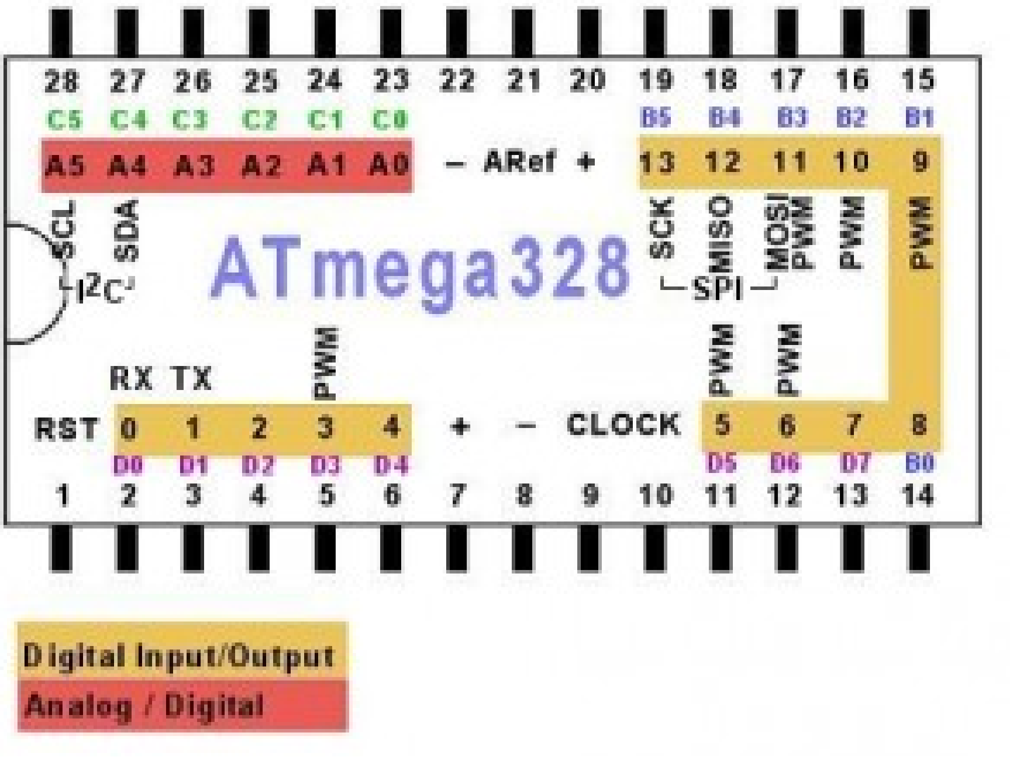
AVR Micro controller on TINAH board

Prasad Mehendale



Foreword

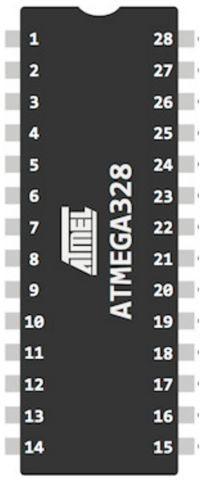


Figure 1: AVR micro controller

AVR is a world famous micro-controller. It is powerful and used in many applications. This book is about a development kit that allows user to write applications quickly around AVR architecture. Atmega328 is used as the brain of the kit.

TINAH (Tinah Is Not Arduino Hardware) is the Arduino Uno compatible board. Arduino is a simple yet powerful language to program an avr micro controller. The TINAH kit fulfills the bare-bone requirements of a micro controller application. You can program the chip using any avr-uploader. TINAH uses open-source USBASP uploader. Simple connectors are given on the board to connect the cpu to the peripherals which an application needs.

The most powerful feature of TINAH is its compatibility with the Arduino IDE. Using the freely available Arduino Software, one can develop an avr application quickly.

Why not to use original Arduino board or its clone then ? Because TINAH is low cost and offers all the features Arduino offers.

The TINAH board design is released under *Creative Commons License.* Anyone can use it for no charge. It is available at http://lms.technoeschool.in The Arduino IDE (Integrated Development Environment) is released under the freee software license (GNU GPL). Know more about free software at http://gnu.org.

Using the kit, I expect the kit user to develop a prototype easily and quickly at low cost.

This book is written keeping in mind those students who are not studying electrical or electronics engineering. They will find the book easy to understand and still practically implement the hardware and programs.

The book is written using LaTeX in *GNU-Emacs* and is released under the Free Documentation License [FDL].

Happy micro-controlling !

–**Prasad Mehendale**

Chapter 1

Introduction to micro controllers

Micro controllers are used almost everywhere. They control humidity, temperature in polyhouses. They help us to control TV and air conditioners remotely at home and office. In industry, automobiles and ships they control very complicated machinery. So study of micro controller has become unavoidable to the students of all engineering branches.

Digital technology results in high precision and adds great reliability to an application. Well-thought *algorithm* [1] devides the whole process in smallest possible steps. Each step is carried out with great accuracy. So micro controller makes a system automatic, saving energy and time.

1.1 What are micro controllers?

Micro controllers are semiconductor integrated circuit chips. (See figure 1.1 on page 6). Thousands of transistors, diodes, resistors and capacitors are integrated to form circuits. When these circuits are powered up, they work. The real beauty of the circuits, however, is that you can change the circuit behaviour by writing *programs.*[2] To store the programs, controllers have program memory.

Suppose you want to implement a simple fixed-time-timer. The timer is

[1]Algorithm means the theme according to which the system works.

[2]A program is a set of executable instructions arranged in an order to implement an algorithm.

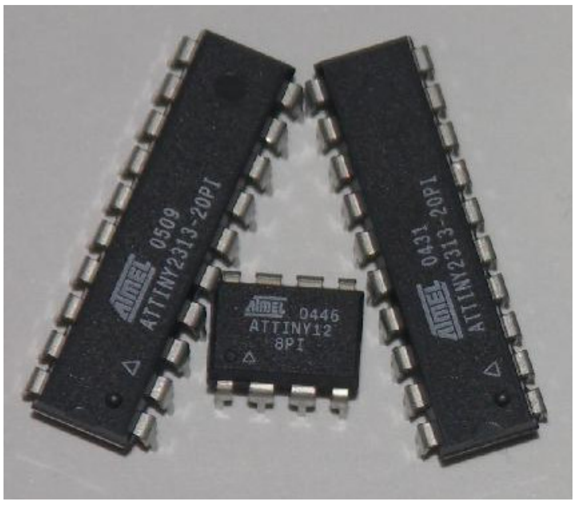


Figure 1.1: AVR micro controllers

started using a key. It will light up an LED for a fixed time (say 50 seconds) and then make the LED off. you will go through the following steps.

1. Connect a key to one of the controller pins configured as input.
2. Connect an LED to one of the controller pins configured as output.
3. Think of an appropriate algorithm:
    (a) Wait till the key is pressed.
    (b) Once the key is pressed, light up an LED for 50 seconds.
    (c) Make the LED off after 50 seconds are over.
    (d) Go back to step 3a above.
4. Now write the code based on the algorithm above.
5. Compile it till it is error free.
6. Dump it into the program memory.
7. Power up the chip. Test the timer.

Now if you want to use the same micro controller to work as an LED flasher, just change the algorithm and the program. Go through the last three steps above. **This will convert the same controller into a flasher.**

This flexibility of the controller device makes it very useful for the developers. They can practically reprogram the memory hundreds of times till the application works as they wish.

1.2 Applications of micro controllers

Micro controller applications are endless. You can use micro controllers in any and every field. Here is a *small* list of micro controller applications:

1. **Automobiles:** Network of micro controllers is used to optimize the performance of the automobile.
2. **Agriculture:** From timely watering, sprinkling on farms, to environment conditioning in polyhuses.
3. **Domestic:** TV and air-conditioner remote control, washing machine automation, home security systems, home automation using xigbee protocols, parking light control.
4. **Marine engg:** Rudder control, safety alarms, fault annunciation systems, engine performance optimization.
5. **Public services:**Automatic street light control, Motor pump control, Traffic signals, water meters.

You can extend this list further.

1.3 Philosophy behind micro controllers

Micro controller is comparatively a recent development in the field of electronics. Before micro controllers or micro processors, digital ICs were used. They implemented combinational and sequential circuits using logic gates. After the advent of micro controllers, the whole scenario changed !

Let us now know the philosophy (basic principles) behind micro controllers.

1.3.1 Continuous and discrete

To measure quantity of wheat, we can count number of grains. It means, we express wheat quantity in terms of a number. How can we express quantity of milk in terms of a number ? Practically it is not possible. There is no natural unit of milk, becuase you can't standardize a drop of milk. So we will say, wheat is discrete and milk is continuous. In reality, if wheat-grain-size is standardized then the quantity will be the correct number.

We can roughly say that wheat is discrete and milk is continuous. *In short, anything which can be expressed in terms of a number, is discrete and the one where it is not possible is continuous.* Digital technology [3] can handle only numbers, i.e. discrete entities. In electronics, using transducers, we can convert anything [to be measured] to an electrical signal. Mostly these signals are coninuous and are called analog signals.

Now, *"bad" news for us is, the real world is mostly analog !* Well, it is not *that* bad because there are electronics circuits which convert the analog voltages into numbers. These circuits are known as *analog to digital converters.*

Micro controllers are number crunchers. Micro controllers can do various operations on numbers. All arithmetic and logical operations can be done. Shifting and rotation of numbers is also possible. All these operations can be done with great speed. This is how digital technology can be used to make the micro controller work.

1.3.2 Eat in small chunks

God: How will you eat an elephant ?
Ant: *In small chunks.*

Well, whether you would like to eat an elephant or not, any huge task can be tackled, dealing with it in small parts (chunks).

Suppose we want to design a fire-alarm annunciation system on a ship. This is a big task and we will deal with it in chunks:

1. Spread the fire sensors on various locations.

2. Connect the sensors to the micro controller at the center using cables.

3. Embed a program in the controller memory to scan all the sensors one after the other continuously.

[3] Digital technology uses only two voltage states +5V and 0V , which represent binary numbers one and zero.

4. Repeat sensor scanning till there is fire.

5. When fire catches, on the annunciator screen, show all the sensor locations catching fire.

6. Flash on the screen, that location which caught fire first. This may help to find the cause of fire quickly.

7. After fire is totally extinguished, annunciator should be reset *manually*.

Any process which we want to control, must be divided in small parts. The actions must be described sequnetially. Micro controller inherently (by default) understands some *instructions* [4]. These instructions must be used to implement the steps 4 to 7 above.

1.3.3 One at a time

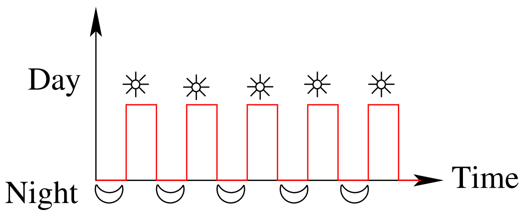


Figure 1.2: Sun as a clock

A clock is something with which you synchronize your activity.

Millions of years ago, our forefathers *had* a clock. They worked all the day and slept at night. Actually, sun worked like a clock to them. When the man became more organized, tasks went smaller, the clock needed lesser period i.e. higher frequency.

Today we, in our daily work use a clock whose smallest measurable period is one second. If you look at the figure 1.2 on page 9, you will observe that it is an oscillator with a period "one-day".

A micro controller uses an electronic oscillator as a clock. The oscillator *should* have these features:

[4]These are known as assembly language instructions. They are further converted in binary (machine language) format for execution.

1. It should produce high frequency (in Mega Hertz).
2. The frequency must be very stable.

A crystal oscillator satisfies both criteria above. The micro controller Atmega328 (which we study here) uses 16 MHz crystal oscillator clock as its *time reference.*

It simply means, for every pulse produced by an oscillator there exists a miniature task which the micro controller completes. We as programmers, must define these miniature tasks and arrange them in a particular order.

1.3.4 Aristotle said it !

Aristotle said, **“The whole is greater than the sum of its parts.”**

A micro controller has following features:

- It can store binary numbers electronically (+5V=1, 0V=0).
- It can operate (+,-,/,*,and, or,xor, rotate, compare etc) on binary numbers.
- It can synchronize “operations” to crystal oscillator clock pulses.

All the features above, when come together (the “whole” as Aristotle says), their overall effect is greater than the sum of its parts ! The device now can control *any* process. Keeping this in mind, we will now see in the next section, what a micro controller can do !

1.4 What can our micro controller do ?

You will understand internal working of micro controllers in the next chapter. In this section you will get a general idea how controllers actually control devices connected to them.

1.4.1 ON & OFF

You can make a bulb ON and OFF. So can do the micro controller. **But this is the only thing it can do to the device connected to its pins.** The only difference is, it can ON and OFF any device with great speed.

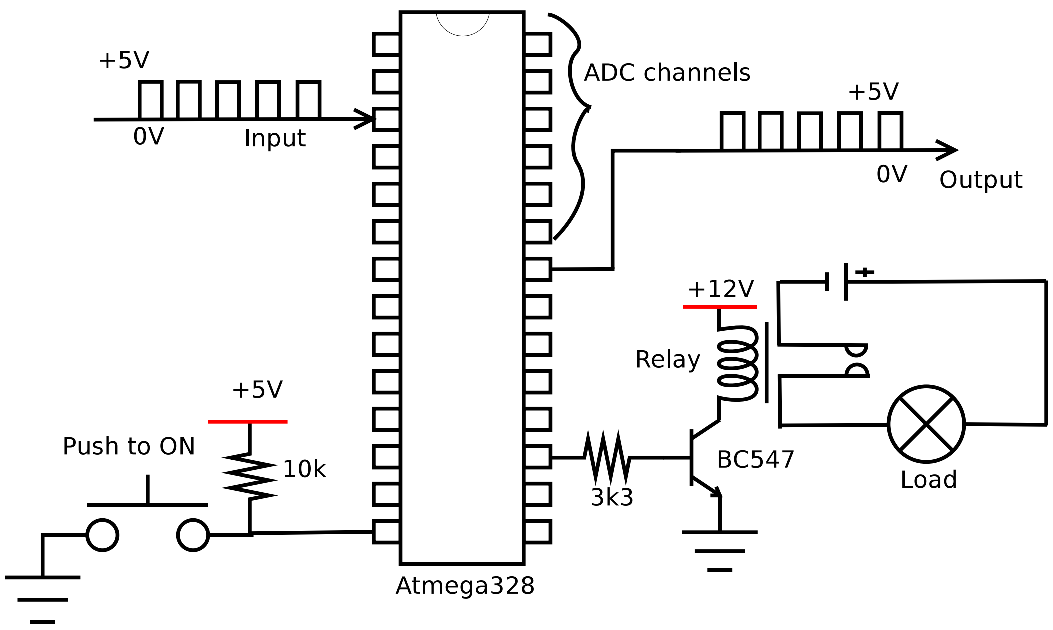


Figure 1.3: Devices connected to micro controller

Can you make a bulb ON & OFF 10 times in one second ? The controller can do this 10000 times (or more) in one second !
See the figure 1.3 on page 11. In this figure following devices are shown connected to the micro controller Atmega328:

- A push to on switch.
- An electromagnetic-relay[5] driven by a transistor.
- A device generating a square wave connected to controller as input.
- A device driven by a square wave connected to the controller as output.

Now remember, that the controller can understand only ON & OFF. We can explain the behaviour of the micro controller for the above devices like this:

1. When you push the button of the **switch**, *that* controller pin is connected to ground [6]. When the button is not pushed, +5V are connected to the pin through a resistor.
   Thus, when button is pushed, controller receives binary number ZERO

[5]Current through the coil produces magnetism. This attracts the contact points together. When no current, no magnetic field and a spring pulls the contact points away.
[6]In electronics, ground means zero potential point.

and when not pushed, controller receives binary number ONE. This is how the controller reads input signals which come from the outer world.

2. To drive a relay, the micro controller makes the pin HIGH from inside. Transistor will be on. Coil will be energized and this will activate the contacts to power up the load.

3. When a device from outside, sends ON-OFF signal very quickly, micro controller *can* sense it and processes it further according to the program inside. (e.g. it will measure number of pulses in one second and declare signal frequency.)

4. The controller can produce a square wave of very high frequency as shown in the figure 1.3 on 11. This can be used as a signal to some other device.

1.4.2 Analog to digital conversion

Our micro controller can convert exernally given anolog (continuous) signal into numbers (digital-binary). It has a 6 channel ADC to sense upto 6 different analog signals. Once the analog signal is converted to a binary number, controller is able to process it accurately and with great speed.

1.5 What does electronics do in micro controllers ?

Electronic circuits work fast. 0 Volts represent binary ZERO. +5 Volts represent binary ONE. Using flip-flops, [7] a circuit can store numbers in binary format. Logic gates, sequential and combinational circuits are used to manage the overall working. Crystal oscillators provide the time reference to this electronic system.

1.6 Binary basics

To understand the AVR architechture better, we brush up some concepts from binary arithmetics.

[7]Flip-flops are bi-stable multi-vibrators implemented using transistors. They can be stable in both states: ONE and ZERO.

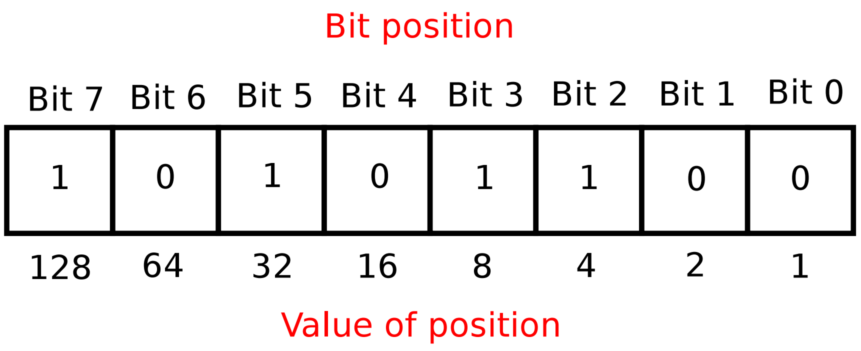


Figure 1.4: 8 bit register

- **A bit** can take only two values, ONE or ZERO.
- **A byte** is an array of 8 bits. Greatest 8 bit number is: 255 Decimal.
- **A word** is an array of 16 bits. Greatest 16 bit number is: 65535 Decimal.

Now, see the figure 1.4 on page 13. Like our normal decimal system, the binary number also has positional values. This means, the leftmost 7th bit is the most significant and its value is 128. The right-most 0th bit is the least significant and has the positional value as 1. To convert the binary number to its decimal equivalent, follow this procedure:

1. Start from the least significant 0th bit.
2. Take the bit value 0 or 1 from each bit.
3. Multiply this 1 or 0, by the correrspondinng position value.
4. Write down the products of (bit value x position value) on a paper.
5. Add these products. This is the decimal equivalent of binary number.

In the diagram 1.4 on page 13, the binary number is 10101100 and its decimal equivalent is 172. This 8 bit arrangement is called as a *byte*. Follow the procedure mentioned above and verify this result.

An AVR micro controller can handle an 8 bit number at a time. This means, all arithmetic and logical operations can be done on any two numbers

1. within the range 0 to 255 using one instruction.
2. using only one micro controller instruction.

Table 1.1: Arduino data types

Type	**Bytes**	**Range**
char	1	-128 to 127
byte	1	0 to 255
boolean	1	false (0) or true (1)
double or float	4	3.4028235E+38 to -3.4028235E38
unsigned long	4	4294967295
long	4	-2147483648 to 2147483647
unsigned int	2	0 to 65535
int	2	-32768 to +32767

If the number is greater than 255, the programmer uses usual arithmetic methods and write a small program. We will use Arduino language[8] to program the kit cpu. We can use following data types directly. That enables us to handle large numbers without any hassels.

Exercises

- Make a list of micro controller applications in your home.
- What is analog signal ? What is digital signal ?
- What is a clock ? How is it used by a micro controller ?
- What is an electromagnetic relay ? How does it work?
- How can a micro controller "read" input signal from a push-button ?
- What can a micro controller basically do ?
- Why is analog to digital conversion necessary ?
- What is the maximum value 1 byte number, 2 bytes number, 4 bytes' number?

[8]A high level language based on C and C++. It is very user-friendly.

Chapter 2

AVR architecture

Atmega328 is our cpu. It is an 8-bit micro controller. Architechture means discription of internal functional blocks without going into the circuit details. Each block has its function and is related to the other blocks. Together, all the blocks make a system.

2.1 Block diagram

See the block diagram of Atmega328 in figure 2.1 on page 16. Various internal parts of the cpu are shown.

2.1.1 General observations

1. Every block is connected to an 8 bit *bus.*[1]

2. Different types of memories are available.

3. Different types of addressing is possible.

4. Input and output lines are available for interface with the real world.

5. Arithmetic and logic unit, along with status & control, form the central part.

2.1.2 Memory types

Every intelligent system needs at least two types of memories.

[1]It is the 8 track path which is used for 8 bit data exchange between different blocks.

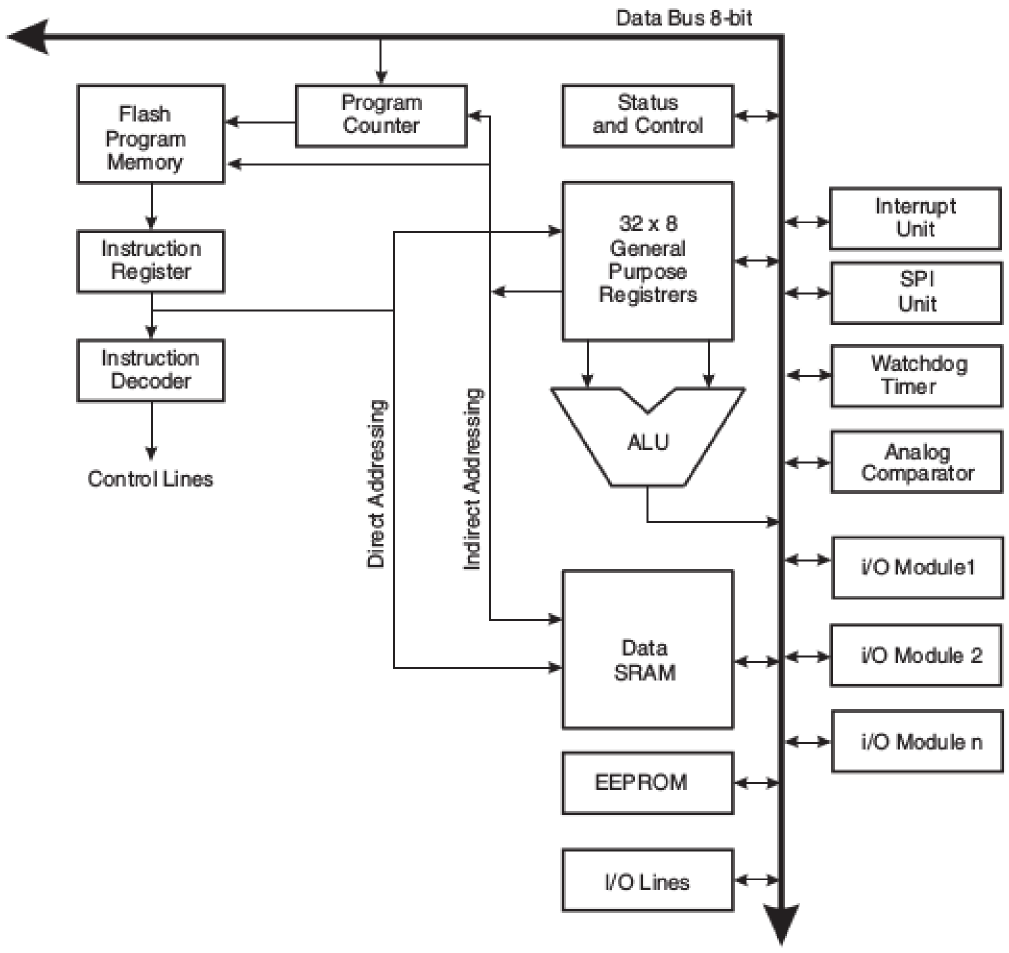


Figure 2.1: Block Diagram of Atmega328

Program memory

AVR micro controllers use flash type of program memory.

- Program memory stores programs written by the programmers.
- The cpu[2] implements the tasks guided by this program.
- This memory is non-volatile[3].
- It takes more time to read and write flash memory, compared to the data memory.

[2]It means arithmetic logic unit and status-control unit

[3]It retains the information even if power is off.

- You can't write one byte of flash program meomory. It must be written in pages[4].
- Atmega328 contains 32 kilo bytes of program memory.

Data memory

- The cpu needs this **temporary memory** when it processes numbers.
- This is known as **data memory.**
- It is volatile memory[5].
- It is faster to access as compared with the program memory.
- You can write a single byte of data memory. No need to access the full page.
- Atmega328 contains 2 kilo bytes of data memory.

In addition to the above two memories one more type of memory is available in AVRs.

EEPROM

- EEPROM means **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory.
- You need to store *data in the non-volatile form* while some process[6] is going on. EEPROM serves this purpose.
- In atmega328, 1 Kilobyte EEPROM is available.
- You can write/erase EEPROM 100,000 times.

2.1.3 General purpose registers

- General purpose registers are part of data memory.
- They are particularly useful for storing global variables and Status Flags.

[4]An array of 128 bytes

[5]When power is off, the data is lost.

[6]For example: room temperature after every 10 minutes throughout the day is sensed and stored

- General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.
- These registers are directly hardware connected to the cpu so they can be accessed very quickly for reading and writing.

2.1.4 Status-Control unit

This is a very complicated part of a micro controller.

- It maintains coordination between various blocks.
- Using PLA technique, it decodes the instructions.
- It synchronizes processes with reference to the crystal clock oscillator.
- It stores status of various registers for events like overflow, carry etc.

2.1.5 Special blocks

Interrupt unit

Any software process can be interrupted by an external hardware signal. *This allows micro controller to implement predefined processes at unscheduled time.*
Let us take an example.

- Suppose you write a program of a simple timer switch.
- You start the timer for 10 minutes.
- In the meanwhile you think that you want to change the time.
- To change the time, you have to interrupt the time counting process by an external (say, high to low) signal.
- You can do it using the interrupt unit.
- *Remember however,*that what to do after you interrupt the process, must be predefined in the original program. This predefinition of the process is stored at the corrosponding "interrupt vector address".

SPI unit

The Serial Peripheral Interface (SPI) allows *high-speed*synchronous data transfer between the atmega328 and peripheral devices or between several AVR devices. The only limitation is of the physical distance between these devices.

Watchdog timer

The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.
This timer helps the system to avoid the possiblity of "hanging" the application software. Time-out value can be chosen by the programmer and is in a range 16mS to 2 Sec.

Analog comparator unit

Atmega328 contains an inbuilt analog comparator. It can be used to decide which of the two analog signals is greater in magnitude. Features of analog comparator can be listed as below:

- The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1.
- When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set.
- The comparators output can be set to trigger the Timer/Counter1 Input Capture function.
- In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle.

Analog to digital converter unit

A six channel analog to digital converter is available in Atmega328. It has following features:

- Each channel gives 10 bit resolution. This means, analog input voltage within the range 0 t o +5V is converted to a proportional number between 0 to 1023.
- Conversion time lies between 13 μ seconds to 260 μ seconds, depending upon crystal clock frequency and chosen prescalar while programming.
- Optional left adjustment for adc result readout. The left adjustment enables the programmer to convert the ten bit result to a byte, which contains the most significant 8 bits of the 10 bit result.

- Optionally, input reference voltage can be chosen to be 1.1V. This enables the programmer to use a sensor (like temperature sensor LM35) giving maximum 1V at the output. This eliminates the need of an additional amplifier.
- Free running of the adc can be enabled so that the micro controller receives continuous values from a rapidly changing signal. Single conversion mode is also a good choice if slowly varying signal is sensed.
- We can produce an interrupt signal after the conversion is complete. This enables the programmer to make next process automatic.

2.2 Relation between various blocks

The internal architecture of atmega328 is like a factory architecture.

1. The program memory works like the book of manufacturing design rules.
2. The control block works like managerial section, coordinating various blocks.
3. Arithmetic and Logic Unit works like a workshop where all machinery is installed.
4. Data memory busses works like floor space and conveyor belts.
5. Input pins work like raw material incoming ports.
6. Output pins work like finished product storage.

2.3 Fetch-Decode-Execute cycle

Refer to the figure 2.2 on page 21. This shows how an instruction from program memory is finally executed. Let us know more about each step:

2.3.1 Fetch

1. A programmer writes a program.
2. An assembler or compiler[7] converts it into an executable hex file.

[7]We use avrgcc to compile.

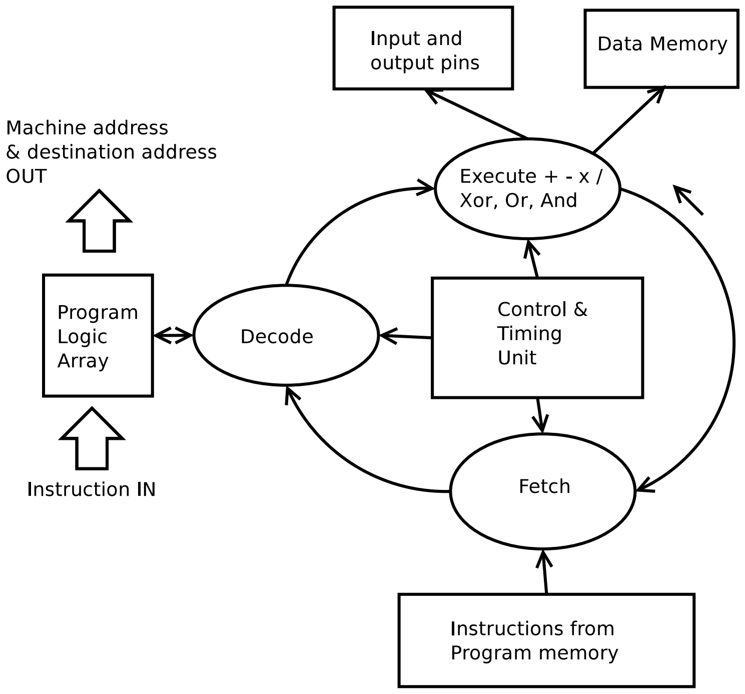

Input and output pins
Data Memory
Machine address & destination address OUT
Execute + - x / Xor, Or, And
Program Logic Array
Decode
Control & Timing Unit
Instruction IN
Fetch
Instructions from Program memory

Figure 2.2: Fetch Decode Execute Cycle

3. This file is dumped into the program memory (non volatile memory) of atmega328 using chip uploader[8].

4. When the chip is powered-up, the reset circuit forces the control unit to fetch the first instruction from the non-volatile program memory. Note that this instruction is a set of 8 bit numbers.

5. The control unit loads this instruction (in number form) into the "instruction decoder".

2.3.2 Decode

The decoding arrangement is in the form of "programmable logic array".

1. The instruction in number form is treated as input to this array.

2. Another number comes out from the array which decides the address of the "machine"[9]

[8]we use "avrdude" software for this purpose.
[9]Adder, subtractor, multiplier, divisor or any other logical operation machine".

3. The decoder also finds addresses of operands for operation.
4. The decoder also finds the destination address for the result of operation.

2.3.3 Execute

1. Once the operands and the operating machines are known, the operands are brought to the machine using data bus.
2. "Machines" operate on the data and output is stored on the destination address. Destination can be an output port or some data memory address.

2.4 Hardware specifications and peripheral features

Atmega328 integrates a large set of digital electronic circuits. In this section, we enlist some important ones.

2.4.1 Chip voltage and power

Atmega328 can work within the range 1.8V to 5.5V.
Power Consumption at 1MHz, 1.8V, 25C

- Active Mode: 0.2mA
- Power-down Mode: 0.1μA
- Power-save Mode: 0.75μA (Including 32kHz RTC)

As the voltage increases, power increases in the square proportion. Also higher clock frequency needs higher power.

2.4.2 Clock frequency

Clock frequency has vital importance. It decides operational speed of the device.

- Clock frequency can be within the range 0 to 20 MHz. At 20 MHz, 20 MIPS[10] thoroughput available.
-

[10]million instructions per second

2.4.3 Program memory

These are the program memory specifications for atmega328.

1. It is non-volatile memory.
2. Flash Program memory of 32K bytes available
3. You can reprogram this memory 10,000 times.
4. Data is retained upto 20 years at 85°C or upto 100 years at 25°C.
5. This memory must be written only block by block (called page).

2.4.4 Data memory

1. Volatile memory. It loses its contents when power is off.
2. 2kB in size.
3. It is static ram so no dynamic refreshing needed.

2.4.5 EEPROM

1. 1KB in size
2. Non-volatile memory
3. Reprogrammable byte by byte. 100,000 reprograming cycles possible.
4. Data is retained upto 20 years at 85°C or upto 100 years at 25°C.

2.4.6 Timers & counters

Any digital timer is actually a binary counter. Binary counters can be used as timers if they are updated using fix frequency pulse triggers.
Suppose a pulse frequency of 1KHz is triggering a binary counter, each update will require one mili second. If it is an 8 bit counter, it will take 255 mS to reach the maximum value. It also means that this timer can measure time upto 255 mS. By changing the triggering frequency, we can change the maximum measureable time.
Timers/counters run simultaneously when program is running. This makes the micro controller a (sort of) multi tasking device.
There are three timers available (in addition to watchdog timer) in atmega328.

8 bit timer/counter 0

- It can generate interrupts at 3 occasions.
- Maximum value is 255
- Bottom value is 0
- Using the output compare match facility, this timer can be used to:
  - Generate waveforms.
  - Implement pulse width modulation.

16 bit timer/counter 1

- Four independant interrupt sources.
- 16 bit PWM.
- Variable frequency period.
- Can implement frequency generator.

8 bit timer/counter 2

- Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV2, OCF2A and OCF2B)
- Allows Clocking from External 32kHz Watch Crystal Independent of the I/O Clock

2.4.7 PWM channels

Using timers metioned in the earlier section, programmer can implement PWM[11] channels. Tinah board uses Arduino pin configurations. So the PWM library functions will operate only on the pins mentioned in the table: 2.1 on page 25.
PWM channels make it easy to use the arduino command **analogWrite**. *PWM can also be used as a simple Digital to Analog converter.*

[11]Pulse Width Modulation means, frequency remains same but On time changes.

Table 2.1: PWM channels

Arduino Pin	**Atmega328 pin**
3	5
5	11
6	12
9	15
10	16
11	17

2.4.8 Analog to digital converters

Atmega328 has 6 channels of 10 bit analog to digital converter. This means:

1. 6 Different analog signals can be sensed (only one at a time).
2. Each signal is converted to a proportional 10 bit binary number[12].This is known as 10 bit *resolution.*
3. Because atmega328 is an 8 bit controller, the 10 bit result *may* be easily adjusted to 8 bitresult (with some sacrifice in accuracy)

2.4.9 USART: Serial communication

Universal Synchronous Asynchronous Receiver Transmitter enables the controller to communicate with any device like a computer or other micro controller. This communication is serial. Bit by bit data is sent and received. Baud rate[13] upto 38400 bits/sec is easily possible. *Standard rate for reliable communication is 9600 Bits/sec.* Use of booster-ICs like max232 or max486 helps long distance communication. See figure 2.3 on page 26.

2.4.10 SPI: Serial peripheral interface

This is another type of serial communication. However, it is useful between two micro controllersplaced near to each other. It is used in a multi-controller system. The baud rate may go upto mega-bytes per second. See the figure 2.4 on page 26. The same interface is used to upload program (sketch in arduino language) into the chip flash memory.

[12]A number between 0 to 1023

[13]Number of bits sent or received per second.

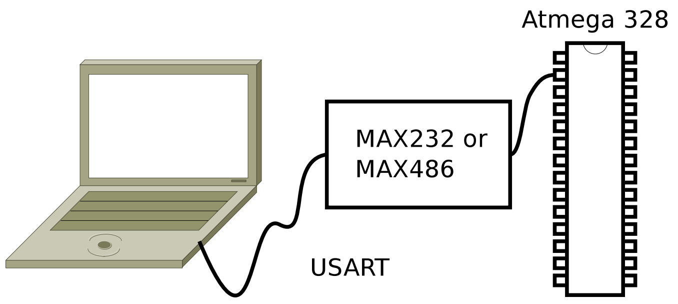


Figure 2.3: USART-communicate with computer

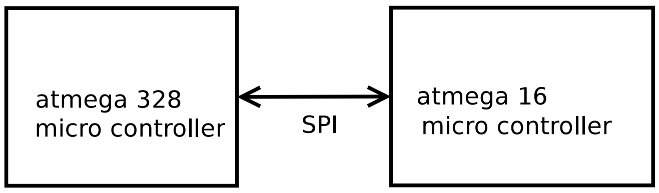


Figure 2.4: Serial peripheral interface

2.4.11 TWI: two wire interface

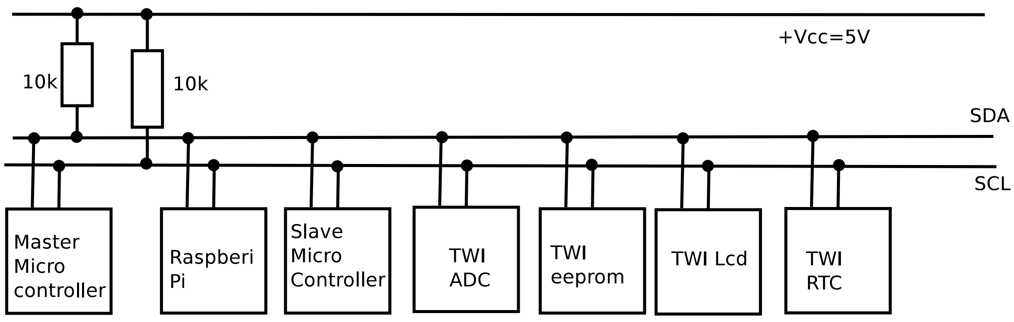


Figure 2.5: TWI devices on common bus

This is a protocol (method) compatible to philips I^2C protocol. Many devices support this protocol. Some of these devices are- micro controllers, memories, real time calendars, analog to digital converters. Using this feature, many such devices can be connected to the micro controller at the same time to form a large network. In atmega328, this protocol is supported by appropriate hardware, handling 8 bit data. See figure 2.5 on page 26. Observe how many TWI devices are connected to each other on the same bus.

Table 2.2: Interrupt Services

Interrupt type	**Description**	**Vector address (hex)**
Reset	External,power-on,brown out,watch-dog	0x000
External interrupt	Interrupt from outer world	0x001,0x002
Pin change	Pin change interrupt	0x003,0x004,0x005
WDT	Watch-dog time-out	0x006
Timer interrupts	compare match,overflow,event capture	0x007 to 0x010
SPI	SPI data transfer complete	0x011
USART	Xmit,Recpn complete,data empty	0x012 to 0x014
ADC	AD conversion complete	0x015
EEPROM	EEPROM ready	0x016
Total	—	23 interrupts

2.4.12 Analog Comparator

An inbuilt analog comparator helps atmega328 to handle two analog signals from the outer world. The analog comparator will decide which one of them is greater than the other, making its output low or high. Depending on this, program may take new mode. This is the response of the controller to the outer analog signals. This output also can raise an interrupt[14] signal to the controller.

2.4.13 Interrupt services

Refer to the table 2.2 on page 27. Different 23 interrupt services are available with atmega328.

2.4.14 Power on reset and brown out detection

Power on reset

Every micro controller must begin executing the program from the first instruction.
Refer to the figure 2.6 on page 28. When a positive pulse is applied to the power-on reset pin[15] micro controller is forced to go and fetch the very first

[14]See section Interrupt unit on page 18
[15]Pin 1 of atmega328.

instruction in the program memory. The shown values of the resistor and the capacitor are typical but are not fixed. The reset pin must be LOWER than typically 1.3Volts to 1.4 Volts for 1.5 μsec minimum.

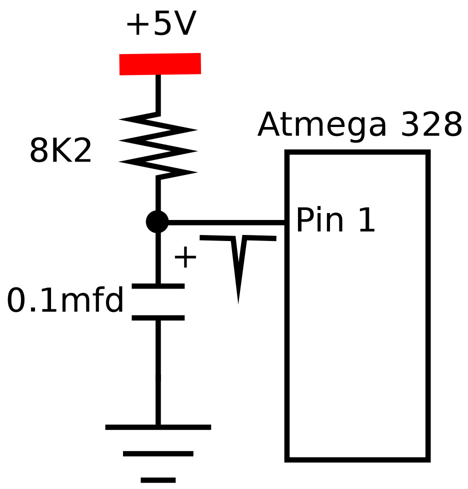


Figure 2.6: Power on reset pulse

Brown out detection

A micro controller works fine when the power supply is within the correct range. Atmega 328 is configured to work well within the range 4.5V to 5.5V. If the voltage goes below the 4.5V, the controller will not understand the logic levels correctly and will malfunction. The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than $t_{BOD} = 2\mu Sec$. After the detection, micro controller is reset. Thus BOD will protect data during the period of *power glitch*[16].

2.4.15 Internal calibrated oscillator

Atmega 328 can work without an external oscillator. By default, an rc oscillator of 8.0MHz is active. Its frequency can be fine tuned by the programmer. Using a calibration register OSCCAL, the programmer can set and tune the frequency within the range 1.0 MHz to 8.0MHz. This is useful in reducing the component count and finally the product price.

[16]Power goes down for very small time and returns to normal value quickly.

Exercises

- Why do we need different types of memories ?
- Explain how a micro controller is similar to a factory.
- What is an FDE cycle ?
- Atmega328 contrller allows the programmer to improvise the program thousands of times. Explain.
- How do timer/counters make the micro controller a multi-tasking device?
- How would you use pwm channel as a simple DAC ?
- What do you mean by *10 bit resolution* of an ADC ?
- Differentiate between SPI and USART.
- What is an interrupt? How does it help the application?
- How is TWI very useful in connecting many devices together ?
- How will you use analog comparator which is inbuilt in atmega328?
- What is the speciality of *power on reset* circuit?
- What is brown-out detection? How is it useful?
- Is external crystal oscillator compulsory to use as a clock ? Why?