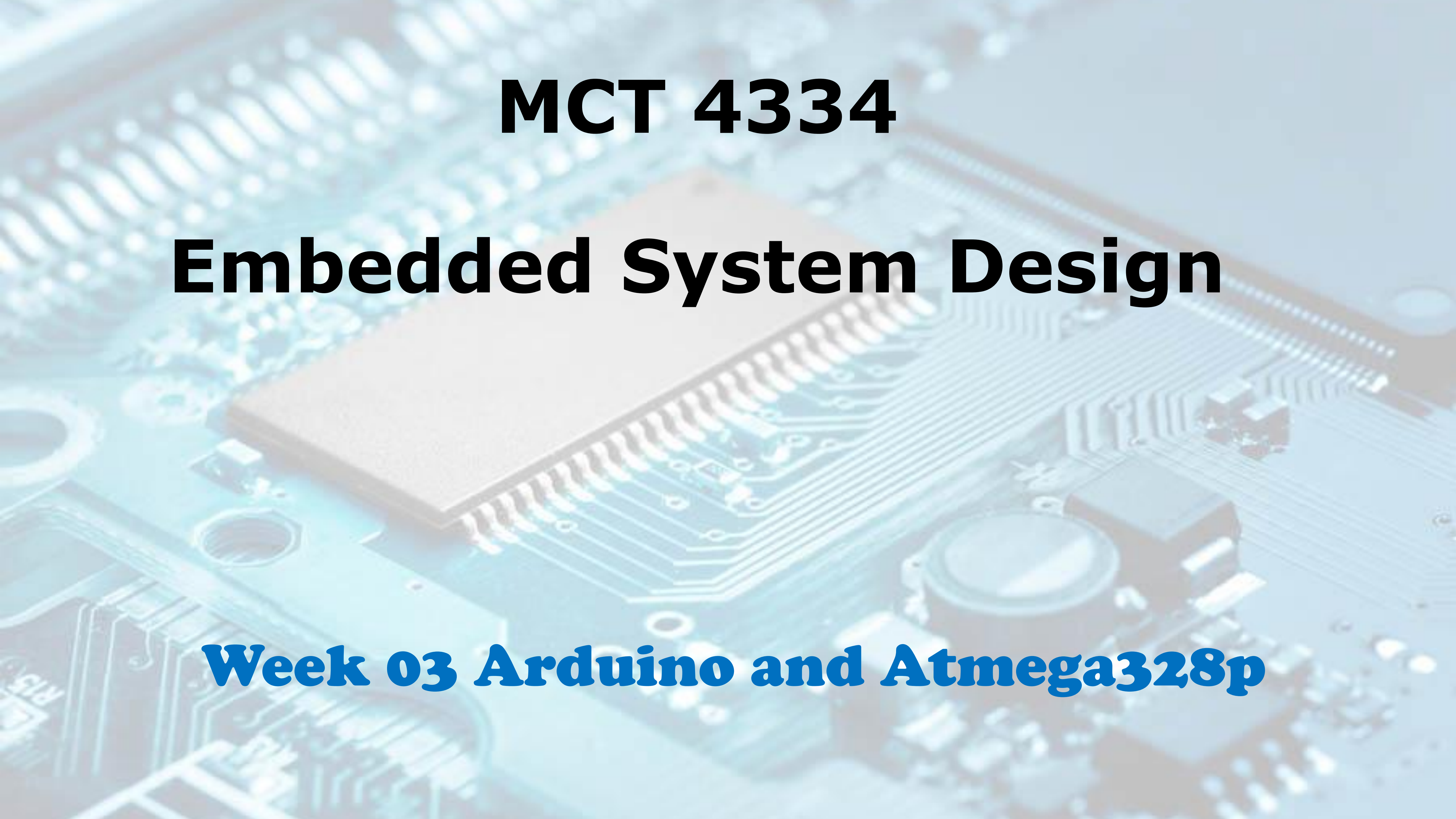


MCT 4334

Embedded System Design

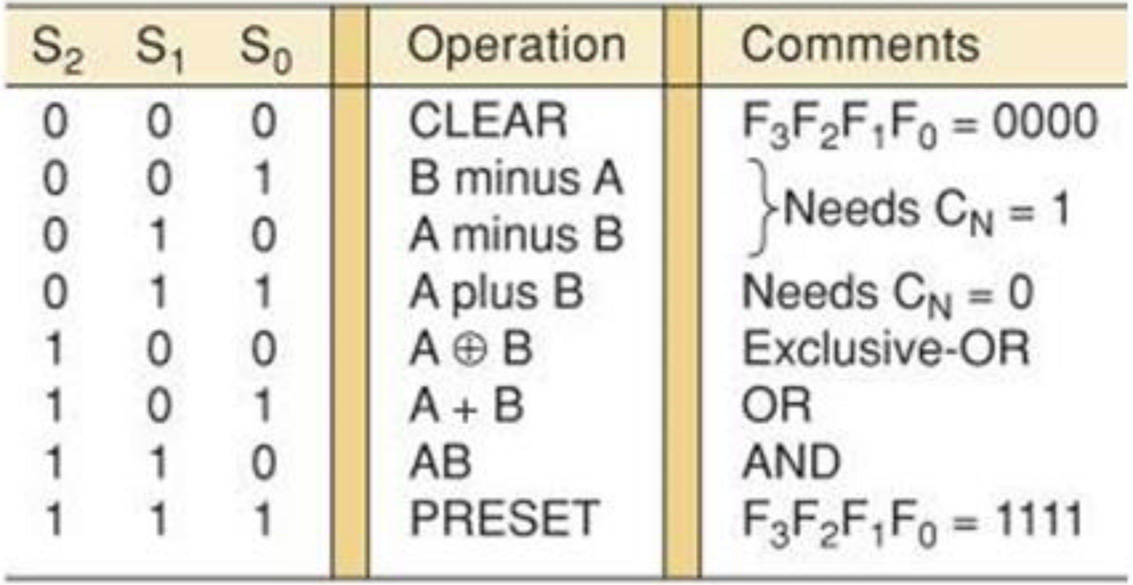
Week 03 Arduino and Atmega328p

Outline

- ATmega328P Architecture
- Introduction to Arduino
- Embedded debugging

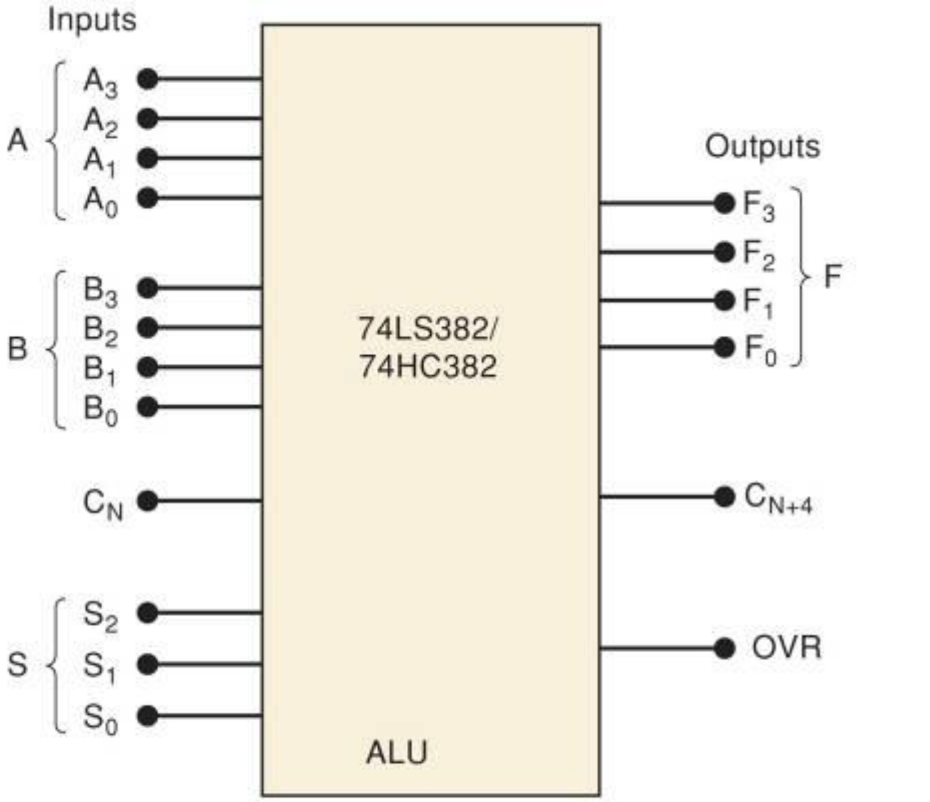
Basics of ALU (Review from DLD)

- The 74LS382 (TTL) or 74HC382 (CMOS) is a 4-bit ALU with 8 possible operations/instructions.



S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	$A \oplus B$	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

The table is the instruction set.

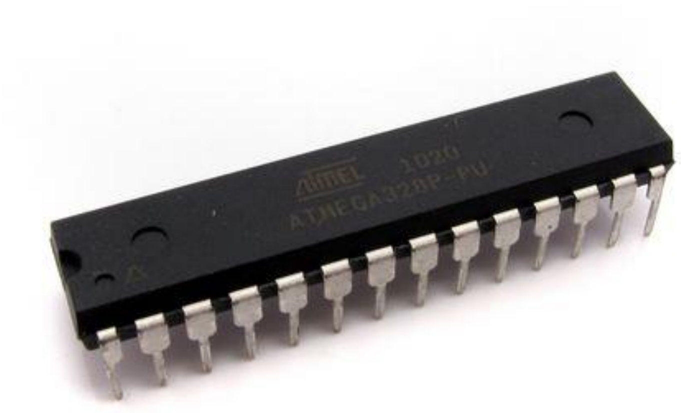


A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs

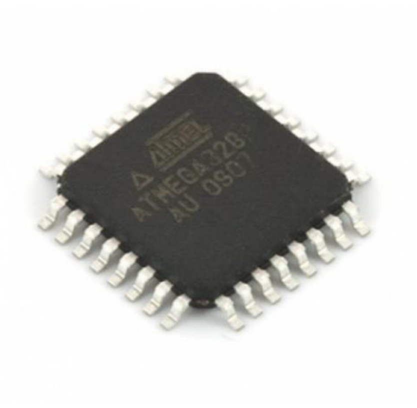
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

Specifications of ATmega328p

The Atmel’s ATmega328P is a low-power CMOS 8-bit microcontroller based on a RISC architecture.



- 131 instructions
- Most instructions are executed in single clock cycles
- 32 x 8 general purpose registers
- Up to 20MHz clock speed
- On-chip 2-cycle multiplier
- 128kHz and 8MHz internal oscillators



RISC vs CISC

- **Reduced Instruction Set Computer (RISC) Architecture = simple and small instruction set.**

  Each instruction only does a small amount of work.

  Example: Atmel microcontrollers, ARM processors used in mobile phones

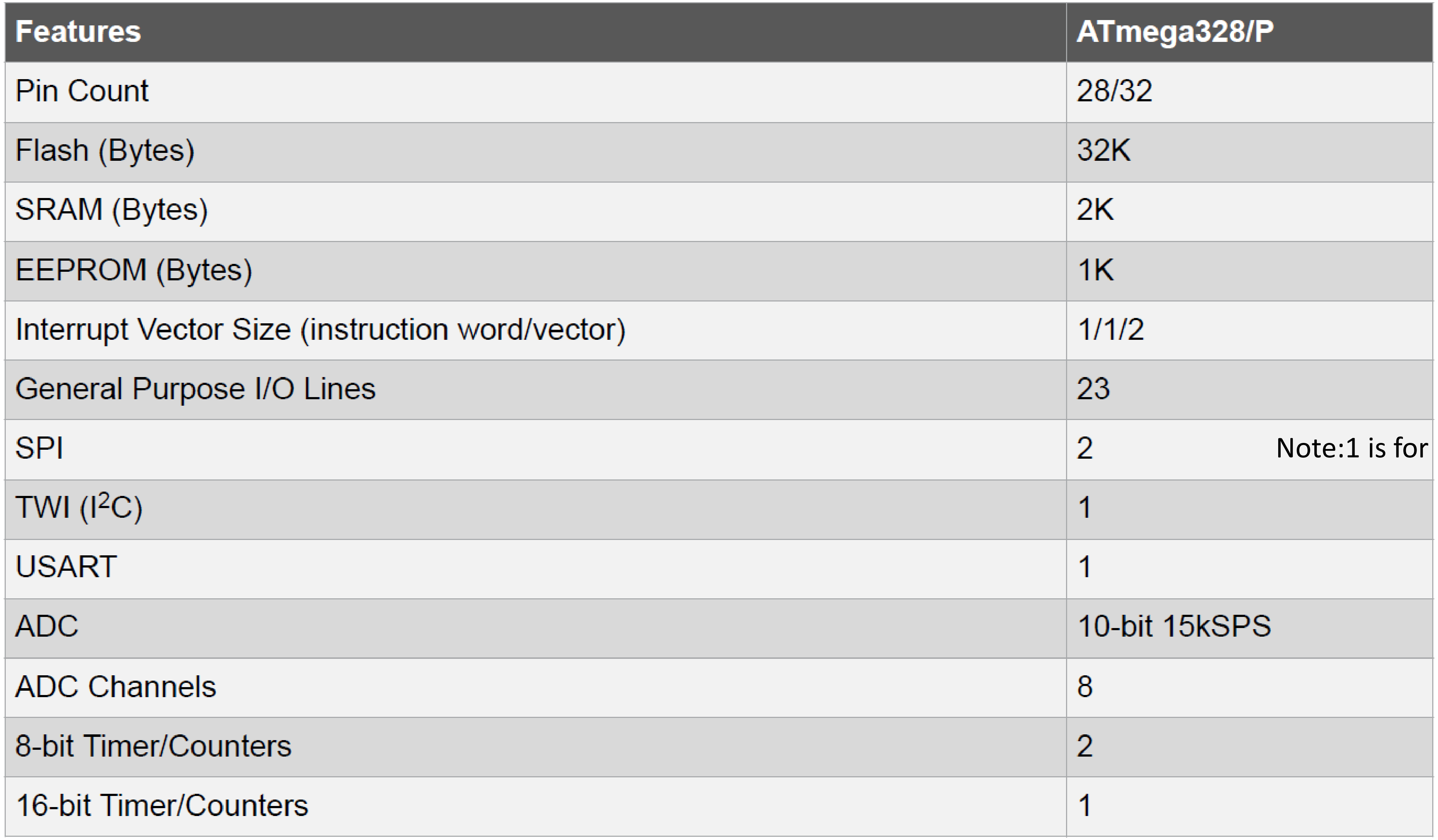
  Emphasizes low cost and power

- **Complex Instruction Set Computer (CISC) Architecture = complex and large instruction set**

  Example: Intel x86, Intel x64

  Emphasizes performance

Specifications of ATmega328p



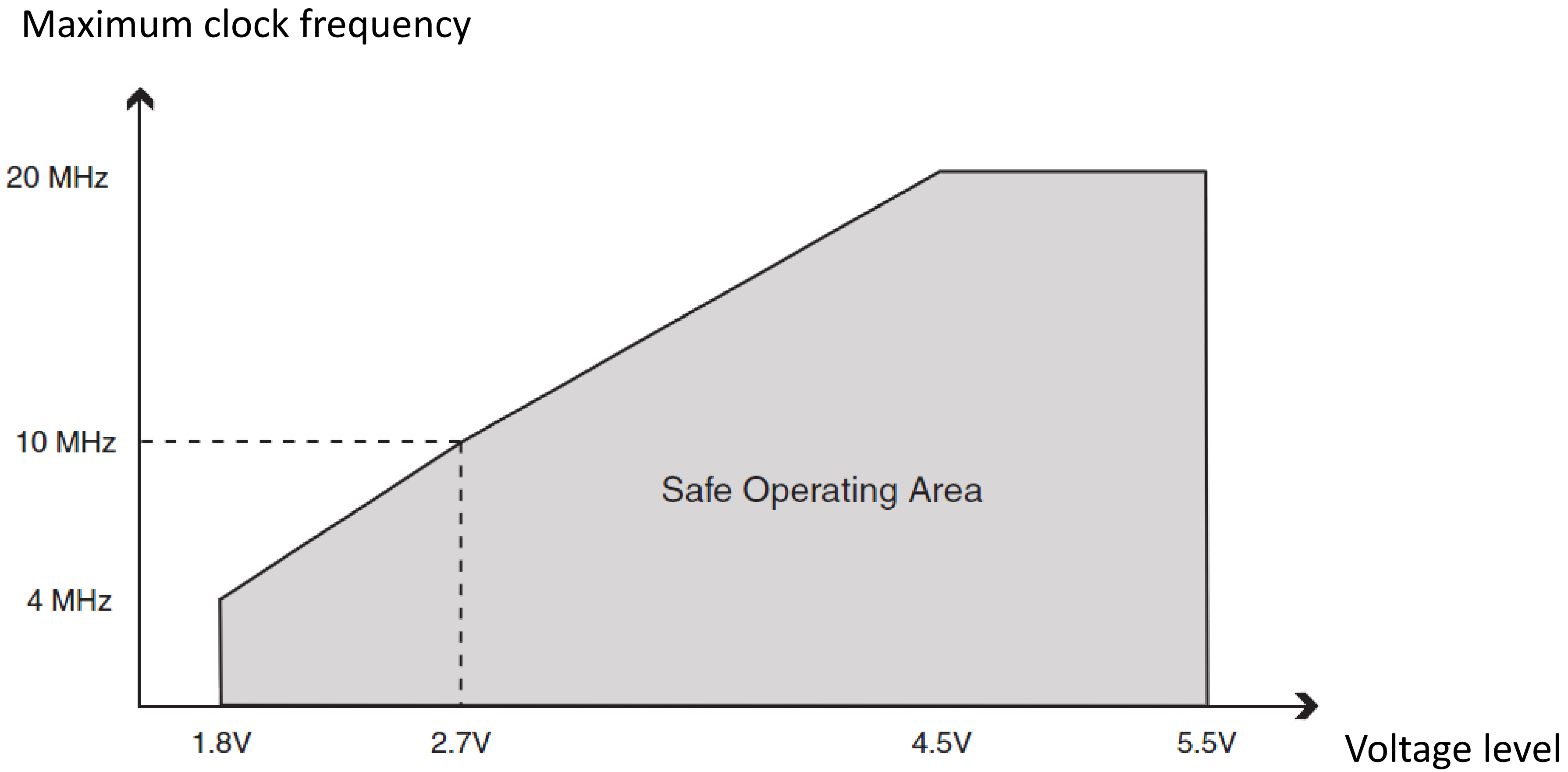
Features	ATmega328/P
Pin Count	28/32
Flash (Bytes)	32K
SRAM (Bytes)	2K
EEPROM (Bytes)	1K
Interrupt Vector Size (instruction word/vector)	1/1/2
General Purpose I/O Lines	23
SPI	2
TWI (I^{2}C)	1
USART	1
ADC	10-bit 15kSPS
ADC Channels	8
8-bit Timer/Counters	2
16-bit Timer/Counters	1

28 for DIP and 32 for SMD

Note:1 is for USART as SPI Master Mode

Clock frequency

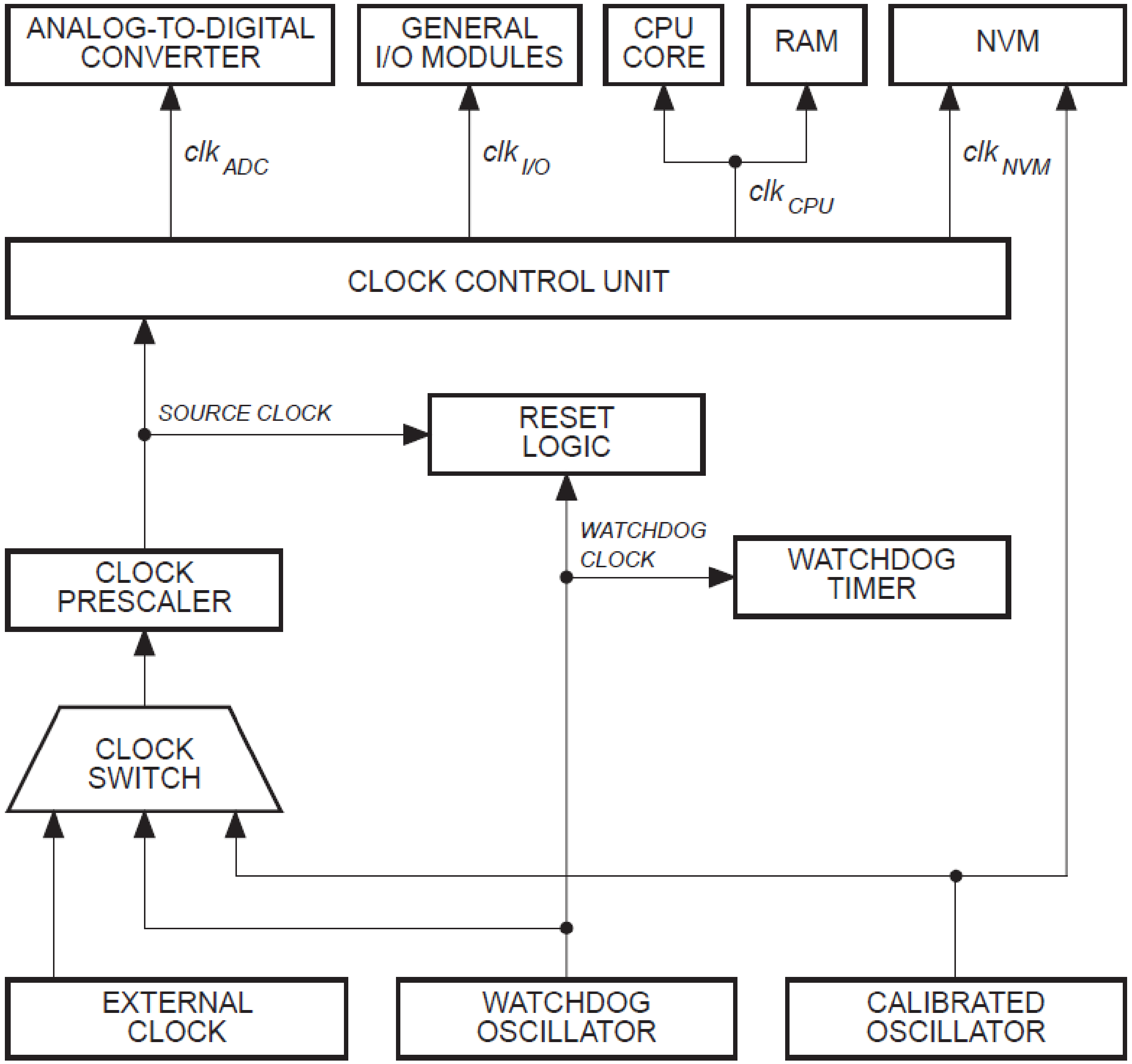
- Operating Voltage: 1.8 - 5.5V
- Temperature Range: -40°C to 105°C
- The lower the operating voltage, the lower the maximum clock frequency.



Clock Frequency

There are three clock sources for ATmega328p

- External clock
- Calibrated Internal 8 MHz oscillator
- Internal 128Khz low-power oscillator



IO

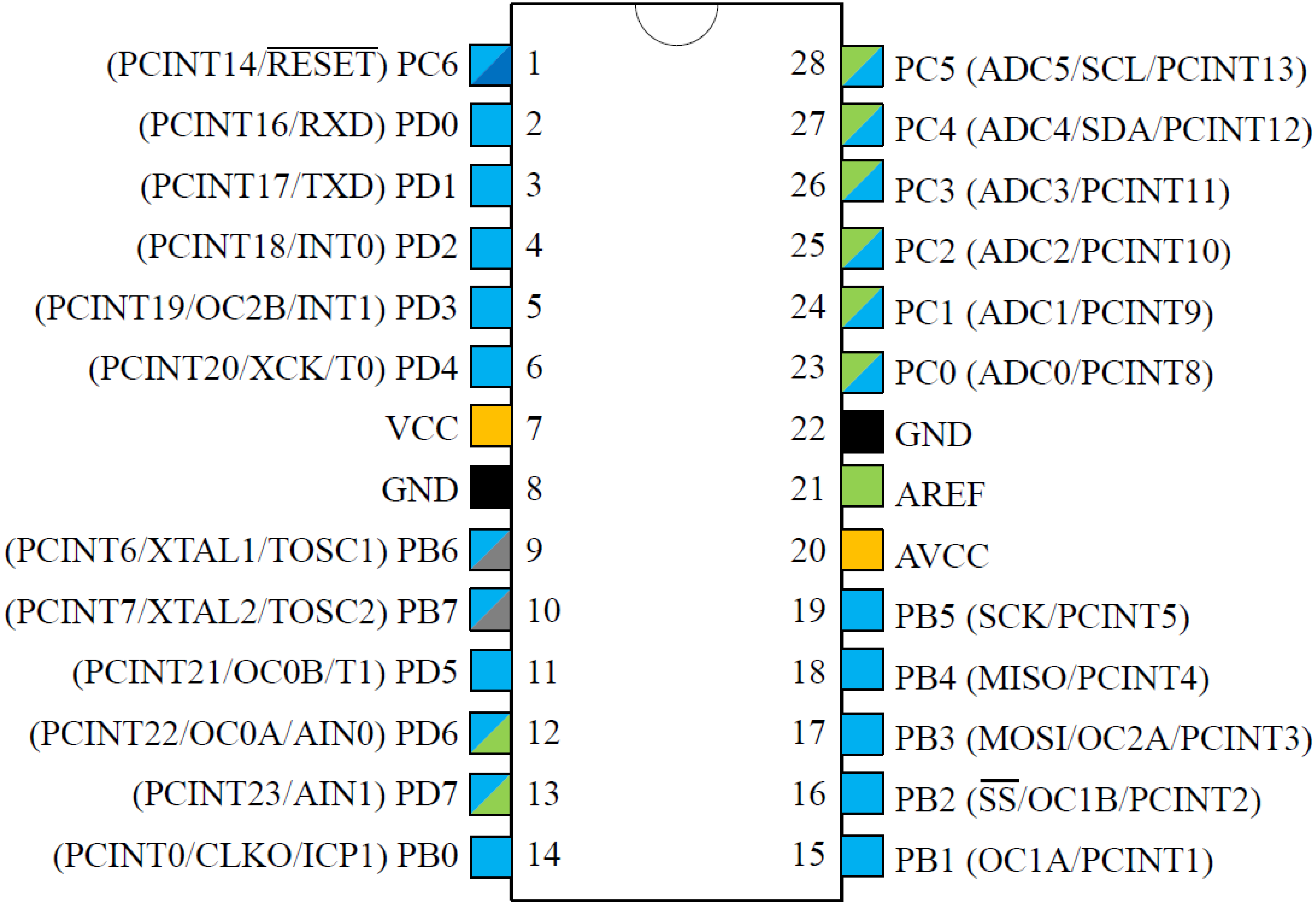
- Atmega328p supports 23 IO lines
- The IO lines can be access through 3 **ports** called (B, C and D)

Port B has 8 lines (PB0 to PB7)

Port C has 7 lines (PC0 to PC6)

Port D has 8 lines (PD0 to PD7)

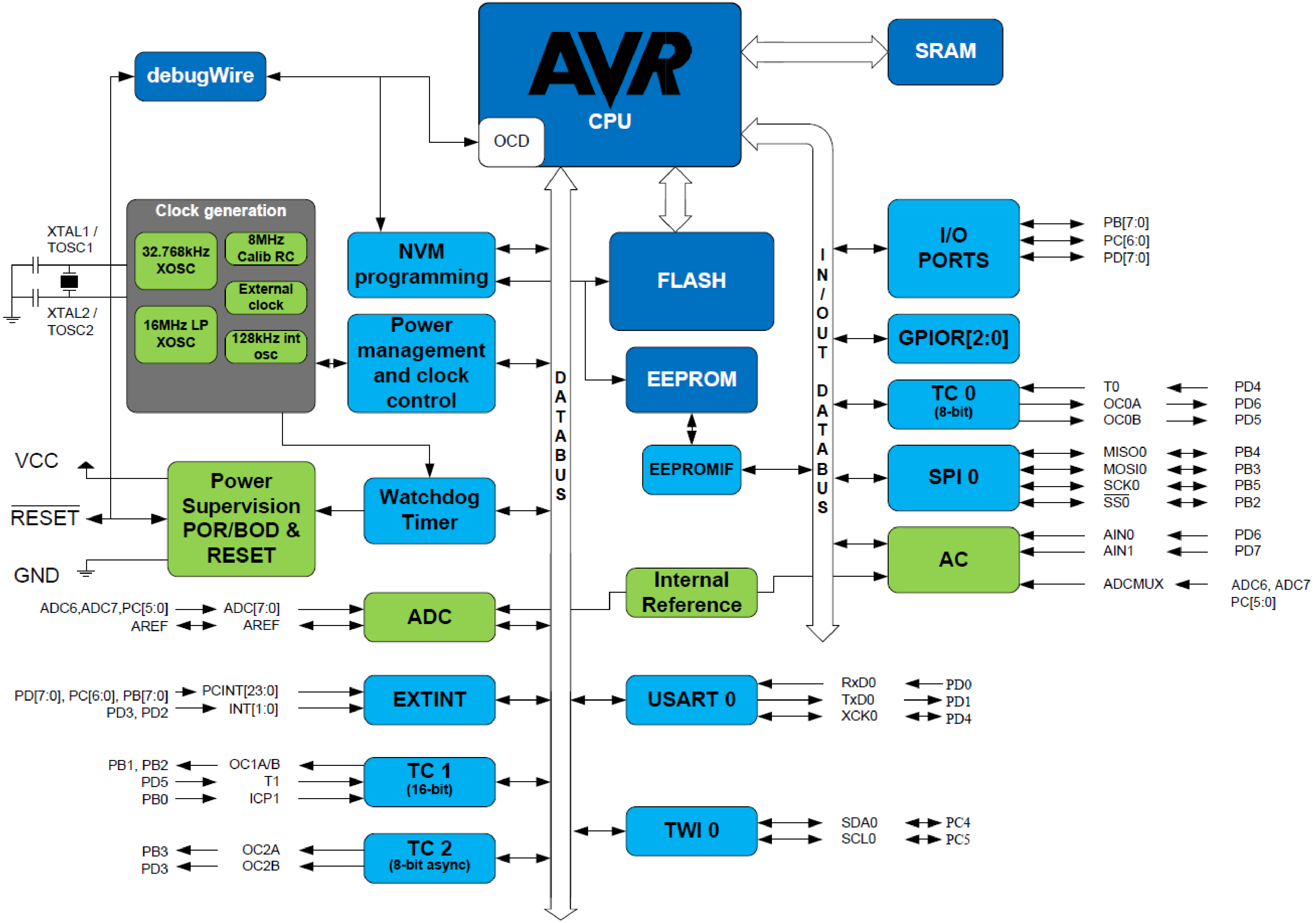
ATmega328p pinout

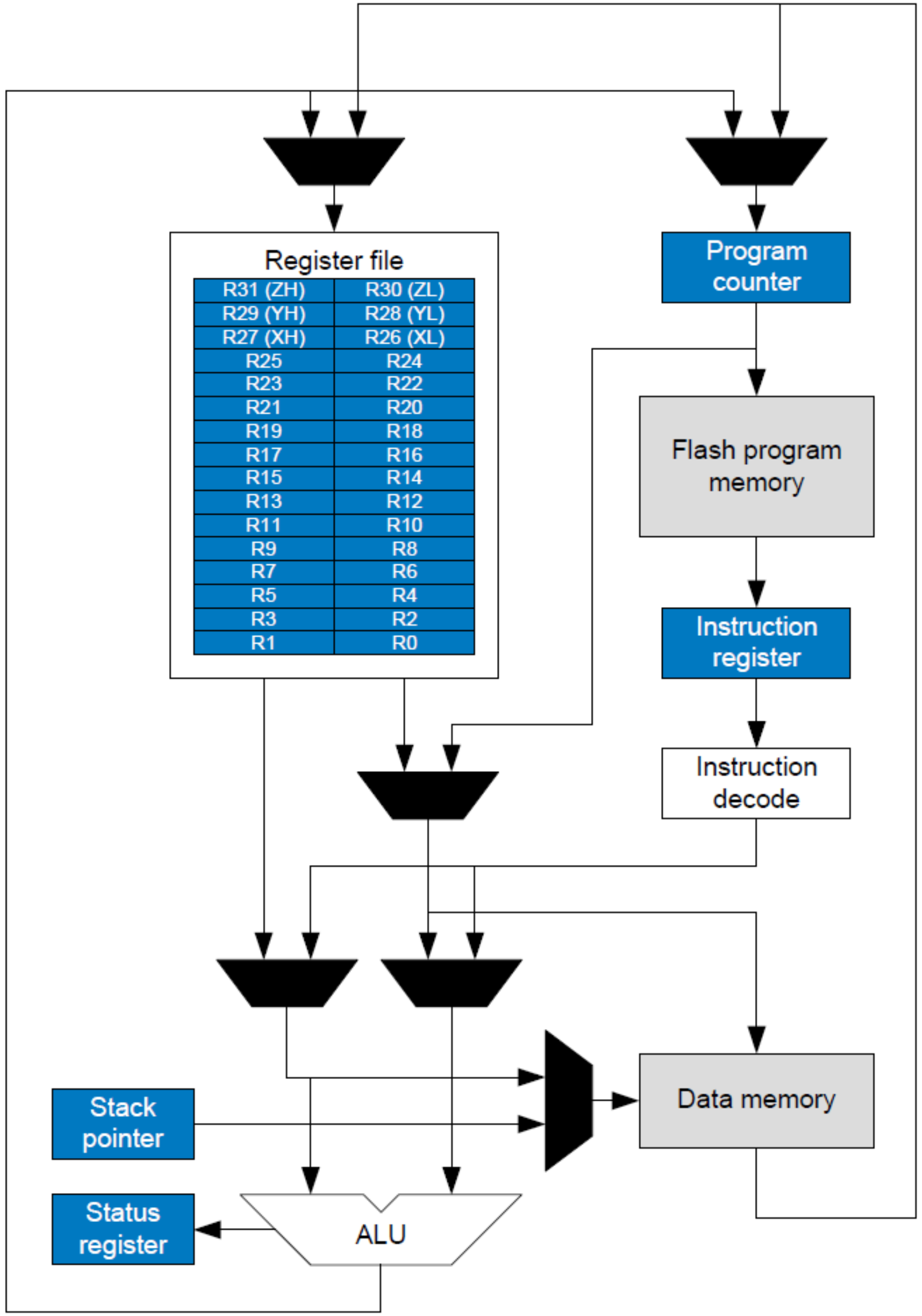


(PCINT14/$\overline{\text{RESET}}$) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 ($\overline{\text{SS}}$/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

🟧	Power
⬛	Ground
🟦	Programming/debug
🟦	Digital
🟩	Analog
⬛	Crystal/Osc

PCINT: pin change interrupt
RXD: receive (for serial com)
TXD: transmit (for serial com)
OCXX: timer output
XCK: External clock for USART
XTALX : External clock
TOSCX timer oscillator
AINX: analog comparator
CLKO: clock output buffer
ICP1: timer input capture
SCL and SDA (for I2C)
MOSI, MISO, $\overline{SS}$: for SPI


AVR
CPU
OCD
SRAM
debugWire
Clock generation
32.768kHz XOSC
8MHz Calib RC
External clock
16MHz LP XOSC
128kHz int osc
XTAL1 / TOSC1
XTAL2 / TOSC2
NVM programming
FLASH
Power management and clock control
EEPROM
EEPROMIF
DATABUS
I/O PORTS
PB[7:0]
PC[6:0]
PD[7:0]
GPIOR[2:0]
IN/OUT DATABUS
TC 0 (8-bit)
T0
OC0A
OC0B
PD4
PD6
PD5
SPI 0
MISO0
MOSI0
SCK0
SS0
PB4
PB3
PB5
PB2
AC
AIN0
AIN1
ADCMUX
PD6
PD7
ADC6, ADC7
PC[5:0]
VCC
RESET
GND
Power Supervision POR/BOD & RESET
Watchdog Timer
Internal Reference
ADC
ADC6,ADC7,PC[5:0]
AREF
ADC[7:0]
AREF
EXTINT
PD[7:0], PC[6:0], PB[7:0]
PD3, PD2
PCINT[23:0]
INT[1:0]
USART 0
RxD0
TxD0
XCK0
PD0
PD1
PD4
TC 1 (16-bit)
PB1, PB2
PD5
PB0
OC1A/B
T1
ICP1
TWI 0
SDA0
SCL0
PC4
PC5
TC 2 (8-bit async)
PB3
PD3
OC2A
OC2B



ATmega328 uses a Harvard architecture with separate memories and buses for program and data.

Instructions in the flash program memory are executed with a single level pipelining.

While one instruction is being executed, the next instruction is pre-fetched from the program memory.

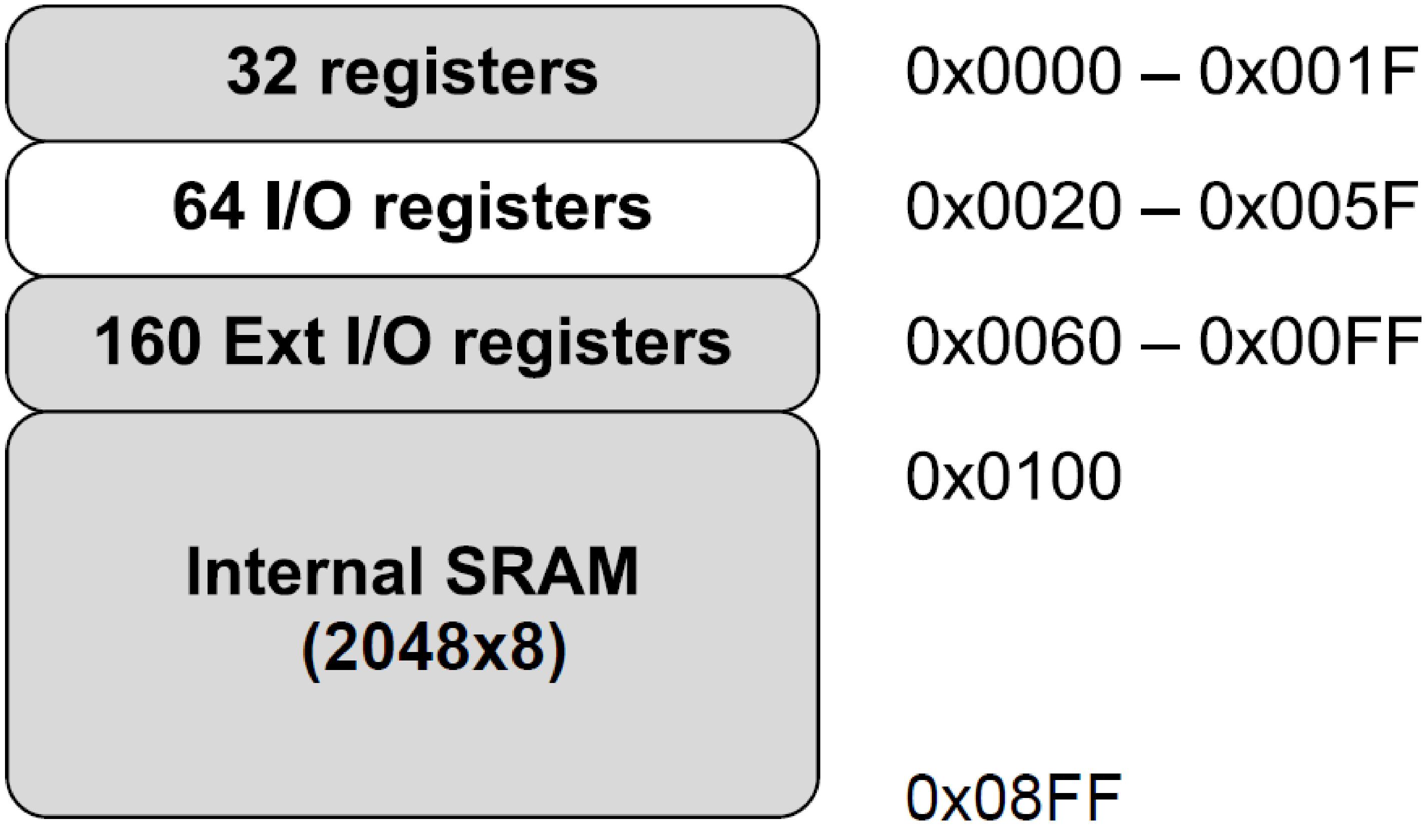
This concept enables instructions to be executed in every clock cycle.

Flash memory

- ATmega328p has 32kB of flash memory for storing the program.
- The Flash memory supposedly has an endurance of over 10,000 write/erase cycles.
- Upon power up, the power-on reset (PoR) triggers a reset signal and the CPU starts executing the program on the flash memory.

Main Memory

- Refer to *Resource 04 List of Registers.pdf*



32 registers 0x0000 – 0x001F

64 I/O registers 0x0020 – 0x005F

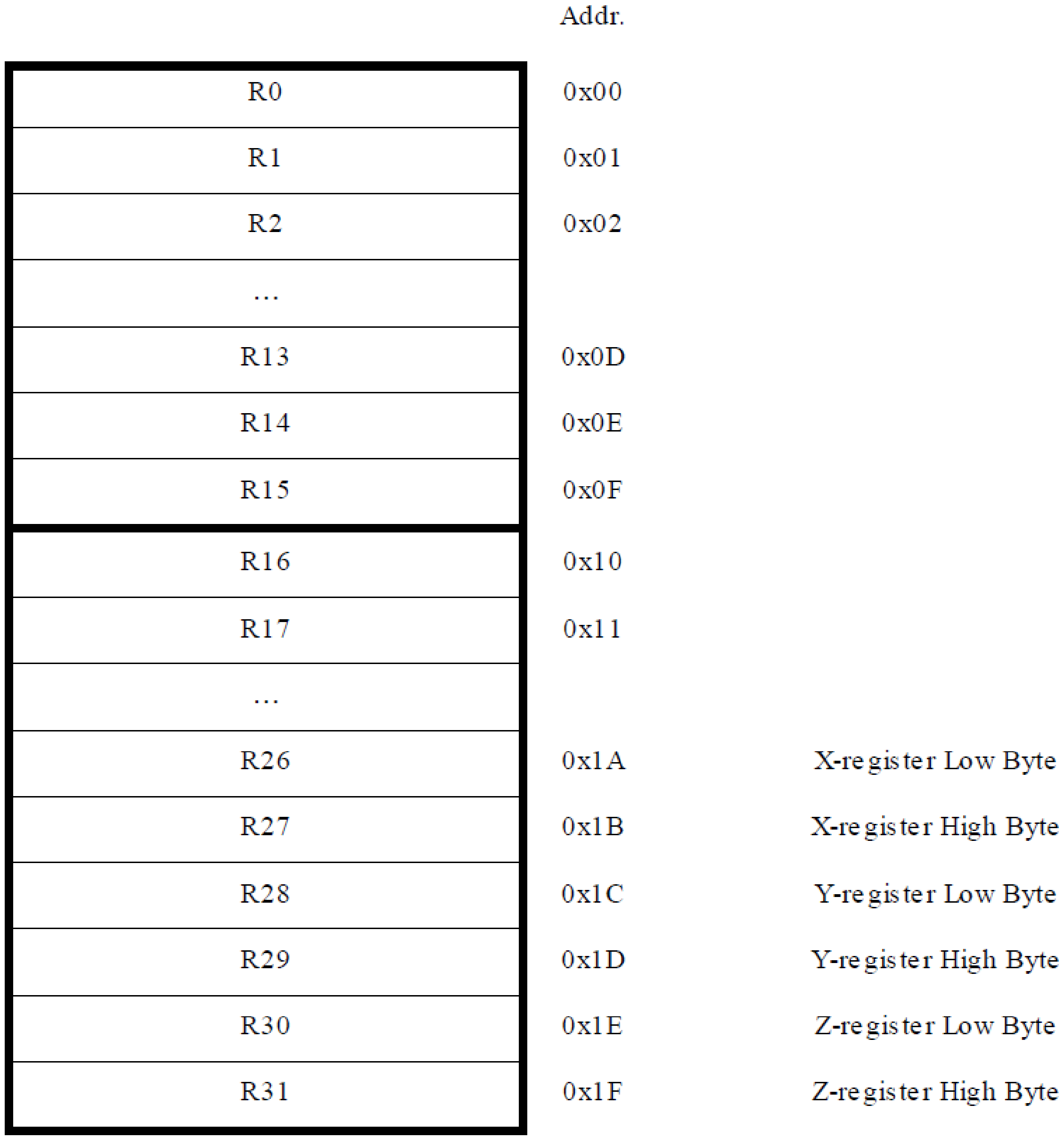
160 Ext I/O registers 0x0060 – 0x00FF

Internal SRAM (2048x8) 0x0100

0x08FF

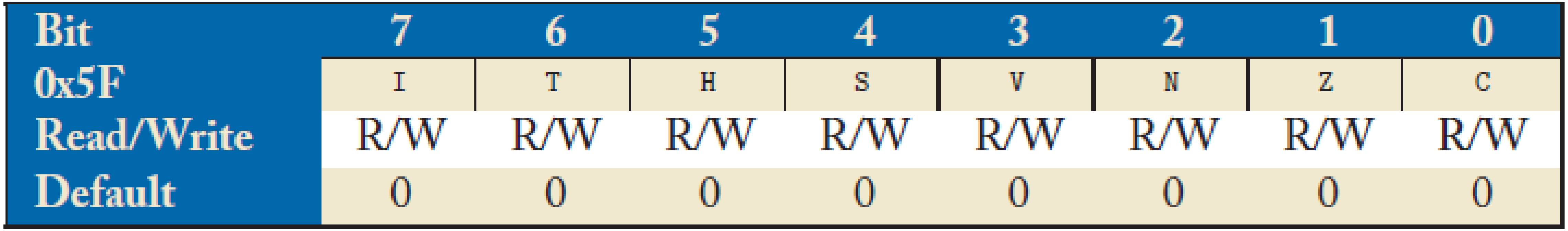
The first 32 registers from 0x0000 to 0x001F are called general-purpose registers

General purpose registers



	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
…		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
…		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte

Status Register (SREG)



Bit	7	6	5	4	3	2	1	0
0x5F	I	T	H	S	V	N	Z	C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

- I: Global interrupt enable
- T: Bit copy storage
- H: Half-carry
- S: Sign-bit
- V: Two’s complement overflow flag
- N: Negative flag
- Z: Zero flag
- C: Carry flag

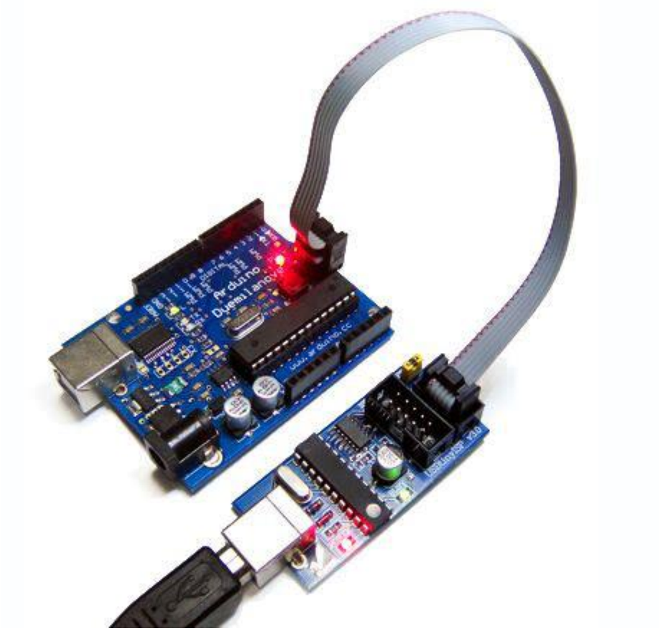
List of Instructions

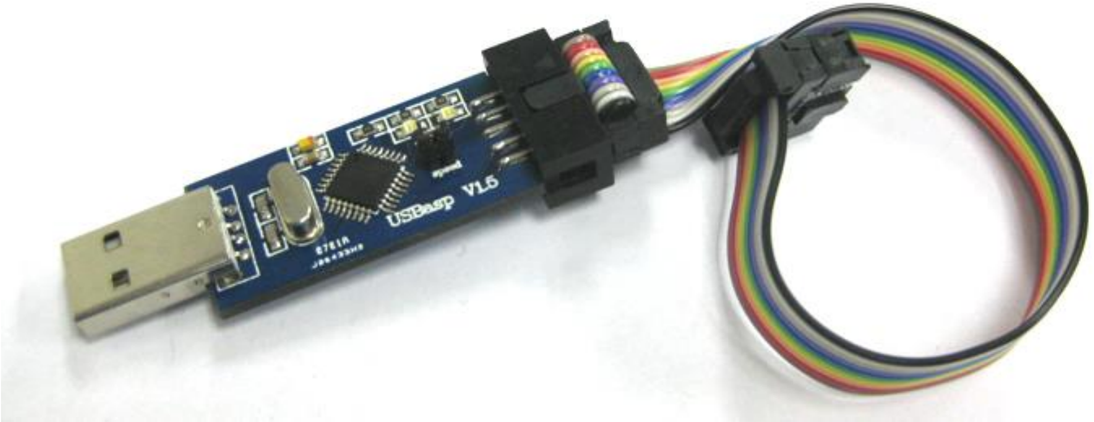
- Refer to *Resource 05 Instruction Set Summary.pdf*

ICSP

- Programs can be uploaded to the flash memory of ATmega328p using in-system programming (ISP), also called In-Circuit Serial Programming (ICSP).

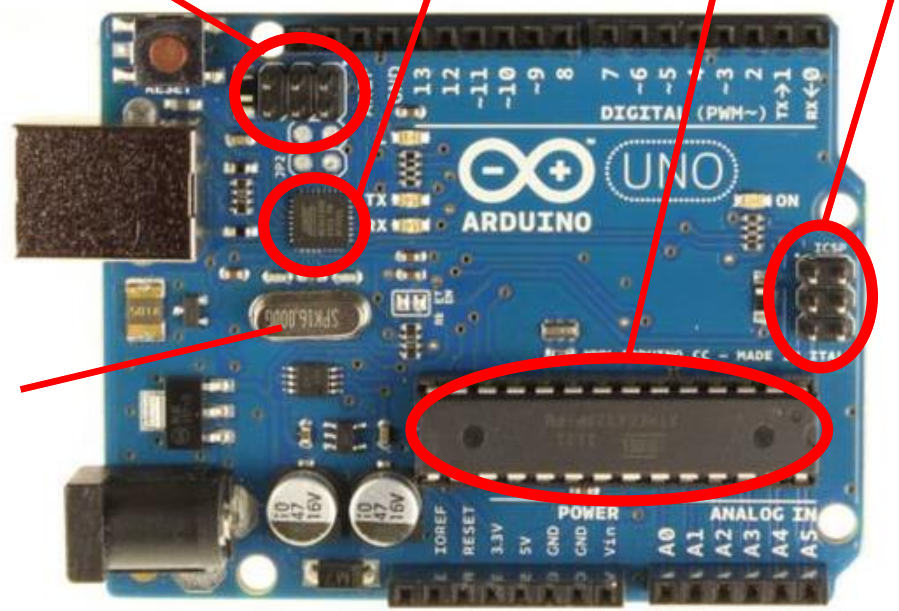
- It requires an extra circuitry/device.





Arduino Uno

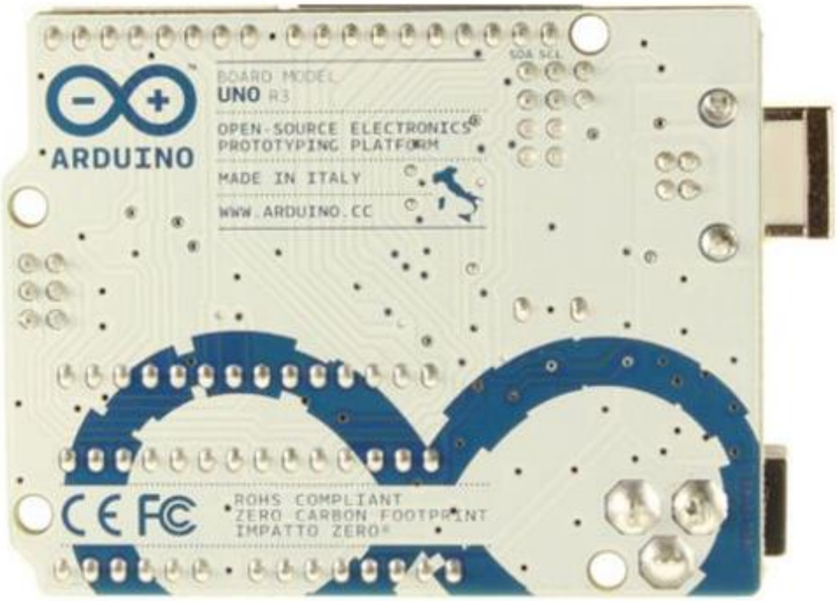
- Arduino Uno is an open-source development board that utilizes ATmega328p.
- It has onboard voltage regulators and 16 MHz crystal.
- It contains two microcontrollers:
  - ATmega16u2 (that handles USB-Serial conversion) and
  - ATmega368p (the main microcontroller)



ICSP header for ATmega16u2

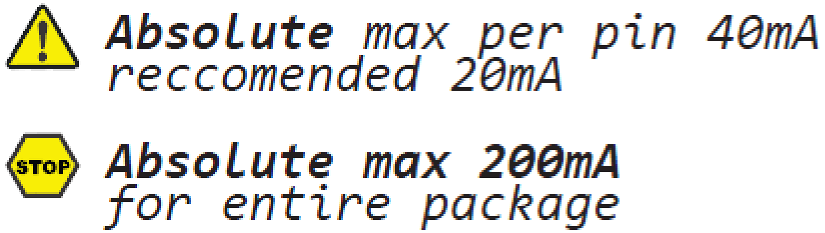
ICSP header for ATmega368p

Clock Crystal



Specifications of Arduino Uno

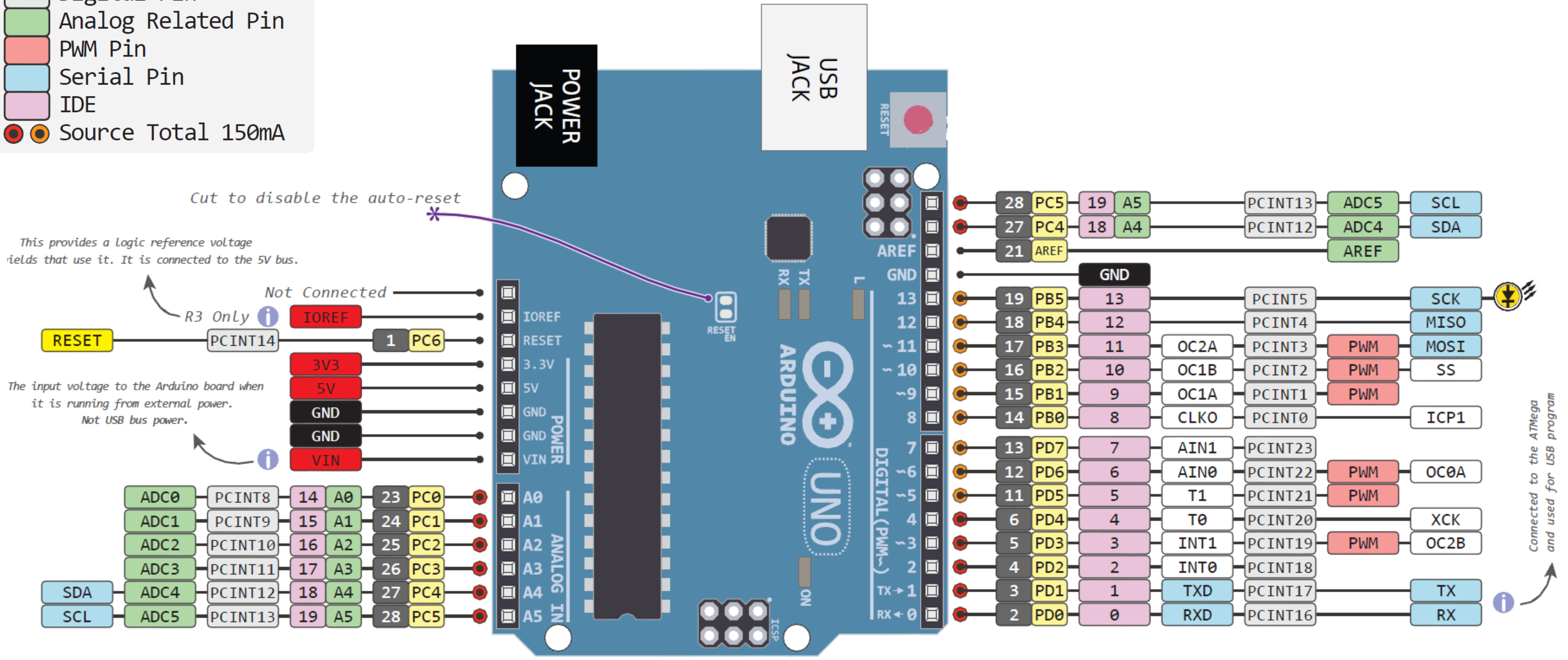
Microcontroller	ATmega328p
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
IO with built-in LED	1 (on pin #13)
Length	68.6 mm
Width	53.4 mm
Weight	25 g



⚠ ***Absolute*** *max per pin 40mA*
reccomended 20mA

STOP ***Absolute max 200mA***
for entire package

⬛	GND
🟥	Power
🟨	Control
▨	Physical Pin
🟨	Port Pin
⬜	Pin Function
⬜	Digital Pin
🟩	Analog Related Pin
🟥	PWM Pin
🟦	Serial Pin
🟪	IDE
● ●	Source Total 150mA



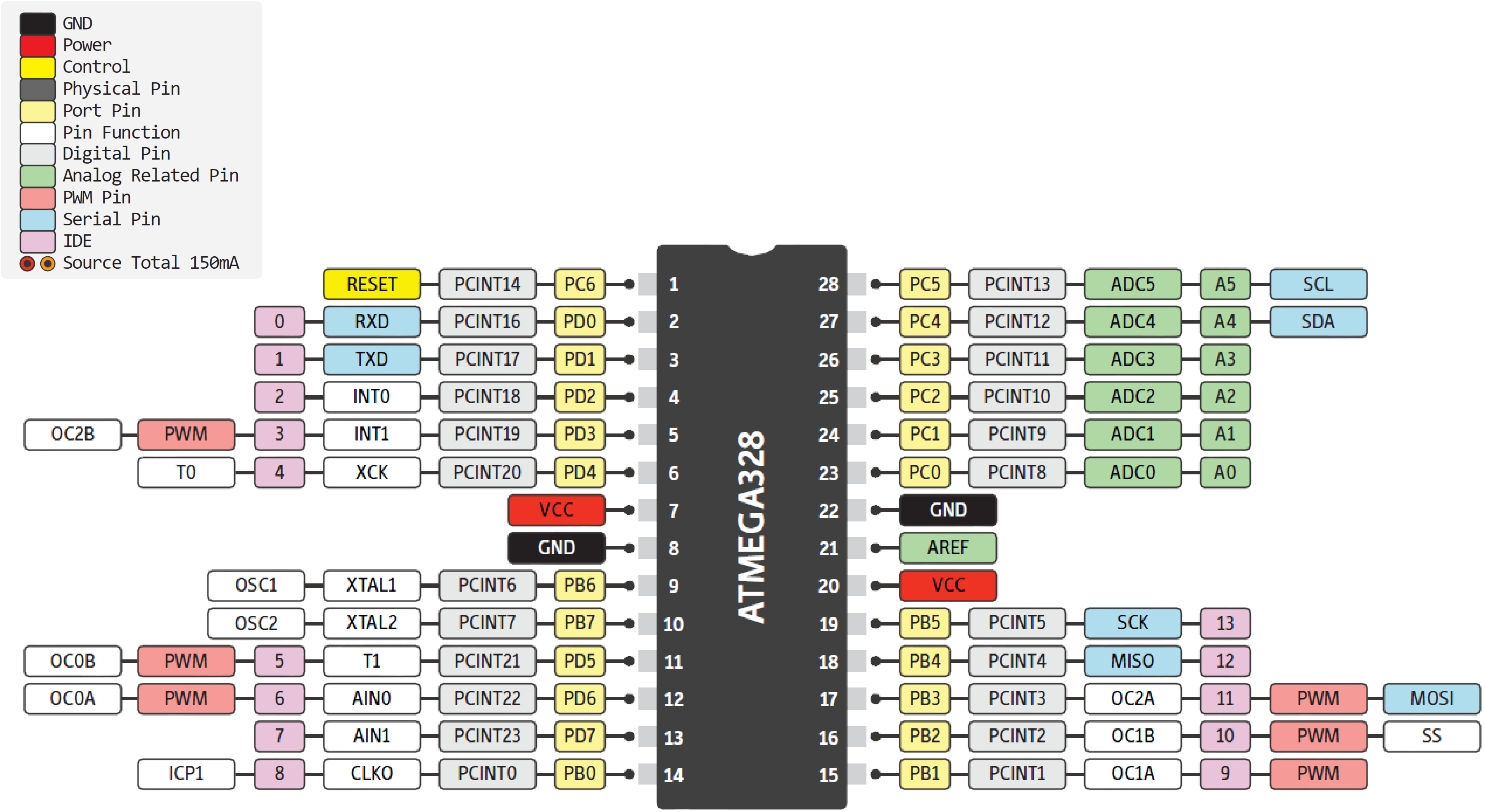
Cut to disable the auto-reset

This provides a logic reference voltage ields that use it. It is connected to the 5V bus.

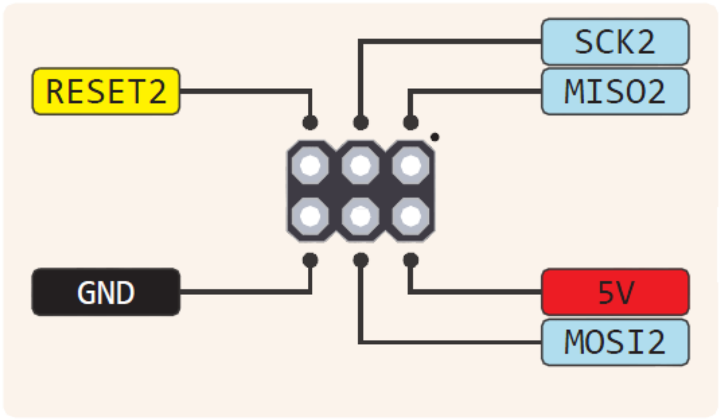
R3 Only

The input voltage to the Arduino board when it is running from external power. Not USB bus power.

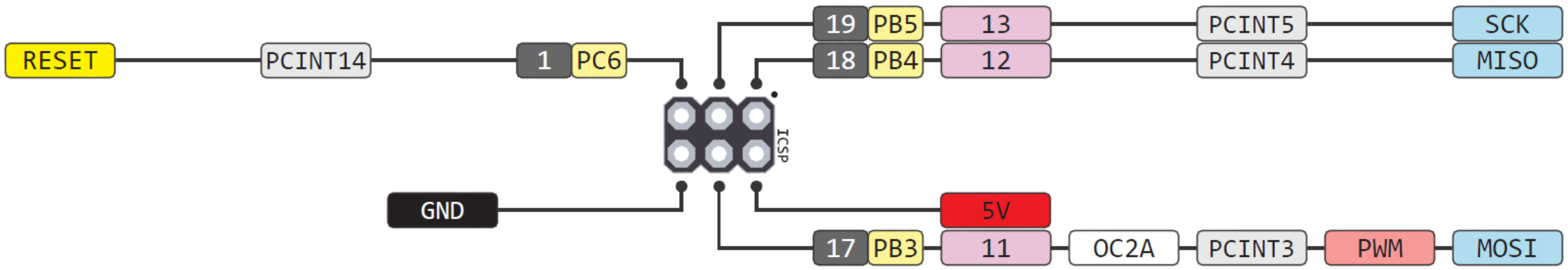
Connected to the ATMega and used for USB program


GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
ATMEGA328
RESET PCINT14 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
28 PC5 PCINT13 ADC5 A5 SCL
27 PC4 PCINT12 ADC4 A4 SDA
26 PC3 PCINT11 ADC3 A3
25 PC2 PCINT10 ADC2 A2
24 PC1 PCINT9 ADC1 A1
23 PC0 PCINT8 ADC0 A0
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 SCK 13
18 PB4 PCINT4 MISO 12
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM

ICSP header for ATmega16u2

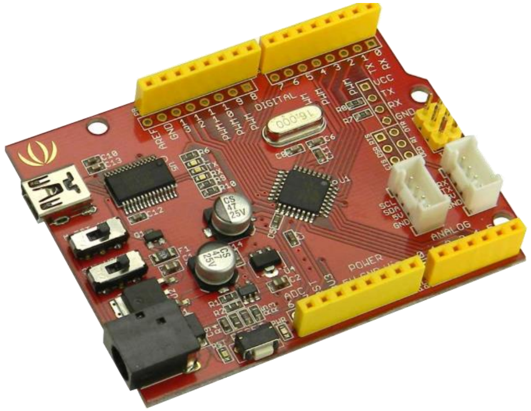


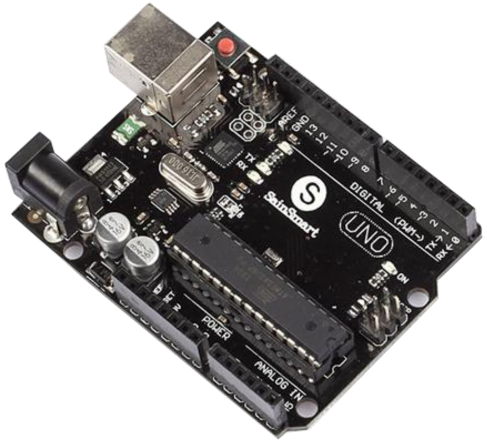
ICSP header for ATmega368p

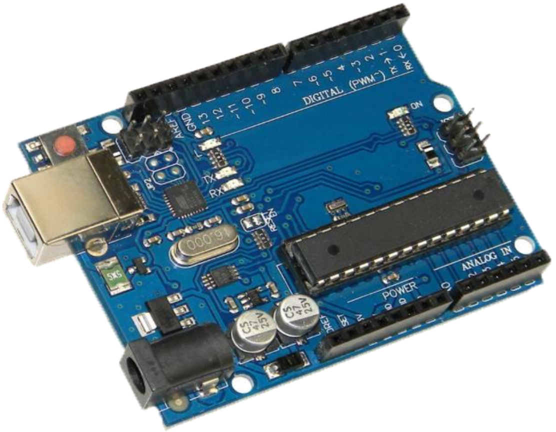


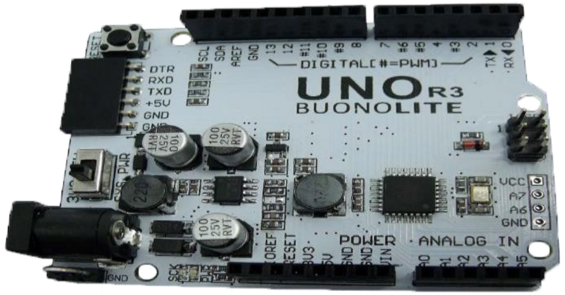
Arduino compatibles

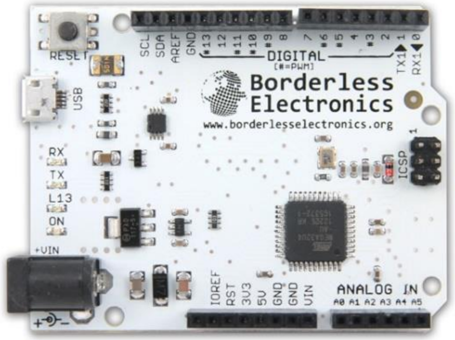
- Arduino is an open-source platform. Any one can freely modify it and produce a new board.
- There are many boards that are **compatible** with the Arduino IDE.



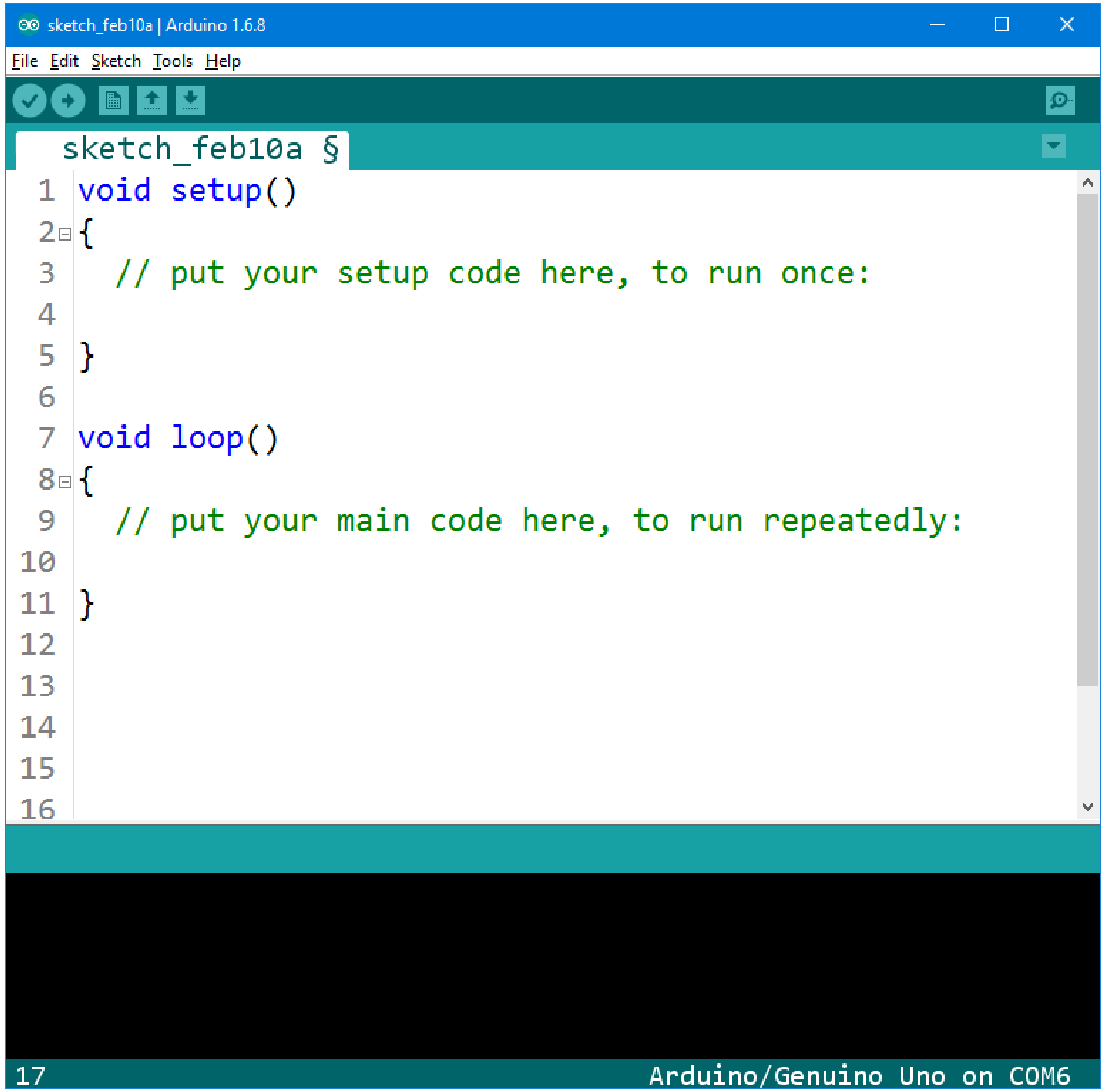








Arduino IDE



sketch_feb10a | Arduino 1.6.8
File Edit Sketch Tools Help
sketch_feb10a §

```
void setup()
{
  // put your setup code here, to run once:

}

void loop()
{
  // put your main code here, to run repeatedly:

}
```

17
Arduino/Genuino Uno on COM6

There are two mandatory functions

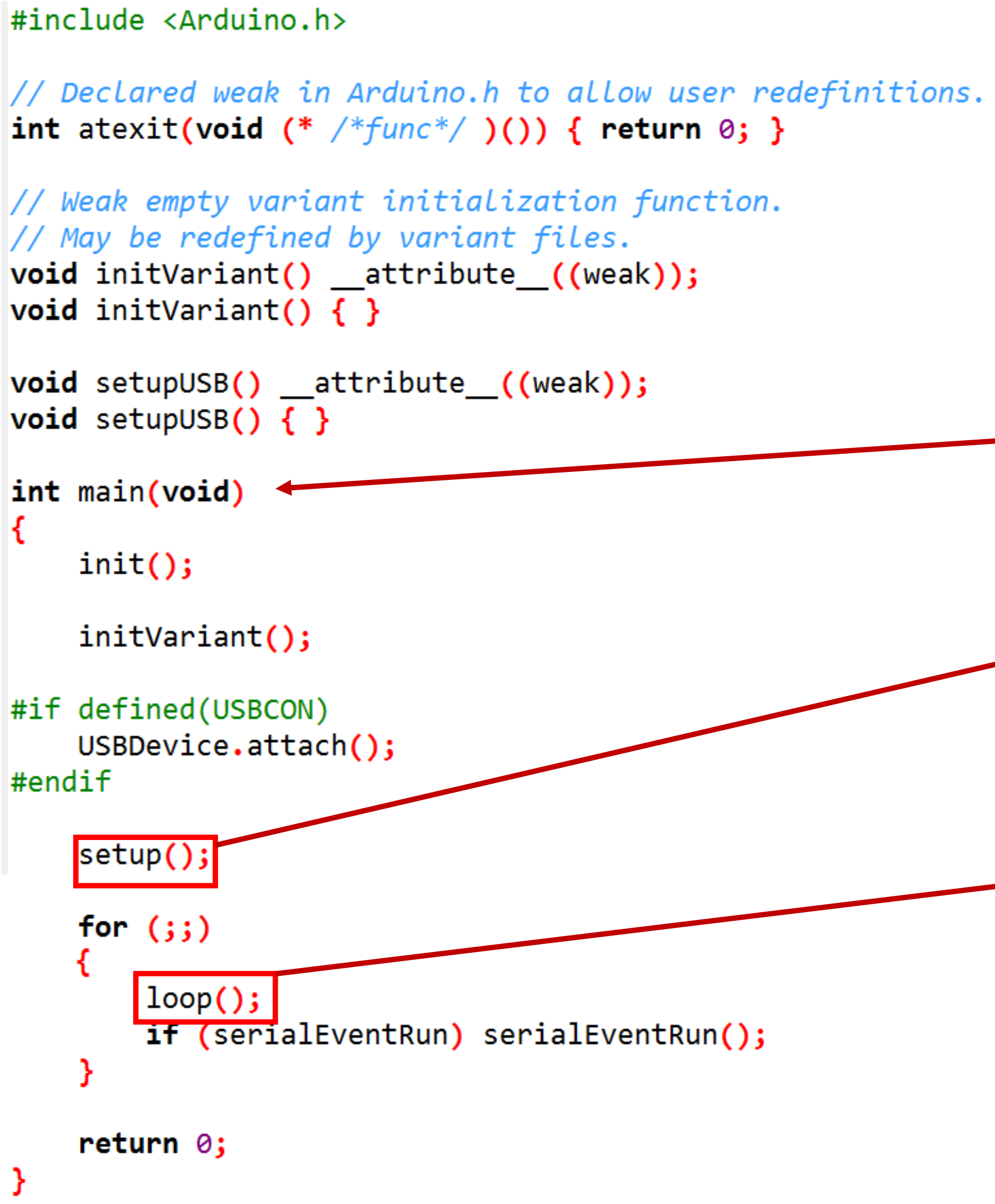
- setup() which gets called once the system boots or resets.
- loop() which gets called repeatedly after the setup function gets executed.

Wait! Where is the main function?

- The Arduino library is written in C++. It is called “Wiring”.
- The library hides the details.
- Library location: C:\Program Files (x86)\Arduino\hardware\arduino\avr\cores\Arduino
  (may differ based on where you installed Arduino)

Arduino under the hood

File name: main.cpp



```
#include <Arduino.h>

// Declared weak in Arduino.h to allow user redefinitions.
int atexit(void (* /*func*/ )()) { return 0; }

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() { }

void setupUSB() __attribute__((weak));
void setupUSB() { }

int main(void)
{
    init();

    initVariant();

#if defined(USBCON)
    USBDevice.attach();
#endif

    setup();

    for (;;)
    {
        loop();
        if (serialEventRun) serialEventRun();
    }

    return 0;
}
```

The main function is here

The set up function gets called here

The loop function gets repeatedly called here

Bootloader

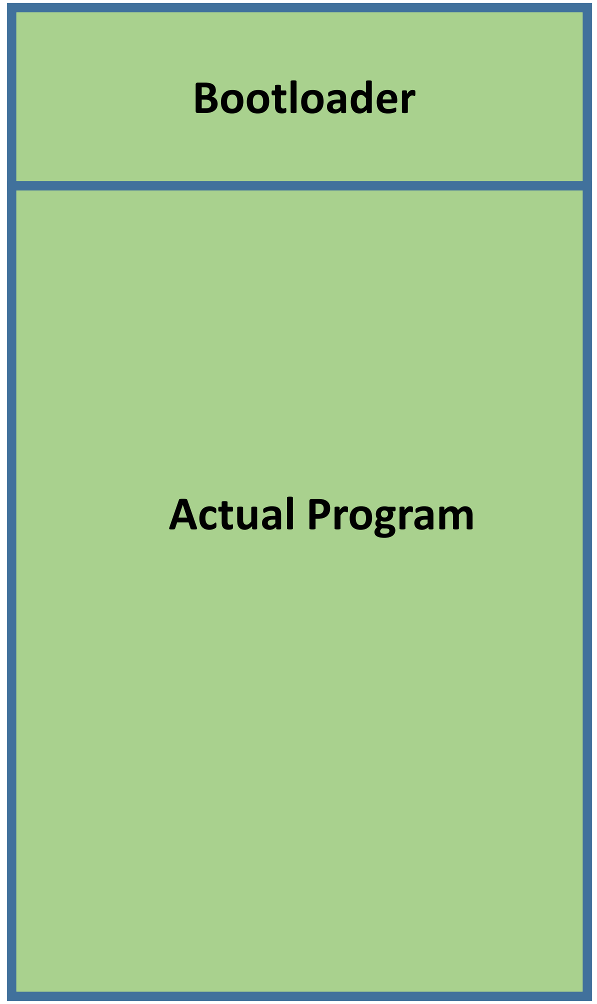
ATmega328p on Arduino is pre-loaded with a special program called "bootloader".

When power is first applied or the reset button is pressed, the CPU jumps into it and executes its instructions.

- If there is no serial programming signal from the USB, the bootloader makes the CPU immediately jump to the actual program.
- If there is serial programming signal from the USB, the bootloader captures the program sent from the host computer and them dumps the program on the flash memory. After that, it makes the CPU jump to the actual program.

This allows programs to be uploaded to the flash memory without ICSP or any special hardware.

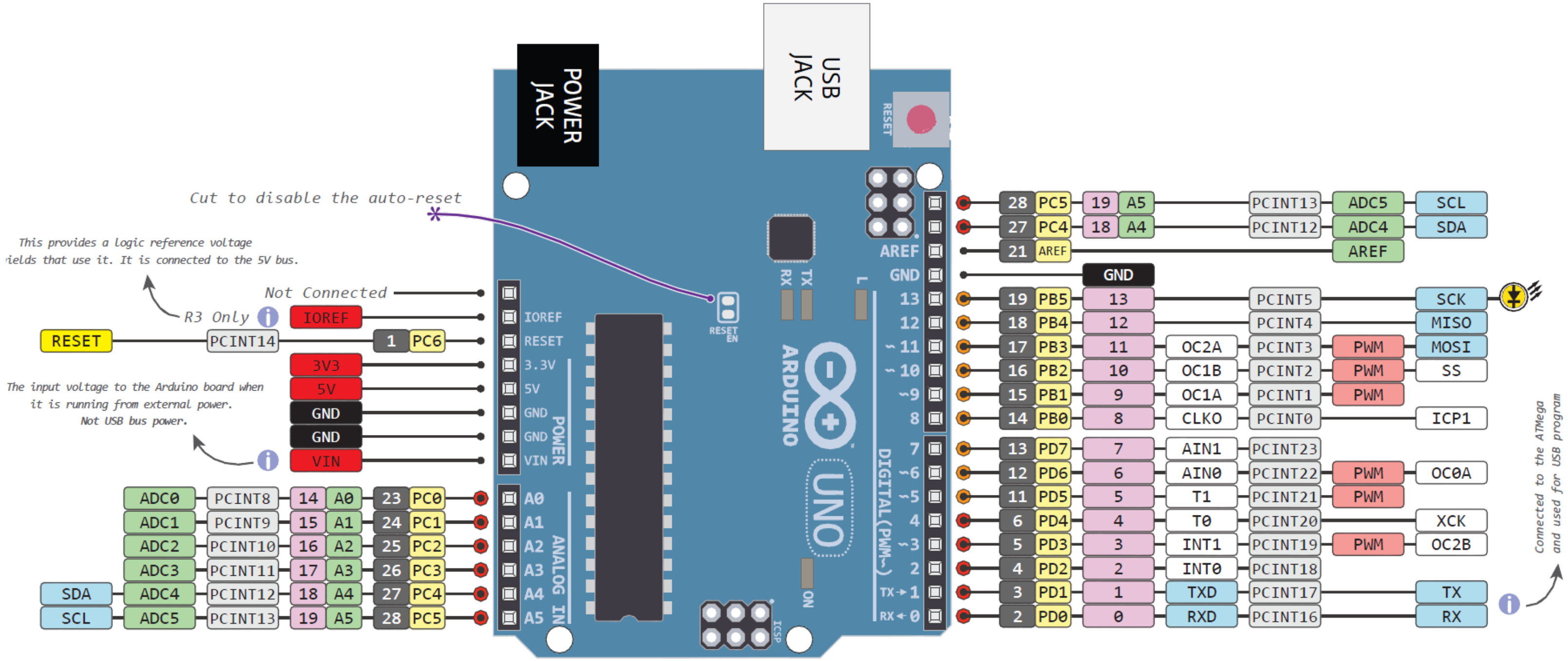
Flash Memory of ATmega328p



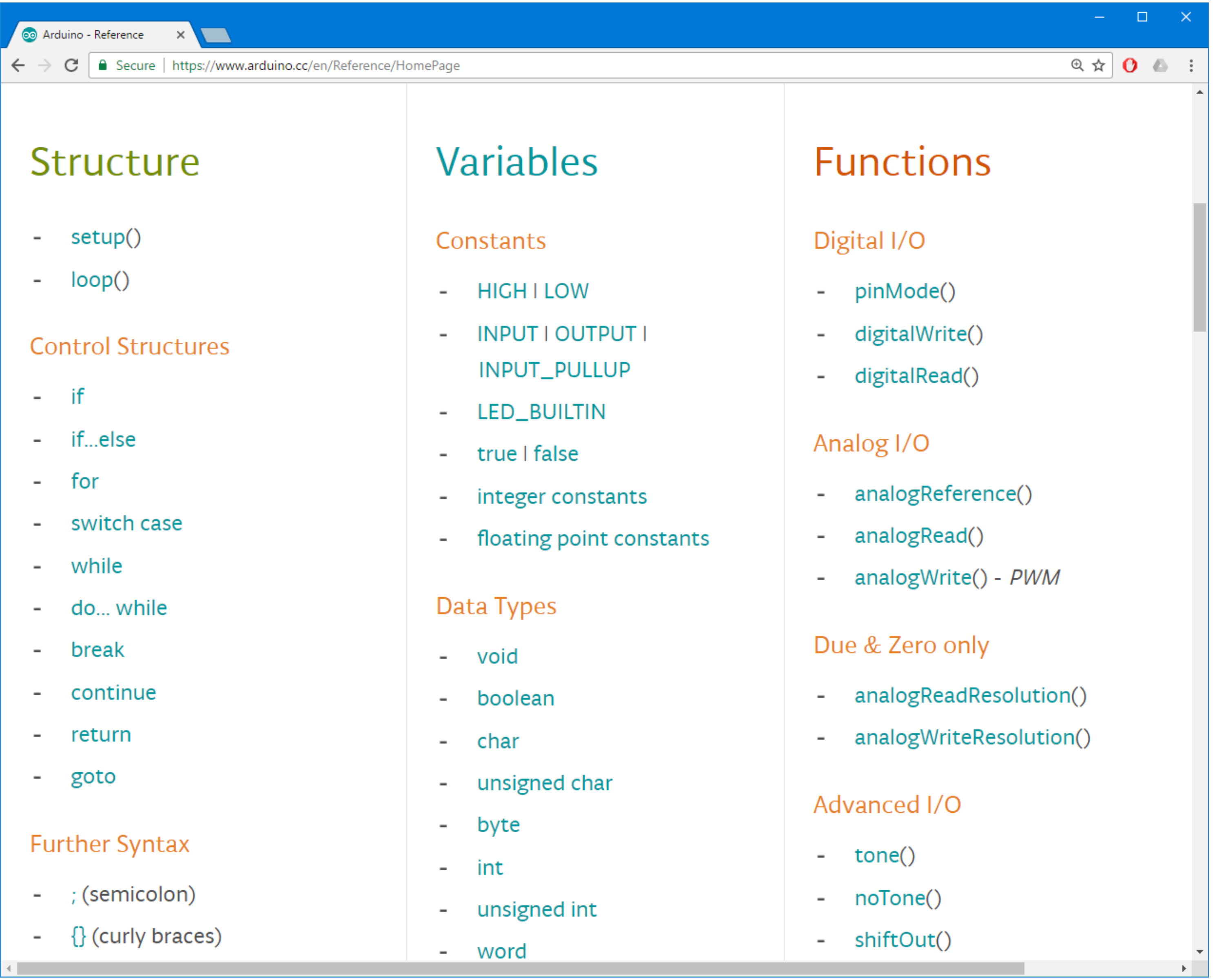
Disclaimer: the above diagram is only for visualization and by no means it is an accurate representation

GPIO

- Genera Purpose Input/Output (GPIO) lines can be used as either input or output.
- Arduino library provides PWM output on certain pins .
- ATmega328p has 23 GPIO pins.
- Arduino only provides 20 GPIO pins (PC6, PB6 and PB7 are unavailable)
- PC6 is connected to the reset/fuse. PB6 and PB7 are used by the external clock crystal.


POWER JACK
USB JACK
RESET
Cut to disable the auto-reset
This provides a logic reference voltage
ields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
IOREF
RESET PCINT14 1 PC6
3V3
5V
GND
GND
VIN
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
ADC0 PCINT8 14 A0 23 PC0
ADC1 PCINT9 15 A1 24 PC1
ADC2 PCINT10 16 A2 25 PC2
ADC3 PCINT11 17 A3 26 PC3
SDA ADC4 PCINT12 18 A4 27 PC4
SCL ADC5 PCINT13 19 A5 28 PC5
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RESET EN
ARDUINO
UNO
ON
ICSP
RX
TX
L
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
28 PC5 19 A5 PCINT13 ADC5 SCL
27 PC4 18 A4 PCINT12 ADC4 SDA
21 AREF AREF
GND
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLKO PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
Connected to the ATMega and used for USB program

Arduino Library Reference



Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due & Zero only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

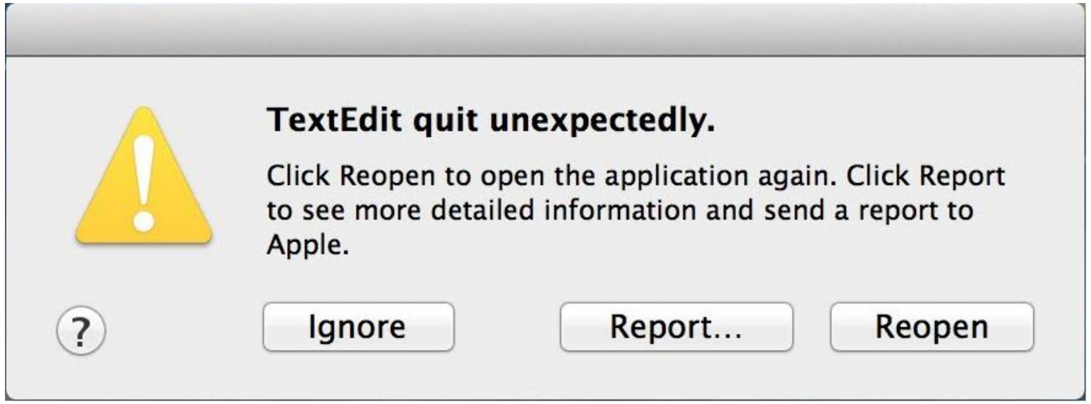
- tone()
- noTone()
- shiftOut()

https://www.arduino.cc/en/Reference/HomePage

Errors

- There are two kinds of errors:
  - Compile-time errors
  - Run-time errors
- Compile time errors occur at compile time (e.g. syntax errors)
- Run time errors occur at run time (they are notorious)

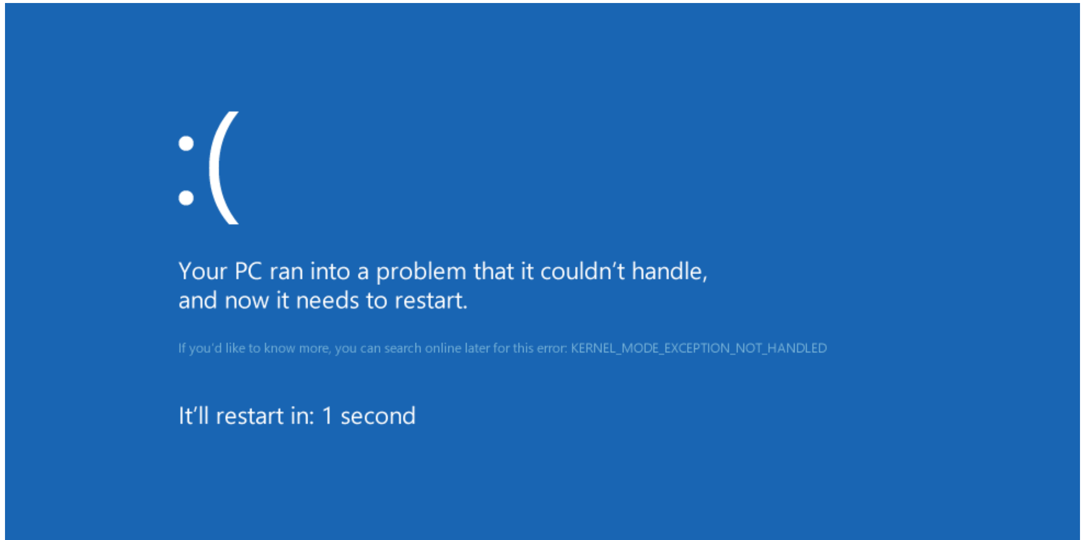
COMPUTER SALES
...AND IN CASE OF PROGRAM CRASHES, THIS MODEL COMES FULLY EQUIPPED WITH AN AIR BAG!
Baloo
Search ID: 122558932


TextEdit quit unexpectedly.

Click Reopen to open the application again. Click Report to see more detailed information and send a report to Apple.

Run-time errors

- Examples of run time errors are division by zero, stack overflow, and running out of memory.
- The program must handle these errors (also called exceptions)
- If the program does not handle, the OS will abruptly end it.
- Operating systems can sometimes experience unexpected errors. But the operating systems are usually programmed in such a way to gracefully end.

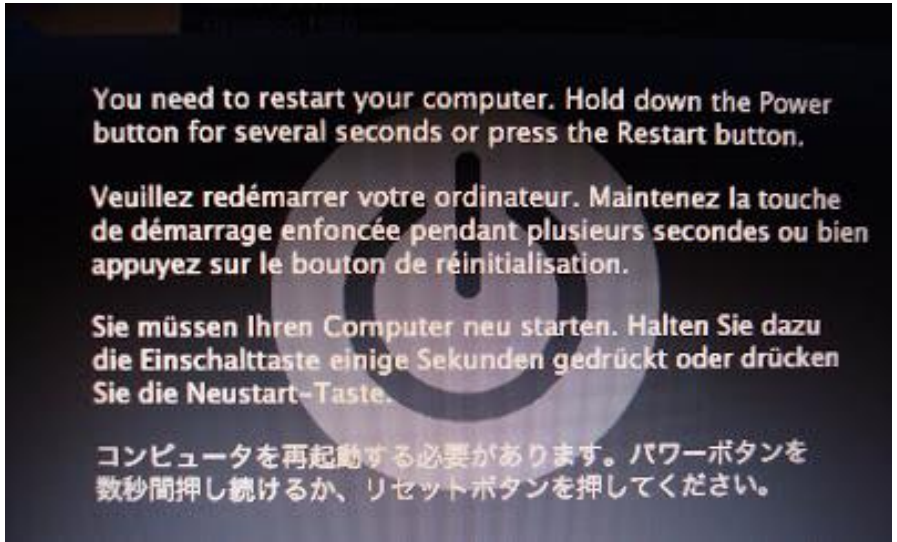


:(

Your PC ran into a problem that it couldn’t handle, and now it needs to restart.

If you’d like to know more, you can search online later for this error: KERNEL_MODE_EXCEPTION_NOT_HANDLED

It’ll restart in: 1 second



You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

Run-time errors

- The effect of run-time errors are usually not detectable on microcontrollers.
- The program must handle all these exceptions. (Avoid division by zero, avoid running out of memory, etc)
- If such errors occur, the microcontroller will usually exhibit unexpected and strange behaviors. The intended output may not be obtained.

Debugging

- Debugging is the process of fixing bugs so that desired output is obtained.

- It is an important aspect of software engineering.

- Some exceptions are difficult to locate and fix.

- Debuggers are usually an Integral part of IDE. For example, in Visual Studio.

- Arduino IDE does not have a debugger.

Ways of debugging

- halt program execution when a specific line of code is reached (i.e.,breakpoints);
- once halted, look at the current contents of memory (i.e.,memory watches);
- once halted, look at the current contents of CPU registers;
- receive notification of critical exceptions with explanations of the offending instruction;
- Execute one instruction (both high-level and assembly) and then halting again (i.e.,stepping);
- resume normal execution

The above techniques are also used if the program does not produce a desired output.

Debugging on Microcontrollers

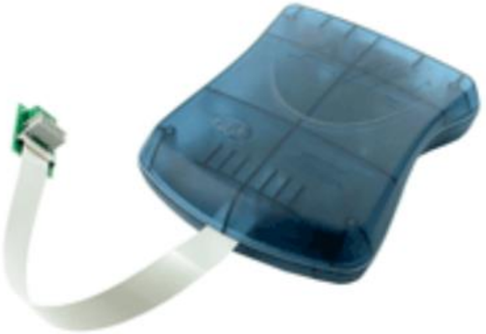
- It is not possible to have debugging functionality on microcontrollers without additional hardware debugger (or advanced OS-like software on a microcontroller) that does things like pausing the execution of the CPU.

- Hardware debuggers are often used in development of embedded systems.

- Hardware debuggers provides the ability to allow to halt the program counter and peek inside registers, memory, etc.

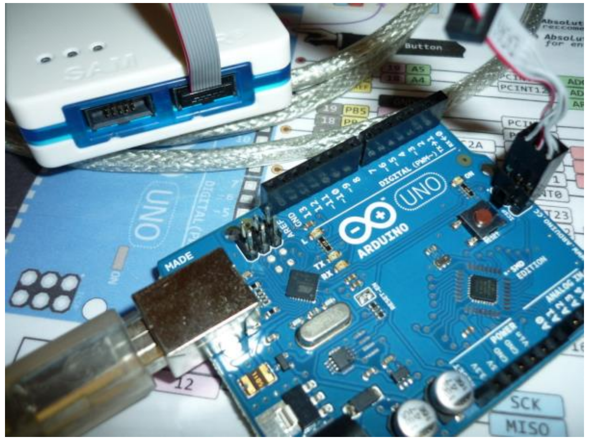
Hardware Debuggers

Atmel ICE

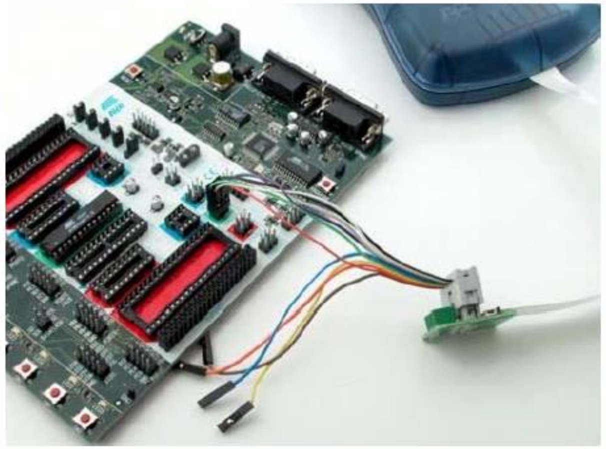


AVR JTAGICE mkII

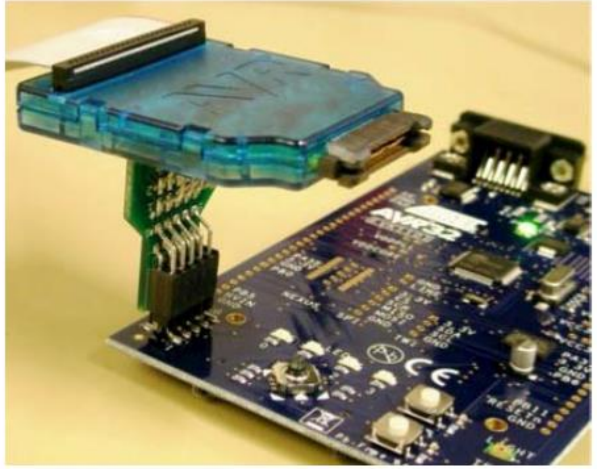
AVR ONE!



Connected to Target



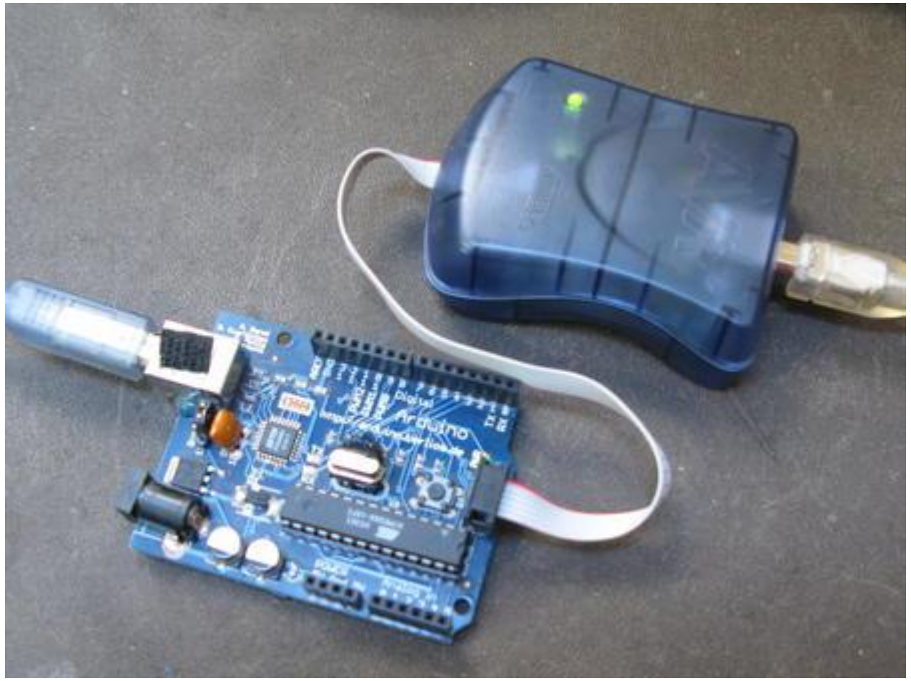
Connected to Target



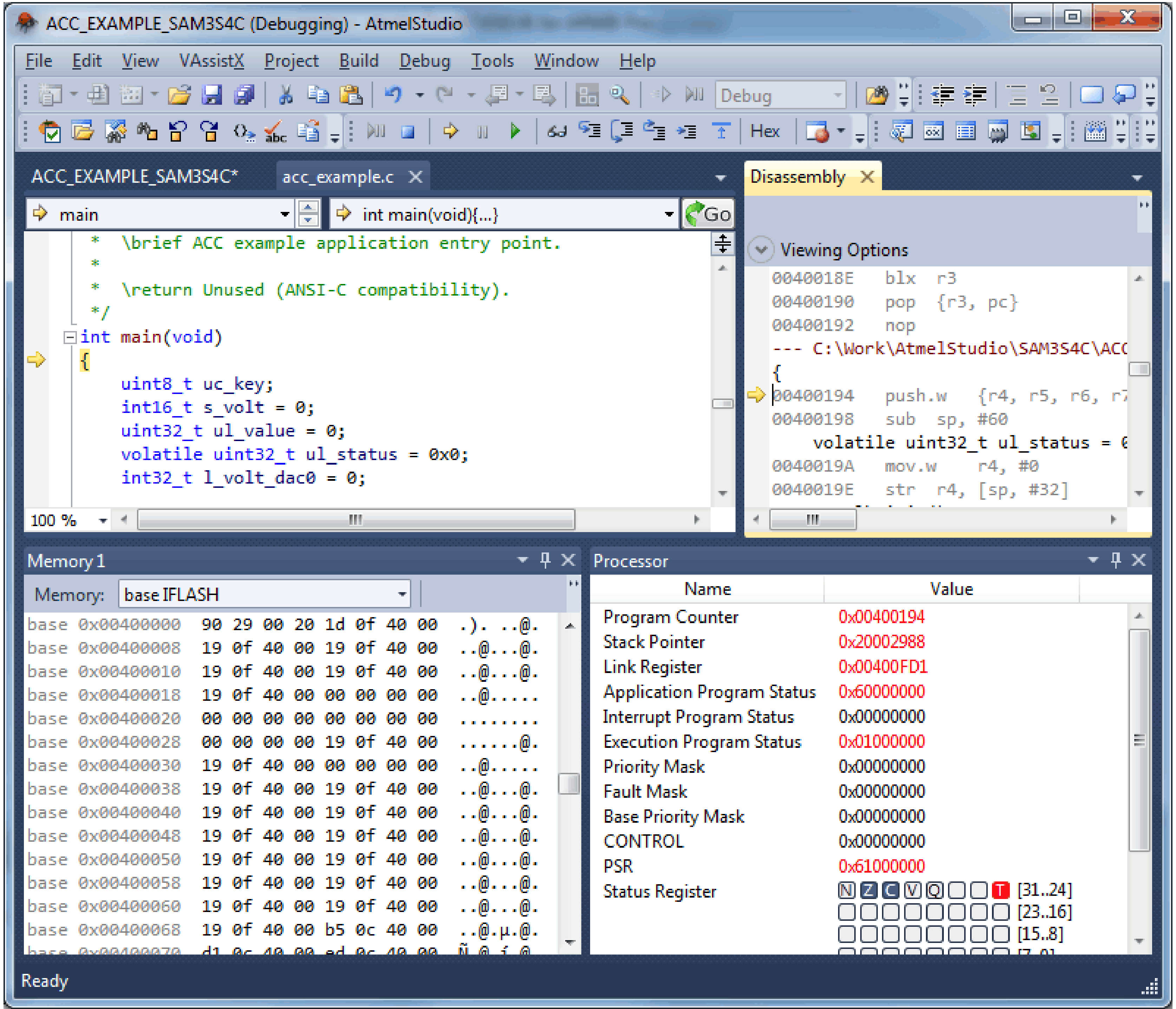
Connected to Target

Hardware Debugger

- The Arduino IDE does not support hardware debuggers.
- Atmel Studio supports hardware debuggers.
- Hardware debuggers use ICSP pins which allow programs to be uploaded without a boot loader.



Debugging on Atmel Studio



```
 *  \brief ACC example application entry point.
 *
 *  \return Unused (ANSI-C compatibility).
 */
int main(void)
{
    uint8_t uc_key;
    int16_t s_volt = 0;
    uint32_t ul_value = 0;
    volatile uint32_t ul_status = 0x0;
    int32_t l_volt_dac0 = 0;
```

```
0040018E    blx   r3
00400190    pop   {r3, pc}
00400192    nop
--- C:\Work\AtmelStudio\SAM3S4C\ACC
{
00400194    push.w    {r4, r5, r6, r7
00400198    sub   sp, #60
     volatile uint32_t ul_status = 0
0040019A    mov.w     r4, #0
0040019E    str   r4, [sp, #32]
```

```
base 0x00400000  90 29 00 20 1d 0f 40 00  .). ..@.
base 0x00400008  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400010  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400018  19 0f 40 00 00 00 00 00  ..@.....
base 0x00400020  00 00 00 00 00 00 00 00  ........
base 0x00400028  00 00 00 00 19 0f 40 00  ......@.
base 0x00400030  19 0f 40 00 00 00 00 00  ..@.....
base 0x00400038  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400040  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400048  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400050  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400058  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400060  19 0f 40 00 19 0f 40 00  ..@...@.
base 0x00400068  19 0f 40 00 b5 0c 40 00  ..@.µ.@.
base 0x00400070  d1 0c 40 00 ed 0c 40 00  Ñ.@.í.@.
```

Name	Value
Program Counter	0x00400194
Stack Pointer	0x20002988
Link Register	0x00400FD1
Application Program Status	0x60000000
Interrupt Program Status	0x00000000
Execution Program Status	0x01000000
Priority Mask	0x00000000
Fault Mask	0x00000000
Base Priority Mask	0x00000000
CONTROL	0x00000000
PSR	0x61000000
Status Register	N Z C V Q □ □ □ T [31..24] □ □ □ □ □ □ □ □ [23..16] □ □ □ □ □ □ □ □ [15..8] □ □ □ □ □ □ □ □ [7..0]

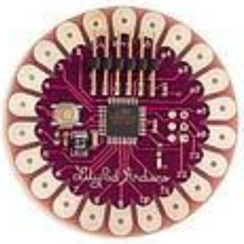
Variants of Arduino

- Arduino has many variants
- Other common models of microcontrollers from Atmel are ATmega168, ATmega328 and ATmega1280.

Arduino Uno

Arduino Leonardo

Arduino Mega 2560



Arduino LilyPad

Arduino Mega ADK

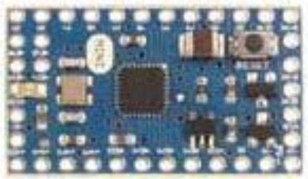
Arduino Fio

Arduino Ethernet

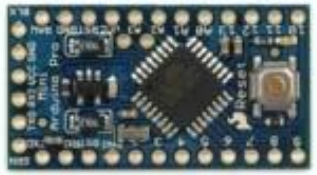
Arduino Pro

Arduino BT

Arduino Nano



Arduino Mini



Arduino Pro Mini

Arduino mega

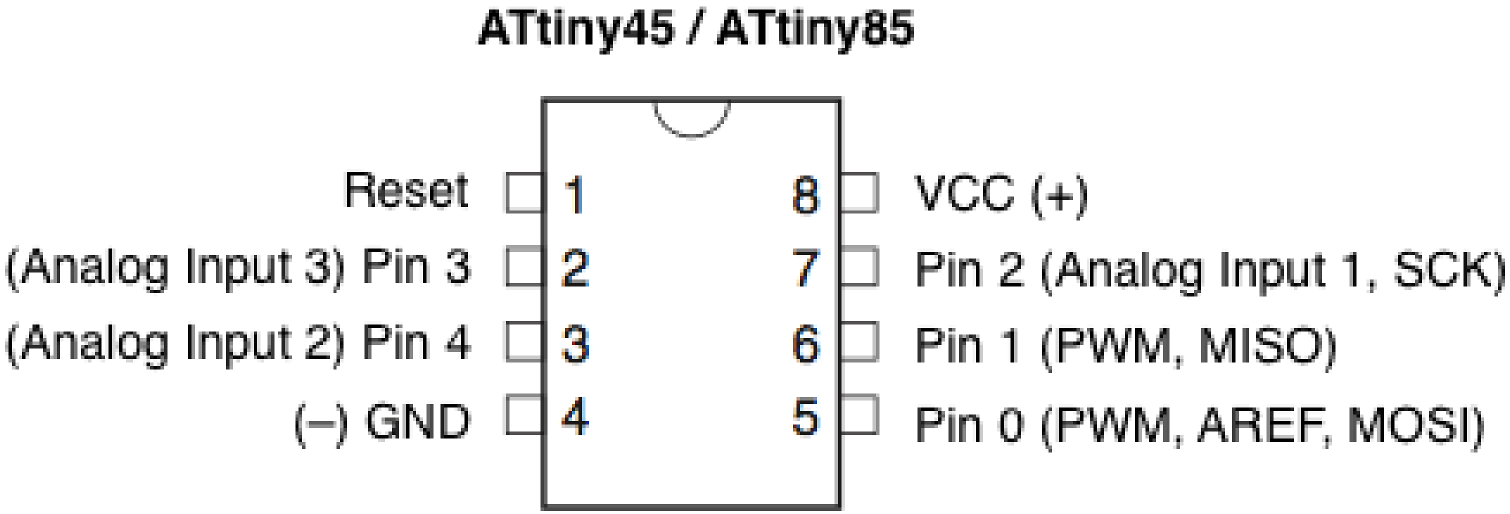
Microcontroller	ATmega1280
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

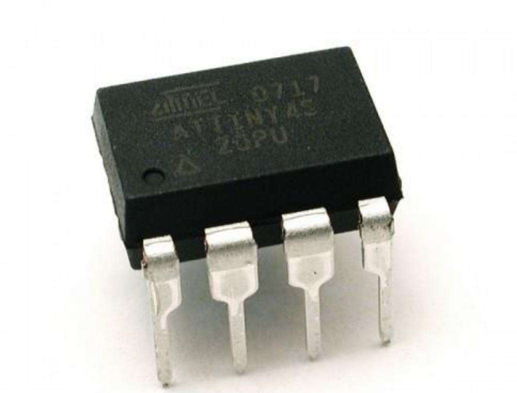
Limitation of Arduino IDE

- Not transparent enough
- Slow
- It is just a text editor with added ability to compile and upload the program to the microcontroller.
- No autocomplete feature (as of v.16.8)
- No debugging capability
- No auto-syntax checking capability

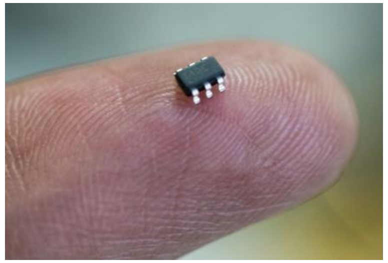
Other models of Atmel microcontrollers

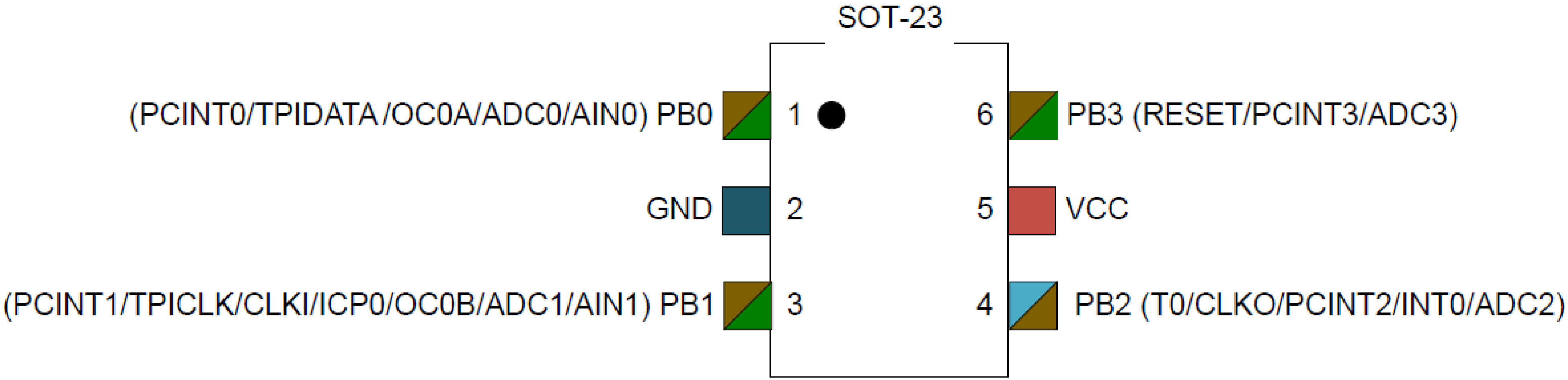
The ATtiny family is very popular for small projects.





ATtiny4 only has 6-pins





SOT-23

(PCINT0/TPIDATA/OC0A/ADC0/AIN0) PB0 1 ● 6 PB3 (RESET/PCINT3/ADC3)

GND 2 5 VCC

(PCINT1/TPICLK/CLKI/ICP0/OC0B/ADC1/AIN1) PB1 3 4 PB2 (T0/CLKO/PCINT2/INT0/ADC2)