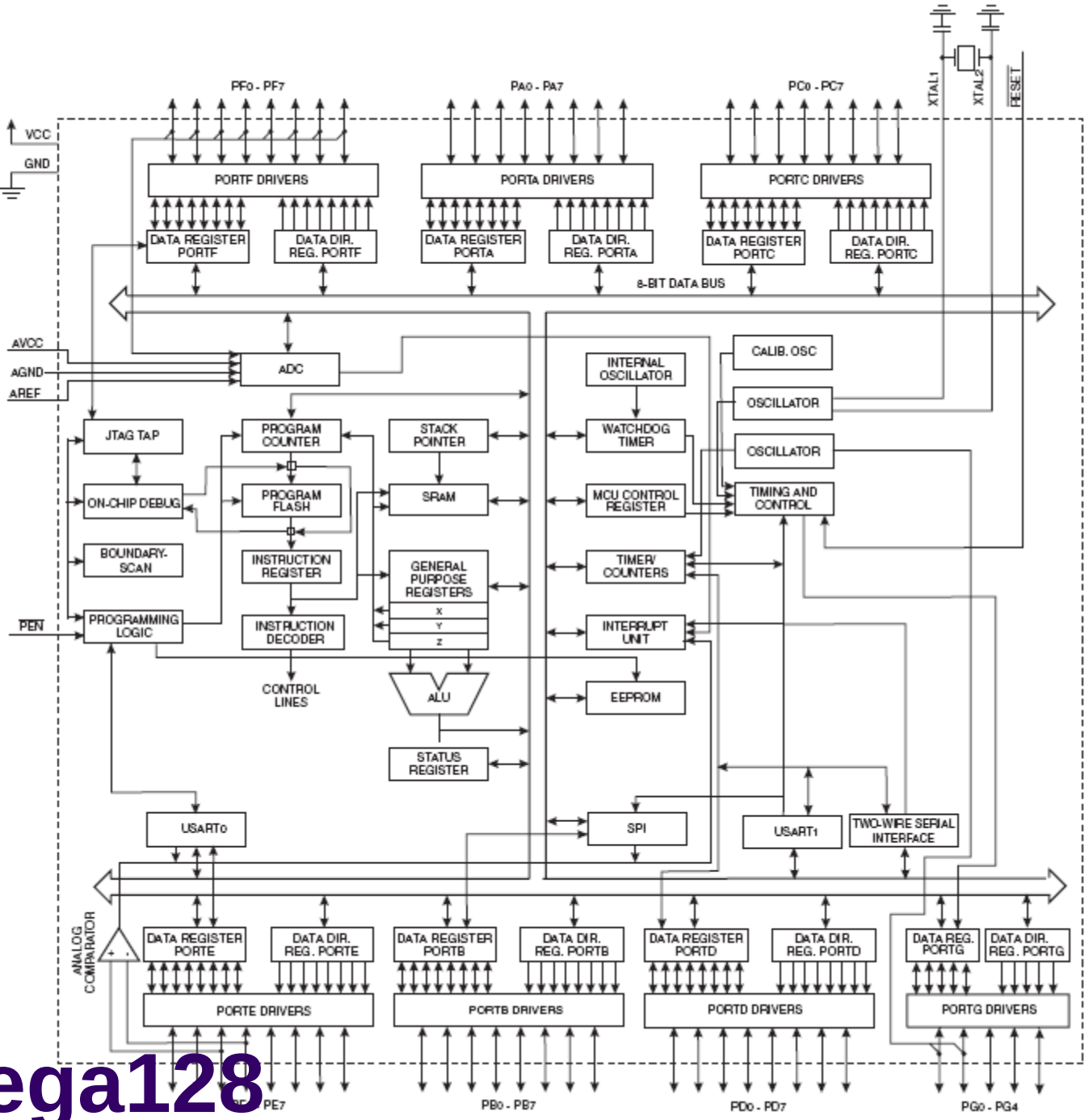
Programování mikrokontrolerů

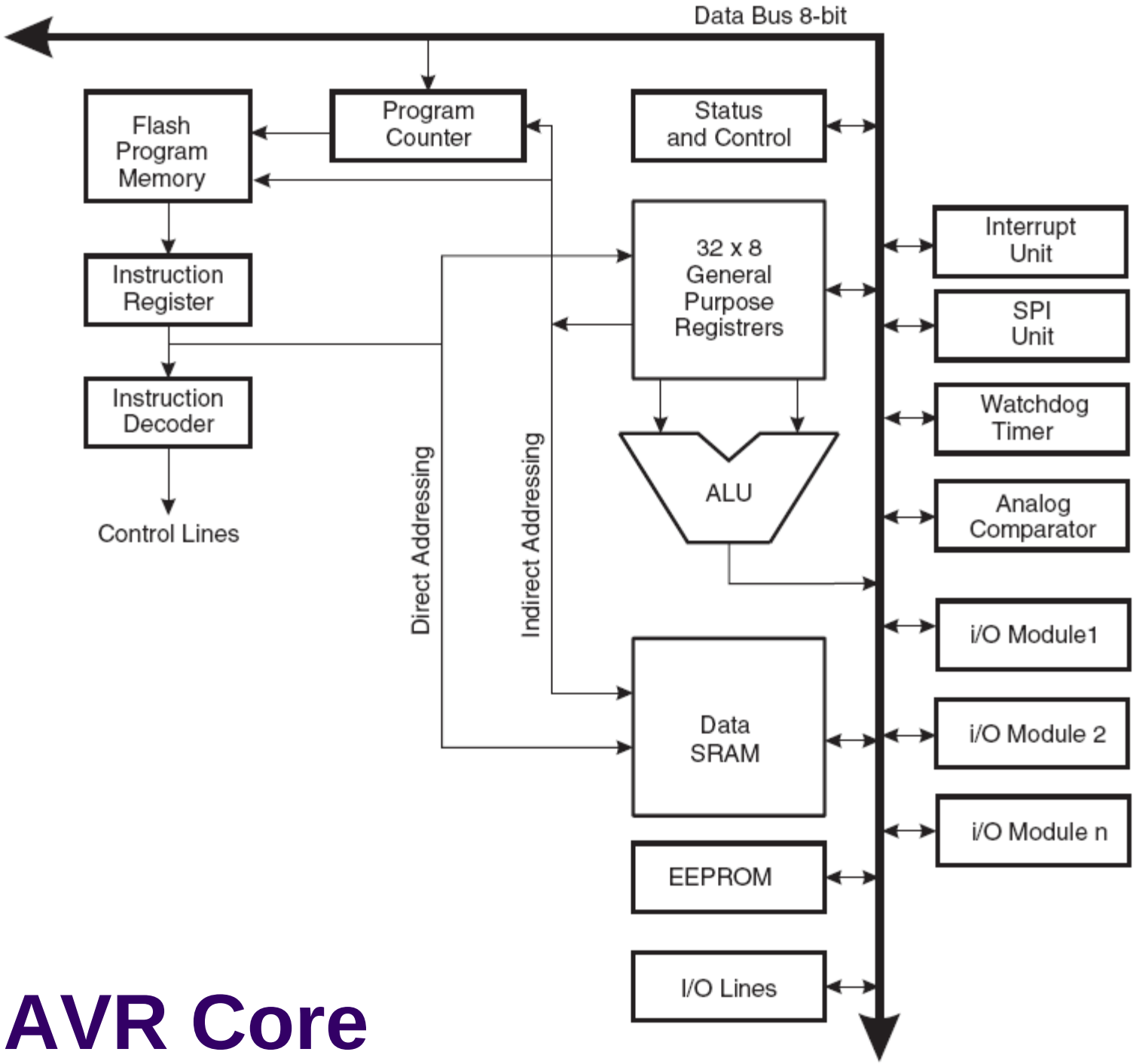
EEPROM

Atmel AVR

ATmega128/328PB


PF0 - PF7
PA0 - PA7
PC0 - PC7
XTAL1
XTAL2
RESET
VCC
GND
PORTF DRIVERS
PORTA DRIVERS
PORTC DRIVERS
DATA REGISTER PORTF
DATA DIR. REG. PORTF
DATA REGISTER PORTA
DATA DIR. REG. PORTA
DATA REGISTER PORTC
DATA DIR. REG. PORTC
8-BIT DATA BUS
AVCC
AGND
AREF
ADC
INTERNAL OSCILLATOR
CALIB. OSC
OSCILLATOR
OSCILLATOR
TIMING AND CONTROL
JTAG TAP
PROGRAM COUNTER
STACK POINTER
WATCHDOG TIMER
ON-CHIP DEBUG
PROGRAM FLASH
SRAM
MCU CONTROL REGISTER
BOUNDARY-SCAN
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
X
Y
Z
TIMER/ COUNTERS
PEN
PROGRAMMING LOGIC
INSTRUCTION DECODER
INTERRUPT UNIT
CONTROL LINES
ALU
EEPROM
STATUS REGISTER
USART0
SPI
USART1
TWO-WIRE SERIAL INTERFACE
ANALOG COMPARATOR
DATA REGISTER PORTE
DATA DIR. REG. PORTE
DATA REGISTER PORTB
DATA DIR. REG. PORTB
DATA REGISTER PORTD
DATA DIR. REG. PORTD
DATA REG. PORTG
DATA DIR. REG. PORTG
PORTE DRIVERS
PORTB DRIVERS
PORTD DRIVERS
PORTG DRIVERS
PE7
PB0 - PB7
PD0 - PD7
PG0 - PG4

ATmega128


Data Bus 8-bit
Flash Program Memory
Program Counter
Status and Control
Instruction Register
Instruction Decoder
Control Lines
32 x 8 General Purpose Registrers
ALU
Direct Addressing
Indirect Addressing
Data SRAM
EEPROM
I/O Lines
Interrupt Unit
SPI Unit
Watchdog Timer
Analog Comparator
i/O Module1
i/O Module 2
i/O Module n

AVR Core

- ATmega128 – 4kB EEPROM
  ATmega328PB – 1kB EEPROM
- Endurance: at least 100.000 write/erase cycles
- Not directly accessible

- EEPROM write
  - blocks all flash writes
  - blocks reading fuses / lock bits

EEPROM Address Registers

	7	6	5	4	3	2	1	0
EEARH	-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8
EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
	R	R	R	R	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	x	x	x	x
	x	x	x	x	x	x	x	x

EEARH7:4 reserved

EEARH11:EEARL0 0-4095 memory address

(EEARH9:EEARL0 i.e. 0-1023 for ATmega328PB)

EEPROM Data Register

EEDR

7	6	5	4	3	2	1	0
MSB							LSB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

EEDR EEPROM data read / to be written

EEPROM Control Register ATmega128

EECR

7	6	5	4	3	2	1	0
-	-	-	-	EERIE	EEMWE	EEWE	EERE
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	x	0

EERIE EEPROM Ready Interrupt Enable

EEMWE EEPROM Master Write Enable

EEWE EEPROM Write Enable

EERE EEPROM Read Enable

EEPROM Control Register ATmega328PB

EECR

7	6	5	4	3	2	1	0
-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	x	0

EEPM EEPROM Programming Mode Bits

- 00 Erase and Write on one operation (atomic)
- 01 Erase Only
- 10 Write Only
- 11 (reserved)

EERIE EEPROM Ready Interrupt Enable

EEMPE EEPROM Master Write Enable

EEPE EEPROM Write Enable

EERE EEPROM Read Enable

Write / read Algorithm

- controlled via EEAR, EEDR, EECR:

- Write:
  - EEWE=0? SPMEN=0?
  - address → EEAR
  - data → EEDR
  - EEMWE=1, EEWE=0
  - EEWE=1 within max 4 clocks

  {2 clocks halt}
  done when EEWE=0 (~8.5ms)

- Read:
  - address → EEAR
  - EERE=1

  {4 clocks halt}
  done

ASM implementation

```
EEPROM_write:
    ; Wait for completion of previous
    ; write
    sbic EECR, EEWE
    rjmp EEPROM_write
    ; Set up address (r18:r17) in
    ; address register
    out EEARH, r18
    out EEARL, r17
    ; Write data (r16) to data
    ; register
    out EEDR, r16
    ; Write logical one to EEMWE
    sbi EECR, EEMWE
    ; Start EEPROM write by setting
    ; EEWE
    sbi EECR, EEWE
    ret
```

```
EEPROM_read:
    ; Wait for completion of previous
    ; write
    sbic EECR, EEWE
    rjmp EEPROM_read
    ; Set up address (r18:r17) in
    ; address register
    out EEARH, r18
    out EEARL, r17
    ; Start EEPROM read by writing EERE
    sbi EECR, EERE
    ; Read data from data register
    in r16, EEDR
    ret
```

C implementation

```
void EEPROM_write(unsigned int uiAddress,
unsigned char ucData)
{
   /* Wait for completion of previous write */
   while(EECR & (1<<EEWE))
        ;
   /* Set up address and data registers */
   EEAR = uiAddress;
   EEDR = ucData;
   /* Write logical one to EEMWE */
   EECR |= (1<<EEMWE);
   /* Start EEPROM write by setting EEWE */
   EECR |= (1<<EEWE);
}
```

```
unsigned char EEPROM_read(unsigned int
uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
       ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start EEPROM read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```