2

Architecture of ATmega328P Microcontroller

External architecture refers to the organization, direction, and function of the 28 pins of the ATmega328P microcontroller. Internal architecture refers to the organization, interconnection, and function of various processing and peripheral modules that are present within the microcontroller.

2.1 Physical Pin Diagram

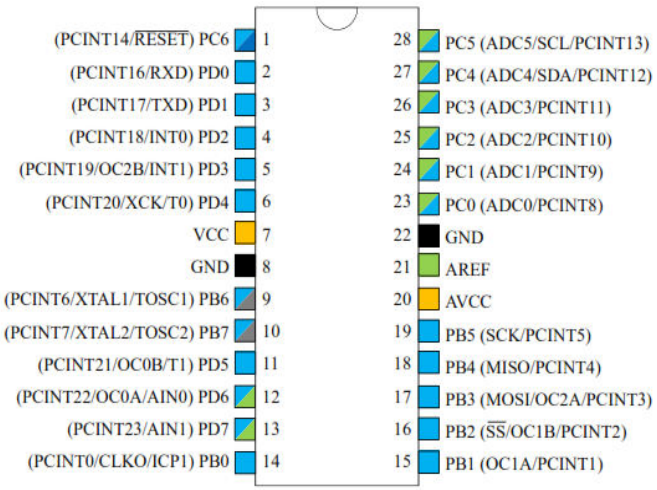

Figure-2.1: *Physical pin diagram of ATmega328P microcontroller*

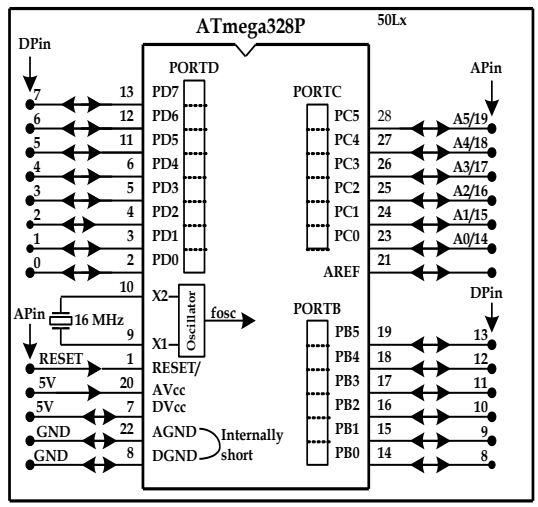

Figure-2.2: *Port structured diagram of ATmega328P*

(1) A pin has a serial number, direction, and attachment with one or more signals of which only one signal is active at a time which is outside the pair of parentheses. For example: In Fig-2.1, Pin-4 has the serial number 4; it is attached with 3 signals viz., PCINT8, INT0, and PD2 of which PD2 signal is outside the pair of parentheses, and it is the **active** signal.

(2) In Fig-2.1/2.2, the active signals are: PB0 – PB6, PC0 – PC5, and PD0 – PD7; they are serving as general purpose digital IO lies to exchange 1-bit data with external devices. These lines can be individually programmed to work either as input or as output. The groupings of the IO pins are depicted in Fig-2.2.

(3) In Fig-2.2, DPin stands for 'Digital Pin'; they refer to the numbers printed near the edge connectors installed on the PCB.

(4) The following codes are used to configure directions and to perform data exchange operations with the digital IO pins.

```
pinMode(DPin, INPUT);         //the pin/signal named by DPin will work as  an input line.
pinMode(DPin, INPUT_PILLUP); //DPin is an input line with internal pull-up resistor enabled.
pinMode(DPin, OUTPUT);        //DPin will work as output line
digitalWrite(DPin, LOW);      //LOW low (0V – 0.9V) is asserted on DPin
digitalWrite(DPin, HIGH);     //HIGH (4.2V – 5.0V) is asserted on DPin
bitSet(PORTX, bitPosition);   //HIGH is asserted on: X = B/C/D; bitPosition = 0 to 7;
bitWrite(PORTDX, bitPosition, bitValue);    //bitValue = HIGH or LOW
bitClear(PORTX, bitPosition);               //LOW is asserted on the target pin
bool n = bitRead(PINX, bitPosition);        //1-bit data is read from an input pin of PINB/PINC/PIND
bool n = digitalRead(DPin);                 //1-bit data is read from DPin when it is an input
```

(5) Create sketch to check that S1 of Fig-2.3 is opened and then blink LD (L) only for 5 times at 1-sec interval.

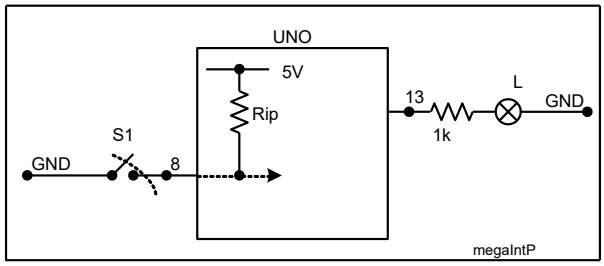

Figure-2.3: *State change event of an IO pin of ATmega328P microcontroller*

2.2 Hardware Block Diagram for the Internal Resources of ATmega328P MCU

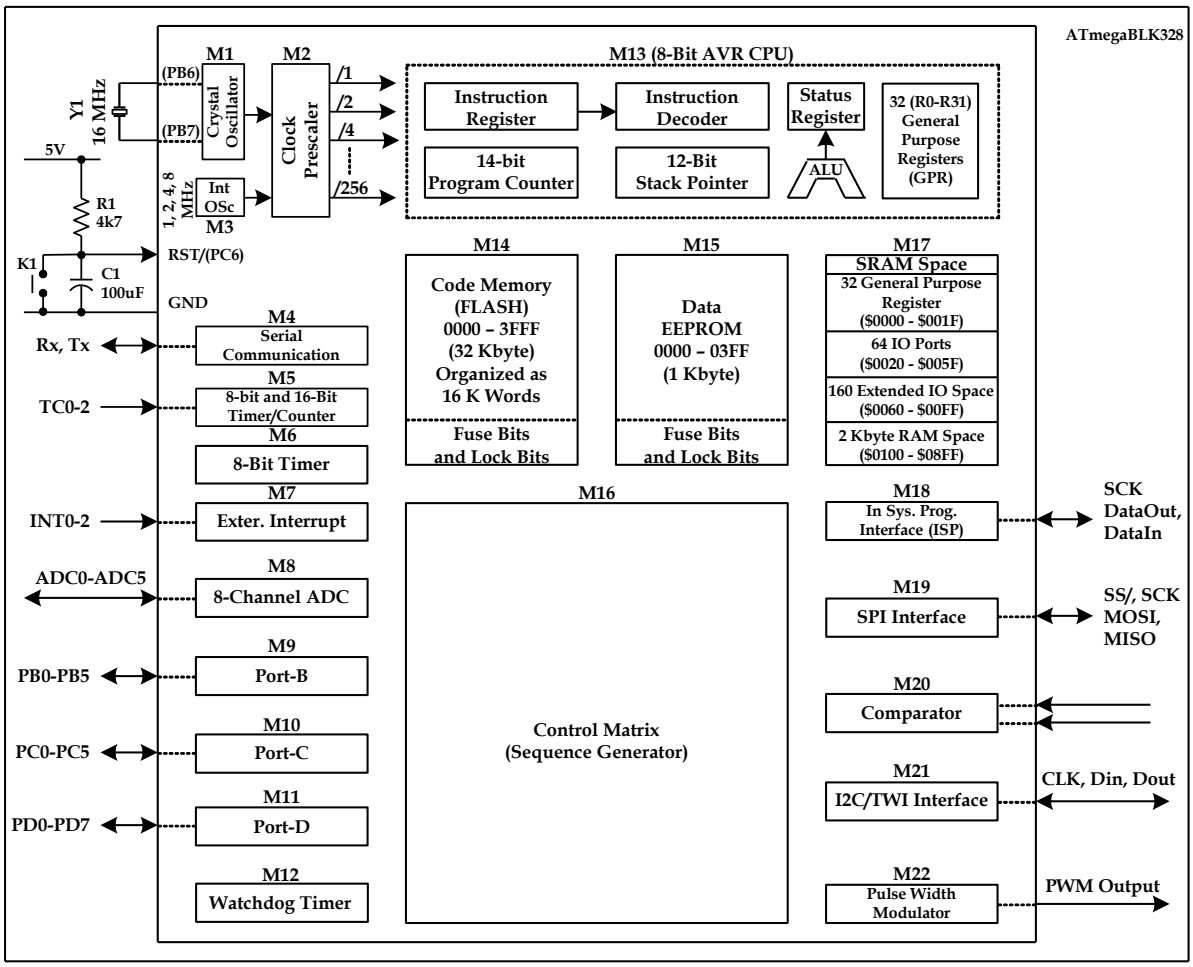

Figure-2.4: *Hardware block diagram for the internal resources of ATmega328P microcontroller*

(1) Clock Oscillator:
(2) System Clock Prescaler:
(3) 8-bit AVR Core:
(4) General Purpose Register:

(5) Standard IO Ports:
(6) Expanded IO Ports:
(7) Flash Memory:
(8) EEPROM Data Memory:
(9) RAM Memory
(10) UART Port
(11) SPI Port
(12) I2C Bus
(13) Analog to Digital Converter
(14) Timer/Counter Module
(15) Pulse Width Modulation
(16) Interrupt Structure
(17) In System Programming Interface:
(18) Analog Comparator:
(19) Fuse Bits
(20) Lock Bits
(21) Watchdog Timer

2.3 System Clock Prescaler

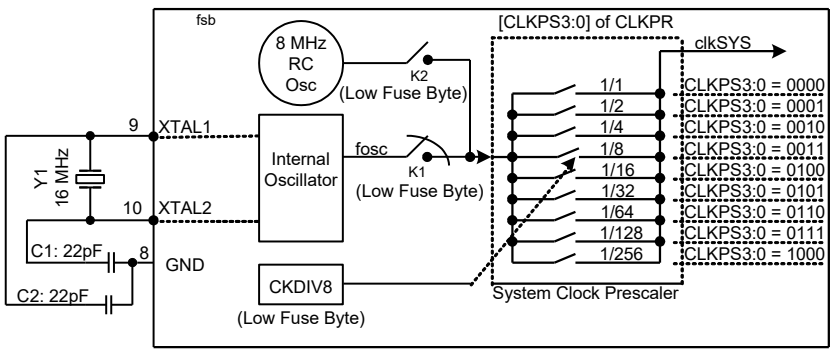

Figure-2.5: *Hardware block diagram for the internal resources of ATmega328P microcontroller*

(1) The MCU needs clock pulses for the generation of timing functions. The clock source could be an external 16 MHz accurate crystal or an internal inaccurate 8 MHz oscillator. When the chip is shipped from the Factory, the setting is made at 8 MHz internal oscillator. In Arduino UNO, the setting has been made at the 16 MHz source. The setting is changed with the help of **Fuse Bits**.

(2) There is a 'System Clock Prescaler' to divide the f_{osc} clock. The user can change the setting through programming codes. In Arduino Platform, the setting is at divide factor 1.

2.4 Flash Memory

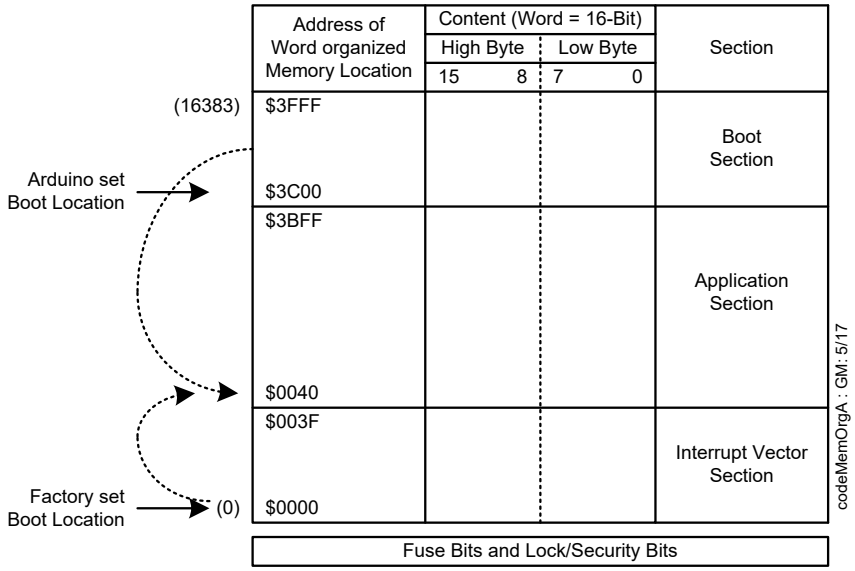

Figure-2.6: *Flash memory organization of ATmega328P microcontroller*

(1) Flash memory or code memory or program memory holds binary codes of our sketches that we create, compile, and upload using Arduino IDE. A sketch may contain both executable codes and non-executable data. Flash memory is non-volatile memory which means that the memory holds its contents even at the absence of power.

(2) The capacity of the flash memory in ATmega328P MCU is 32 Kbytes (32 x 1024 bytes). But from code/data storage point of view, it is organized as WORD (2 bytes = 16 bits) location; where, each WORD location has its own address.

(3) The flash space is divided into three sub-spaces: Interrupt Vector Section, Application Section, and Boot Section.

- **(a)** Interrupt vector section offers a location to link between an ISR (located in application section) and its interrupting source.
- **(b)** Application section holds the binary codes of the user application program.
- **(c)** Boot section contains a **Boot Loader Program** which interacts with the IDE and uploads the binary codes of the sketch from the IDE into the application section. After uploading, the MCU jumps to the beginning address of the application program.

(4) **Boot Location:** After power up reset, the MCU jumps to a particular location/address of the flash to begin with program execution. This location is known as Boot Location. When the MCU shipped from the factor, the boot location is at 0x0000 ($0000). In Arduino UNO, the boot location has been reset at location 0x3C00 with the help of **Fuse Bits**.