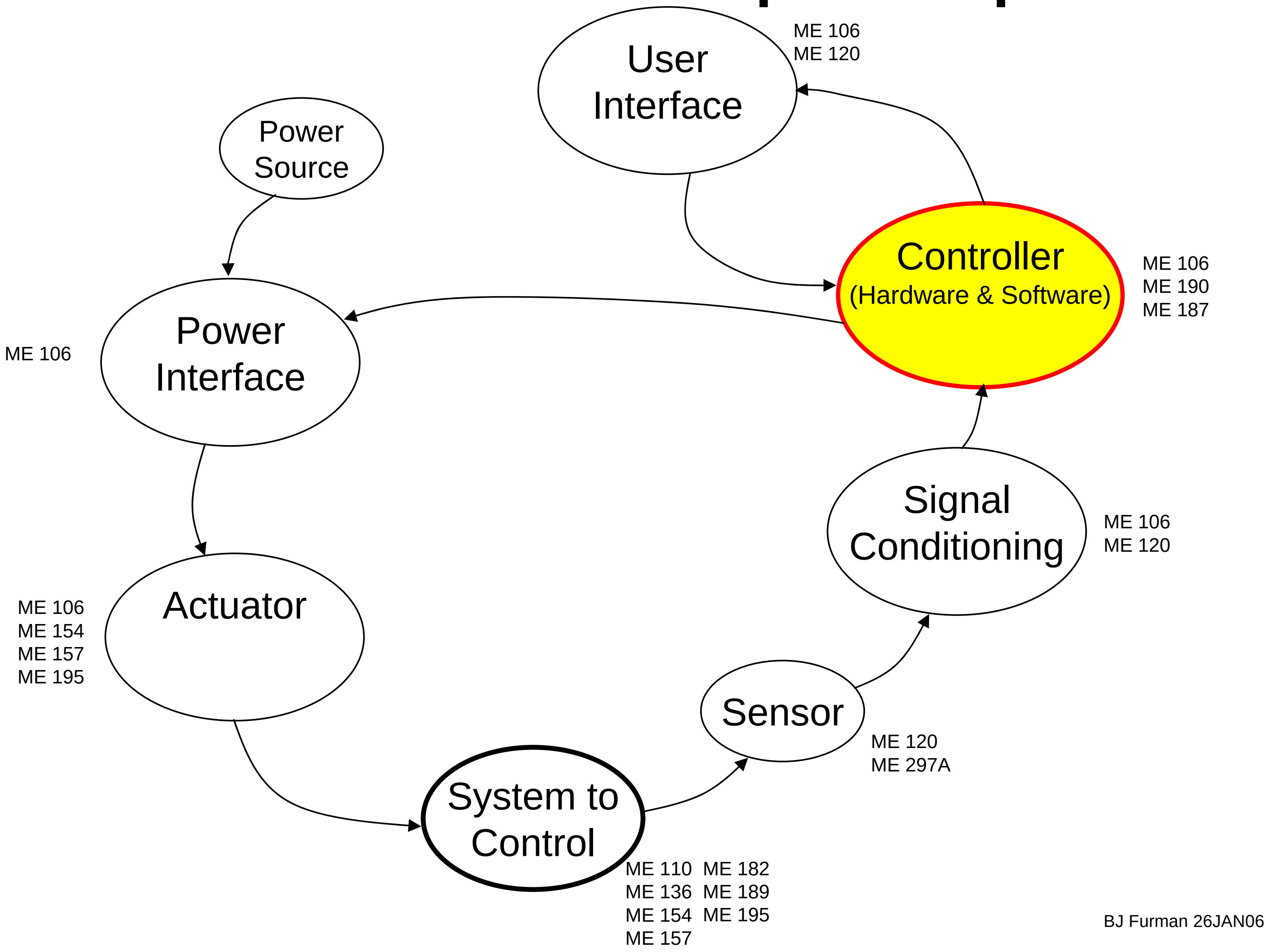
Programming the ATmega128

B. Furman

12FEB2007

Mechatronics Concept Map

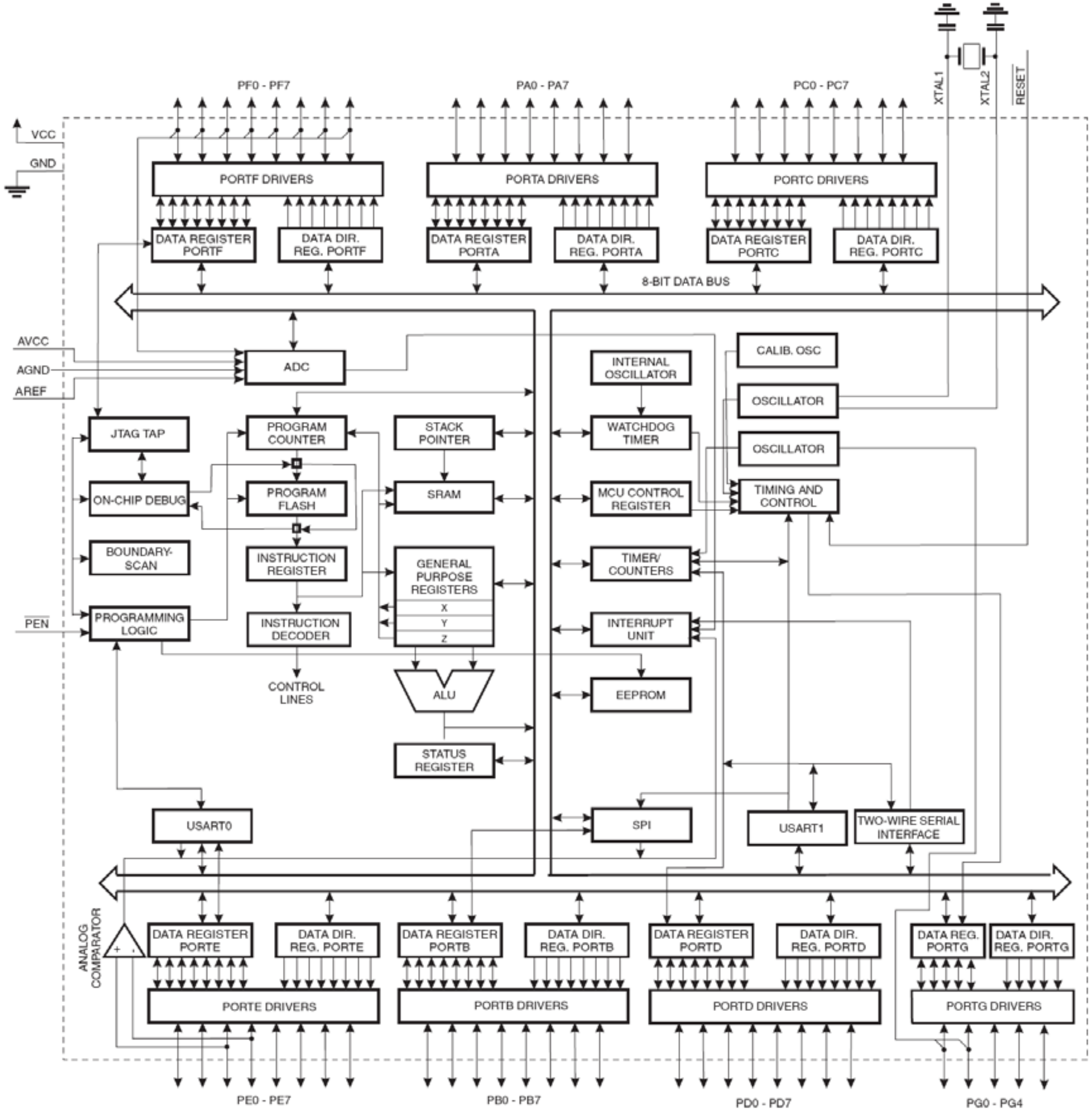


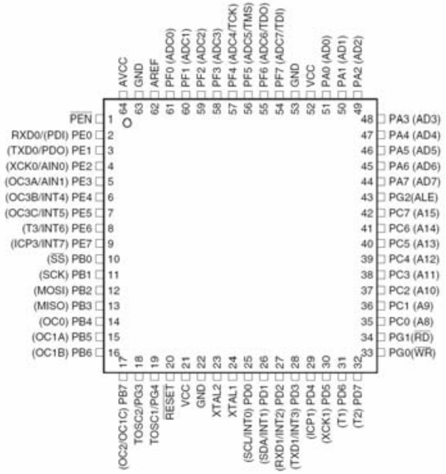
BJ Furman 26JAN06

ATmega 128 Internals

- See the data sheet, p. 2-3
- Pins and Ports for Digital I/O
  - Inputs
    - External connections determine pin voltage
  - Outputs
    - Microcontroller sets pin voltage
  - Controlled by three corresponding ***registers*** (memory locations)
    - 'Direction' set by Data Direction Register (DDRx) – bi-dir.
      - Logic 1 → Output   Logic 0 → Input
      - Pins are set to be 'inputs' on reset
    - Data Register (PORTx) – bi-dir.
      - Writing to PORTx when a pin is configured as an input turns on internal 'pull up' resistor (will read as logic 1 until pulled low)
    - Port input pins (PINx) – Note: read only

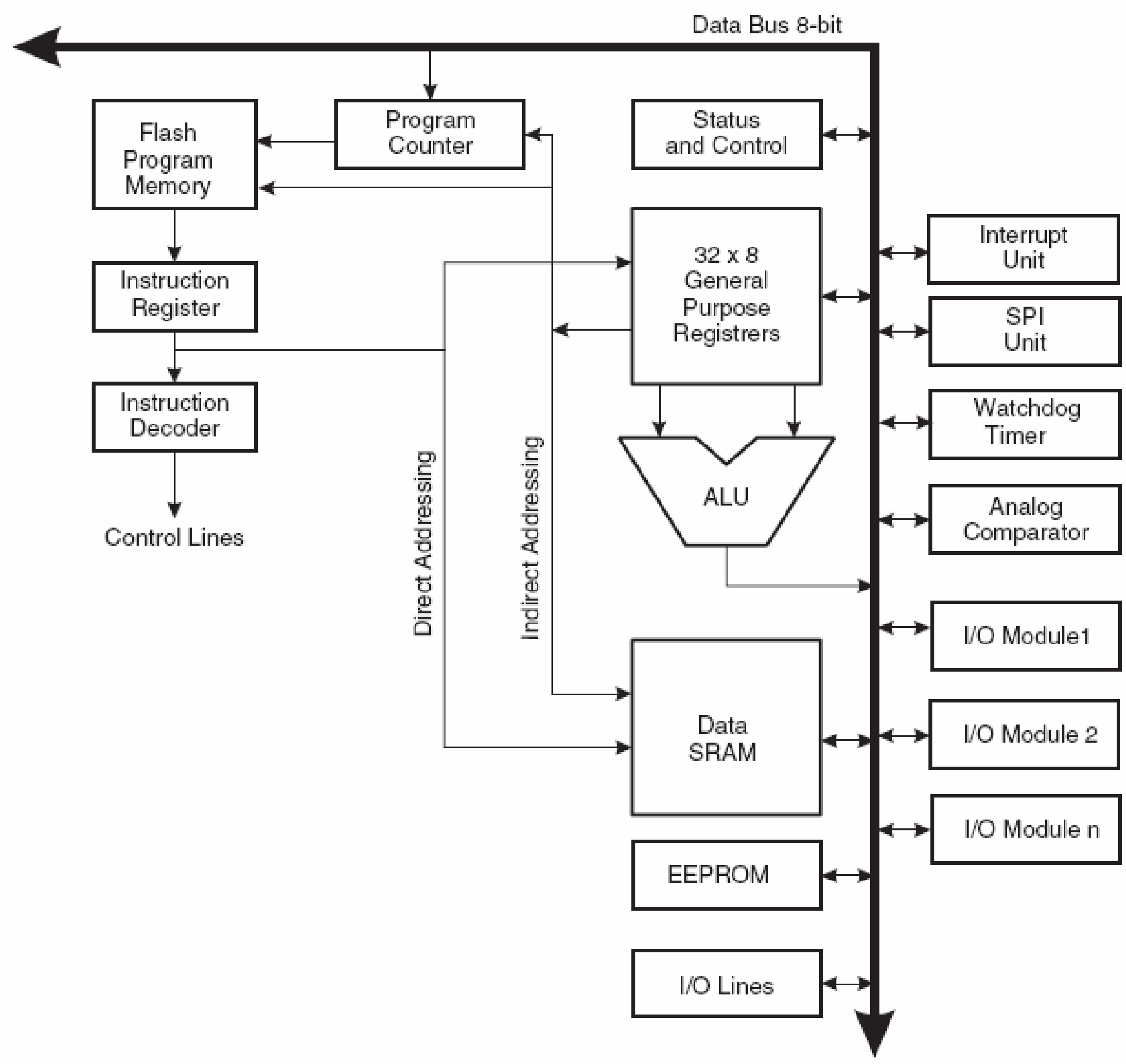
ATmega 128 Internal Architecture - 1



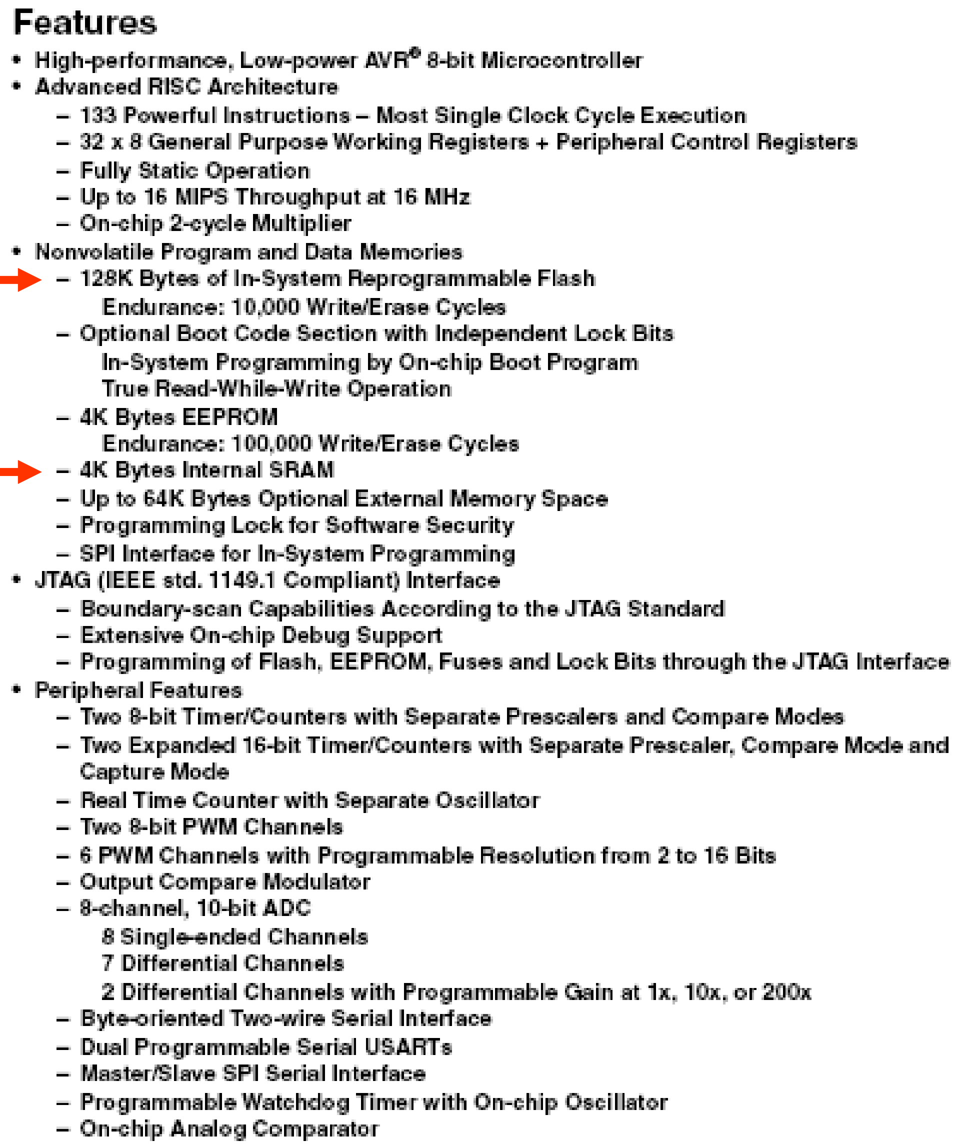


http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

ATmega 128 Internal Architecture - 2



ATmega128 Features



Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 128K Bytes of In-System Reprogrammable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 4K Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 4K Bytes Internal SRAM
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator

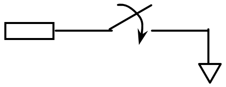
http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

Recap ATmega 128 Digital I/O

- Pins are bi-directional. Can configure as:
  - □ **Inputs** – ________ determines the pin voltage
  - □ **Outputs** – _______ determines the pin voltage
  - □ Direction determined by bits in **DDR**x register
    - Where x is A – G (and corresponds to *all* 8 pins associated with the port)
  - □ If configured as *output*:
    - Program can specify a pin to be high (V_{TG}) or low (GND) by writing a corresponding 1 or 0 (respectively) to PORTx register
      - □ Ex. To make Port C pins 7, 3, and 4 low, and the rest high
      - □ PORTC=__________; (write in binary, then in hex)

Recap ATmega 128 Digital I/O, cont.

- If pins configured as *input*, this means:
  - External device can pull pin voltage high or low
    - i.e. take up to VTG or take down to GND



  - You can determine the state of the port pins by reading the PINx register
    - Grabs all eight logic levels at the same time
    - Ex. PORTA configured as inputs

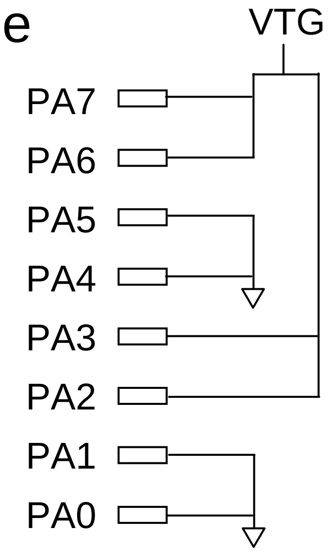
```
unsigned char a_pins;

a_pins=PINA;
```

What is the content of a_pins:

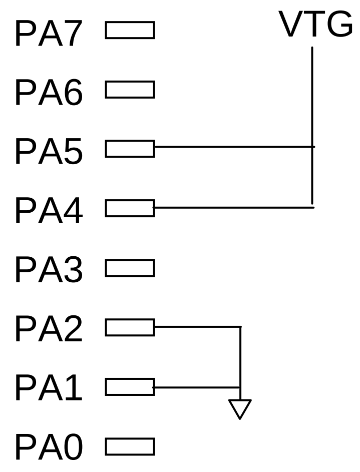
binary:___________

hex:_____



Recap ATmega 128 Digital I/O, cont.

- If pins configured as *input*, cont.:
  - Can turn pull-up resistors on or off by writing a 1 or 0 to corresponding pins in PORTx
    - A pull-up resistor internally connects a pin to VTG to give it a defined state (logic high, i.e., 1)
  - Ex. Write the code that will:
    - Make Port A pins *inputs*
    - Turn on pull-up resistors
    - Read the voltages on the pins and store them in a variable, testA
      - What is the value of testA in binary and hex?



Reading Port A Pins Example

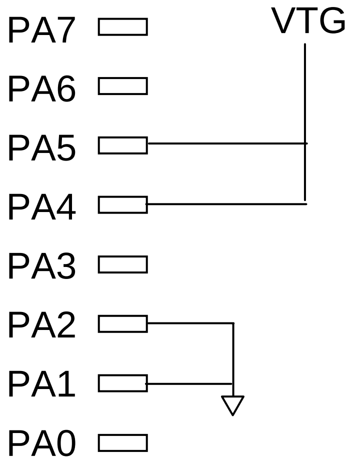
```
unsigned char testA;

DDRA=0;

testA=PINA;
```

What is the content of testA?

- binary: 11111001
- hex: F9


PA7
VTG
PA6
PA5
PA4
PA3
PA2
PA1
PA0