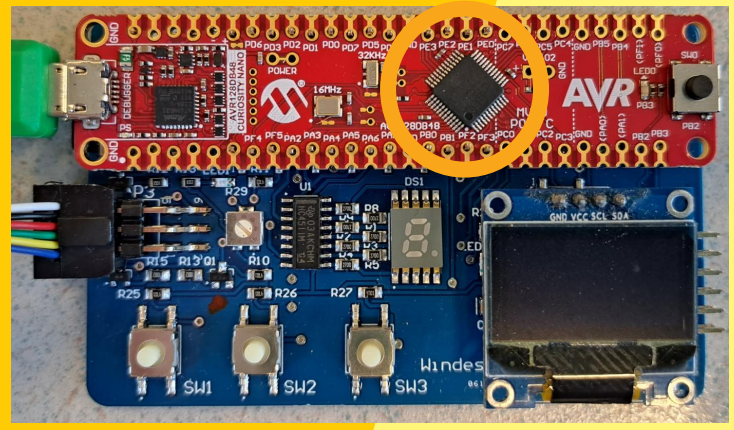
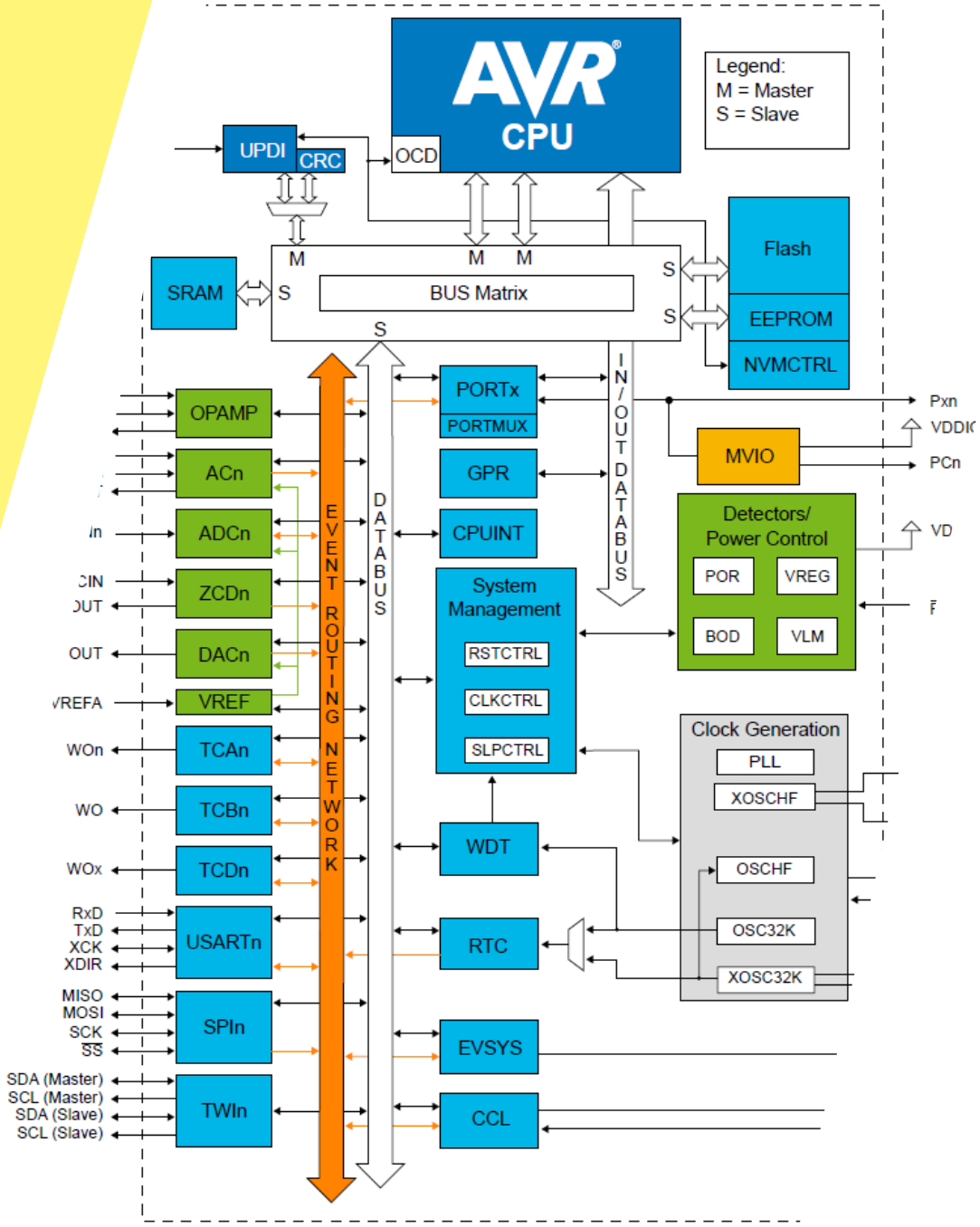
hogeschool Windesheim
Dichter bij jou

Microcontroller Programmeren

Lesweek 1

Docenten:





Wie is wie?

Lab-beheerders

Henk Bouwman

Marcello Roosenstein

Docenten

Douwe Schotanus

Marco Winkelman

Richard Rosing

Wat gaan we doen?

Week 1:	Inleiding, Microcontroller Architectuur, GPIO, debuggen
Week2:	Programma flow, toestanden, state machine
Week3:	Interrupts
Week4:	Timers
Week5:	PWM
Week6:	ADC

Wat wordt er van je verwacht?

Lesvoorbereiding

- Iedere week zijn er een aantal ‘thuisopdrachten’. Maak deze thuis ter voorbereiding aan de les.

Actieve deelname aan de practicumlessen

- Voer de ‘labopdrachten’ op school uit
- Help elkaar
- Stel vragen

Toetsing

Er zijn elke week labopdrachten

- Deze moeten afgetekend zijn om aan de eindtoets te mogen deelnemen.

De eindtoets is een practicumtoets

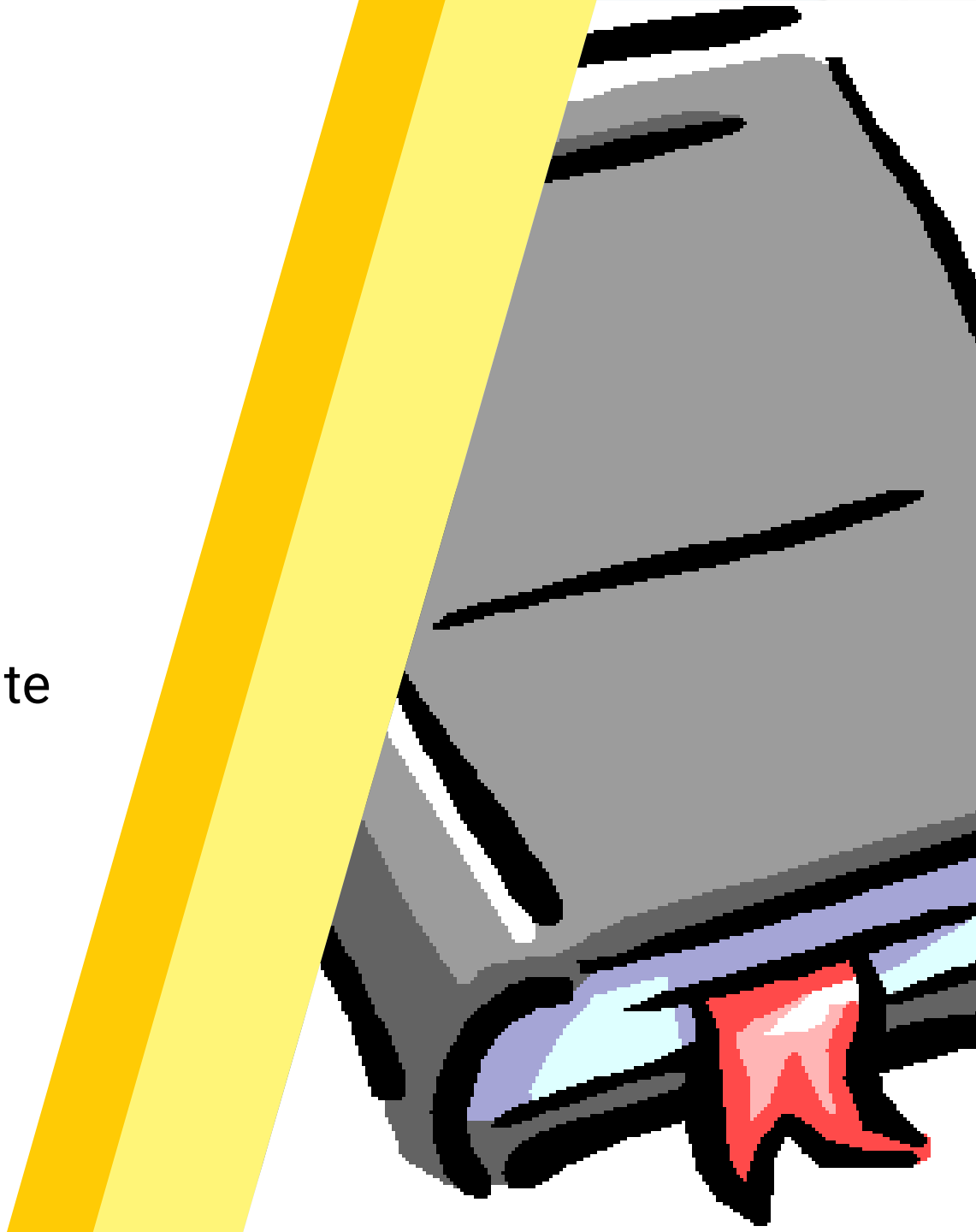
- Theorie en vaardigheden die je tijdens het project hebt geleerd worden getoetst.

Iedereen wordt individueel getoetst!

- Je mag wel samenwerken tijdens de lessen.

Logboek

- Een logboek/labjournaal is een soort technisch dagboek waarin je voor jezelf aantekeningen maakt.
- Je schrijft er alles in op wat je denkt nodig te hebben als je de opdracht nog eens zou moeten maken.
- Je mag dit digitaal of analoog doen:
  - Digitaal met het word-document met opgaven als uitgangspunt.
  - Analoog: opschrijven in een schrift.



Labregels

Lees het document met labregels op de ELO!

- Werk rustig, netjes en ordelijk.
- Niet roken, eten of drinken in de werkplaatsen en laboratoria.
- De student is verplicht gebruikte apparatuur, gereedschap of hulpmiddelen na gebruik in goede staat achter te laten op de bestemde plaats, de werkplek netjes op te ruimen.
- Verlaat je als laatste het lokaal, sluit dan ramen en deuren, schakel alle apparatuur uit en schakel de hoofdschakelaar van de ruimte uit.
- Het gebruik mobiele telefoons en ‘social media’ is verboden tenzij t.b.v. de practica.
- Het afspelen van media is verboden tenzij t.b.v. de practica.

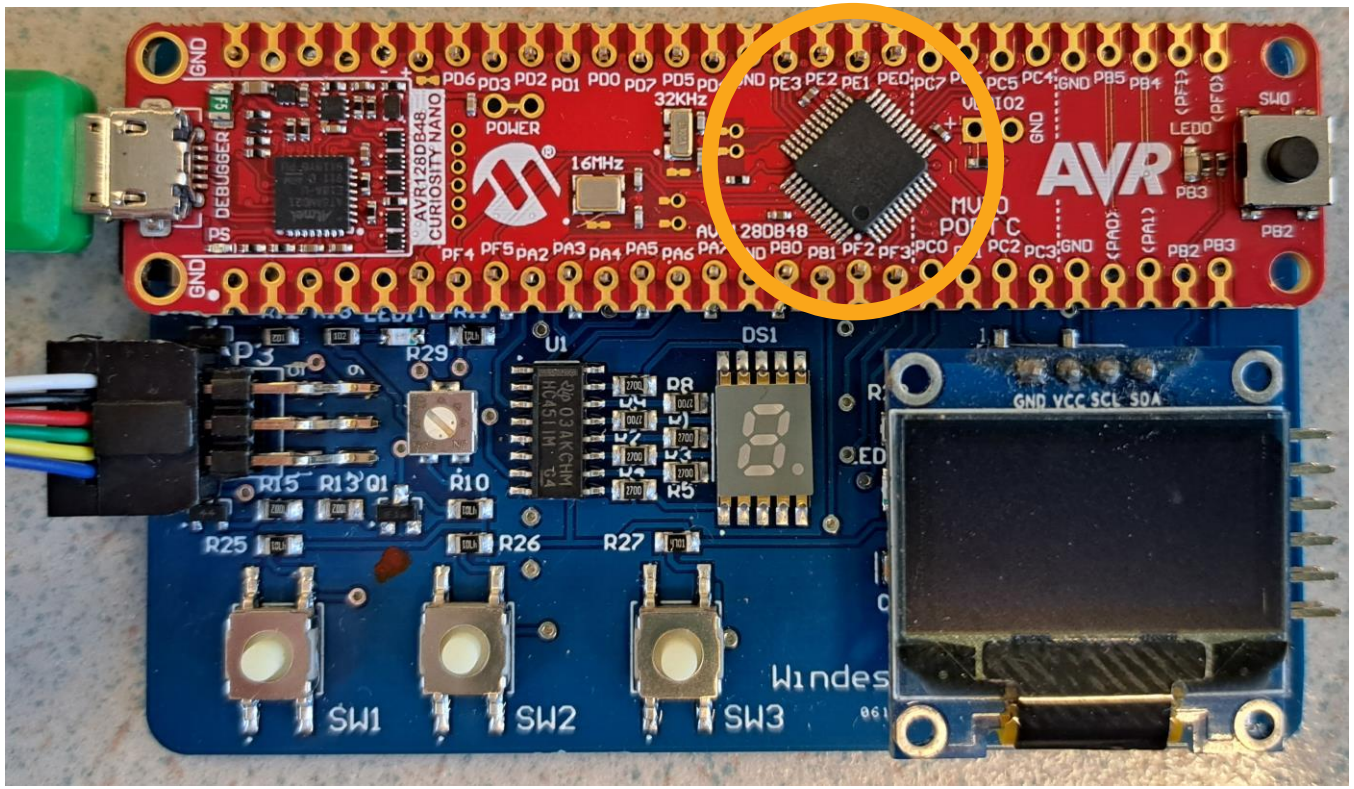
- Er wordt gewerkt met elektrische spanningen, waarbij een fout consequenties kan hebben voor componenten, apparatuur of zelfs personen; werk veilig en overzichtelijk.
- Blus elektriciteitsbranden uitsluitend met de blusser! (niet met water vanwege schokgevaar)
- Defecte of beschadigde apparatuur en snoeren niet gebruiken en melden bij de beheerder.

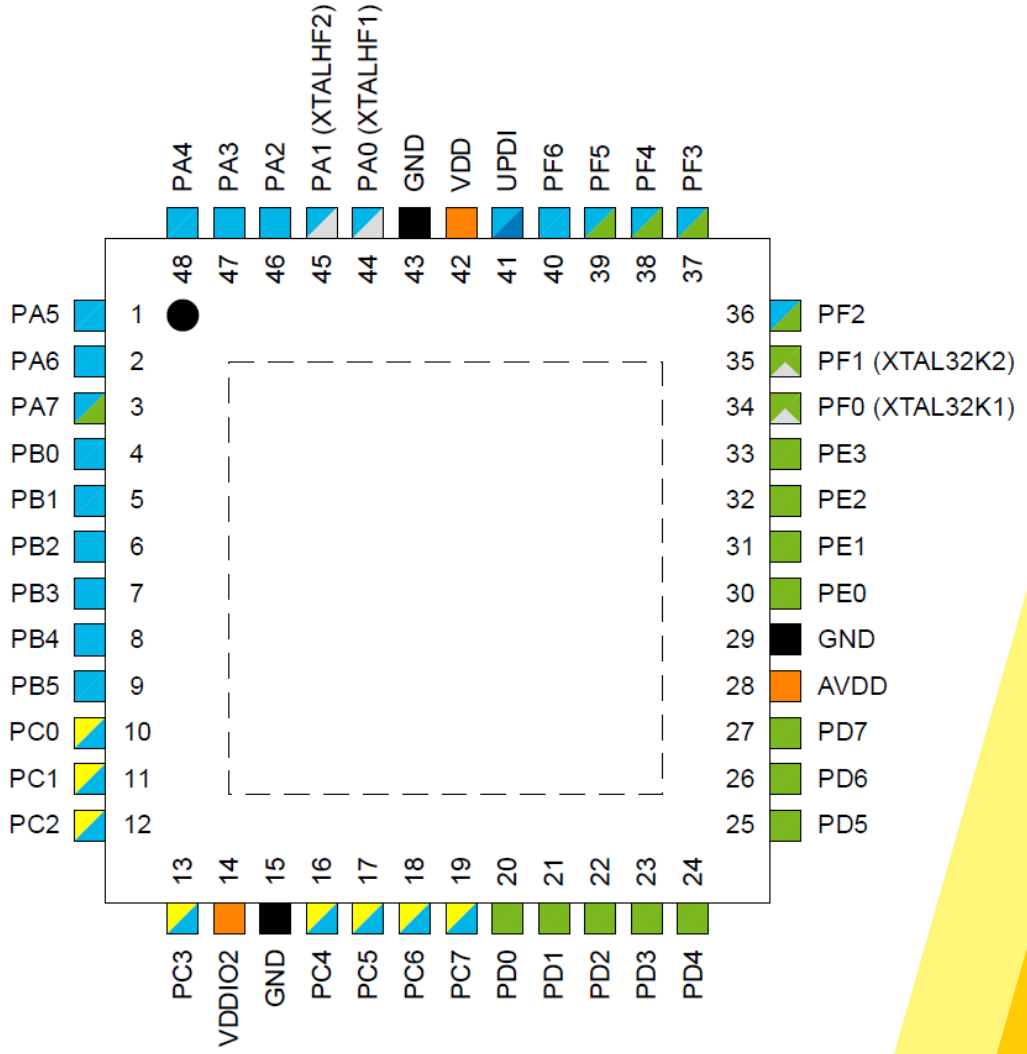
hogeschool Windesheim
Dichter bij jou

Kennismaking met de Microcontroller

(Dit is een stukje herhaling van Digitale Techniek periode 1)

Onze microcontroller

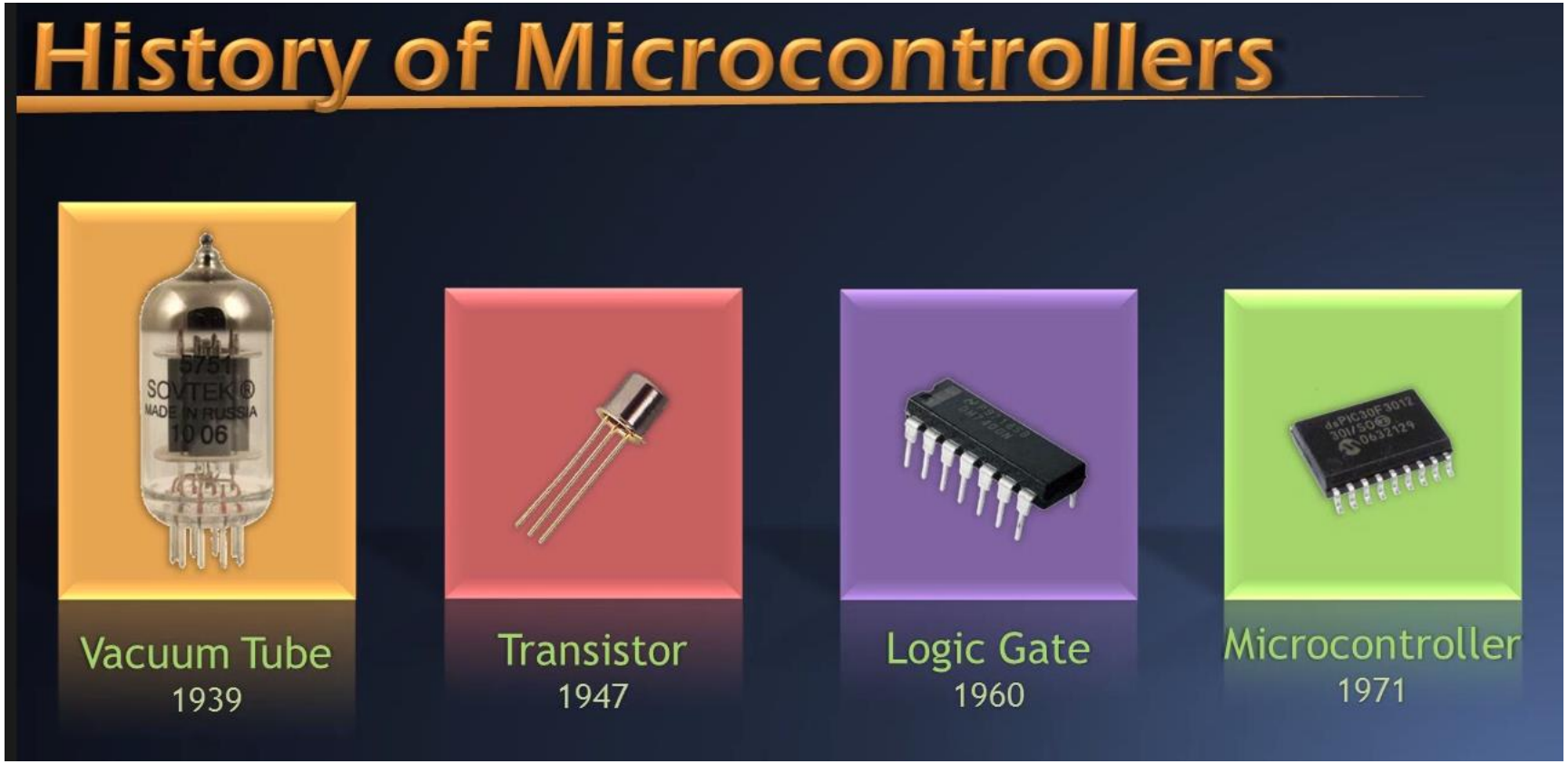




Microcontroller

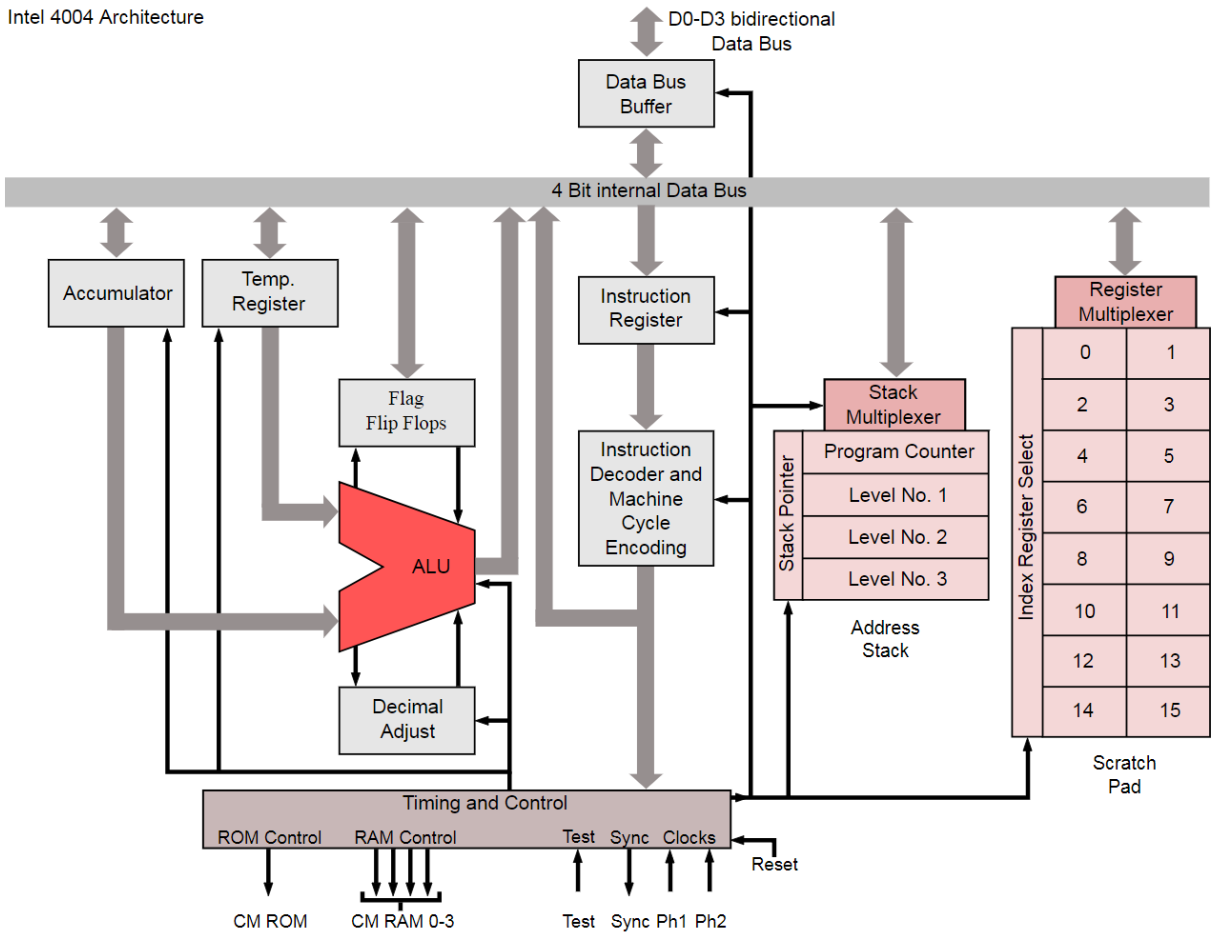
- Geïntegreerd Circuit (IC) dat geprogrammeerd wordt voor een specifieke taak
- Zijn een soort “mini computers” met veel geïntegreerde mogelijkheden
- Zijn te vinden in vele miljarden apparaten (tv, koelkast, magnetron, toetsenbord, mobiele telefoon, routers, oortjes etc.)
- Toegepast in specifieke applicaties
- Snelheid is niet nodig
  - Paar MHz voldoende
  - Weinig geheugen nodig
- Moet worden geprogrammeerd om te functioneren
  - Is dus nooit slimmer dan zijn programma!

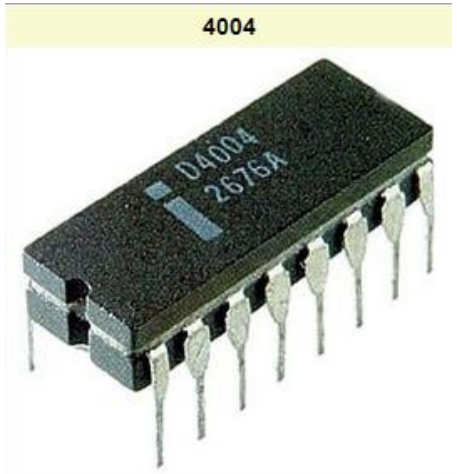
	Laptop	Microcontroller
RAM geheugen	8 Gb	16 kB
Schijf (installeren programma’s)	SSD 256 Gb	Flash 128 kB
Kloksnelheid	2,85 Ghz	24 MHz


History of Microcontrollers
5751
SOVTEK®
MADE IN RUSSIA
10 06
Vacuum Tube
1939
Transistor
1947
Logic Gate
1960
Microcontroller
1971

Een van de eerste 1-chip microprocessors (I): Intel 4004

Released: 1971 als eerste uit de x86 familie

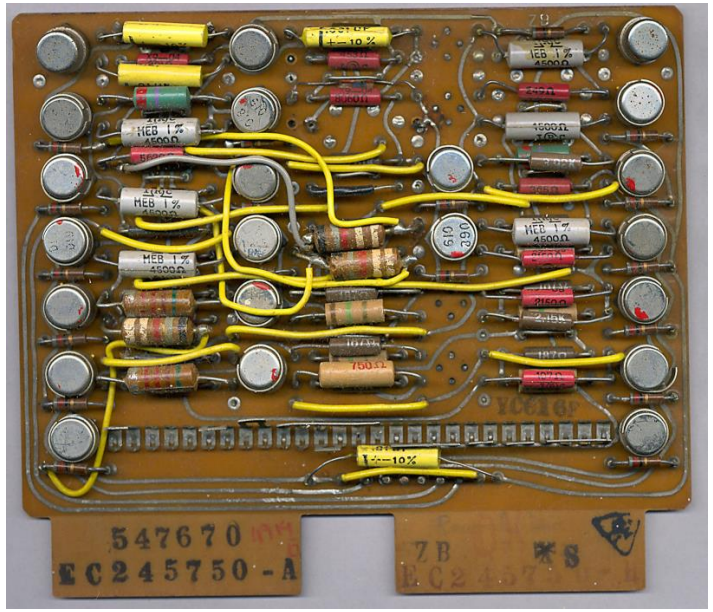




4004	
Intel 4004D	
Registerbreedte	4 bits
Fabrikant	Intel
Begonnen in	1971
Geëindigd in	1978
Klokfrequentie	740 kHz
Schaal	10 µm
Instructieset	4-bit BCD oriented
Level-1 cache	geen
Voedingsspanning	15 volt
Aantal pinnen	16
Details	
Aantal transistors	2300
Aantal registers	16
Breedte databus	4 bits
Adresruimte	1 KB
Aantal instructies	46

https://nl.wikipedia.org/wiki/Intel_4004

Wat hadden we vóór de 1-chip CPU ?



http://ummr.altervista.org/before_microprocessors.htm

De microprocessor(II)

- Texas Instruments
- Start: begin jaren 1970
- 1974: TMS 1000. microcontroller met 4 bits brede bus.

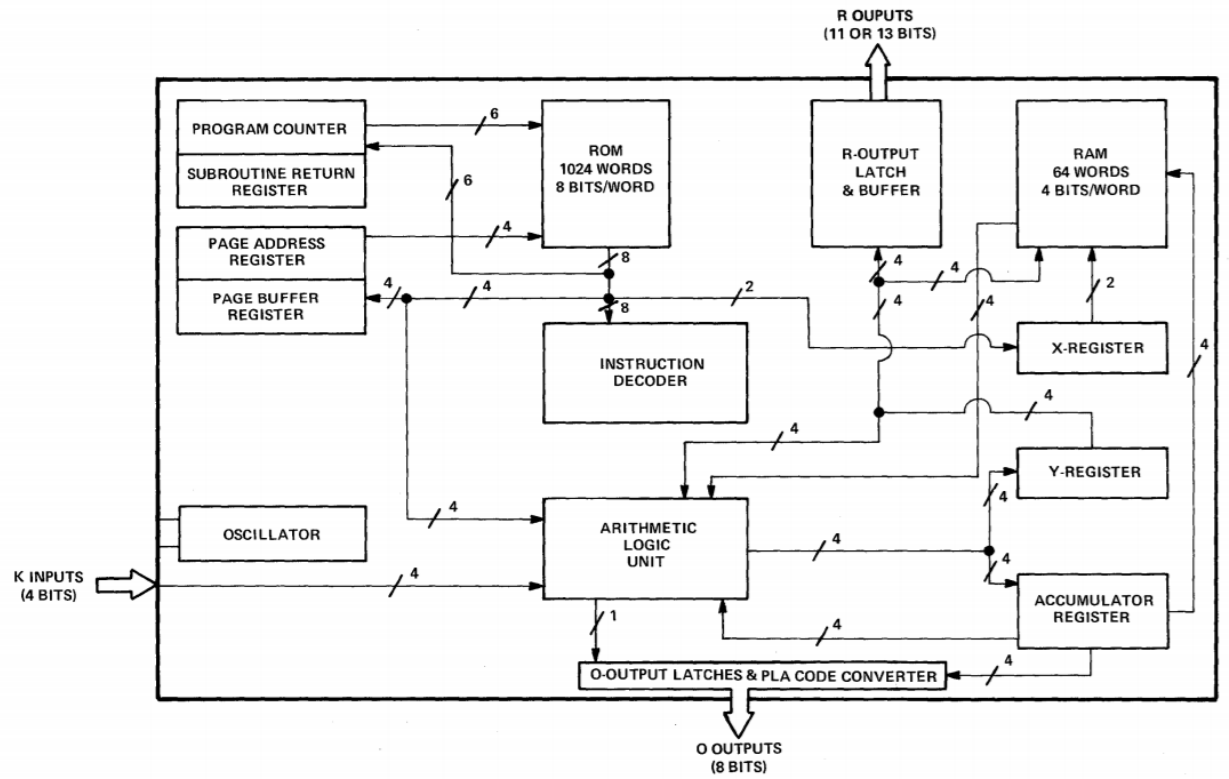
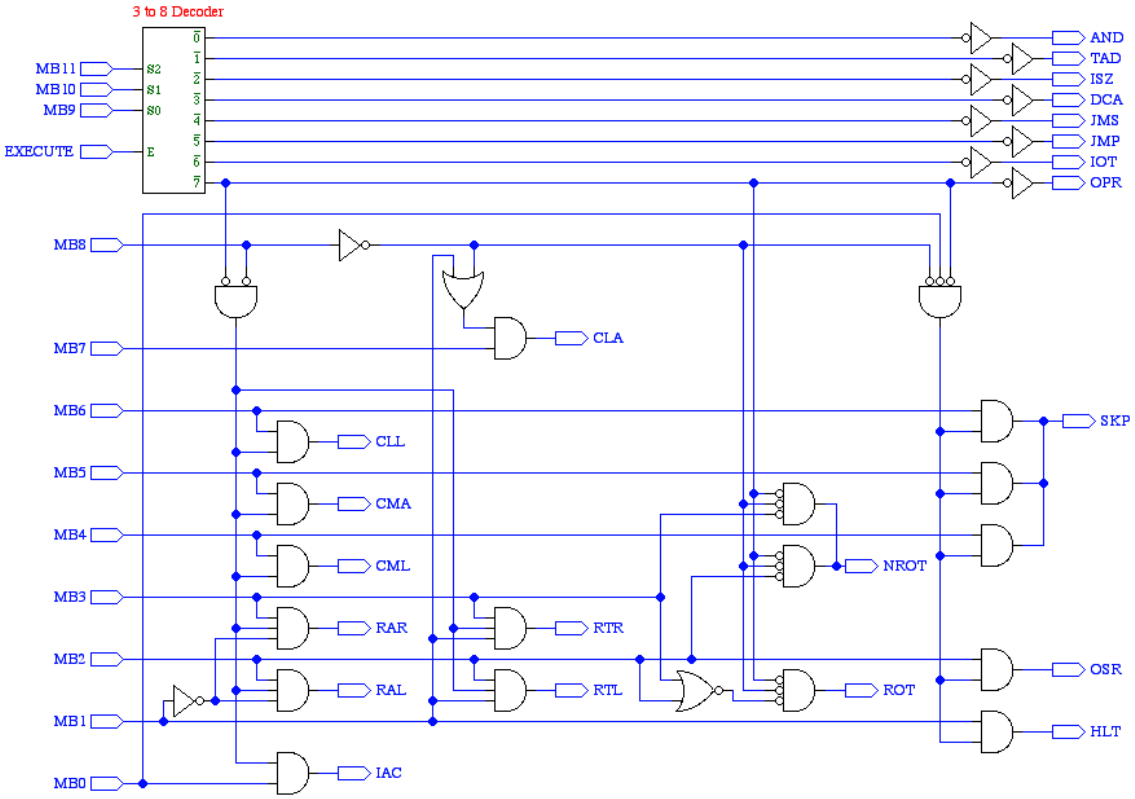
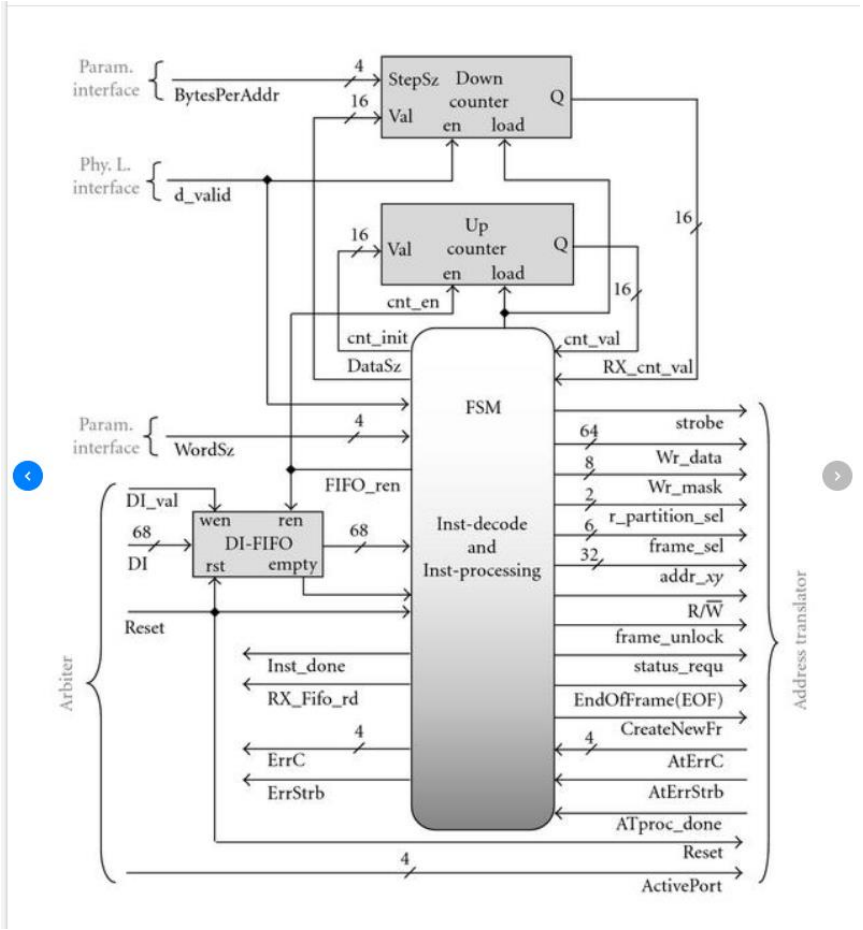


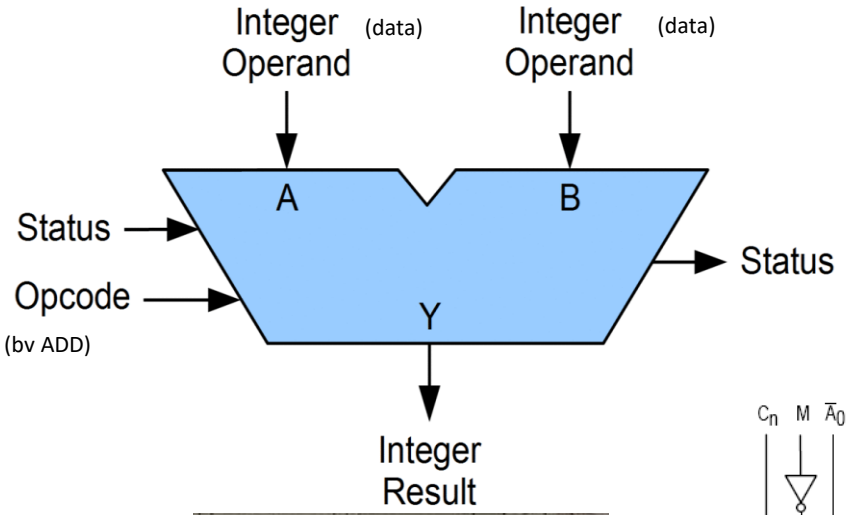
FIGURE 1—TMS1000-SERIES LOGIC BLOCKS

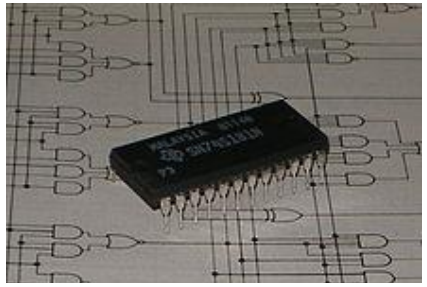
Instruction decoder = hardware

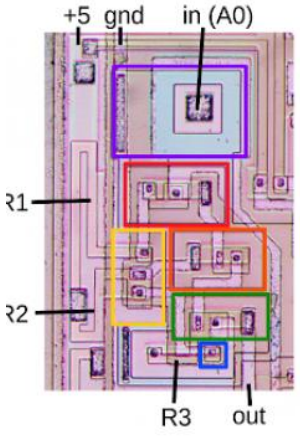


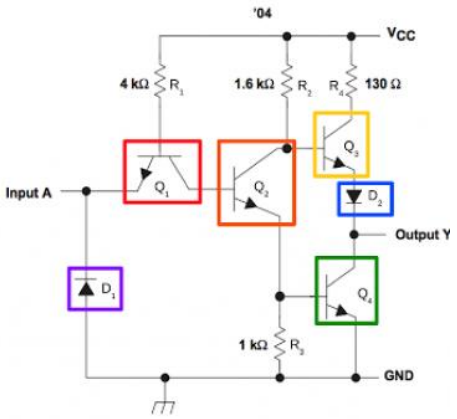


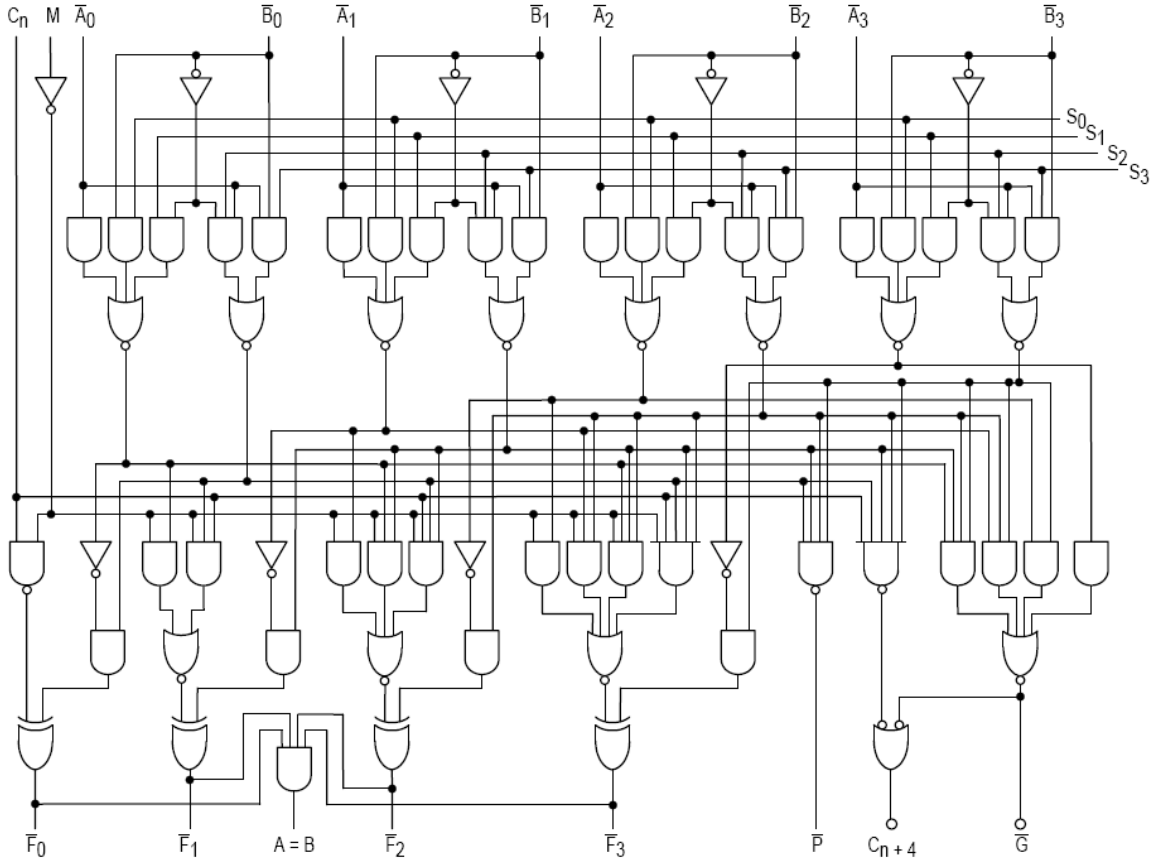
ALU = ook hardware











Functies ALU (1)

Arithmetic operations
Add: A and B are summed and the sum appears at Y and carry-out.
Add with carry: A, B and carry-in are summed and the sum appears at Y and carry-out.
Subtract: B is subtracted from A (or vice versa) and the difference appears at Y and carry-out. For this function, carry-out is effectively a "borrow" indicator. This operation may also be used to compare the magnitudes of A and B; in such cases the Y output may be ignored by the processor, which is only interested in the status bits (particularly zero and negative) that result from the operation.
***Subtract with borrow*:** B is subtracted from A (or vice versa) with borrow (carry-in) and the difference appears at Y and carry-out (borrow out).
***Two's complement (negate)*:** A (or B) is subtracted from zero and the difference appears at Y.
***Increment*:** A (or B) is increased by one and the resulting value appears at Y.
***Decrement*:** A (or B) is decreased by one and the resulting value appears at Y.
***Pass through*:** all bits of A (or B) appear unmodified at Y. This operation is typically used to determine the parity of the operand or whether it is zero or negative, or to load the operand into a processor register.

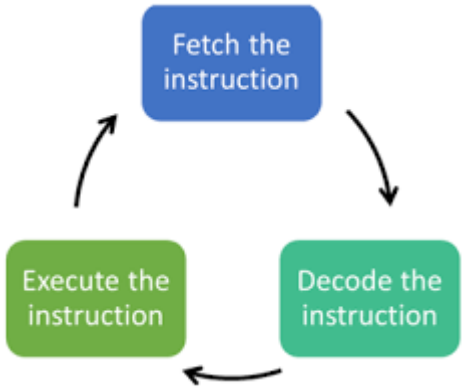
Bitwise logical operations[edit]
AND: the bitwise AND of A and B appears at Y.
OR: the bitwise OR of A and B appears at Y.
Exclusive-OR: the bitwise XOR of A and B appears at Y.
Ones' complement: all bits of A (or B) are inverted and appear at Y.

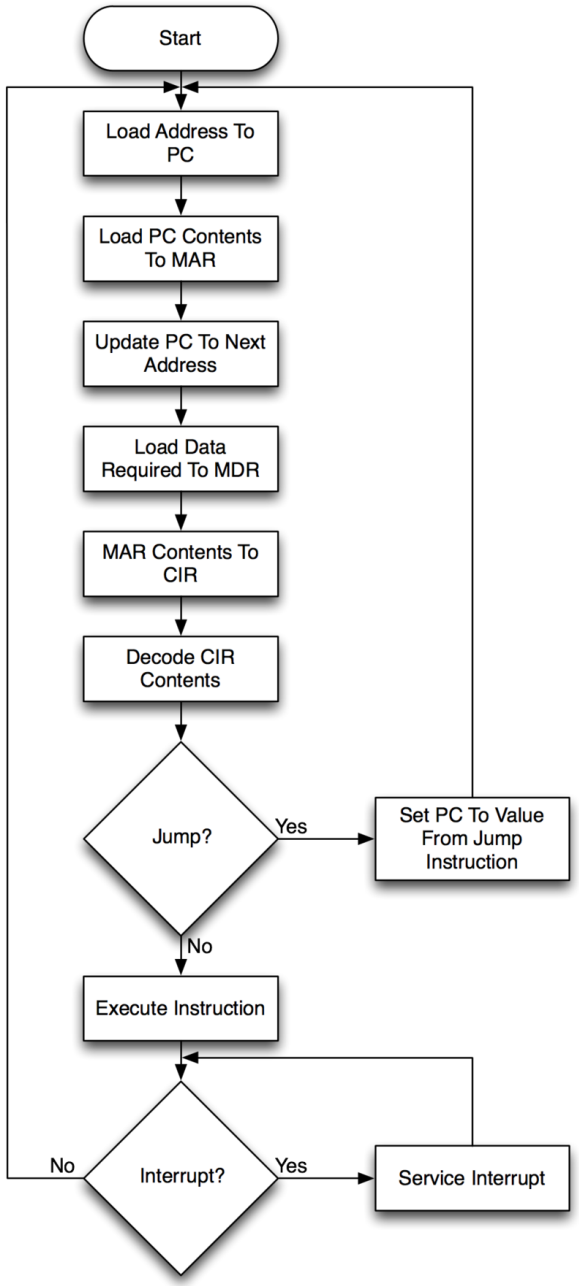
Functies ALU (2)

ALU shift operations cause operand A (or B) to shift left or right (depending on the opcode) and the shifted operand appears at Y. Simple ALUs typically can shift the operand by only one bit position, whereas more complex ALUs employ barrel shifters that allow them to shift the operand by an arbitrary number of bits in one operation. In all single-bit shift operations, the bit shifted out of the operand appears on carry-out; the value of the bit shifted into the operand depends on the type of shift.

- *Arithmetic shift*: the operand is treated as a two's complement integer, meaning that the most significant bit is a "sign" bit and is preserved.
- *Logical shift*: a logic zero is shifted into the operand. This is used to shift unsigned integers.
- *Rotate*: the operand is treated as a circular buffer of bits so its least and most significant bits are effectively adjacent.
- *Rotate through carry*: the carry bit and operand are collectively treated as a circular buffer of bit

Instruction cycle





PC = Program Counter

(locatie van de instructie)

MAR = Memory Address Register

(locatie van de data)

MDR = Memory Data Register

(bevat de data)

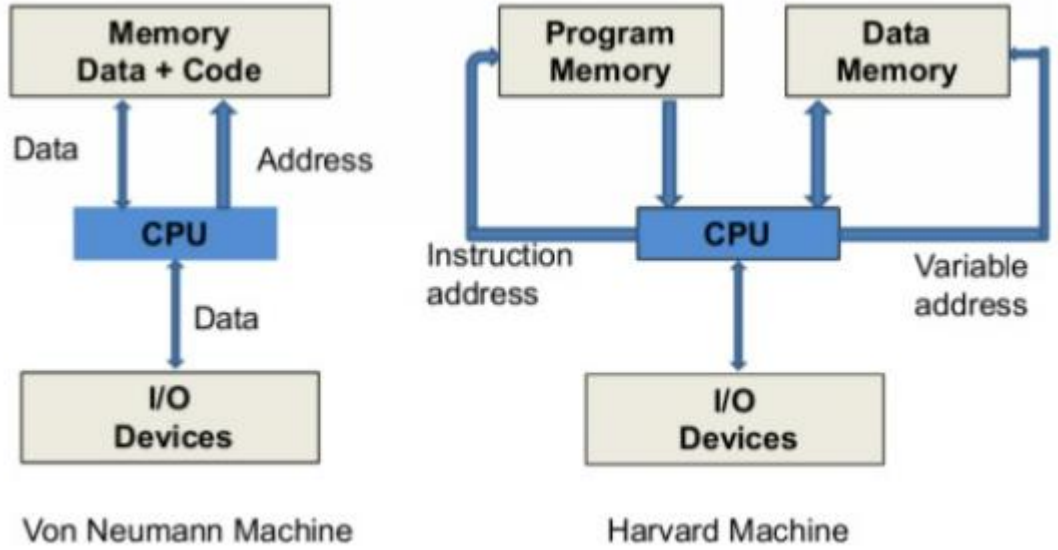
CIR = Current Instruction Register

(bevat instructie die uitgevoerd moet worden door de ALU)

CPU, ALU, MBR, MDR, MAR, CIR

- https://web.archive.org/web/20090611211308/http:/www.cs.montana.edu/~bosky/cs518/ife/IFE.pdf

Harvard of Von Neumann



Von Neumann (1944)

CPU moet bus delen

- Data en instructies via één bus
- Instructie ophalen en data-verwerking kan niet tegelijk

Harvard:

CPU heeft verschillende verbindingen (bussen) met:

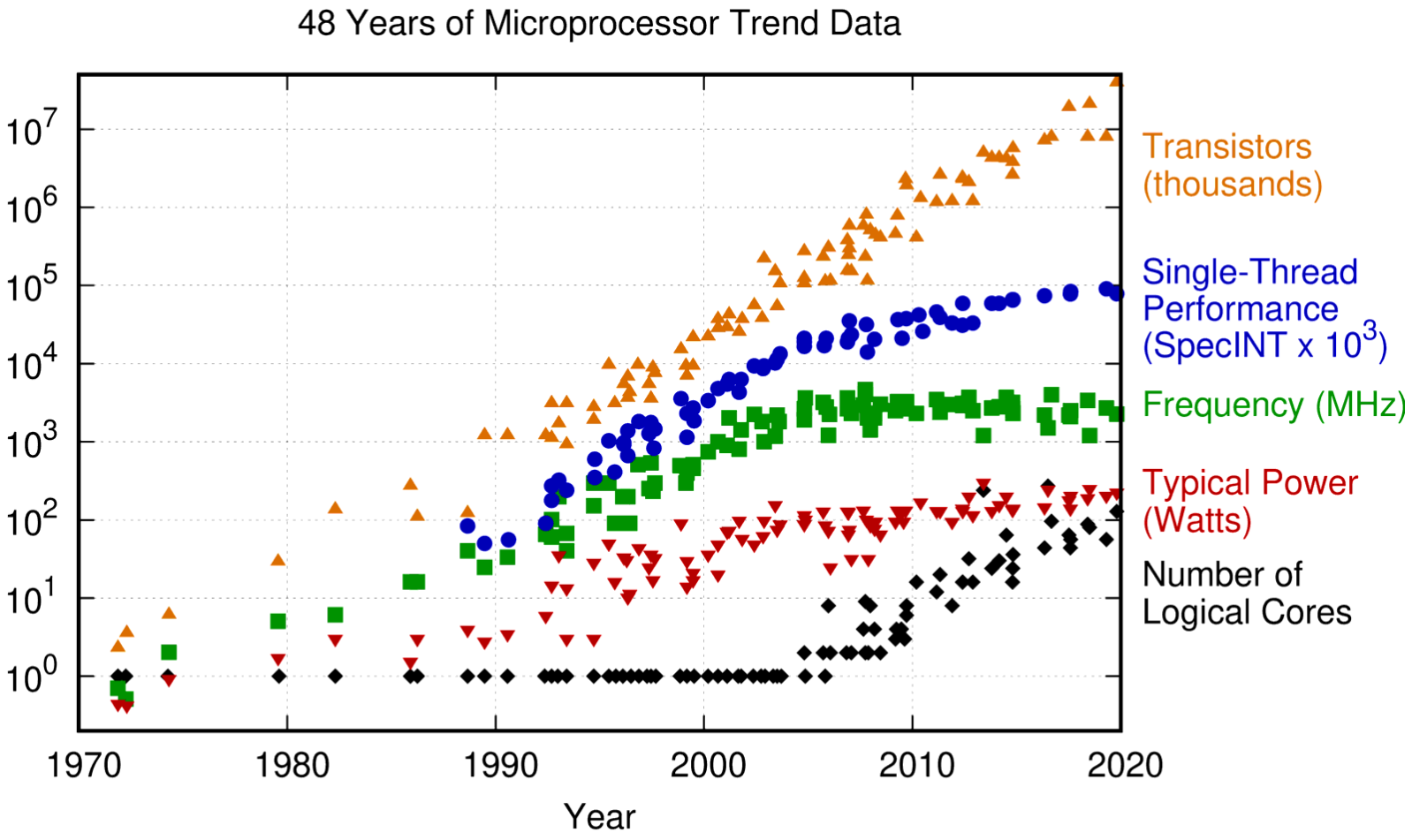
- Instructie geheugen
- Data geheugen

Hierdoor sneller: instructie kan worden opgehaald tijdens bewerking van data

7.3 Architecture

To maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate buses for program and data. Instructions in the program memory are executed with a single-level pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This enables instructions to be executed on every clock cycle.

Moore's Law



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

Semiconductor device fabrication

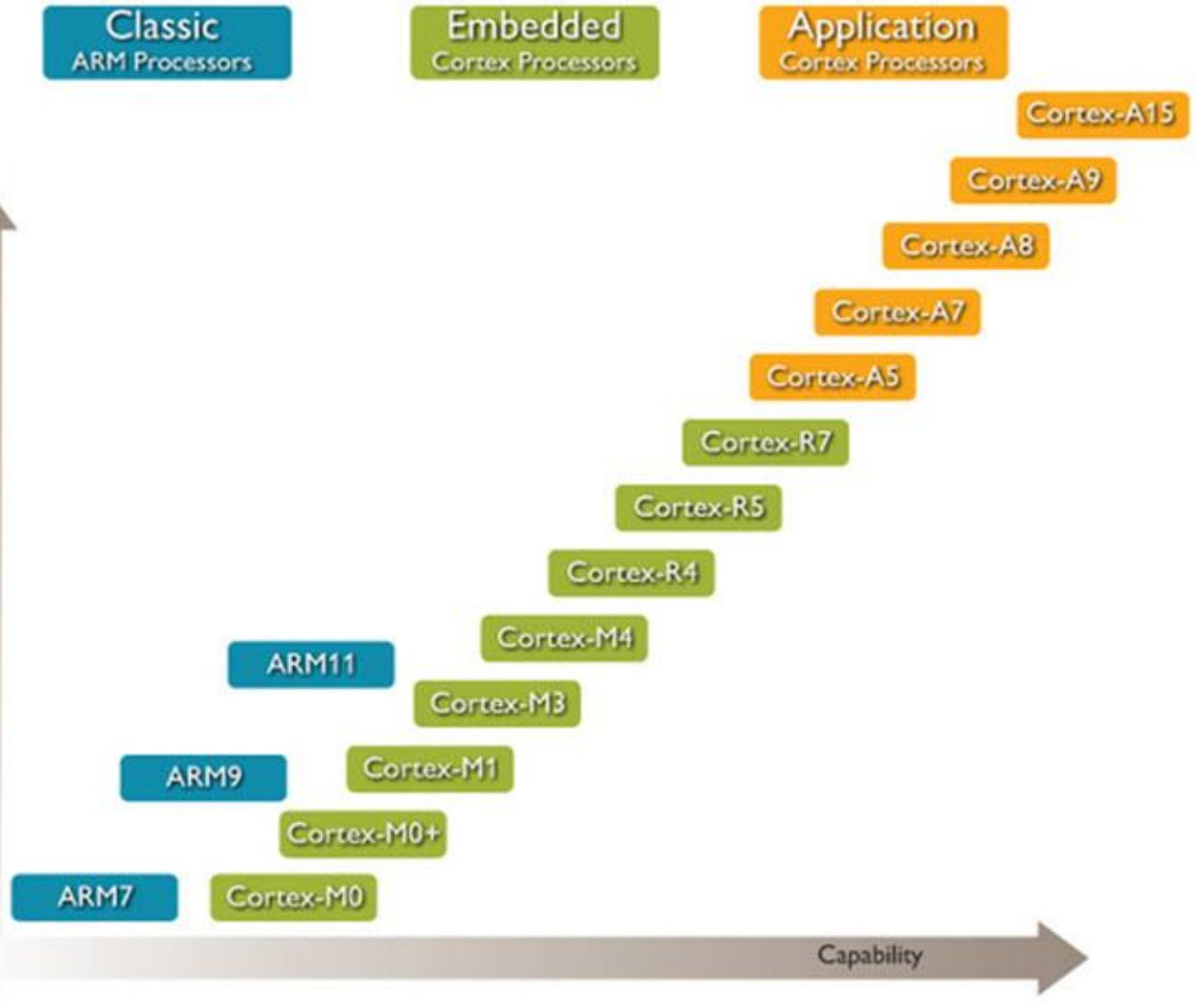
MOSFET scaling (process nodes)

10 µm – 1971
6 µm – 1974
3 µm – 1977
1.5 µm – 1981
1 µm – 1984
800 nm – 1987
600 nm – 1990
350 nm – 1993
250 nm – 1996
180 nm – 1999
130 nm – 2001
90 nm – 2003
65 nm – 2005
45 nm – 2007
32 nm – 2009
22 nm – 2012
14 nm – 2014
10 nm – 2016
7 nm – 2018
5 nm – 2020

Future

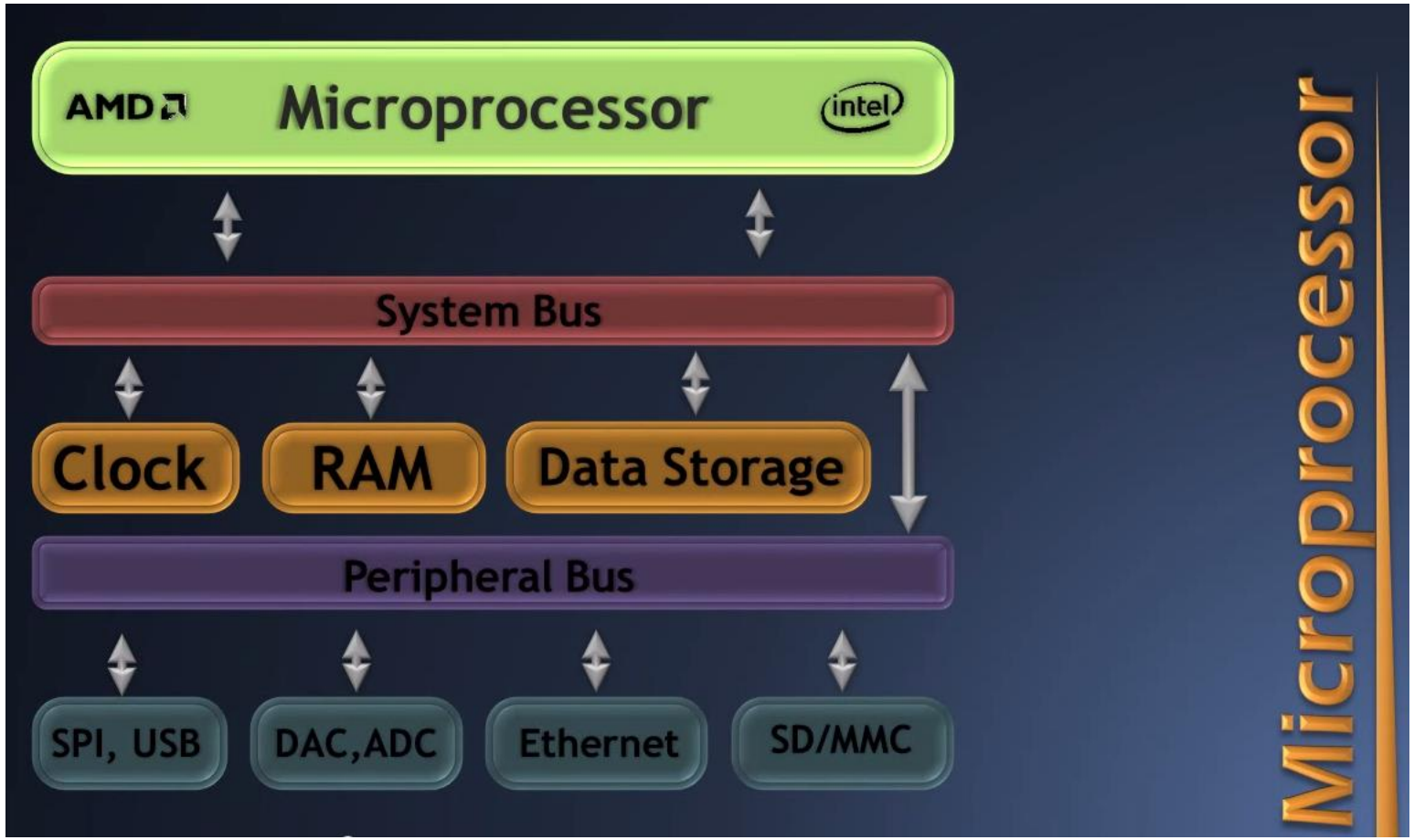
3 nm – ~2022

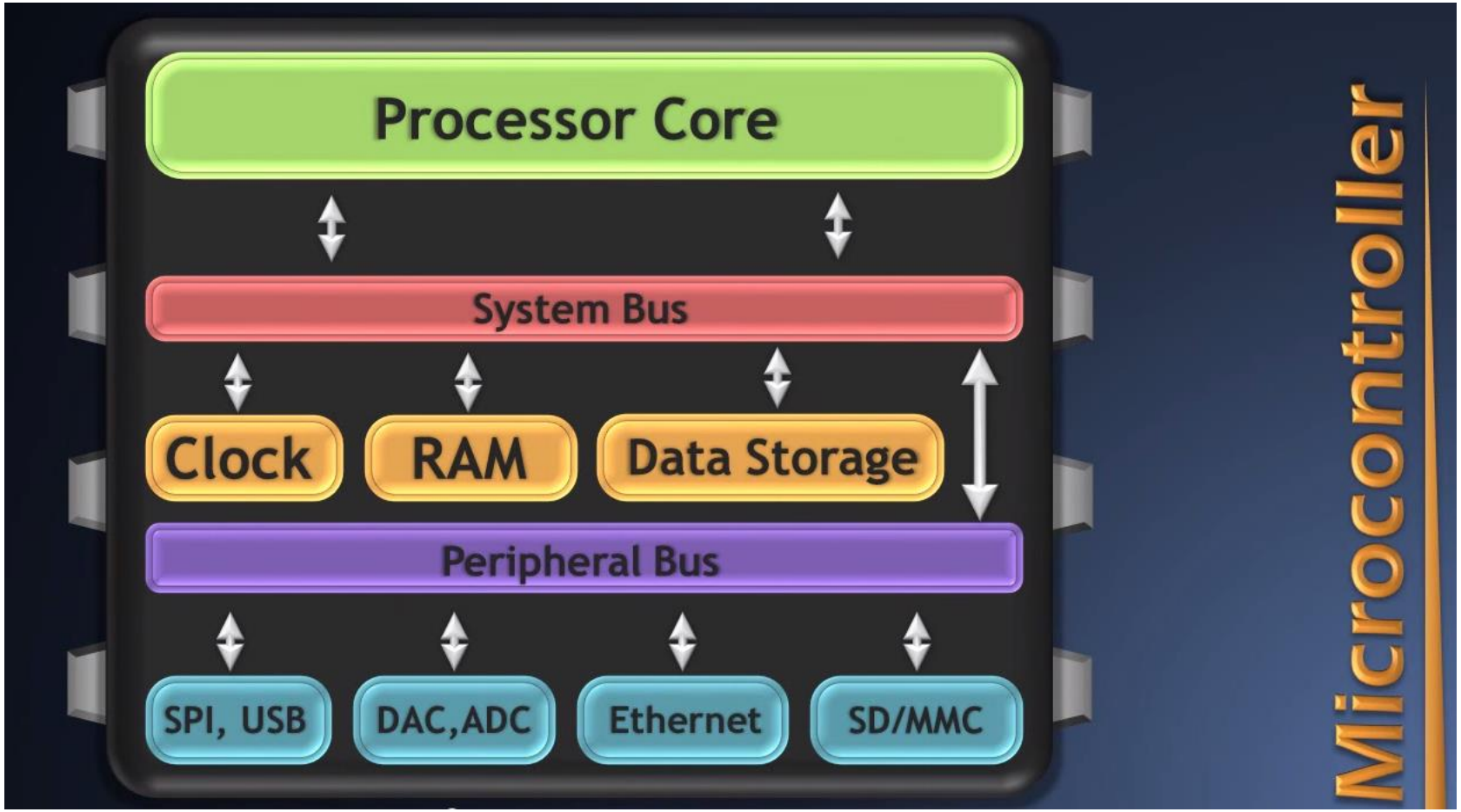
ARM family


Classic
ARM Processors
Embedded
Cortex Processors
Application
Cortex Processors
Cortex-A15
Cortex-A9
Cortex-A8
Cortex-A7
Cortex-A5
Cortex-R7
Cortex-R5
Cortex-R4
Cortex-M4
ARM11
Cortex-M3
Cortex-M1
ARM9
Cortex-M0+
ARM7
Cortex-M0
Performance, Functionality
Capability

Microprocessor vs Microcontroller

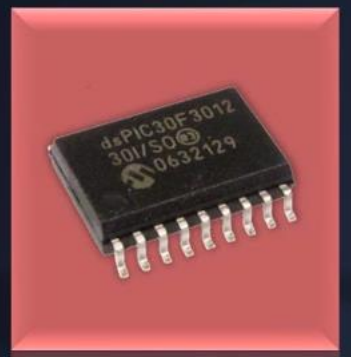
	Microprocessor	Microcontroller
Applications	General computing (i.e. Laptops, tablets)	Appliances, specialized devices
Speed	Very fast	Relatively slow
External Parts	Many	Few
Cost	High	Low
Energy Use	Medium to high	Very low to low
Vendors	intel AMD ARM	ATMEL ST Texas Instruments Microchip


AMD
Microprocessor
intel
System Bus
Clock
RAM
Data Storage
Peripheral Bus
SPI, USB
DAC,ADC
Ethernet
SD/MMC
Microprocessor


Processor Core
System Bus
Clock
RAM
Data Storage
Peripheral Bus
SPI, USB
DAC,ADC
Ethernet
SD/MMC
Microcontroller

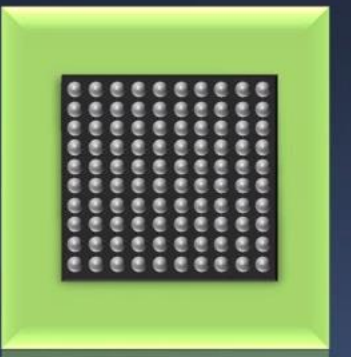
Microcontroller Packaging

DIP
(Dual Inline Package)
Through hole
8 pins
9mm x 6mm
0.15pins/mm^2



SOIC
(Small Outline IC)
Surface Mount
18 pins
11mm x 7mm
0.23pins/mm^2

QFP
(Quad Flat Package)
Surface Mount
32 pins
7mm x 7mm
0.65pins/mm^2



BGA
(Ball Grid Array)
Surface Mount
100 pins
6mm x 6mm
2.78pins/mm^2

Merken Microcontrollers

- Atmel (overgenomen door Microchip)
  - de microcontroller die op de bekende Arduino borden zit
- Microchip:
  - veel gebruikt in hobby toepassingen
- STMicroelectronics (Nucleo)
  - populaire familie microcontrollers
- Texas Instruments
  - MSP430, Goede documentatie en trainingsmateriaal
- Freescale (iMX)
- Maxim

System on Chip, microcontrollers met zelfs (wireless) transceivers erop, zoals Bluetooth, Wifi, CAN, ethernet, 4G, RF, GPU etc:

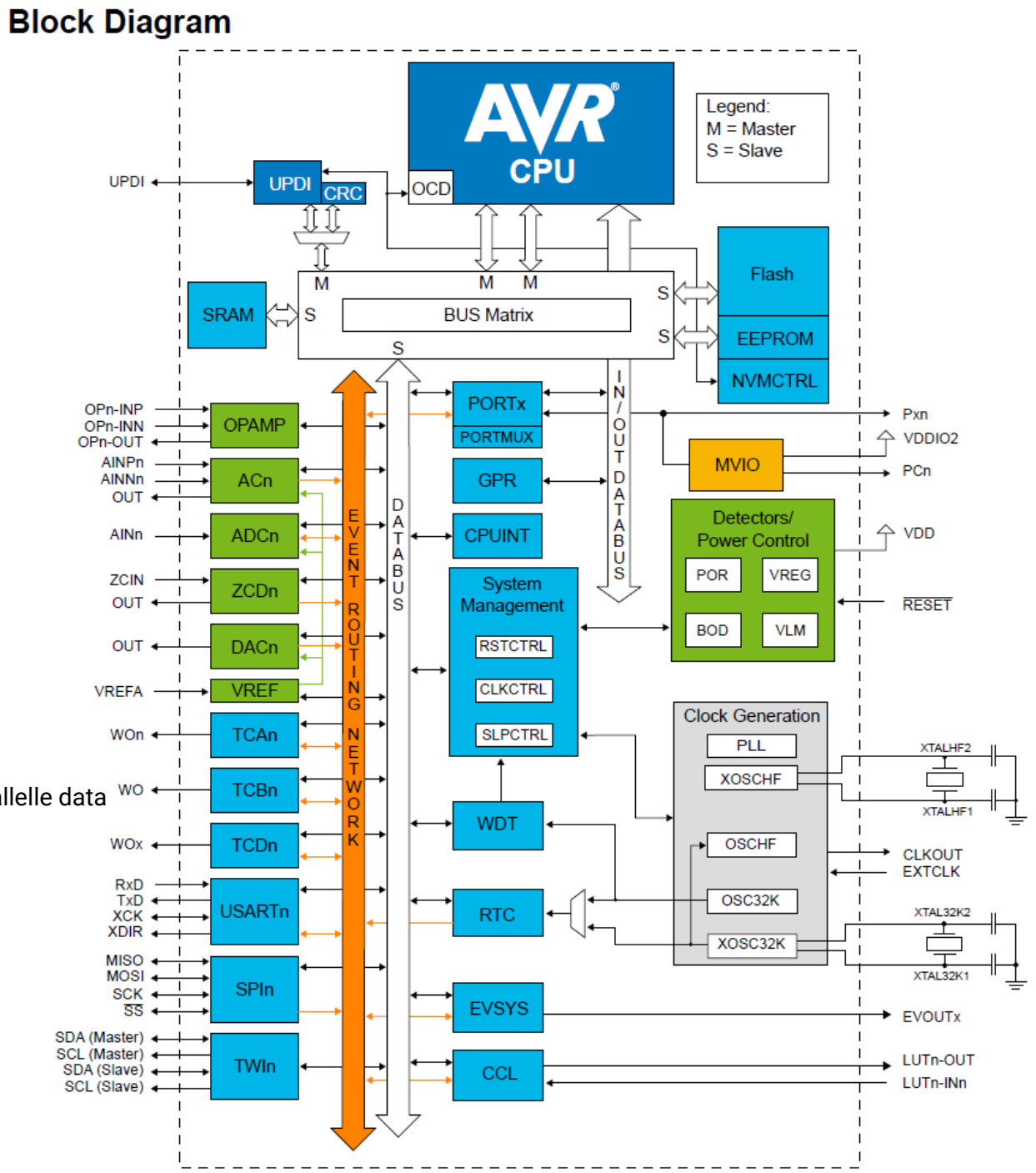
- Nordic Semicondustors (nRF)
- Espressif (ESP32, ESP8266)
- Qualcomm

Microcontroller

Een microcontroller is een chip met daarin een rekenunit (CPU, central processing unit) en daarbij allerlei handige hardware.

Deze on-chip hardware noemen we de peripherals:

- Geheugen (Status geheugen, RAM, (EEP)ROM, FLASH)
- Timers
- Analoog Digitaal Converter en evt Digitaal Analoog Converter
- Digitale Input/Output (kortweg I/O)
- Oscillatoren en klok
- Communicatie zoals SPI, I2C en USART
  - SPI: Serial Peripheral Interface is een synchrone seriële data link tussen bv ICs
  - USART: (Universal (A)Synchronous Transmitter and Receiver) converteert parallelle data naar een seriële datastroom
- Real Time Clock
- Power Management
- Event bus en poort mux
- Zelfs opamps



De AVR CPU

7.1 Features

- 8-bit, High-Performance AVR RISC CPU:
  - 135 instructions
  - Hardware multiplier
- 32 8-bit Registers Directly Connected to the ALU
- Stack in RAM
- Stack Pointer Accessible in I/O Memory Space
- Direct Addressing of up to 64 KB of Unified Memory
- Efficient Support for 8-, 16-, and 32-bit Arithmetic
- Configuration Change Protection for System-Critical Features
- Native On-Chip Debugging (OCD) Support:
  - Two hardware breakpoints
  - Change of flow, interrupt, and software breakpoints
  - Run-time read-out of Stack Pointer (SP) register, Program Counter (PC), and Status Register (SREG)
  - Register file read- and writable in Stopped mode

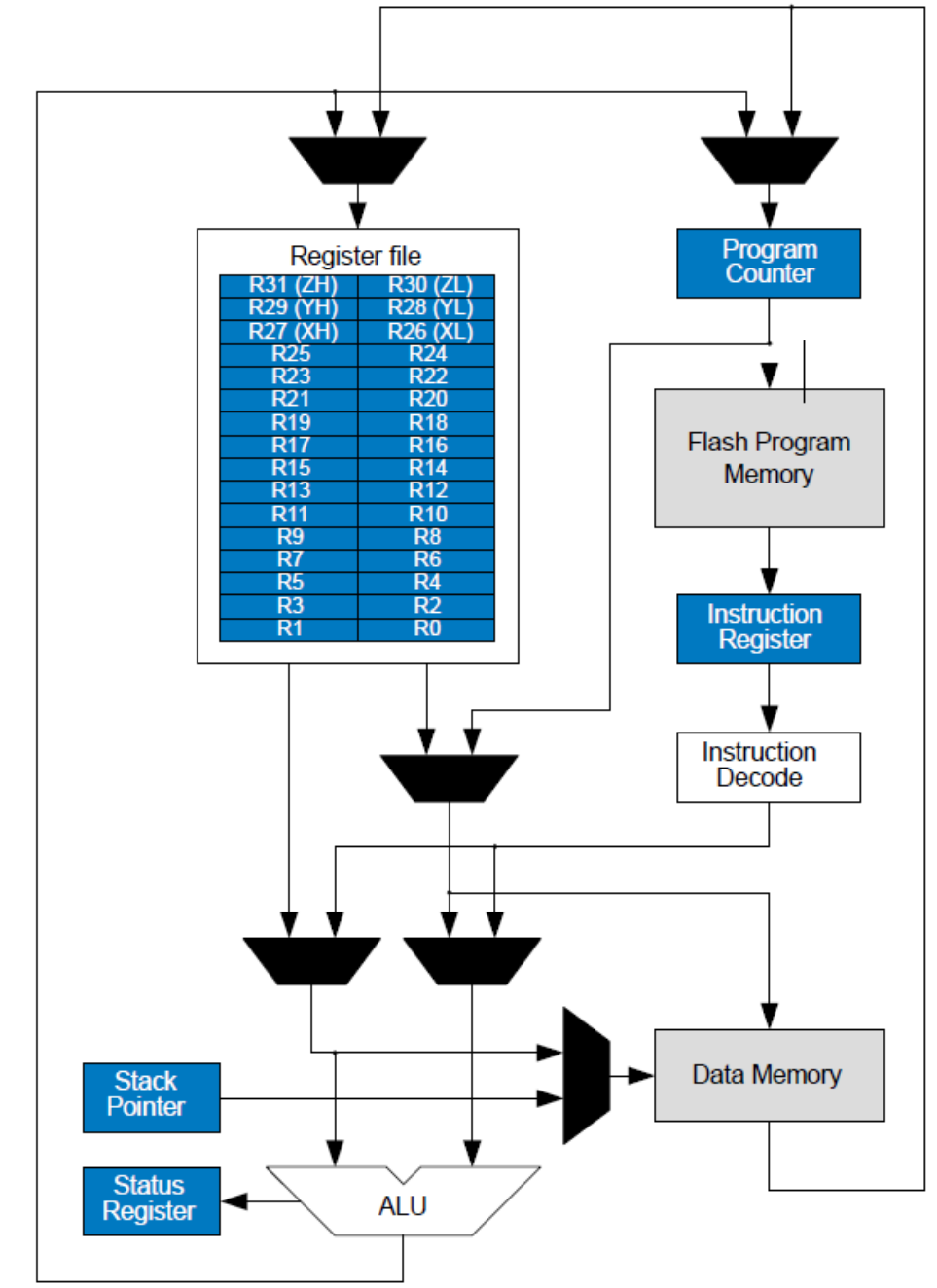
7.2 Overview

The AVR CPU can access memories, perform calculations, control peripherals, execute instructions from the program memory, and handling interrupts.

7.3 Architecture

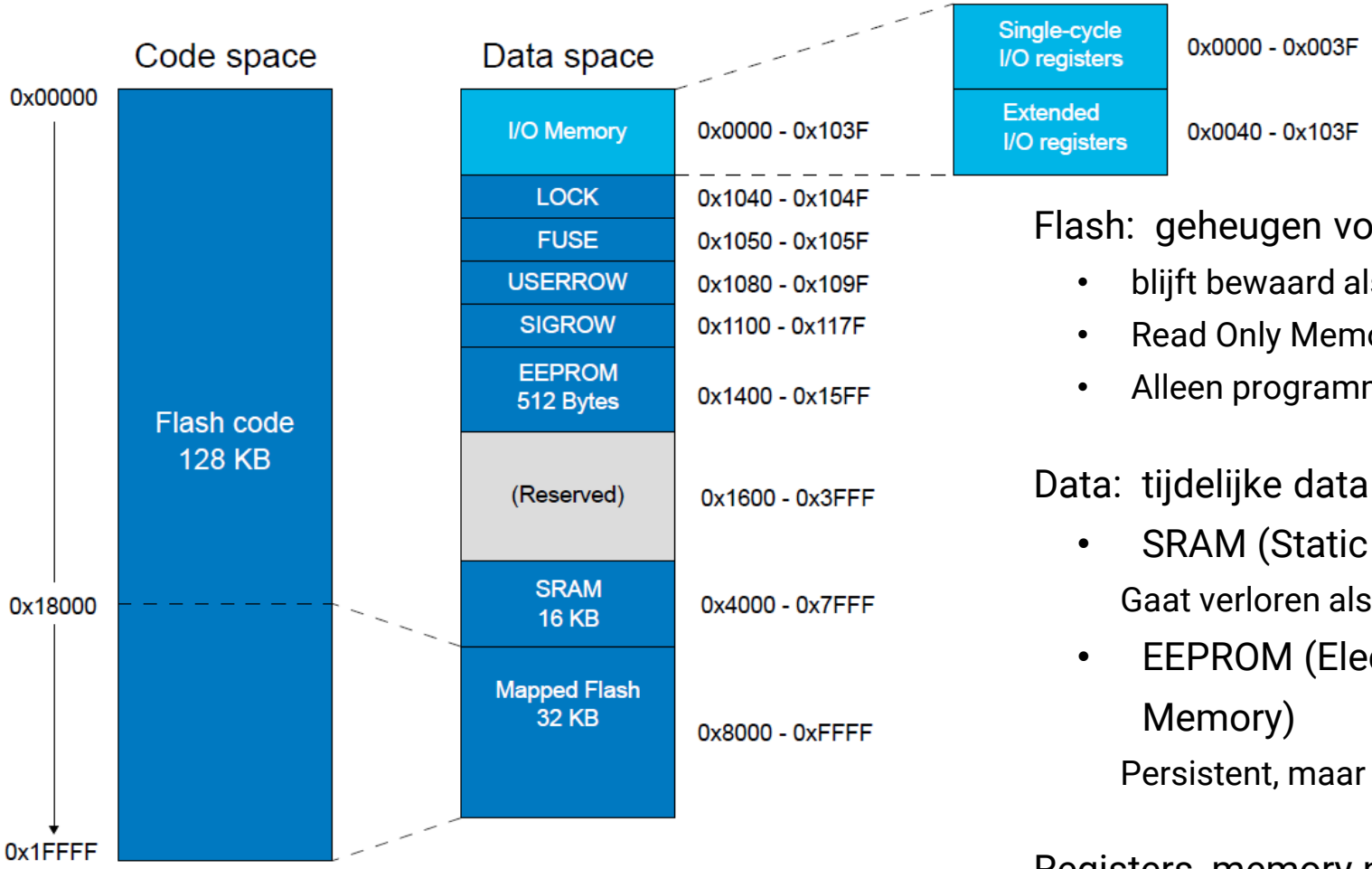
To maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate buses for program and data. Instructions in the program memory are executed with a single-level pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This enables instruct to be executed on every clock cycle.

Figure 7-1. AVR® CPU Architecture



Memory

Figure 8-1. Memory Map



Flash: geheugen voor instructies en data (code geheugen)

- blijft bewaard als de voedingspanning eraf is (persistent),
- Read Only Memory
- Alleen programmeerbaar via bv debugger

Data: tijdelijke data voor variabelen, arrays etc

- SRAM (Static Random Access Memory)

  Gaat verloren als de voedingsspanning eraf is (volatile)
- EEPROM (Electrically Erasaeble Programmable Read Only Memory)

  Persistent, maar beperkt aantal keren schrijven vanuit programma

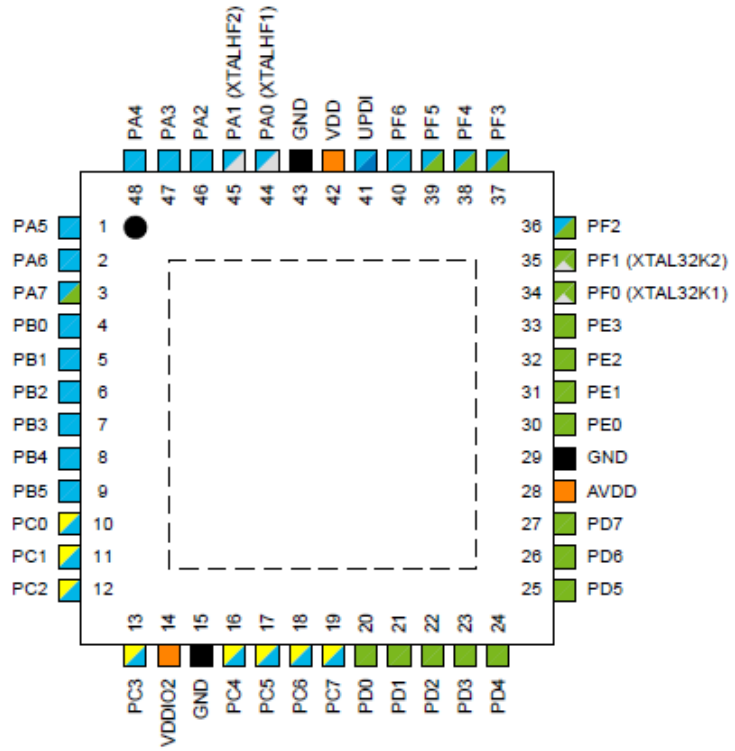
Registers, memory mapped (volatile)

- configuratie van peripherals (I/O, ADC, Timer, UART)
- tijdelijke data van peripherals

I/O Multiplexing

- I/O pinnen kunnen meerdere functies hebben

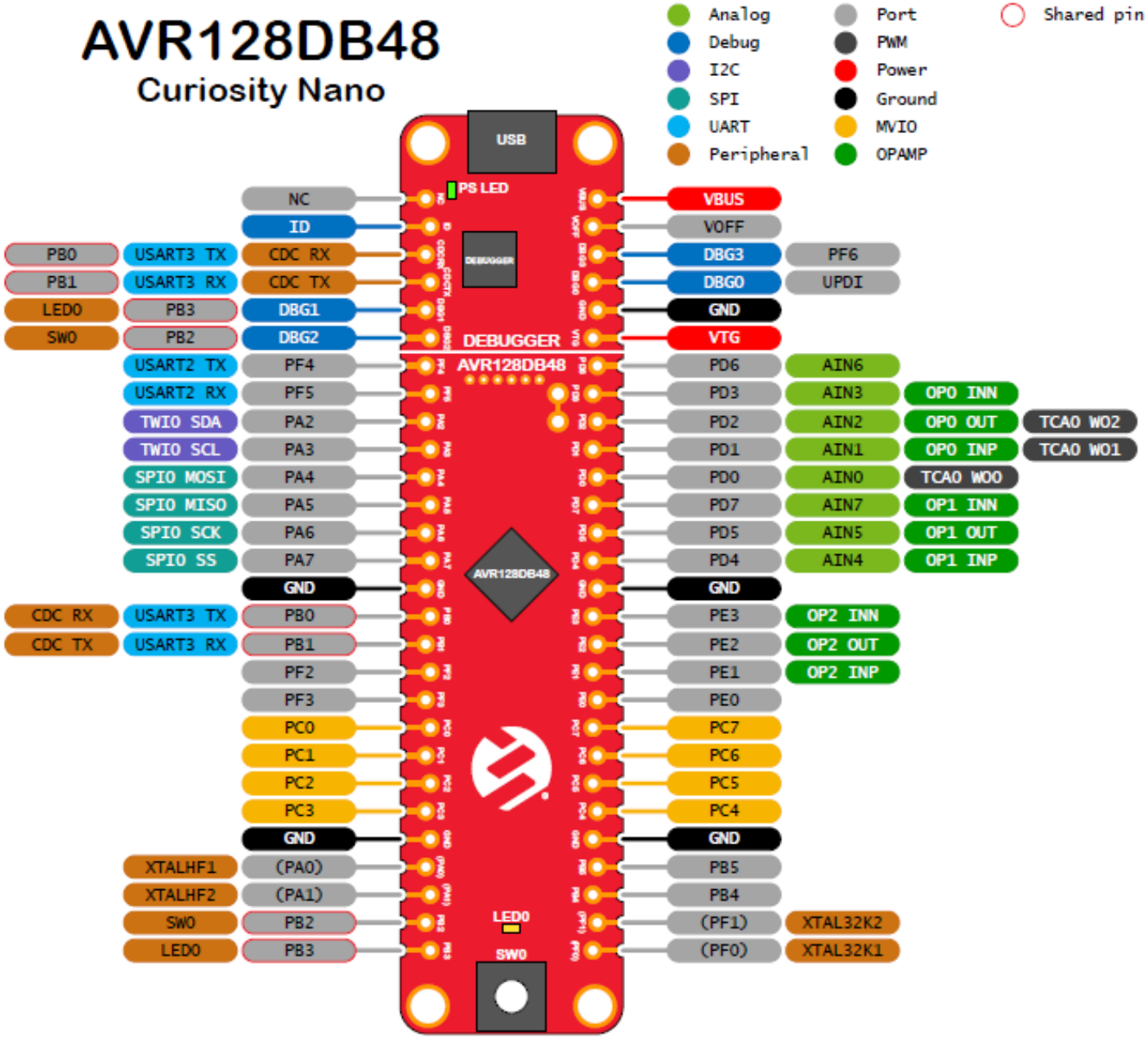
2.3 48-pin VQFN and TQFP



Note: For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

Power	Functionality
Power Supply	Programming/Debug
Ground	Clock/Crystal
Pin on VDD Power Domain	Digital Function Only
Pin on AVDD Power Domain	Analog Function
Pin on VDDIO2 Power Domain	

Figure 5-1. AVR128DB48 Curiosity Nano Pinout



De microcontroller programmeren

```
 * MyFirstProject.c
 *
 * Created: 16-2-2022 09:24:33
 * Author : jk0050363
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Nodig om bij registers te kunnen komen.

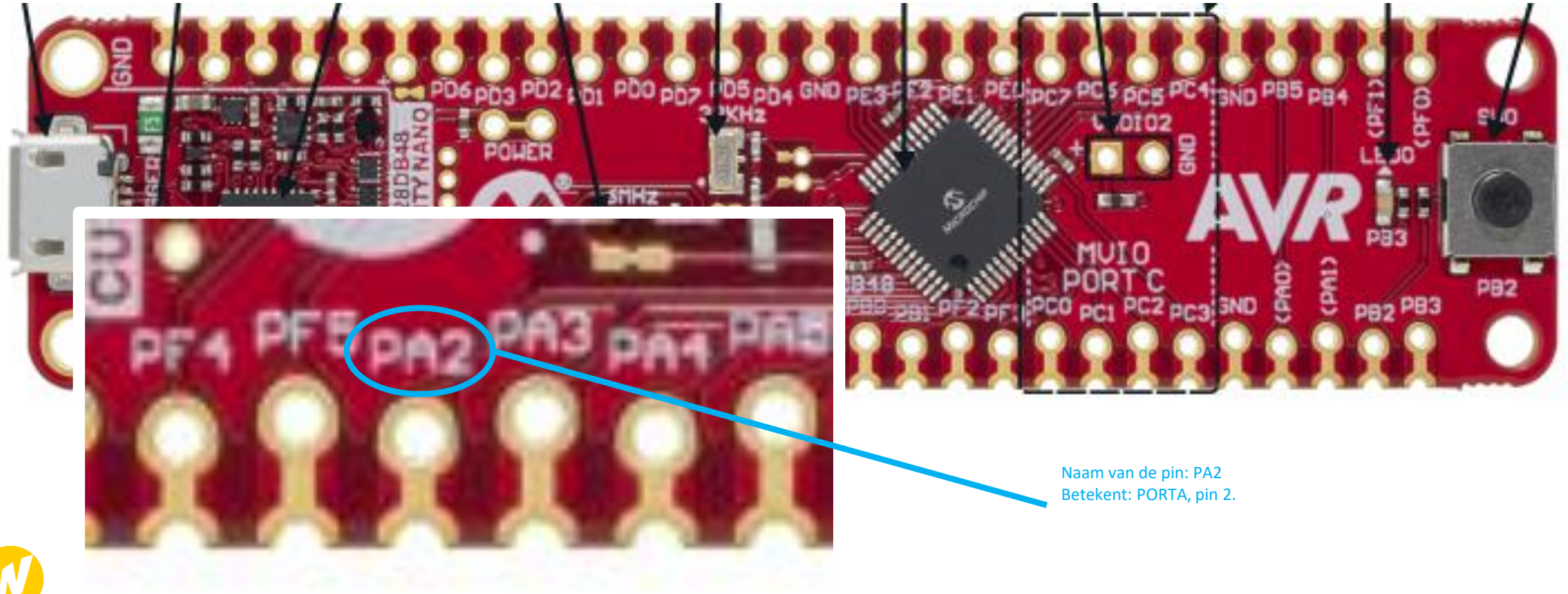
Hier begint je programma zodra de microcontroller wordt aangezet.

Alles binnen deze loop wordt constant herhaald.

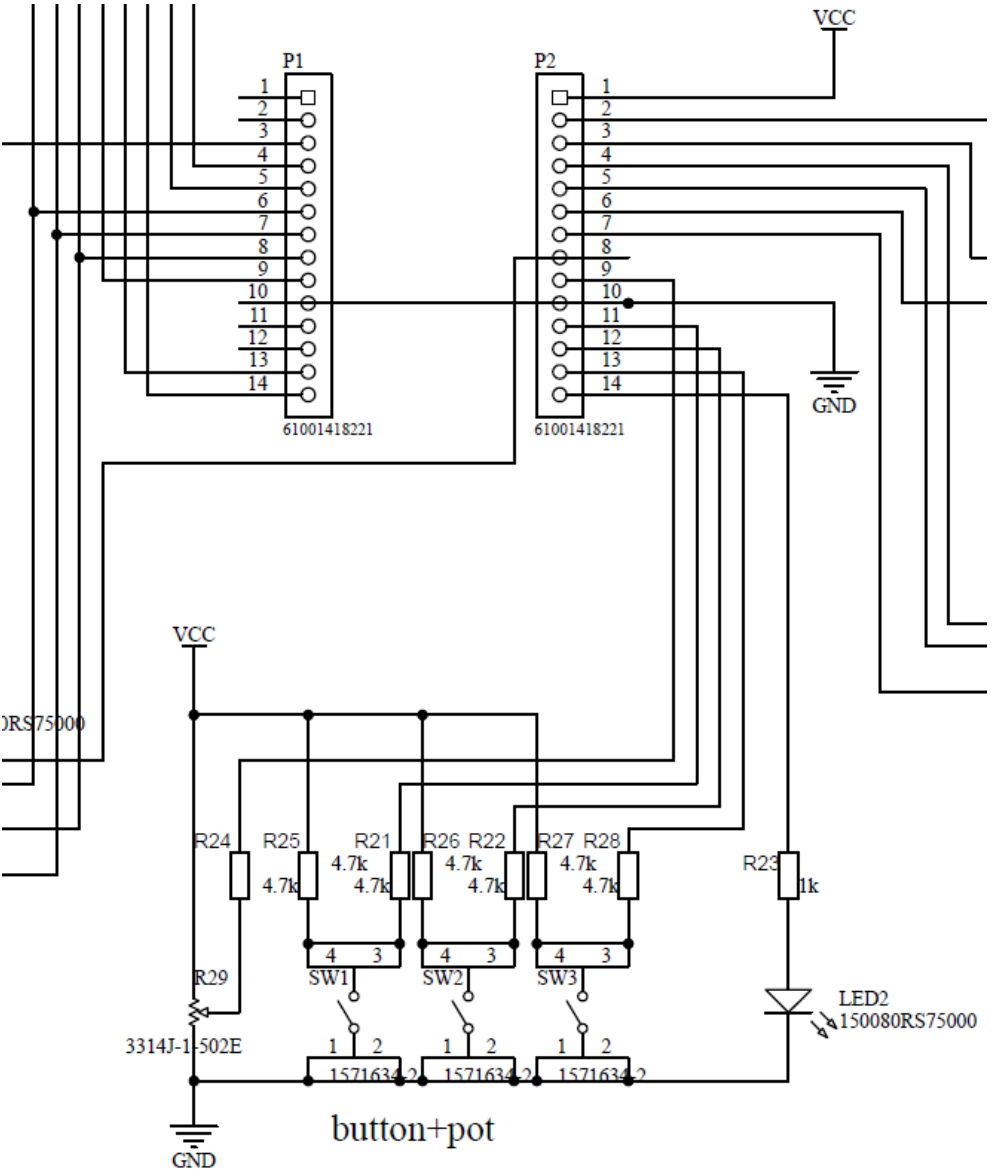
Input en output pinnen instellen

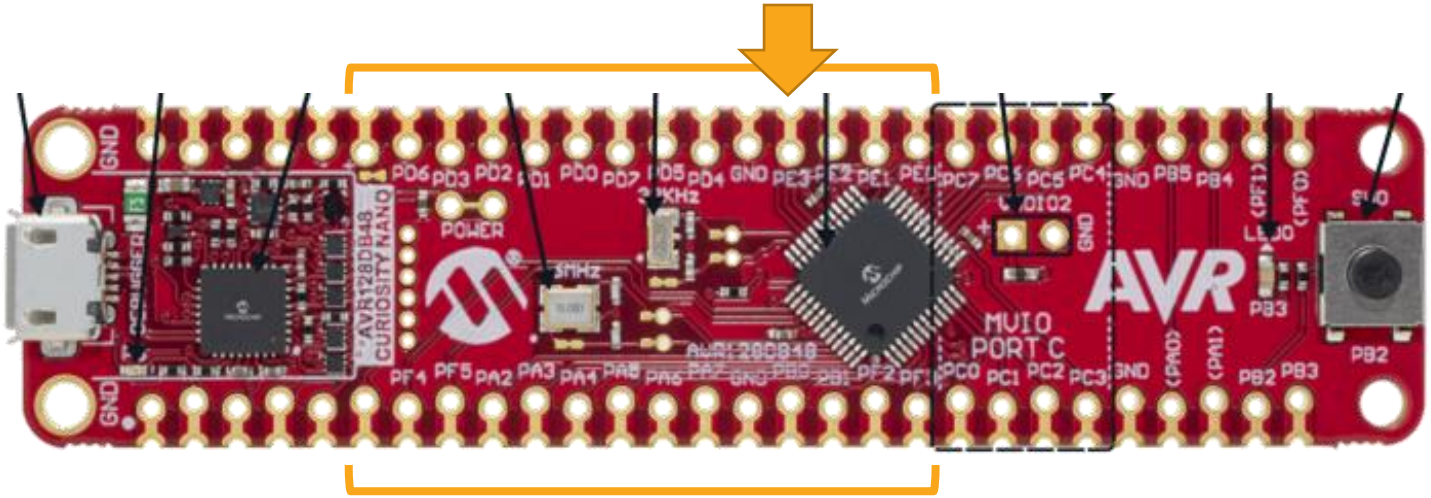
Datasheet (18.2):

The I/O pins of the device are controlled by instances of the PORT peripheral registers. Each PORT instance has up to eight I/O pins. The PORTs are named PORTA, PORTB, PORTC, etc. Refer to the *I/O Multiplexing and Considerations* section to see which pins are controlled by what instance of PORT. The base addresses of the PORT instances and the corresponding Virtual PORT instances are listed in the *Peripherals and Architecture* section.



Input en output pinnen instellen





Op welke pin is SW1 aangesloten?

→ PE3

Input en output pinnen instellen

Pinnen staan standaard geconfigureerd als input.

Wil je een pin als output gebruiken, dan moet je dat in het register instellen (datasheet 18.5.1):

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIR[7:0] Data Direction
This bit field controls the output driver for each PORTx pin.
This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available configuration for each bit n in this bit field is shown in the table below.

Value	Description
0	Pxn is configured as an input-only pin, and the output driver is disabled
1	Pxn is configured as an output pin, and the output driver is enabled

Input en output pinnen instellen

Pinnen staan standaard geconfigureerd als input.

Wil je een pin als output gebruiken, dan moet je dat in het register instellen:

Om pin PD5 als output te gebruiken typ je in:

```
PORTD.DIR = PORTD.DIR | PIN5_bm;
```

Logische OR

Of:

```
PORTD.DIR = PORTD.DIR | 0x20;
```

Hexadecimaal getal

Hexadecimale getallen (0x…)

0x staat voor hexadecimaal
Elk volgend getal is een (veelvoud van) een macht van 16.

VB: $0\text{xC4A} = \text{C} \cdot 16^2 + 4 \cdot 16^1 + \text{A} \cdot 16^0$

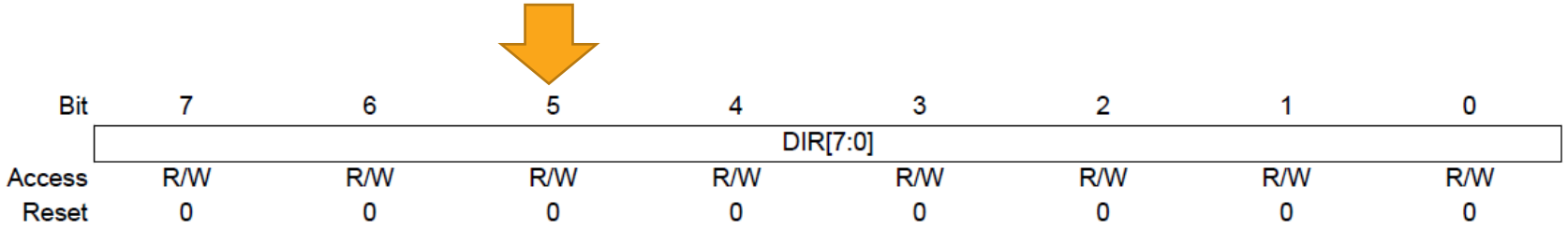
$0\text{xC4A} = 12 \cdot 256 + 4 \cdot 16 + 10 \cdot 1 = 3146$

Hexadecimaal getal	Macht	Decimaal getal
0x01	16^0	1
0x02	2 x 16^0	2
0x10	16^1	16
0xC0	12 x 16^1	192
0xF8	15 x 16^1 + 8 x 16^0	248

Symbool	Waarde
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Hexadecimale getallen (0x…)

Waarom gebruik je 0x20 om pin5 hoog te maken?



Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

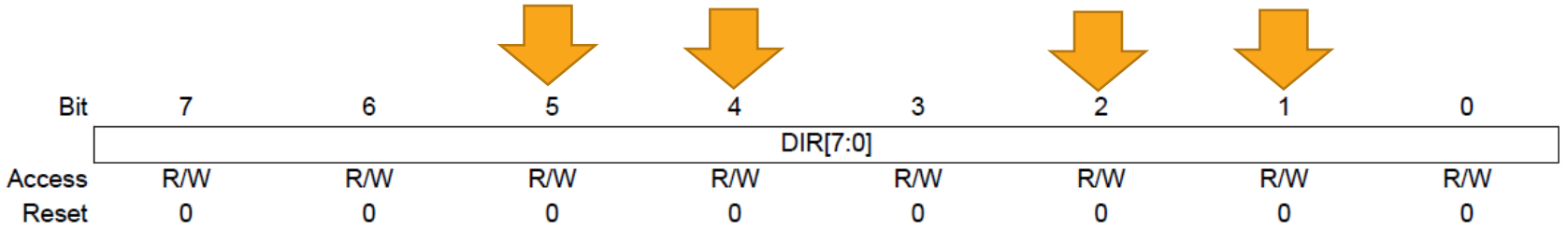
Binair moet er dus in het register komen te staan: “00100000”

Om van binair naar hexadecimaal te gaan deel je op in groepjes van 4 bits:
“0010” “0000”

En zet je elk groepje om naar een hexadecimaal getal:
0x20

Meerdere outputpinnen tegelijk instellen

Stel, ik wil pin PA5, PA4, PA2 en PA1 als output gebruiken, dan kan ik dat in 1 regel instellen:



Binair moet er dus in het register komen te staan: “00110110”

Om van binair naar hexadecimaal te gaan deel je op in groepjes van 4 bits:
“0011” “0110”
En zet je elk groepje om naar een hexadecimaal getal:
0x36

```
PORTA.DIR = PORTA.DIR | 0x36;
```

Opbouw code

```
 * MyFirstProject.c
 *
 * Created: 16-2-2022 09:24:33
 * Author : jk0050363
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Hier moeten de instellingen voor in- en outputpinnen komen te staan.

Hier moet komen te staan wat de in- en outputpinnen moeten doen.

Opbouw code - voorbeeld

```
#include <avr/io.h>

int main()
{
    PORTA.DIR = PORTA.DIR | PIN7_bm;
while(1)
    {
        if (PORTF.IN & PIN0_bm)
        {
      PORTA.OUTCLR = PIN7_bm;
        }
        else
        {
      PORTA.OUTSET = PIN7_bm;
        }
    }
}
```

Operatoren

Relationele operator	Functie
==	Gelijk aan
!=	Ongelijk aan
>	Groter dan
>=	Groter of gelijk aan
<	Kleiner dan
<=	Kleiner of gelijk aan

Rekenkundige operator	Functie
+	Optellen
-	Aftellen
*	Vermenigvuldigen
/	Delen
=	Toekenning

Logische operator	Functie
&&	AND
||	OR
!	NOT

Bit manipulaties

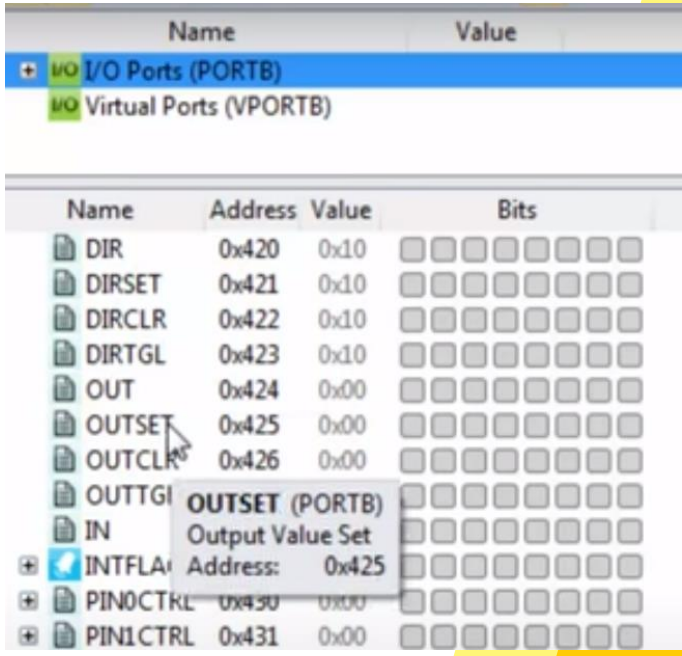
Logische operator	Functie
&	AND
|	OR
~	NOT
^	EXOR (toggle)
>>	Bit-shift rechts
<<	Bit-shift links

Kijk eens in ioavr128db48.h voor wat voorbeelden van bit manipulaties…

```
PIN7_bm  = 0x10 = 0b00010000 = 1<<4 = ~0xEF
```

Let op:
Veel uC hebben geen snelle “hardware read-modify-write” registers zoals OUTSET, OUTCLR, OUTTGL, leer ook te werken met de bit manipulaties!
[AVR® MCU Simulator Debugging]

Pin testen:
```
PORTA.IN & PIN4_bm
```
Pin hoog maken:
```
PORTA.OUTSET = PIN4_bm   of
PORTA.OUT |= PIN4_bm
```
Pin laag maken:
```
PORTA.OUTCLR = PIN4_bm    of
PORTA.OUT &= ~PIN4_bm
```
Pin togglen:
```
PORTA.OUTTGL = PIN4_bm    of
PORTA.OUT ^= PIN4_bm
```



Debugging:

Bekijken van gedrag van de code binnen de microcontroller met de debugger

Figure 1-1. AVR128DB48 Curiosity Nano Board Overview

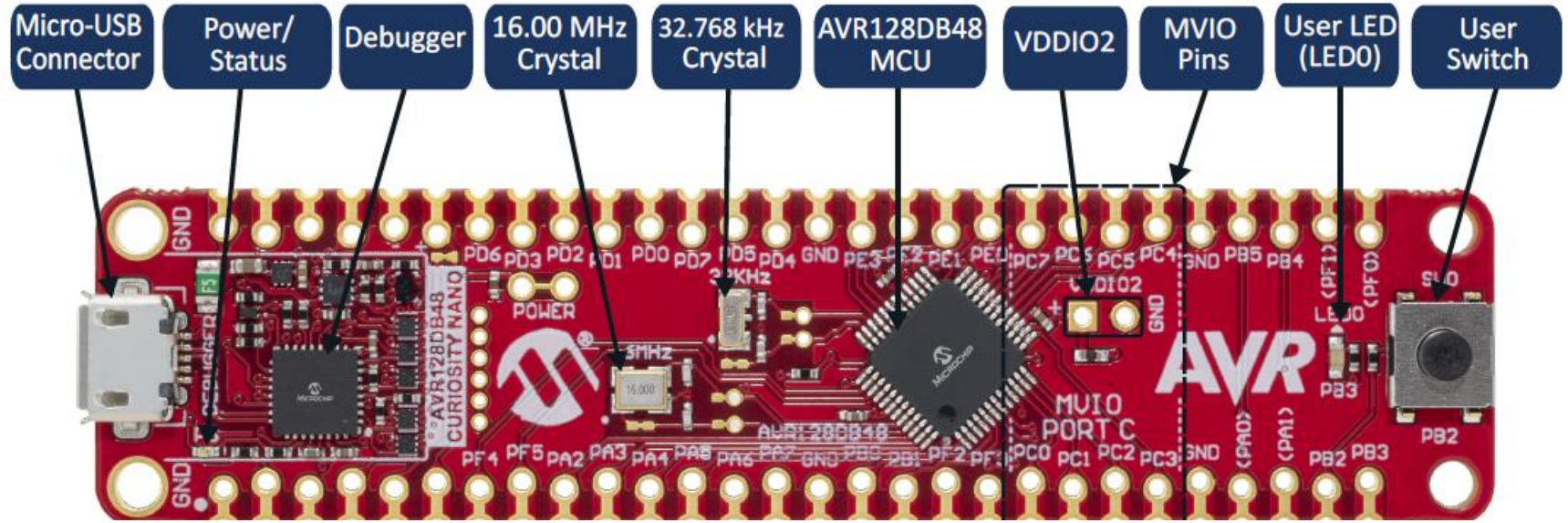
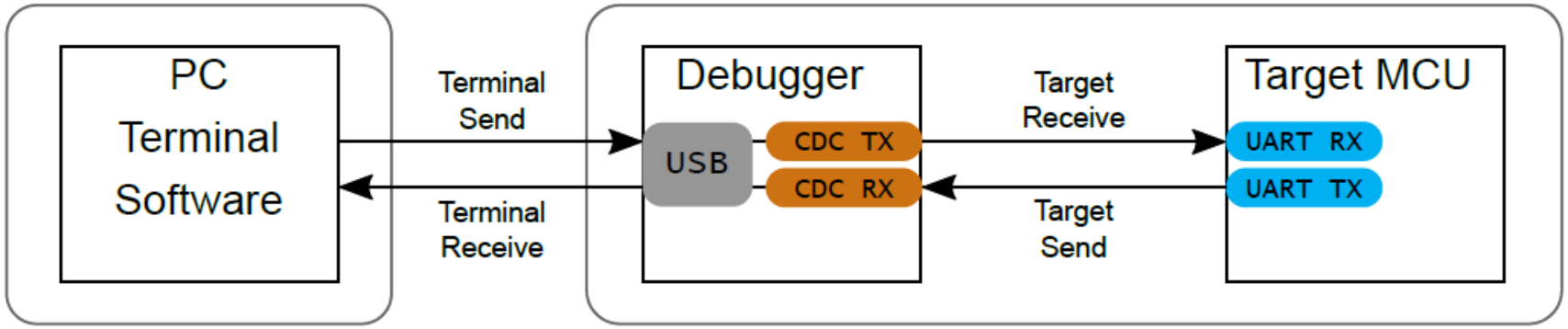
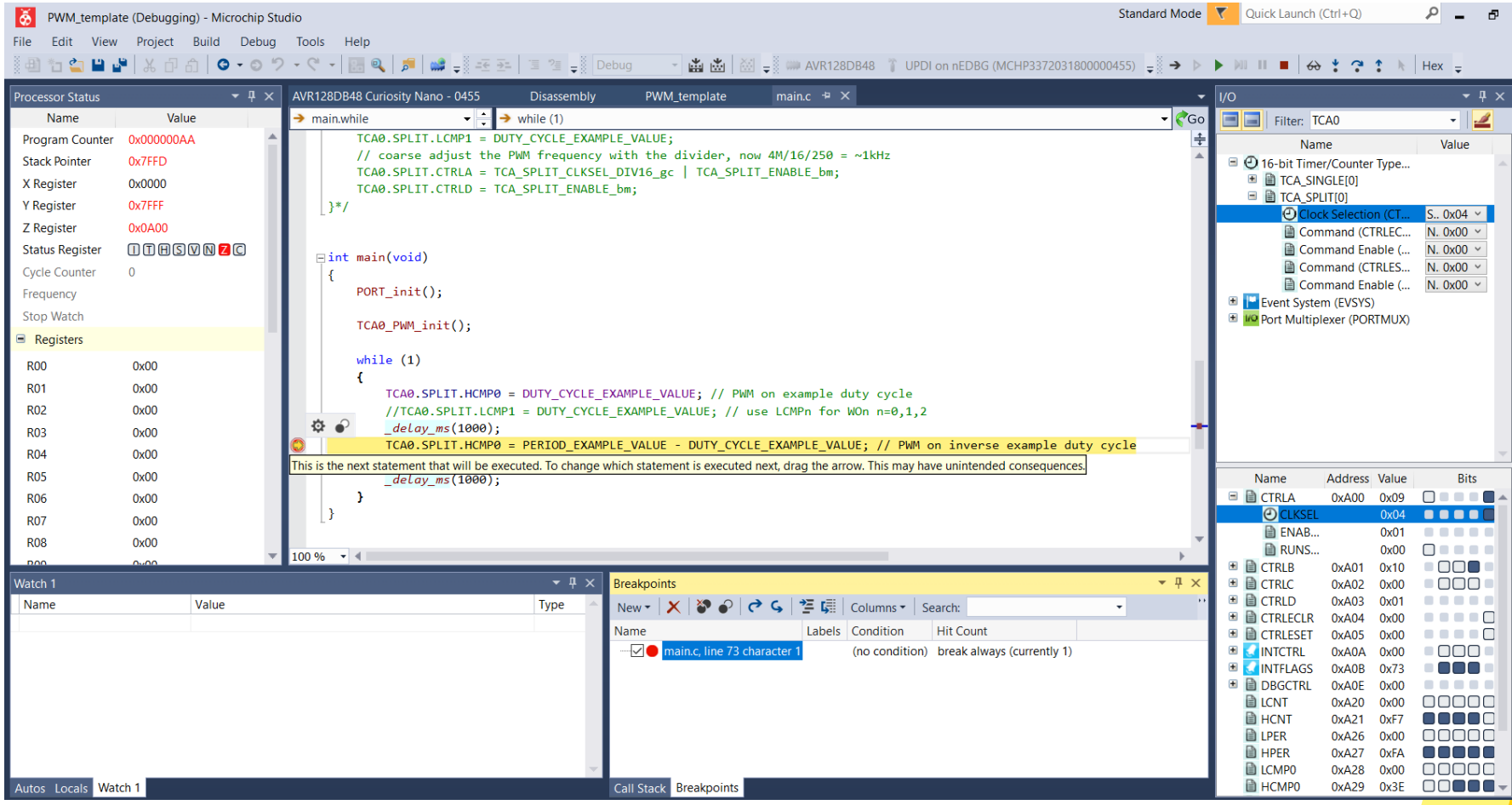


Figure 4-1. CDC Connection



Demo:

- Inline Debugging
- Breakpoints
- Step/continue
- Monitoring I/O
- Watch



- [Getting started with Atmel Studio]
- [Episode 10 - I/O View & Bare Metal Programming]
- [Episode 11 - Editor Visual Assist]
- [Episode 12 - AVR® MCU Simulator Debugging]
- [Episode 13 - Debugging Pt. 1]

Aan de slag!

- Ga naar leren.windesheim.nl

  (zoek de cursus EDPR.22, microcontroller programmeren)

- Bekijk Getting-Started-with-GPIO

- Voer de opdrachten van week 1 uit.

- Ben je klaar? Ga vast verder met de voorbereiding van week 2!