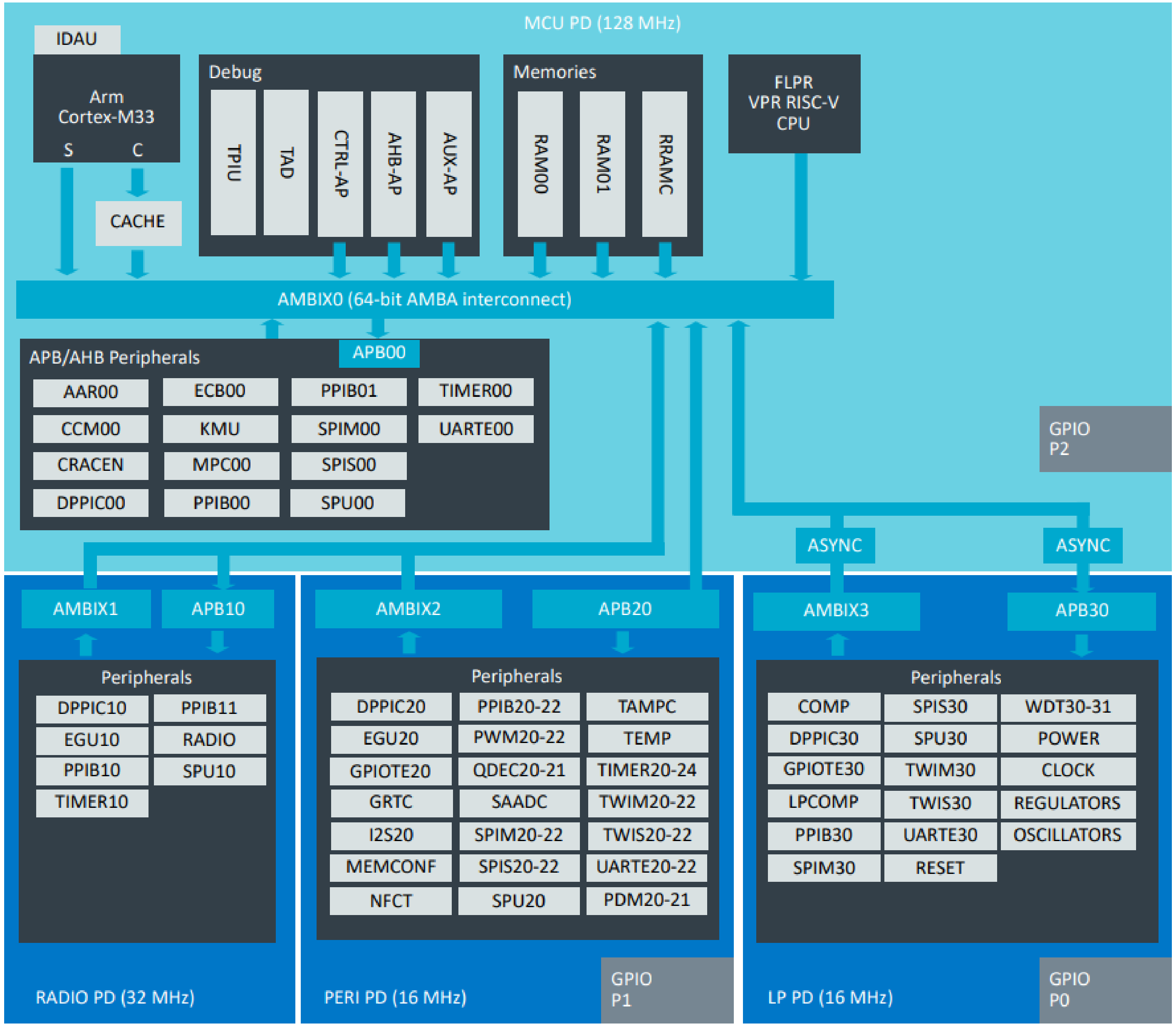


Figure 25-2. Up Mode

Bonus: nRF54L15

- ARM Cortex-M33
  - 128 MHz
- RISC-V EMC
  - 128 MHz
  - Co-processor
  - “Soft Peripherals”
    - Load code so this acts like an arbitrary peripheral
- Memory
  - 256 KB RAM
  - 1524 KB RRAM (non-volatile)



nRF54L15 Temperature Peripheral

Instance	Domain	Base address	TrustZone			Split access	Description
			Map	Att	DMA		
TEMP : S TEMP : NS	GLOBAL	0x500D7000 0x400D7000	US	S	NA	No	Temperature sensor TEMP

Register overview

Register	Offset	TZ	Description
TASKS_START	0x000		Start temperature measurement
TASKS_STOP	0x004		Stop temperature measurement
SUBSCRIBE_START	0x080		Subscribe configuration for task START
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
EVENTS_DATARDY	0x100		Temperature measurement complete, data ready
PUBLISH_DATARDY	0x180		Publish configuration for event DATARDY
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
TEMP	0x508		Temperature in °C (0.25° steps)

nRF54L15 peripherals

- GPIO (very similar) and GPIOTE (very similar)
  - Three ports now, split by: low-power, normal, or high-speed
- TIMER (very similar), improved RTC (1 μs ticks, 52 bits)
- LPCOMP, COMP, and SAADC (all very similar)
- PWM (basically identical)
- UART, TWI, SPI (very similar)

nRF54L15 changes

- Power domains
  - Makes it very clear which peripherals are active in different active/sleep modes
- Peripheral interconnect
  - nRF52 allows chaining Tasks and Events together
  - nRF54 expands this to a full publish/subscribe architecture
    - Timer event -> trigger ADC
    - Timer OR GPIO event -> disable GPIO and Timer

Outline

- What haven’t we talked about?
  - Microbit
  - nRF52833
- **Other hardware systems**
  - Other microcontroller peripherals
  - **Microprocessors**

Microprocessor system design

- What about modern microprocessors? (really SoC’s)
  - Similar idea, but a LOT more stuff

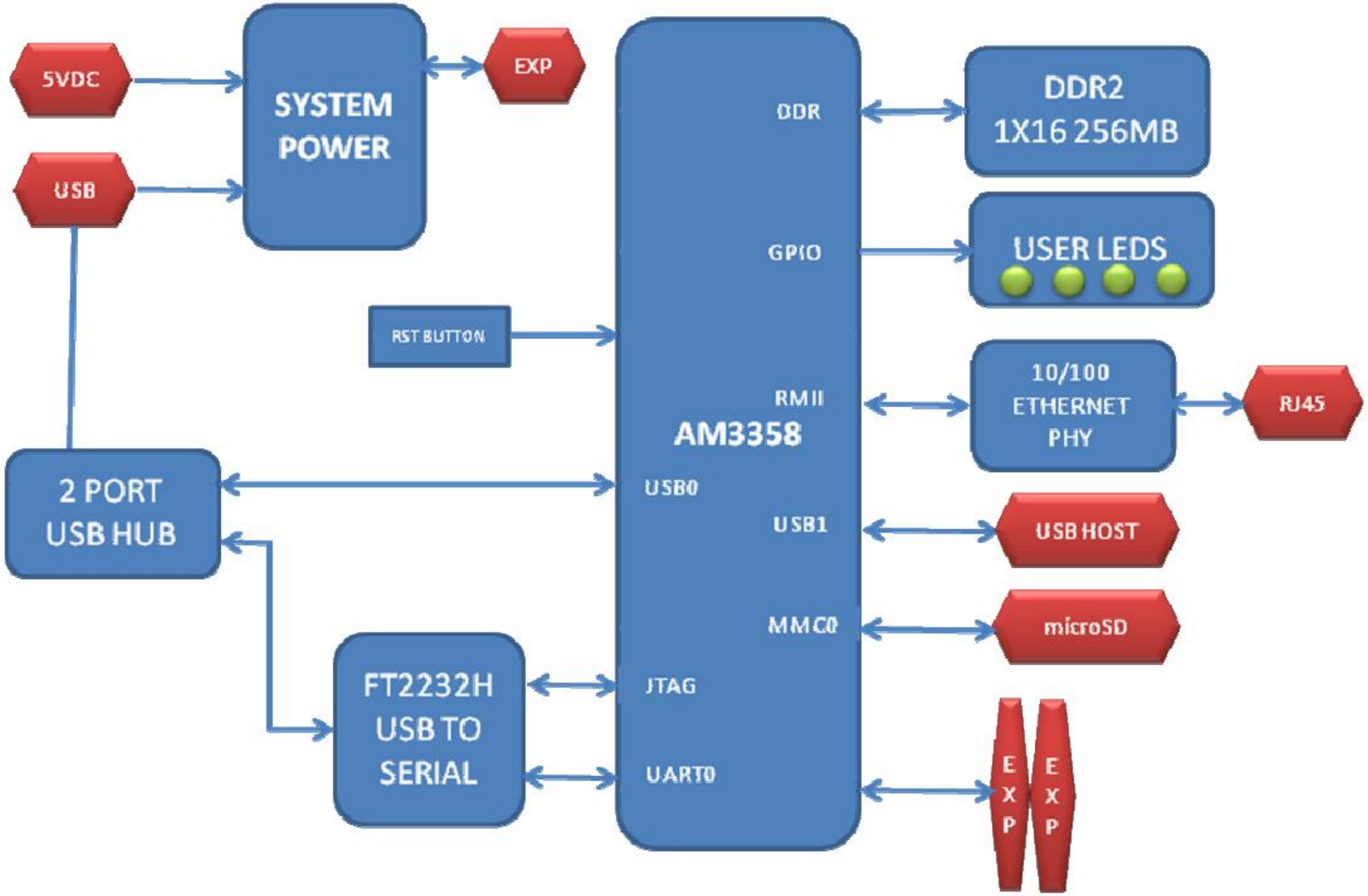
- Various peripherals, which might not be focused on sensor I/O
  - Less ADCs and I2C more Ethernet and Graphics

- External memory busses
  - Microprocessors expect external memory to exist on the system
  - Which means that they need pins for an external memory bus

Beaglebone

- Cheap, single board computer
  - Like Raspberry Pi

Gross diagram that's actually in their datasheet



Resources on the Beaglebone and its microprocessor

- https://beagleboard.org/bone-original
- https://beagleboard.org/static/beaglebone/BEAGLEBONE_SCHEM_A3.pdf
- https://transistor-man.com/files/emu/beaglebone_hardware/BONE_SRM.pdf

- https://www.ti.com/product/AM3358

AM3358 processor

- ARM Cortex A8
  - 32-bit processor
  - 1 GHz clock
  - 300 pins!!

- Internal memory
  - 176 kB ROM
  - 64 kB RAM

- External memory
  - DDR2/3, 400 MHz RAM
  - NAND or NOR Flash

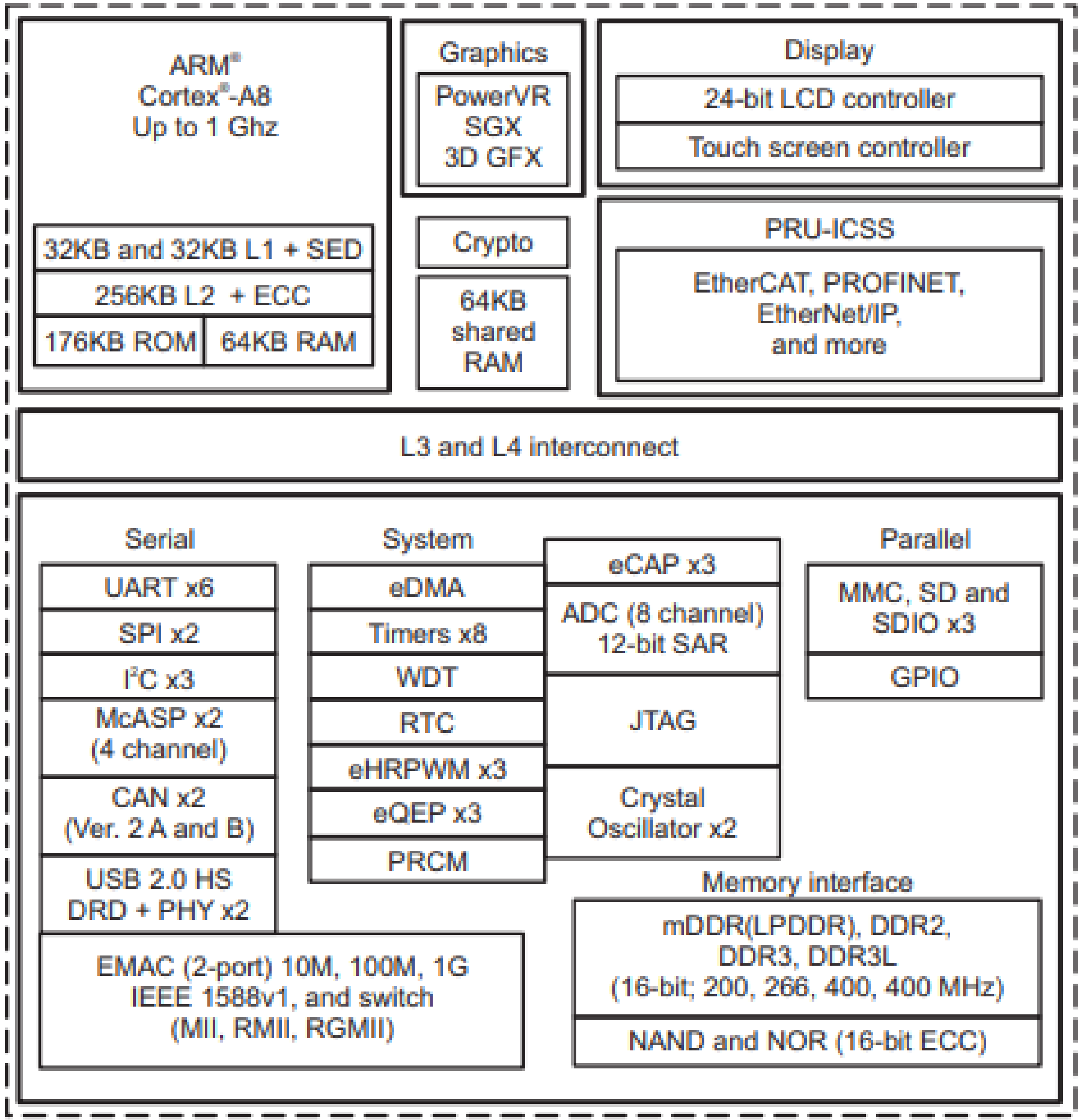


Figure 1-1. AM335x Functional Block Diagram

AM3358 peripherals

- Advanced peripherals
  - Graphics
  - Ethernet
  - USB, CAN

- Regular peripherals
  - GPIO, Timers
  - ADC
  - UART, SPI, I2C

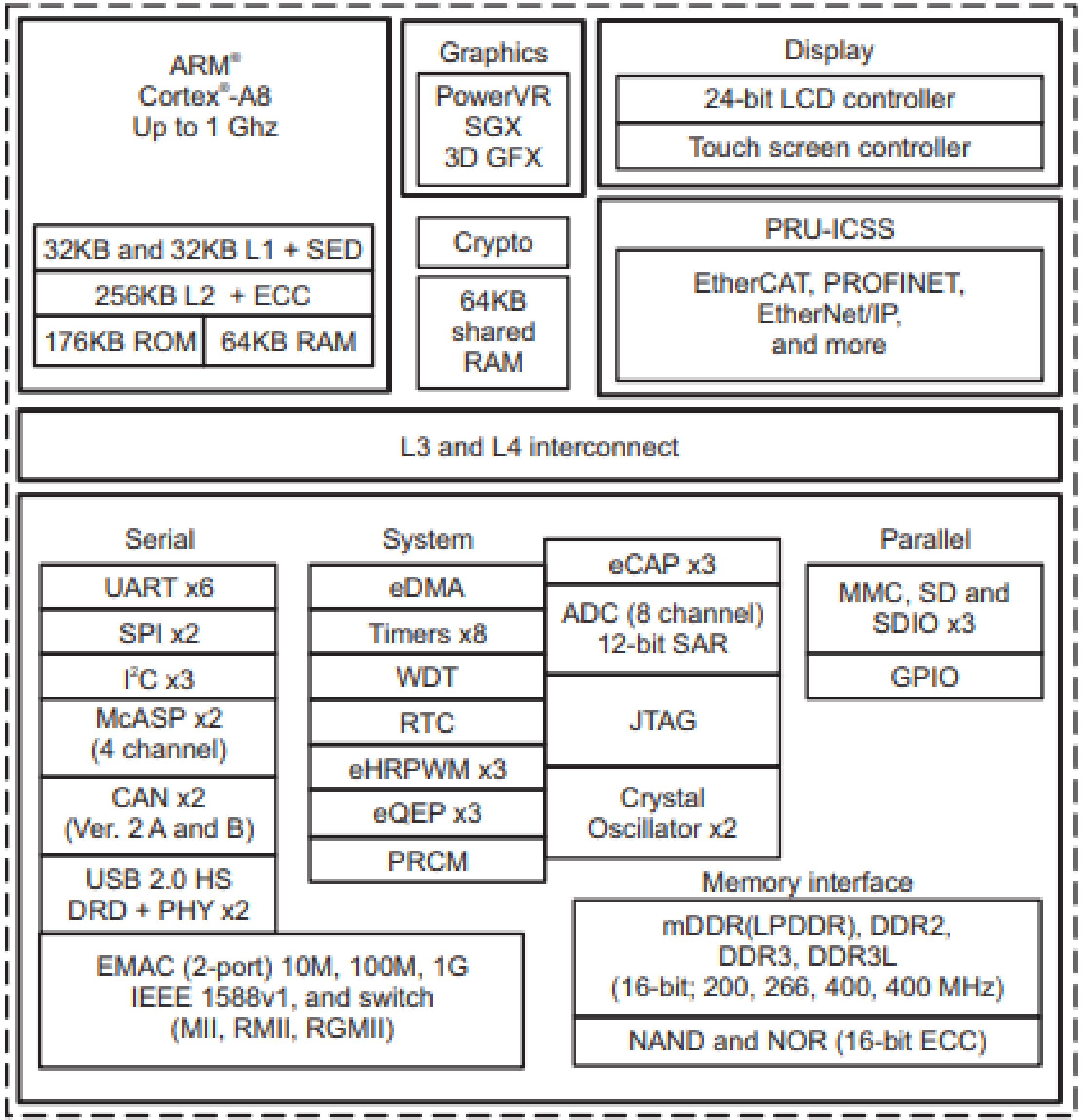


Figure 1-1. AM335x Functional Block Diagram

How much bigger is a microprocessor?

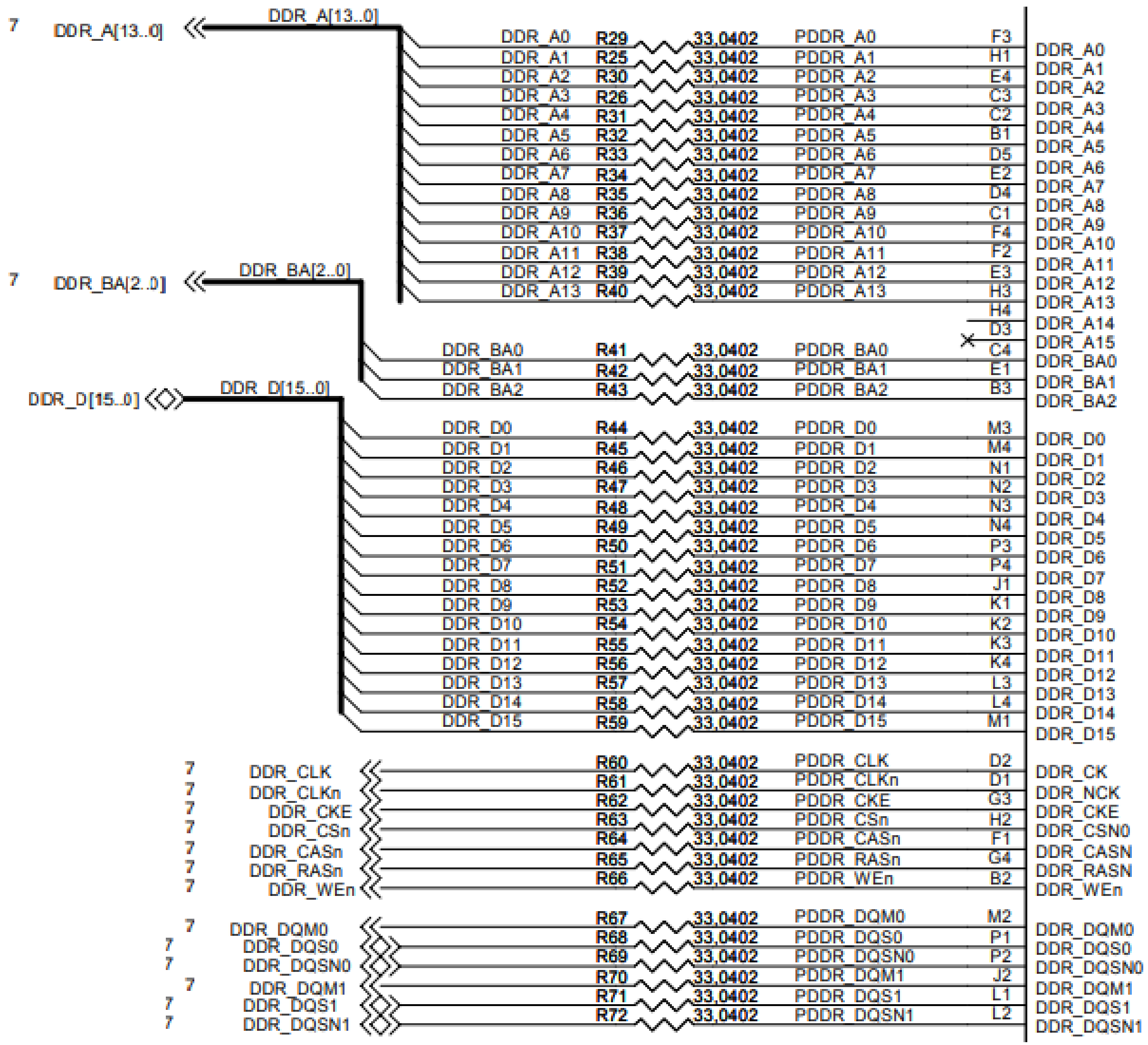
- nRF52833 datasheet:
  - 600 pages total

- AM3358 datasheet:
  - 250 pages of electrical information
  - 5000 pages of peripheral information
    - Lots of peripherals + very complicated peripherals

  - And it’s not that the basic peripherals are much larger
    - UART: 60 pages (okay, maybe a little longer)
    - I2C: 20 pages
    - Four different timers at ~30 pages each

Memory interface - DDR

Pinout on the Beaglebone



- Double Data Rate (DDR)
- Address bits
  - 17 pins (3 bank, 14 address)
  - Bank + Address selects a 2 kB chunk out of 2 GB
- Data bits
  - 16 pins
- Clocks, Chip select
- Various sequencing signals

Break + Question

- Wires (traces) for high-speed busses like memory
  must be match, to ensure that all are exactly the same length.

  Why?

Break + Question

- Wires (traces) for high-speed busses like memory
  must be match, to ensure that all are exactly the same length.

  Why?

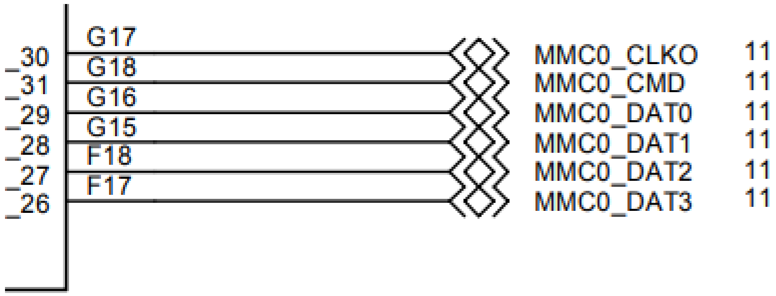
- Speed of electricity in the wires matters
- If one wire is a foot longer than the others, its signal arrives 1 ns late
  - Even fractions of nanoseconds could matter for high-speed communication

Flash interface - MMC

- Multi-media Card interface
  - Used to communicate with SD Cards
  - Also for eMMC Flash chips (which act like SD Cards, but soldered in place)

- For SD cards SPI can usually be used
- For eMMC 4-bit version of interface
  - Clock
  - Command
  - Data 0, 1, 2, 3

Pinout on the Beaglebone

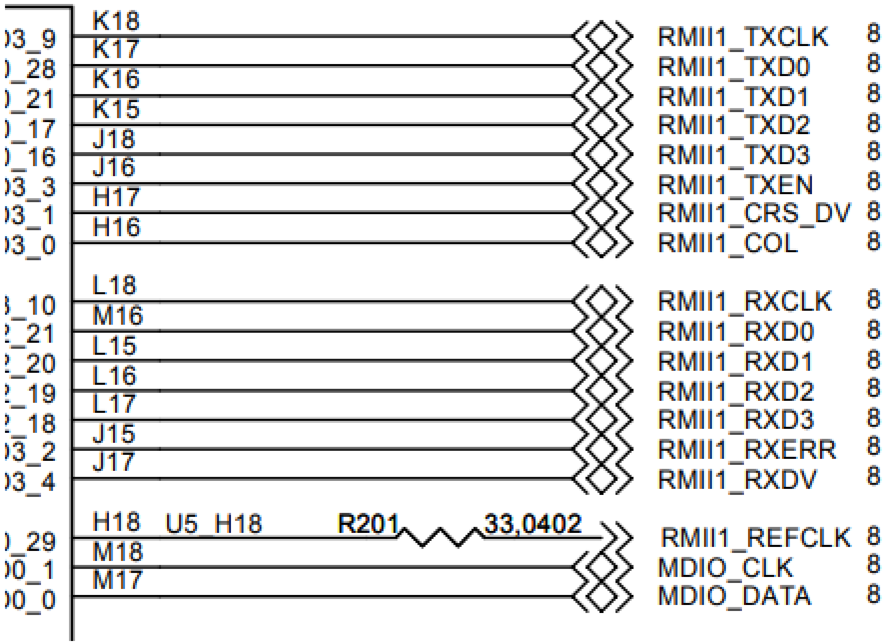


Ethernet interface - RMII

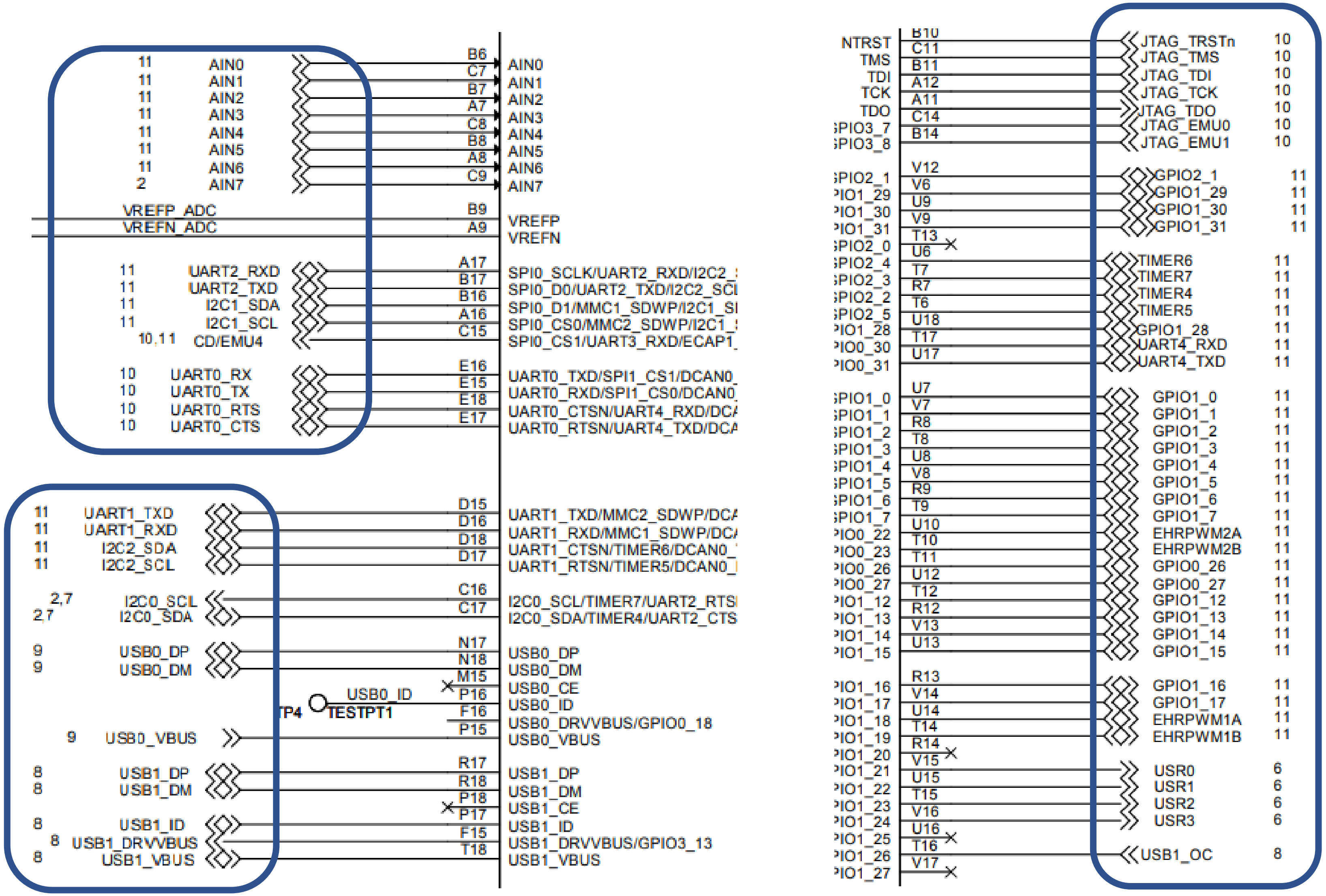
- Reduced Media-Independent Interface (RMII)

Signal name	Description
REF_CLK	Continuous 50 MHz reference clock
TXD0	Transmit data bit 0 (transmitted first)
TXD1	Transmit data bit 1
TX_EN	When high, clock data on TXD0 and TXD1 to the transmitter
RXD0	Receive data bit 0 (received first)
RXD1	Receive data bit 1
CRS_DV	Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
RX_ER	Receive error (optional on switches)
MDIO	Management data
MDC	Management data clock.

Pinout on the Beaglebone



Other processor pinouts on the Beaglebone



- Lots of acronyms that we recognize!
- Analog inputs
- UARTs
- I2C
- GPIOs
- Timers

Outline

- What haven't we talked about?
    - Microbit
    - nRF52833
- Other hardware systems
    - Other microcontroller peripherals
    - Microprocessors

Outline

- **Bonus: Other hardware systems**
  - **FPGAs**

FPGA systems

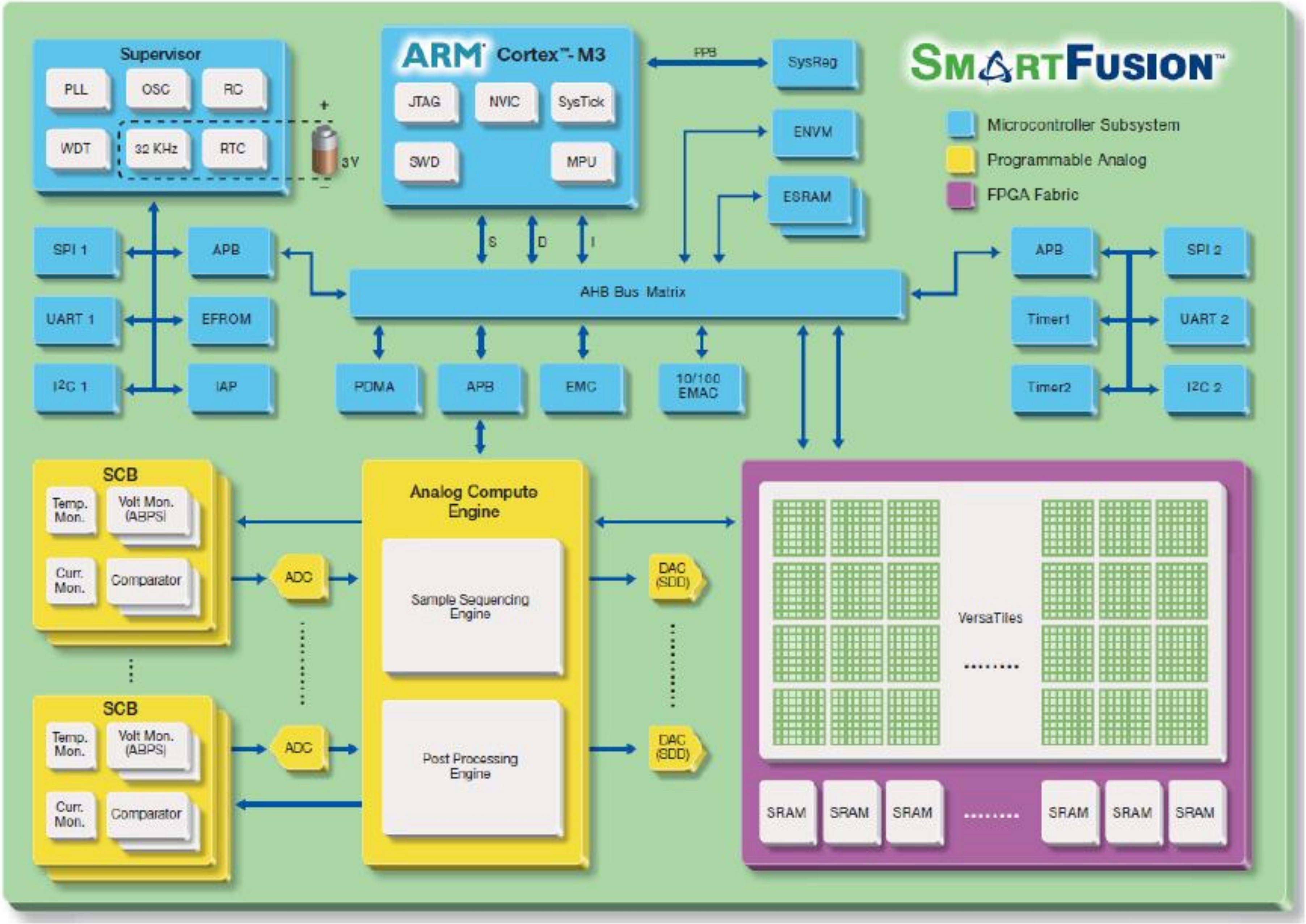
- When else might you run into large busses like memory?

- On an FPGA!
  - Field-Programmable Gate Array (FPGA)

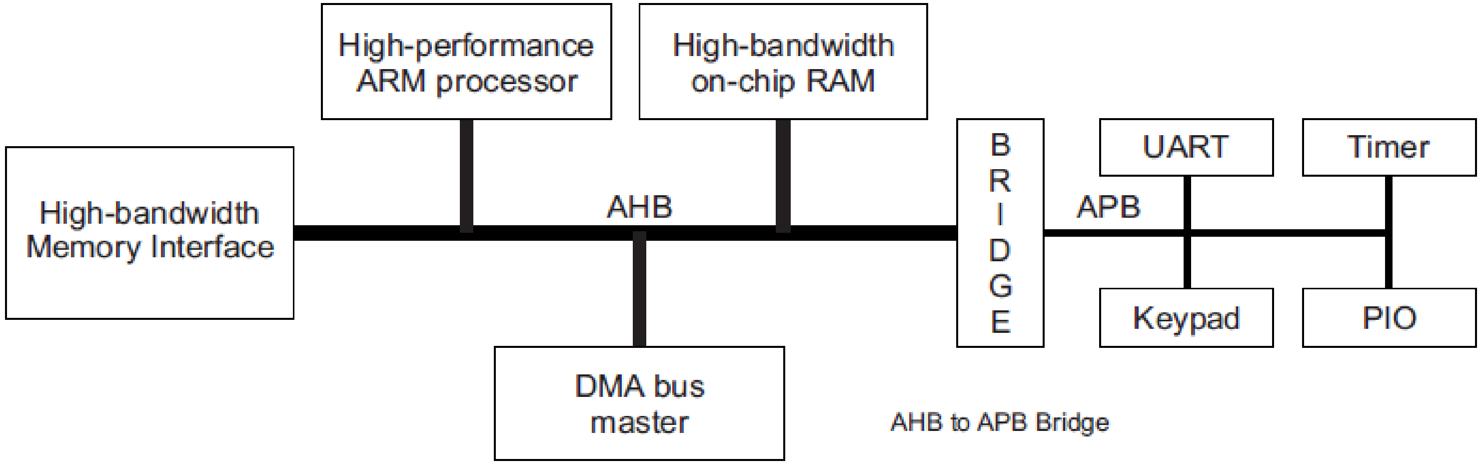
  - Configurable hardware
    - Engineers write a description of the hardware (Verilog or VHDL)
    - FPGA implements that hardware

  - Much faster than software, but still slower than a custom chip

Combined FPGA + Microcontroller



Busses: AHB vs APB



- Advanced High-performance Bus (AHB)
  - Multiple bus controllers
  - Pipelined operation
  - Burst transfers of data
  - Used for high-speed memory operations
- Advanced Peripheral Bus (APB)
  - Simpler interface
  - Lower power
  - Used for most peripheral communication

With an FPGA you can create your own peripherals

- Busses can be exposed to the FPGA for it to use

- Example use case
  - Custom camera with selectable pixel lines and an ADC
  - Don’t want to manually control it with software (too slow)

  - Create a hardware peripheral on the FPGA that takes software commands like “grab a frame” and implements them

Example register map

- Verilog HDL that describes a register map
- Reads in the address from the bus to determine which register is written to
- Modifies internal registers based on the bits of the data

```
// Interact with bus
if (bus_write) begin
    case (PADDR[7:0])
        `GLOB_START: begin
            cam0_frame_capture_start <= PWDATA[0];
            cam1_frame_capture_start <= PWDATA[1];
        end
        `CAM0_FRAMEMASK: begin
            cam0_mask_write_enable <= 1;
            cam0_mask_addr <= {PWDATA[30:24], PWDATA[18:16]};
            cam0_mask_data <= PWDATA[15:0];
        end
        `CAM0_SETTINGS1: begin
            cam0_vref_value   <= PWDATA[29:24];
            cam0_config_value <= PWDATA[21:16];
            cam0_nbias_value  <= PWDATA[13:8];
            cam0_aobias_value <= PWDATA[5:0];
        end
        `CAM0_SETTINGS2: begin
            cam0_val_offset <= PWDATA[27:16];
            cam0_vsw_value  <= PWDATA[15:8];
            cam0_hsw_value  <= PWDATA[7:0];
        end
```

Performing peripheral operations

- On every clock edge, check status registers

- If they are active, iterate through a state machine
  - Possibly many states to make the actual operation occur

```
`IDLE: begin
    if (frame_capture_start) begin
        mask_pixel_col_nxt = 0;
        mask_pixel_row_nxt = 0;
        main_state_nxt = `CAPTURE;
    end else begin
        frame_capture_done_nxt = 0;
        main_state_nxt = `IDLE;
    end
end
```