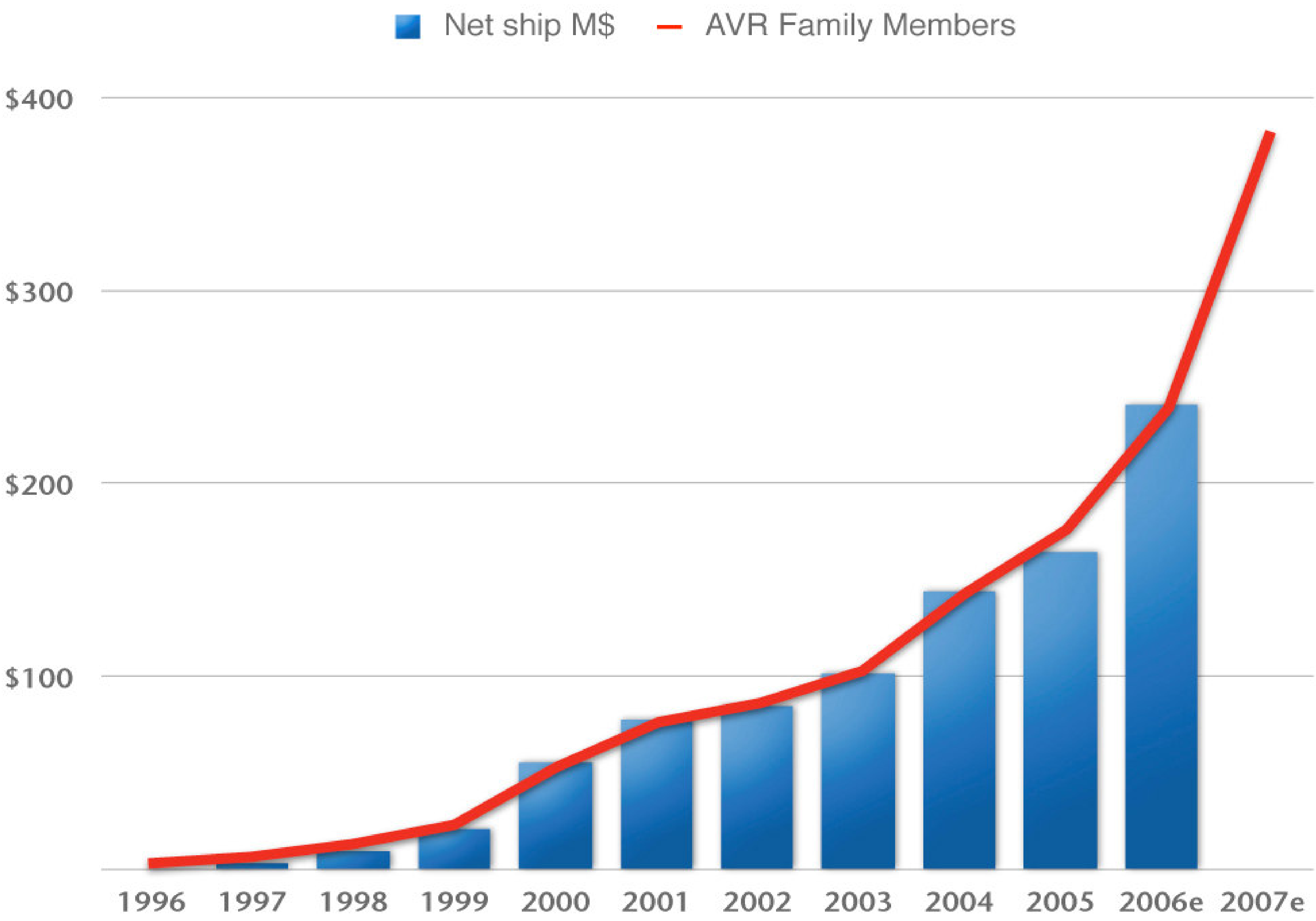
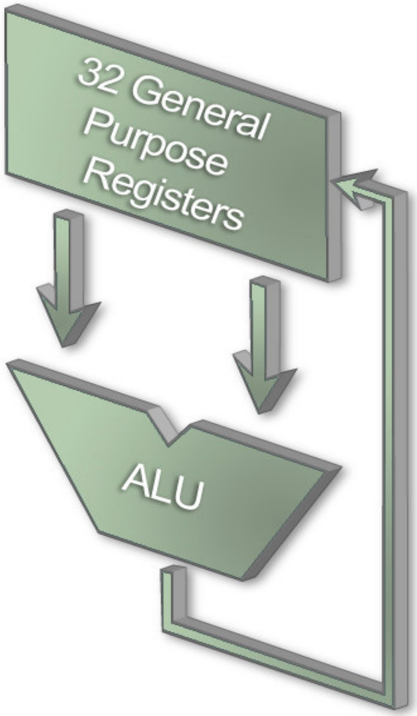
AVR Introduction

AVR Microcontrollers



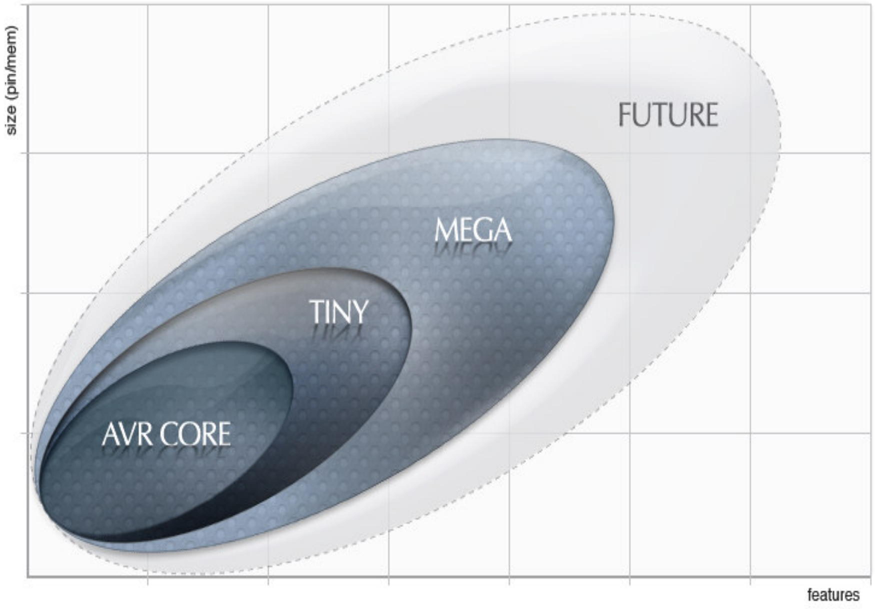
The Growing AVR Family

- TINY AVR family
  - 8 - 32 pin general purpose microcontrollers
  - 16 family members
- MEGA AVR family
  - 32 - 100 pin general purpose microcontrollers
  - 23 family members
- ASSP AVRs
  - USB, CAN and LCD
  - Motor Control and Lighting
  - Automotive
  - Battery Management
  - 8 family members



Scalable

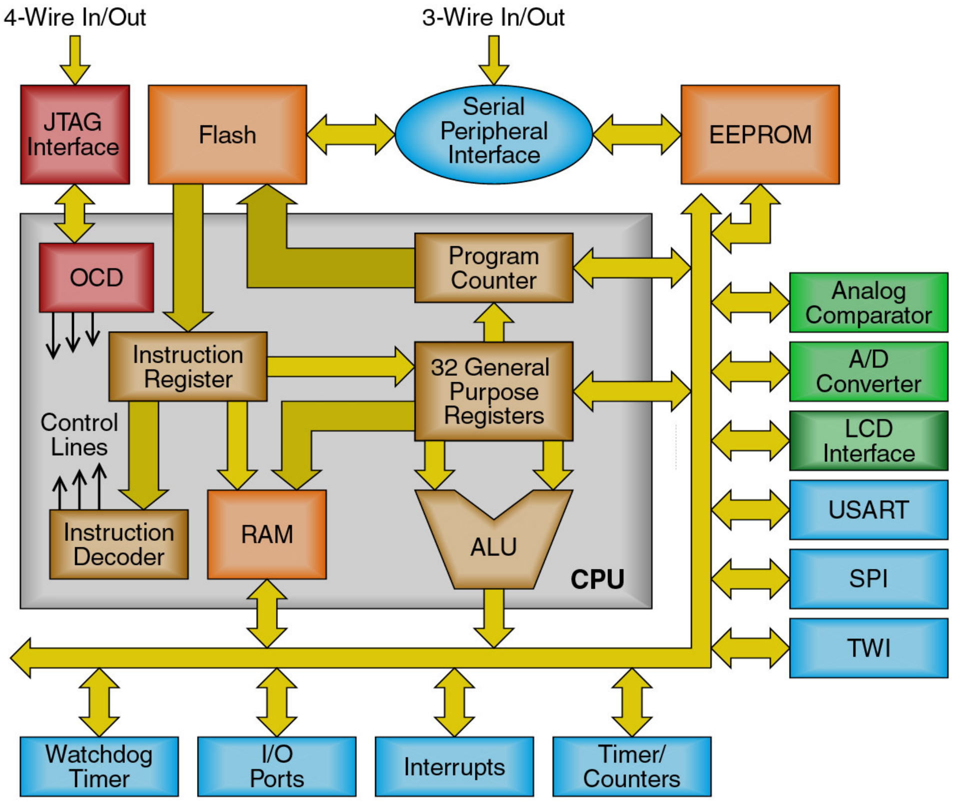
- Devices range from 1 to 256KB
- Pin count range from 8 to 100
- Full code compatibility
- Pin/feature compatible families
- One set of development tools



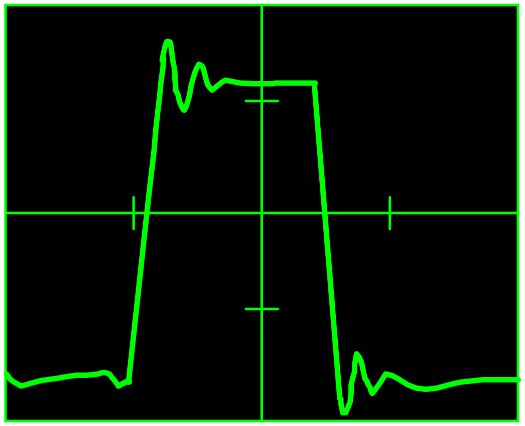
= Roadmap for the future

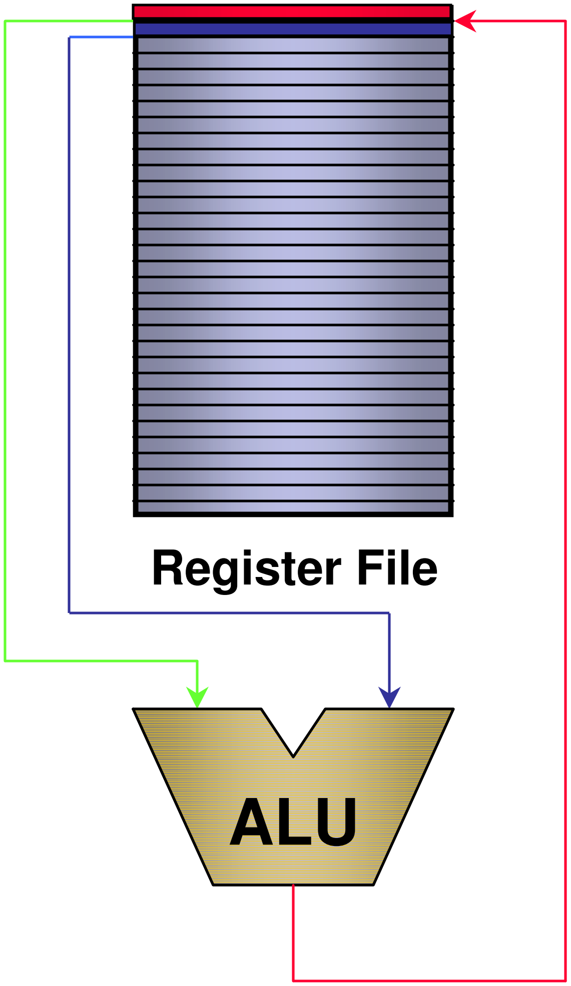
AVR Architecture

- RISC architecture with CISC instruction set
  - Powerful instruction set for C and Assembly
- Scalable
  - Same powerful AVR core in all devices
- Single cycle execution
  - One instruction per external clock
  - Low power consumption
- 32 Working Registers
  - All Directly connected to ALU!
- Very efficient core
  - 20 MIPS @ 20MHz
- High System Level Integration
  - Lowest total system cost



Single Cycle Execution





Register operations take ONE clock pulse on the EXTERNAL clock input

20MIPS @ 20MHz

High Code Density

- Architecture designed for C
- 32 general registers
- C-like addressing modes
- 16- and 32-bit arithmetic support
- Linear address maps

AVR Influenced by IAR

- Architecture and Instruction Set co-designed with IAR systems through several iterations:
  - Compiler development project initiated before architecture and instruction set frozen
  - Compiler experts’ advice implemented in hardware
  - Potential HLL bottlenecks identified and removed

```
Auto Increment/Decrement Example:

C Source:
   unsigned char *var1, *var2;
   *var1++ = *--var2;

Generated assembly code:
   LD   R16,-X
   ST   Z+,R16
```

Example: A Small C Function

```
/* Return the maximum value of
   a table of 16 integers */

int max(int *array)
{
  char a;
  int maximum=-32768;

  for (a=0;a<16;a++)
     if (array[a]>maximum)
        maximum=array[a];
  return (maximum);
}
```

Code Size and Execution Time

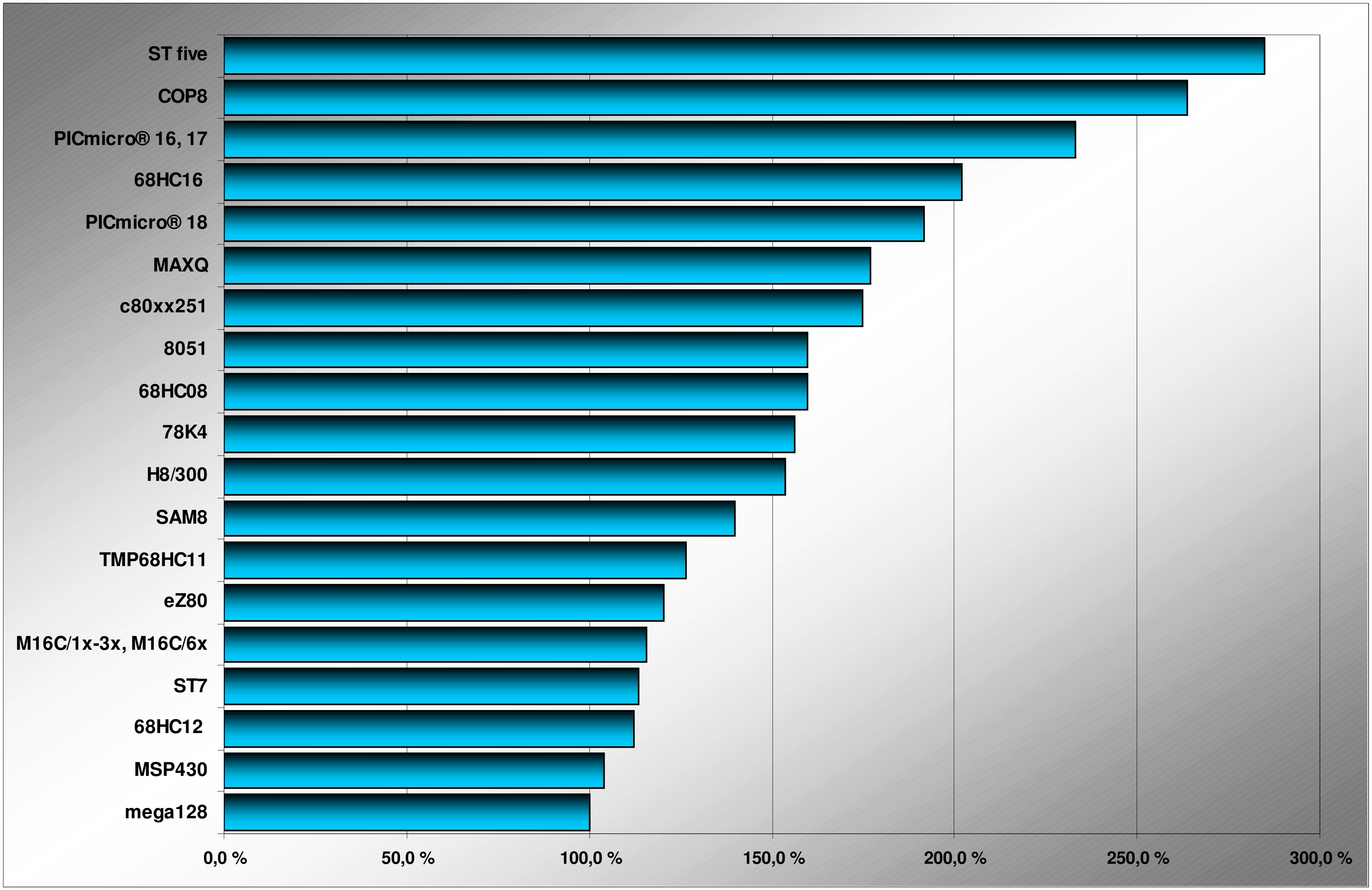
Device	Max Speed [MHz]	Code Size [Bytes]	Cycles	Execution Time [uS]
ATmega16	16	32	227	14.2
MSP430	8	34	246	30.8
T89C51RD2	20	57	4200	210.0
PIC18F452	40	92	716	17.9
PIC16C74	20	87	2492	124.6
68HC11	12	59	1238	103.2

- MSP430 and AVR are running a close race
  - But max speed on MSP430 is only 8MHz
- The C51 would have to run at 296 MHz to match the 16 MHz AVR
- PIC18 seems fast but require 3 times as much code space.

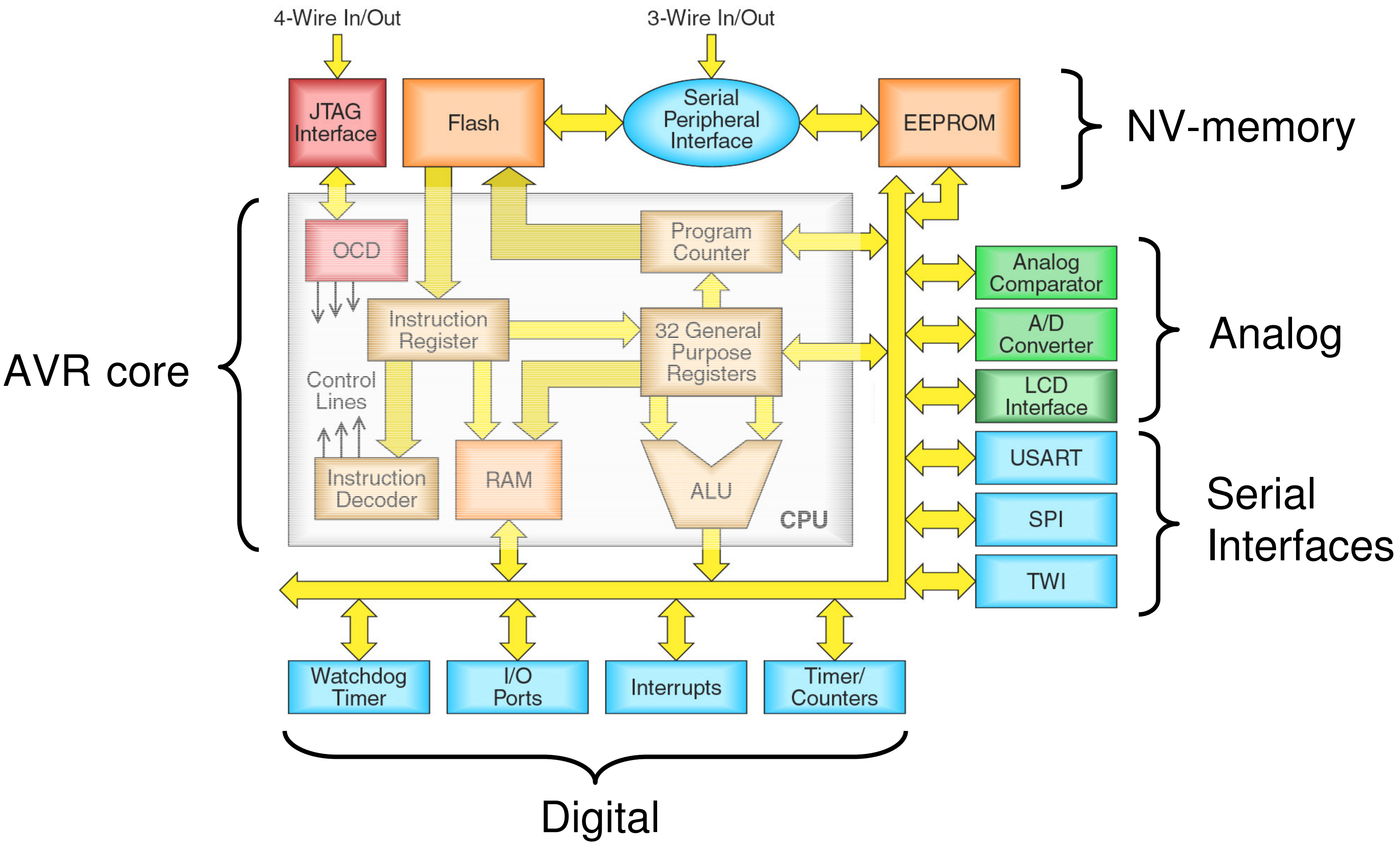
Benchmark: Real-life Applications

- Complete navigation application
- C bitfields
- Car Radio control
- DES encryption / decryption
- Three different modules from analog telephones
- Reed-Solomon (error correction) encoder/decoder
- Pager protocol
- Refridgerator control
- Battery charger
- Embedded web server
- Label/recite printer

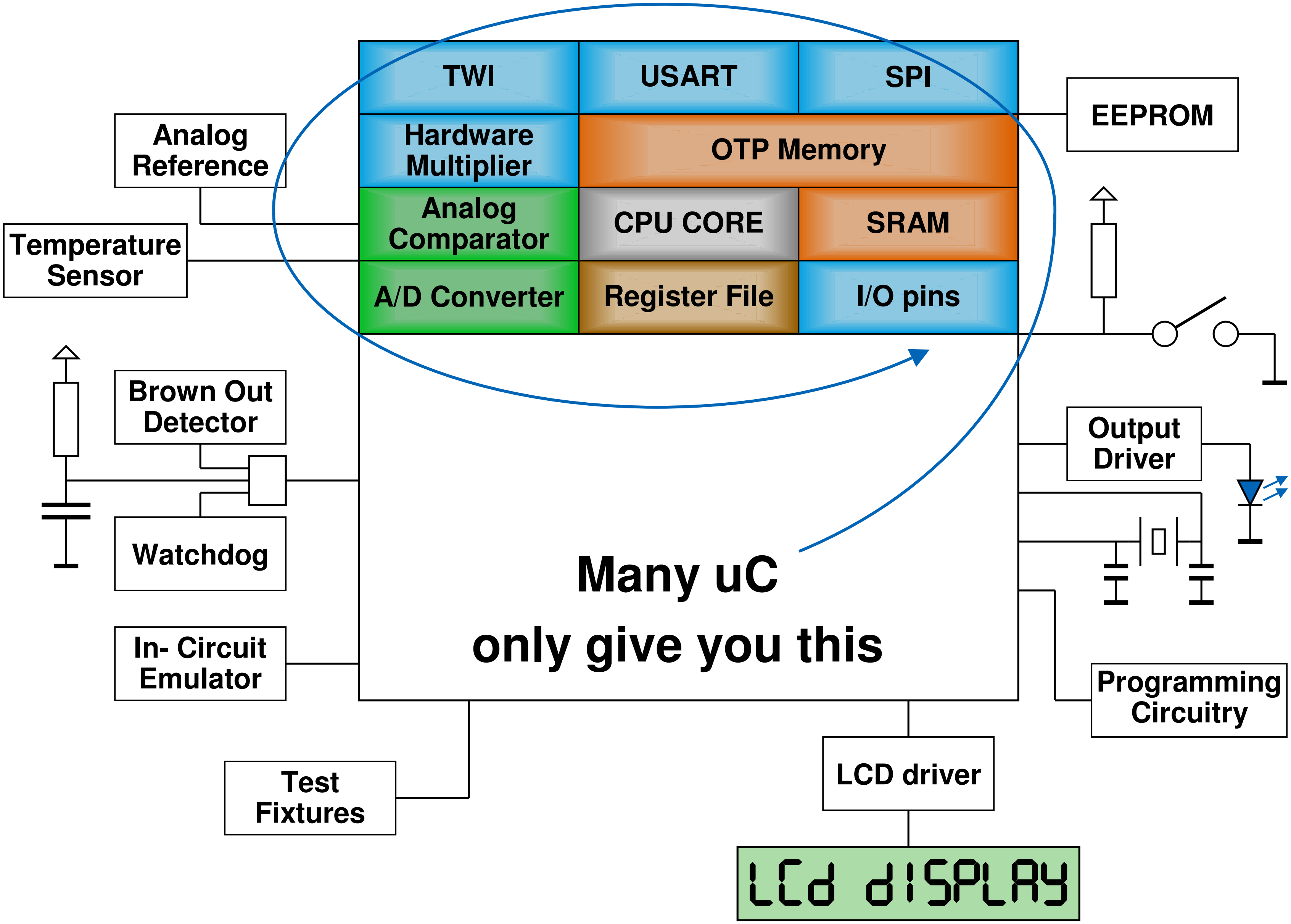
Benchmark – Code size



High Integration

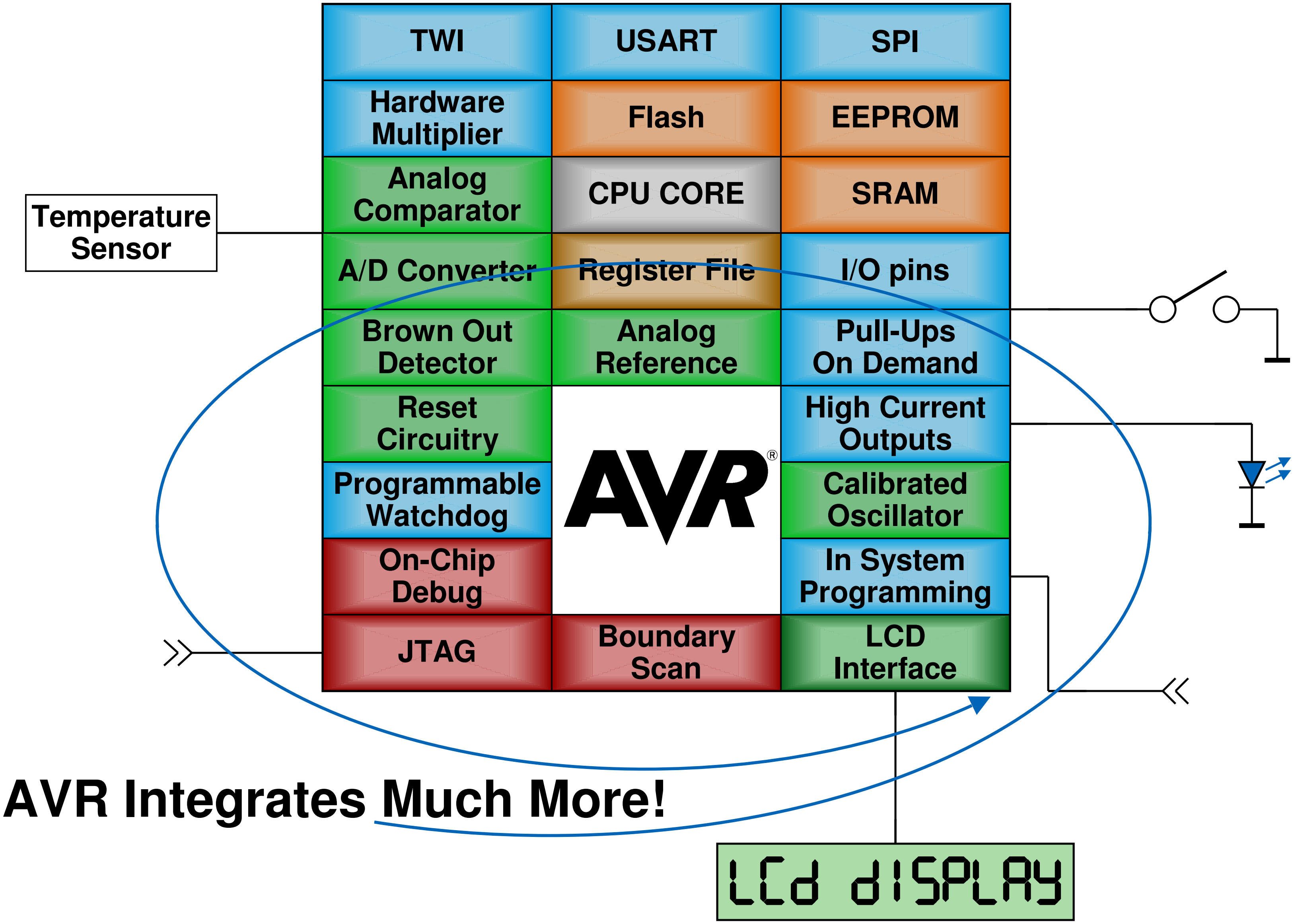


AVR – Single-Chip Solution



Many uC
only give you this

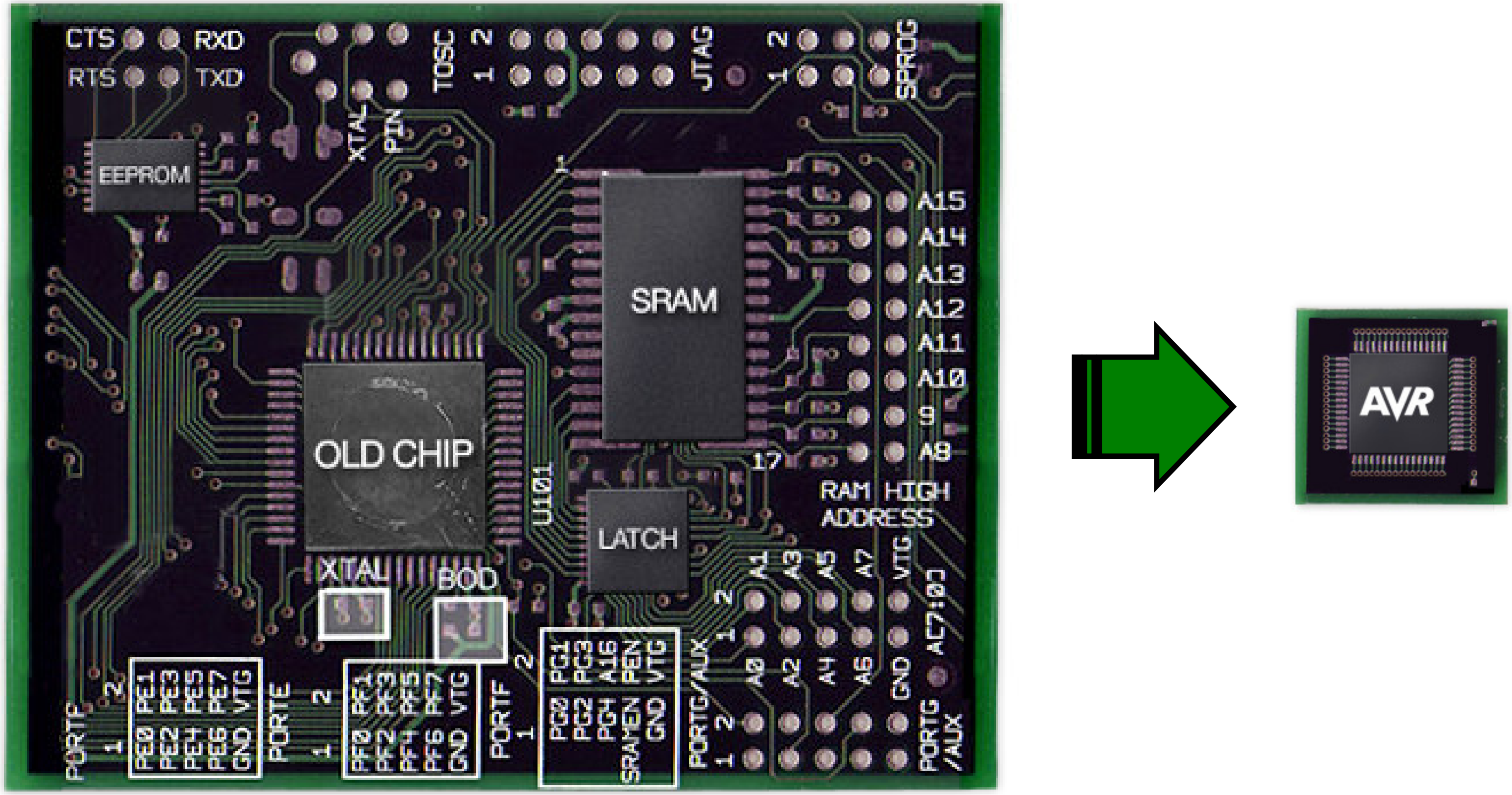
AVR – Single-Chip Solution



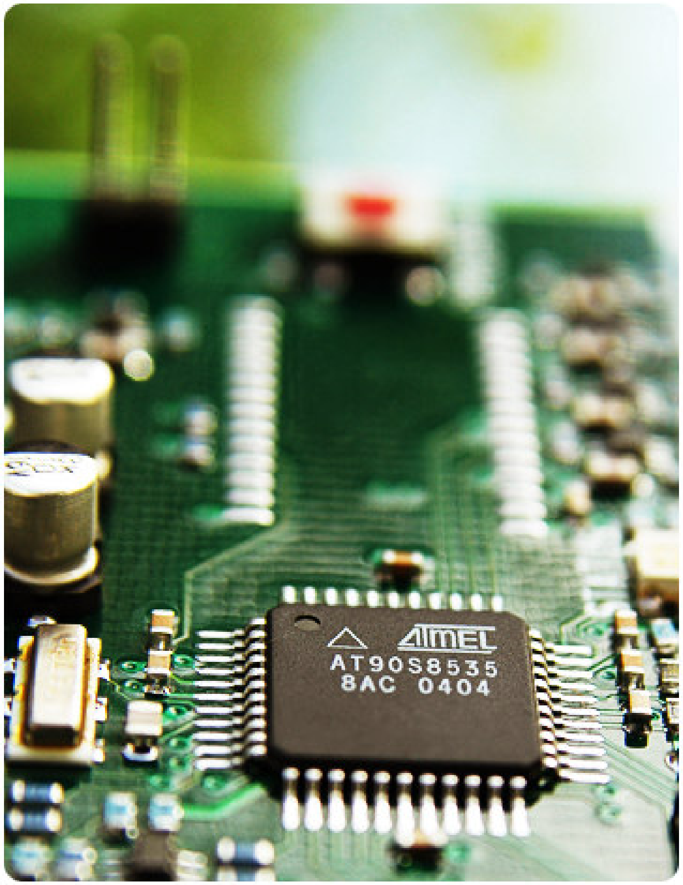
TWI	USART	SPI
Hardware Multiplier	Flash	EEPROM
Analog Comparator	CPU CORE	SRAM
A/D Converter	Register File	I/O pins
Brown Out Detector	Analog Reference	Pull-Ups On Demand
Reset Circuitry	AVR®	High Current Outputs
Programmable Watchdog		Calibrated Oscillator
On-Chip Debug		In System Programming
JTAG	Boundary Scan	LCD Interface

AVR Integrates Much More!

Highest System Level Integration



In-System Development



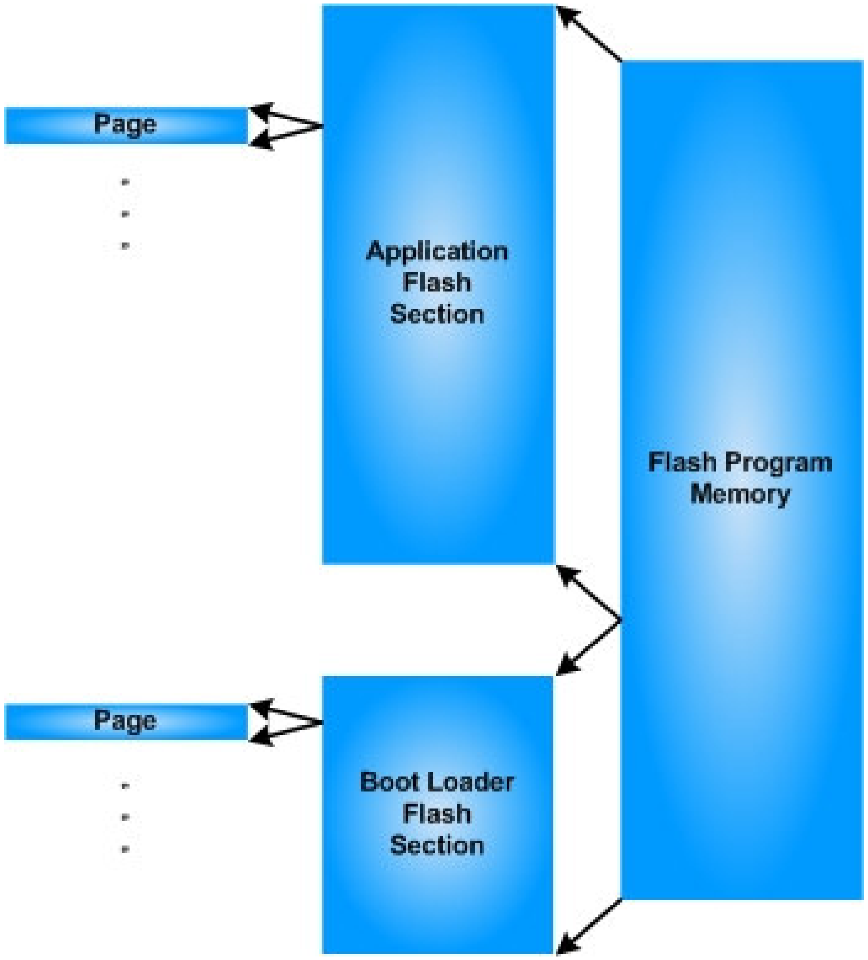
- In-System Programming
- In-System Debugging
- In-System Verification

AVR Self-Programming

- Redefining ISP → Self-Programming
  - The AVR reprograms itself
  - Any existing communication interface
  - Any voltage
  - Any frequency
- Critical functions still operating
  - Run code during programming (Read-While-Write)
- Software controlled programming
  - Firmware updates
  - Parameter updates

Boot Loader and Application Section

- The Flash program memory
- is divided into two sections
  - Application Section
  - Boot Loader Section
- The two sections enables the AVR to handle two independent applications
  - The Application section contain the main application
  - The Boot Loader section contain a Flash programming application
- Note that small AVRs does not divide the Flash
  - The whole Flash can be considered as a Boot Loader
  - Only on devices with 4K Flash or less

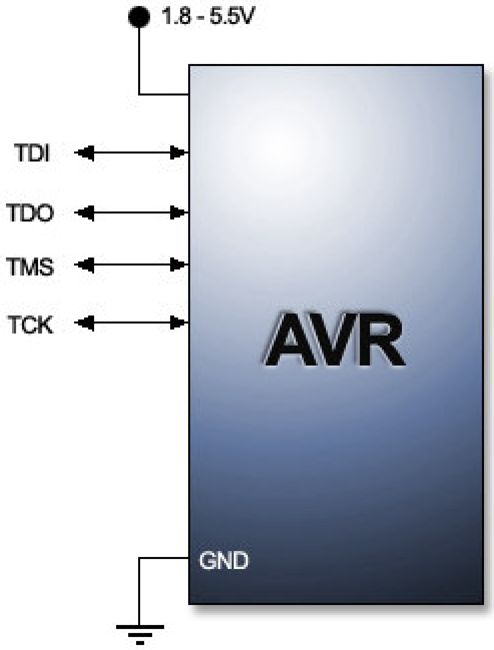


Self-Programming Flexibility

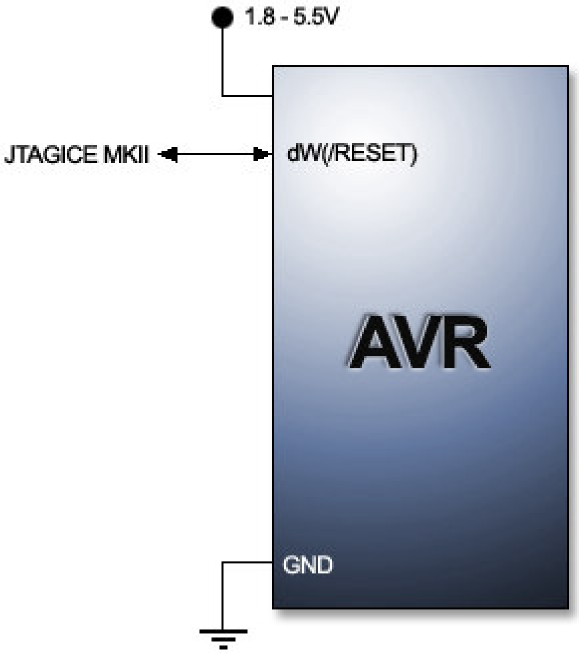
- AVR Self-Programming is controlled by SW
  - SPM instruction controls self-programming
  - SPM is an AVR core feature
  - Not a hard-coded firmware, but a part of the customer application
- The AVR updates its own Flash while running
  - Similar to AVR EEPROM access
  - Critical functions in the customers application can be maintained
- The upgrade data can be received from any interface
  - No restricted communication protocol or interface
  - No external hardware
- No restrictions to Vcc or Clock frequency

All AVRs have In System Debugging

- JTAG interface
  - On High pin-count devices
  - Uses 4 general I/O pins



- debugWIRE interface
  - On Low pin-count devices
  - Uses only Reset pin



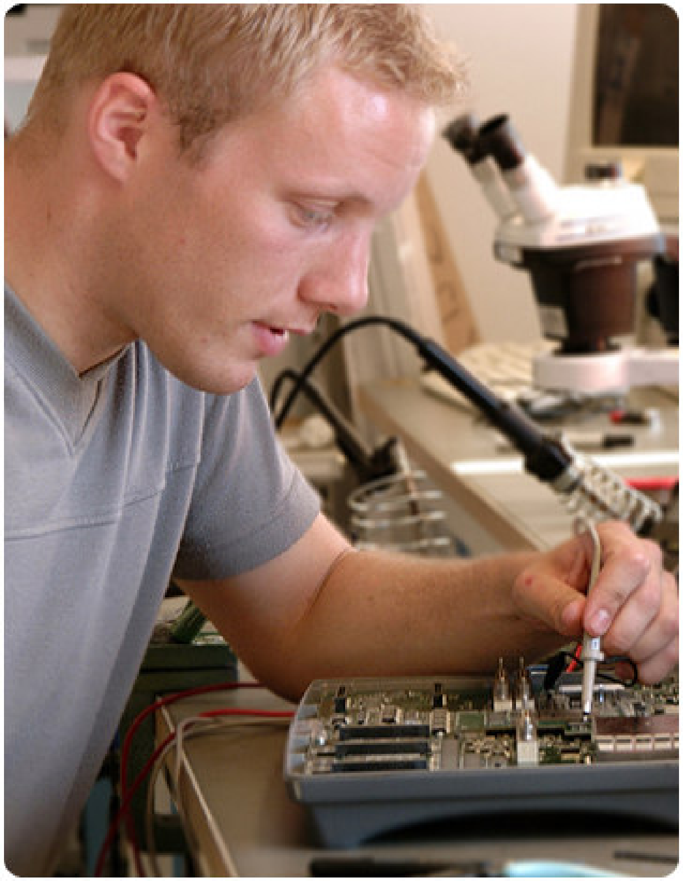
Compared to JTAGs four pins, debugWIRE uses only one; Reset.

This is a big advantage on low pin count devices

Development Tools

- AVR Studio - front end for all AVR tools
  - Free
- Starter kits and evaluation boards
  - From $19
- On-Chip Debuggers and Emulators
  - From $299

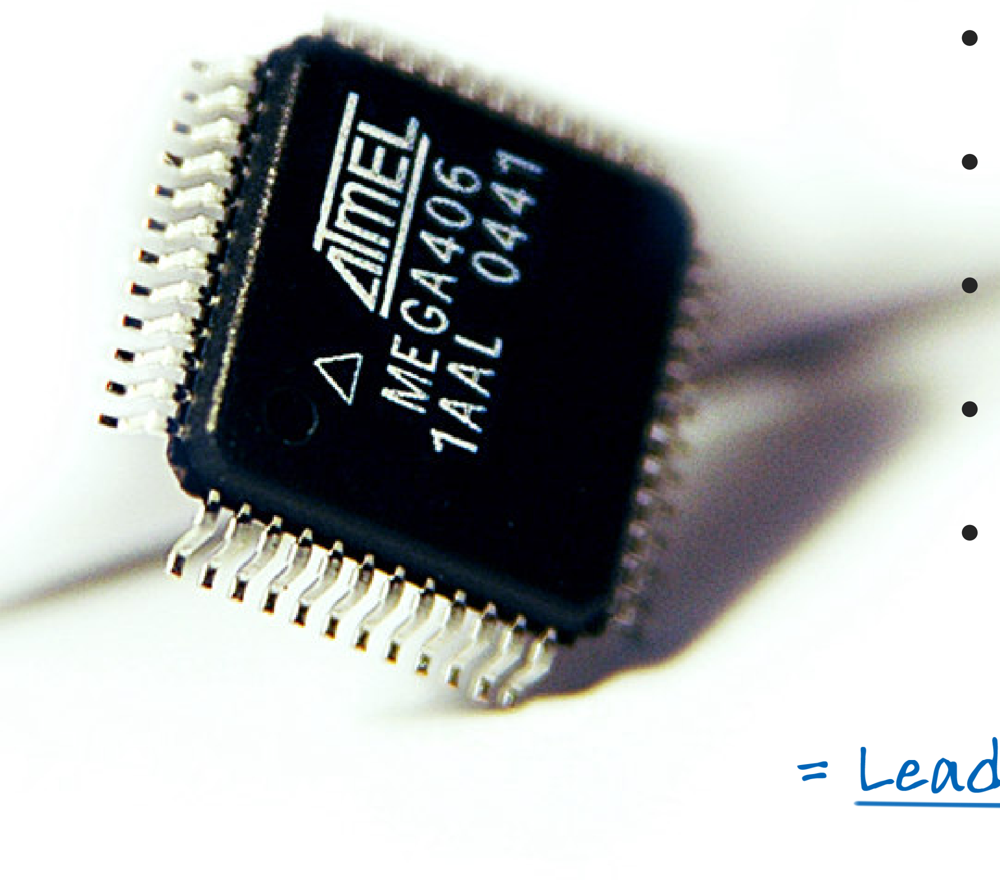
Support



- Fully updated product web
- Highly skilled Field Application Engineers
- Support mail handled by AVR experts
- Reference designs
- Application notes
- AVRfreaks community website

= Ensures no slip in schedule

Summary



- High performance
- Low power consumption
- High code density
- Advanced memory technology
- High integration

= Leading 8-bit microcontroller