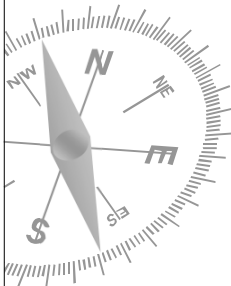




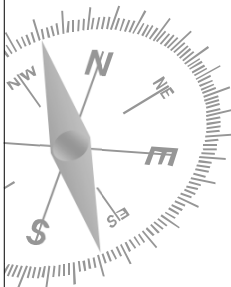
# Atmel AVR Basics

A. A. Arroyo



# Overview

- ▶ Basic AVR Knowledge
- ▶ AVR Studio
- ▶ Programming





## AVR Basics

► The AVR microcontrollers are divided into three groups:

- 1. *tinyAVR*
- 2. AVR (Classic AVR)
- 3. *megaAVR* ☒

► The difference between these devices lies in the available features. The *tinyAVR*  $\mu$ C are usually devices with lower pin-count or reduced feature set compared to the *megaAVR*'s. All AVR devices have the same instruction set and memory organization.



## AVR Basics

► AVR's contain SRAM, EEPROM, External SRAM interface, ADC, Hardware Multiplier, UART, USART, etc. A *tinyAVR* and a *megaAVR* stripped off all the peripheral modules, leaves us with the AVR Core — the same for all AVR's.

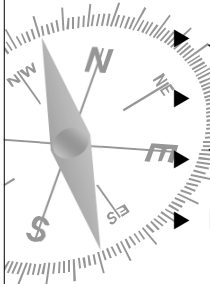
► Datasheets are complete technical documents — a reference on how a given peripheral/feature works.

- 1. One Page—Key information and Feature List
- 2. Architectural Overview
- 3. Peripheral Descriptions
- 4. Memory Programming
- 5. Characteristics
- 6. Register Summary
- 7. Instruction Set Summary
- 8. Packaging Information



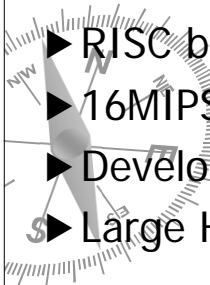
## Peripherals

- ▶ UART
  - Provides serial communication interface
  - Typically RS232
- ▶ Analog/Digital Channels
  - Many sensors are analog
  - 4 channels minimum
- ▶ SPI/I2C/CAN interfaces
  - Synchronous serial, very common interfaces
- ▶ JTAG
  - Allows in-circuit debugging and programming
- ▶ ISP
  - In-system programming, serial communication
- ▶ Timers
  - The more the better
- ▶ PWM / frequency generation modules
  - Allows hardware generation of motor driving signals



## ATMega 128

- ▶ Most popular here at UF
- ▶ Kitchen sink of peripherals
  - 2 UARTS, 8 channel A/D, multiple counters and PWM channels, SPI/I2C/CAN, ISP, and JTAG
- ▶ RISC based 8Bit core
- ▶ 16MIPS @ 16Mhz
- ▶ Development Software free from Atmel
- ▶ Large Hobbyist user base on the internet





# AVR Studio....

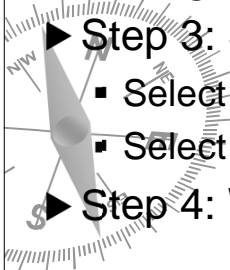
- ▶ Step 1: Create a New Project
- ▶ Step 2: Configure Project Settings

What kind of project we want to create, and setting up filenames and locations.

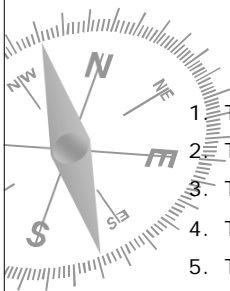
- ▶ Step 3: Selecting Debugging Platform

- Select JTAG ICE
- Select ATmega128

- ▶ Step 4: Write your code



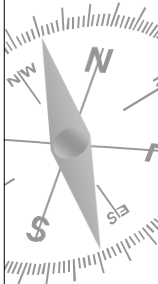
# AVR Studio....



1. The "Menus" - Windows and AVR Studio specific
2. Toolbars - "shortcuts" to commonly used functions.
3. The Workspace - Files in your Project, IO view, and AVR info
4. The Editor window.
5. The Output Window. Status information is displayed here.
6. The System Tray - information on which mode AVR Studio is in.



# AVR Studio....



University of Florida, EEL 5666  
© Dr. A. Antonio Arroyo

```

AVR Studio - C:\Arroyo\EEL5666\Spring 2009\Atmel\Test_Atmel_ASM\Test_Atmel_ASM.asm
-----
*
* Title:      test_atmel_asm.asm
* Programmer: A. Antonio Arroyo
* Date:      February 10, 2009
* Version:   1.0
*
* Description:
* Test the Atmel ATMEGA128 using the Assembler
*
*
=====
#include "8515def.inc" ;Includes an Atmega128 compatible definitions file
.org 0x0000           ;Places the following code from address 0x0000
rjmp RESET           ;Take a Relative Jump to the RESET Label
RESET:               ;Reset Label (start of Main)
ldi R16, 0x0A        ;Store 10 in R16 (Repeat Count)
ldi R17, 0x32        ;Store 50 in R17 (i, Outer Loop)
ldi R18, 0xFF        ;Store 255 in R18 (j, Middle Loop)
ldi R19, 0xFF        ;Store 255 in R18 (k, Inner Loop)
ldi R20, 0xFF        ;Store 255 in R19 (Output Data)
out DDRB, R20        ;Store FF in DDRB (All outputs)

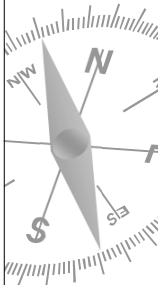
; Set all PORTB pins high
Loop:                ;Loop Label
out PORTB, R20       ;Write Data to PORTB
dec R19              ;Decrement R19
brne Loop            ;If not zero jump to the Loop label
ldi R19, 0xFF        ;Restore 255 in R19
dec R18              ;Decrement R18
brne Loop            ;If not zero jump to the Loop label
ldi R18, 0xFF        ;Restore 255 in R18
ldi R19, 0xFF        ;Restore 255 in R19
dec R17              ;Decrement R17
brne Loop            ;If not zero jump to the Loop label
; Set all PORTB pins low
ldi R17, 0x32        ;Store 50 in R17
ldi R18, 0xFF        ;Store 0xFF in R18
ldi R19, 0xFF        ;Store 0xFF in R18
ldi R20, 0x00        ;Store 0 in Data (R20)

```

9



# Assembly Code Sample



University of Florida, EEL 5666  
© Dr. A. Antonio Arroyo

```

C:\Arroyo\EEL5666\Spring 2009\Atmel\Test_Atmel_ASM\Test_Atmel_ASM.asm - Notepad2
-----
*
* Title:      test_atmel_asm.asm
* Programmer: A. Antonio Arroyo
* Date:      February 10, 2009
* Version:   1.0
*
* Description:
* Test the Atmel ATMEGA128 using the Assembler
*
*
=====
#include "8515def.inc" ;Includes an Atmega128 compatible definitions file
.org 0x0000           ;Places the following code from address 0x0000
rjmp RESET           ;Take a Relative Jump to the RESET Label
RESET:               ;Reset Label (start of Main)
ldi R16, 0x0A        ;Store 10 in R16 (Repeat Count)
ldi R17, 0x32        ;Store 50 in R17 (i, Outer Loop)
ldi R18, 0xFF        ;Store 255 in R18 (j, Middle Loop)
ldi R19, 0xFF        ;Store 255 in R18 (k, Inner Loop)
ldi R20, 0xFF        ;Store 255 in R19 (Output Data)
out DDRB, R20        ;Store FF in DDRB (All outputs)

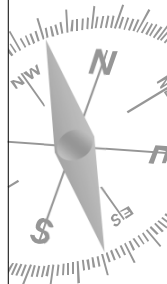
; set all PORTB pins high
Loop:                ;Loop Label
out PORTB, R20       ;Write Data to PORTB
dec R19              ;Decrement R19
brne Loop            ;If not zero jump to the Loop label
ldi R19, 0xFF        ;Restore 255 in R19
dec R18              ;Decrement R18
brne Loop            ;If not zero jump to the Loop label
ldi R18, 0xFF        ;Restore 255 in R18
ldi R19, 0xFF        ;Restore 255 in R19
dec R17              ;Decrement R17
brne Loop            ;If not zero jump to the Loop label
; Set all PORTB pins low
ldi R17, 0x32        ;Store 50 in R17
ldi R18, 0xFF        ;Store 0xFF in R18
ldi R19, 0xFF        ;Store 0xFF in R18
ldi R20, 0x00        ;Store 0 in Data (R20)

```

10



# Assembly Code Sample



```
* C:\Arroyo\EEL5666\Spring 2009\Atmel\Test_Atmel_ASM\Test_Atmel_ASM.asm - Note...
File Edit View Settings ?
[Icons]
Loop2:
  out PORTB, R20      ;Write Data to PORTB
  dec R19             ;Decrement R19
  brne Loop2         ;If not zero jump to the Loop label
  ldi R19, 0xFF       ;Restore 255 in R19
  dec R18             ;Decrement R18
  brne Loop2         ;If not zero jump to the Loop label
  ldi R18, 0xFF       ;Restore 255 in R18
  ldi R19, 0xFF       ;Restore 255 in R19
  dec R17             ;Decrement R17
  brne Loop2         ;If not zero jump to the Loop label

;Repeat Count times
  ldi R17, 0x32       ;Restore 50 to R17
  ldi R18, 0xFF       ;Restore 255 to R18
  ldi R19, 0xFF       ;Restore 255 to R19
  ldi R20, 0xFF       ;Store 0xFF in Data
  dec R16             ;Decrement COUNT
  brne Loop          ;If not zero jump to the Loop label

Here:
  rjmp Here

Ln 37 : 61 Col 1 Sel 0      2.27 KB ANSI CR+LF INS Assembler
```

University of Florida, EEL 5666  
© Dr. A. Antonio Arroyo