3 Digital Input/Output (IO) Controller

3.1 Internal Structure of an Output Port Line

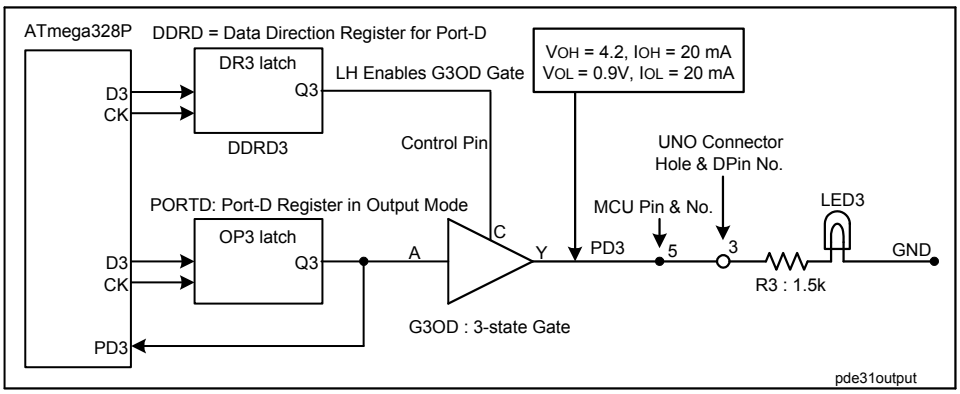


Figure-3.1: *Internal structure of an output port line of ATmega328P MCU*

(1) In Fig-3.1, we observe that Pin-5 of MCU is connected with DPin-3 (Digital Pin 3) of UNO.

(2) To operate Pin-5 as 'output line', we enable gate G3OD (Gate for Output bit-3 of portD) by placing HIGH at its Control Pin (C). The data for the control pin of the gate comes from the MCU via DR3 latch. DR3 latch is a part of Data Direction Register for Port-D (DDRD).

(3) Now, Pin-5 takes over the symbolic name – PD3 or PORTD3. Data (HIGH or LOW) for DPin-3/PD3 comes from the MCU via OP3 latch. OP3 latch is a part of PORTD Register.

(4) The data (HIGH or LOW) that is already written into PD3 is available to the MCU from the output of OP3 latch.

(5) When HIGH is written on PD3, the PD3 line can deliver/source maximum 20 mA current to an external load by maintaining 4.2V at its terminal. The symbolic name of this voltage level is V_{OH}, and the symbolic name for the current is I_{OH}.

(6) When LOW is written on PD3, the PD3 line can sink maximum 20 mA current to be delivered by the output of an external load by maintaining 0.9V at its terminal. The symbolic name of this voltage level is V_{OL}, and the symbolic name for the current is I_{OL}.

3.2 Internal Structure of an Input Port Line

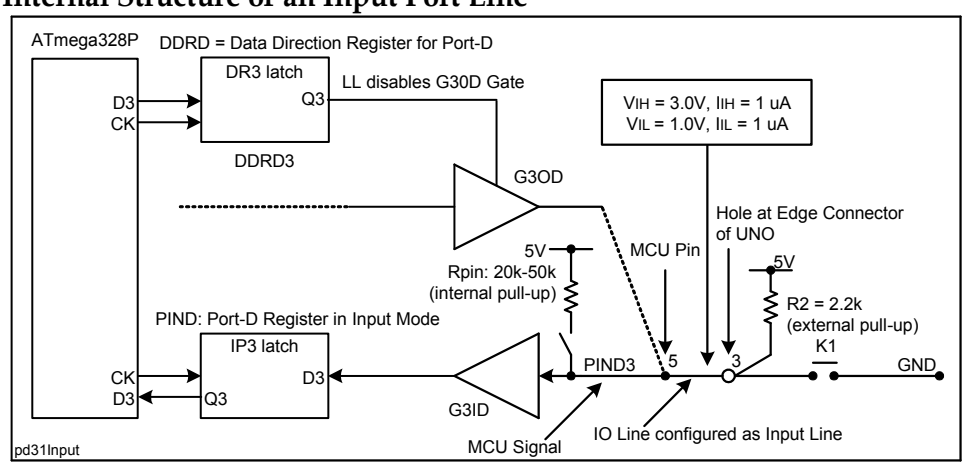


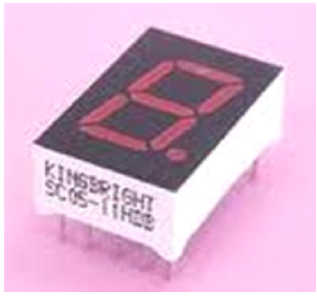
Figure-3.2: *Internal structure of an input port line of ATmega328P MCU*

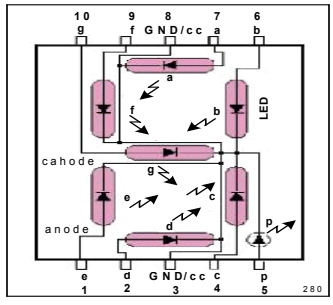
(1) In Fig-3.2, we observe that Pin-5 of MCU is connected with DPin-3 (Digital Pin 3) of UNO.
(2) To operate Pin-5 as 'input line', we disable gate G3OD by placing LOW at its Control Pin (C).
(3) Now, Pin-5 takes over the symbolic name – PIND3. Data (HIGH or LOW) from an input device (K1) goes to the MCU via 'gate G3ID' and IP3 latch.
(4) The PIND3 line has an internal pull-up resistor Rip which is (at the moment) is at disconnected position. An input line must always be terminated to 5V using internal pull-up or external pull-up resistor. In Fig-3.2, we have terminated PIND3 line using external pull-up R2. If desired, the PIND3 line could be terminated to GND by using external pull-down resistor.

3.3 Arduino Codes/Instructions for Programming Digital IO Lines

(1) To set the direction of PD3 line of Fig-3.1 as output
```
pinMode(3, OUTPUT);          //DPin-3 will work as output line
```
(2) To set the direction of PD3 line of Fig-3.1 as output with the help of Bit-3 of DDRD Register.
```
DDRD = 0b00001000;           //Makes DPin-3's direction as output
```
(3) To write/send HIGH at PD3 line of Fig-3.1
```
digitalWrite(3, HIGH);       //Logic-high (4.2V to 5V) will appear at DPin-3
```
(4) To write/send LOW at PD3 line of Fig-3.1
```
digitalWrite(3, LOW);        //Logic-low (0V to 0.9V) will appear at DPin-3
```
(5) To set the direction of PIND3 line of Fig-3.2 as input without internal pull-up resistor Rip. In this case, we must place external pull-up resistor R2.
```
pinMode(3, INPUT);           //DPin-3 works as input line; not pull-up/pull-down resistor
```
(6) To read 1-bit logic-value of PIND3 line of Fig-3.2
```
bool n = digitalRead(3);     //logic value of DPin-3 enters into Boolean variable n
bool n = PIND3;              //Logic-value of DPin-3 (Pin-5 of MCU) will enter into n
```
(7) To set the direction of PIND3 line of Fig-3.2 as input with internal pull-up connected. In this case, we need to disconnect the external pull-up resistor R2.
```
pinMode(3 INPUT_PULLUP);     //DPin-3 works as input line with internal pull-up connected
```
(8) To set the direction of PD3 line of Fig-1.4 as output with the help of Bit-3 of DDRD Register.
```
DDRD = 0b00001000;           //Makes DPin-3's direction as output
```

3.4 Internal Structure of CC7SD Device





(a) Pictorial View *(b) Internal Structure*

Figure-3.3: *Structure of Common Cathode Type 7-Segment Display Device*

We call it 7-segment display device; but in reality, the device contains 8 segments which are named as segment-a (seg-a), seg-b, …,seg-p. Each segment is a LED (Light emitting Diode), and it emits light of optimum brightness when a current of about 15 mA passes through it, and a voltage drop of about

2.0 V is maintained across it. There are two terminals/pins of it — anode (A) and cathode (K). To ignite a LED/segment, we need to apply logic-high (5V) at the A-pin and GND/logic-low at the K-pin. In Fig-3.3b, we observe that all the 8 K-terminals of the LED devices are shorted together; this is the reason to call it a cc-type (common cathode type) 7-segment display device. This voltage/current constraint dictates us to put a 'current limiting resistor' of suitable value (560 Ω to 2.2 kΩ) in series with the segments line (Fig-3.6) to keep the current at the desired safe level.

3.5 Formation of cc-codes for Hexadecimal Digits

Table-3.5.1: Character Vs CC-code Table (1 indicates logic-high and 0 indicates logic-low)

Character	p	g	f	e	d	c	b	a	cc-code
0	0	0	1	1	1	1	1	1	3F
2	0	1	0	1	1	0	1	1	5B
---	---	---	---	---	---	---	---	---	---
F	0	1	1	1	0	0	0	1	71

Figure-3.4: *Lookup Table (LUT) for the hexadecimal characters 0 – F*

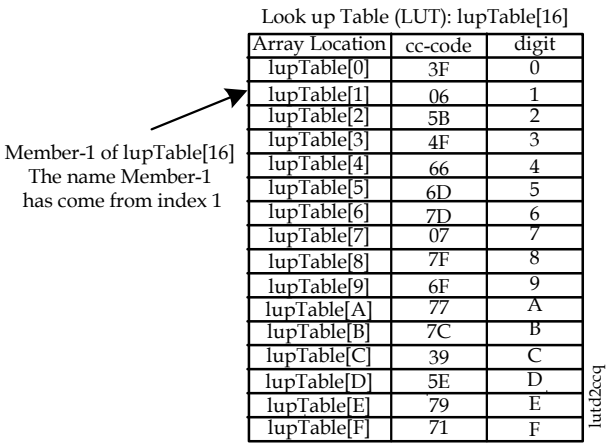


Figure-3.5: *Lookup Table (LUT) as array for the hexadecimal characters 0 – F*

There are 16 symbols/digits in the hexadecimal base system to represent a number. These symbols/digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F; where, A is for decimal 10, B for decimal 11, C for decimal 12, D for decimal 13, E for decimal 14, and F for decimal 15. Using the Table-3.5.1 of Fig-3.4, we can construct cc-code for every digit on the paper. If we arrange these cc-codes in a Table of Fig-1.3.2, we say that we have made a Lookup Table (LUT). Why do we say that it is LUT Table? It is a Table into which we look at the desire digit (say: 6) and immediately know/get the cc-code (7D = 01111101).

In Programming Language/Software, the LUT Table is known as **array**. An array always contains items/members/elements of the same type. For example: the array of Fig-3.5 contains 16 members which are all of the same type — the cc-codes. The features of an array are:

(1) It has a name, data type, and dimension. 'Name (here: lupTable)' refers to all the 16 members of the array; 'Data type (here: byte)' refers to the size (here: byte means 8-bit) of a member; 'dimension (here: 16)' refers to the number of members present in the array.

For example: *byte lupTable[16]*

(2) Every member has a name with an 'index number' in the 'dimension field'. The index refers to its position in the array.

For example:*lupTable[1]* refers to Member-1 of the array.

3.6 Program Codes to Manipulate Array in RAM and Flash Memory

(1) To create array in the RAM memory where members can be read, modified, and written back.

```
bytelupTable[16] =   //contains cc-codes for the digits: 0 to 9 and A to F
{0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
```

(2) To create array in the RAM memory where members can be read but cannot be modified and written back.

```
Const byte lupTable[16] =
{0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
```

(3) To create array in the Code Memory/Flash memory where items can be read but cannot be modified and written back. This procedure saves RAM memory comparing the size of RAM (only 2048 bytes) with that of Flash (32x1024 Bytes).

```
Const PROGMEM byte lupTable[16] =
{0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
```

(4) To read a member from an array, modify the member, and write it back in its original position.

```
Byte myData[4] = {0x31, 0x32, 0x33, 0x34};
byte x1 = lupTable[3];        // x1 = 0x34;
x1 = x1 + 0x80;               // x1 = 0xB4
lupTable[3] = x1;             // lupTable[3] = 0xB4
```

3.7 Sketch to show the Character 2 on a CC7SD Device

In this Subsection, we will create a sketch to show the character/digit **2** on the display device DP0 of the following Fig-3.6. The sketch will be tested using Arduino UNO Learning Kit/System. Let us first list the tasks that are involved in this programming:

(1) The cc-code of 2 is to be constructed using Fig-3.4, 3.5. The cc-code is: 5B. (Only once)
(2) The directions of the IO (input/output) of Port-D are to set as output lines. (Only once)
(3) The direction of PB0-line of PORTB is to be set as output line. (Only once)
(4) Write cc-code of Step-1 on PORTD. (Only once)
(5) Write logic-low (LOW) on cc-pin of DP0 via PB0-pin. (Only once)

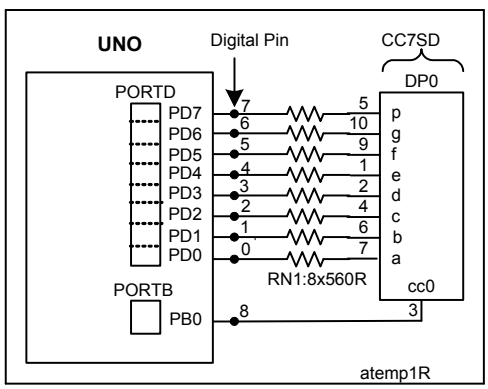


Figure-3.6: *Connection diagram between UNO and CC7SD device*

(6) Let us place the codes/instructions for above tasks (Step-1 to 5) in the following sketch.

```
void setup()
{
     Serial.begin(9600);      //command to establish connection/communication among UNO, PC, IDE, SM

     pinMode(0, OUTPUT);      //pinMode(DPin, value); PD0 = DPin-0 = 0, …, PD7 = DPin=7 = 7
     pinMode(1, OUTPUT);
     pinMode(2, OUTPUT);
     pinMode(3, OUTPUT);
     pinMode(4, OUTPUT);
     pinMode(5, OUTPUT);
     pinMode(5, OUTPUT);
     pinMode(6, OUTPUT);

     pinMode(8, OUTPUT);      //PB0 = Dpin-8
     //-------------------------------------
     PORTD = 0x5B;            //cc-code of 2 goes to DP0-position via PORTD Register
     //-------------------------------------
     digitalWrite(8, LOW);    //cc-pin of DP0 is at logic-low state.

}

Void loop()
{
     //the MCU is looping here = waiting and waiting for ever (infinite loop)
}
```

(7) Upload the sketch of Step-6 into UNO. After uploading, press the RESET button of the UNO and check that 2 have appeared on the display device.

3.8 Sketch to show 2 and 5 simultaneously on two CC7SDD Devices

In Fig-3.7, we have connected two CC7SD devices in parallel; where, the pins of the identical segments are shorted together and the cc-pins are kept separated. This kind of display system is known as 'multiplexed display unit'. In a multiplexed display unit, only one digit comes on the display unit at a time. It means that when 2 will appear at DP0-position, 3 will not appear at DP1-position. After a while (after some time delay) 3 will appear at DP1-position and 2 will not appear at DP0-position. What will we do if we want to see both digits on the display unit at the same time? We will decrease the 'after a while time (the time delay)' to such a level so that both digits are seen to have appeared at the same time. In fact, the digits are still appearing one after another; but as the switching time from one digital to another digit is so small (about 1 to 5 ms), both digits are seen to have appeared at the same time on the display unit. Let us list the tasks of this programming:

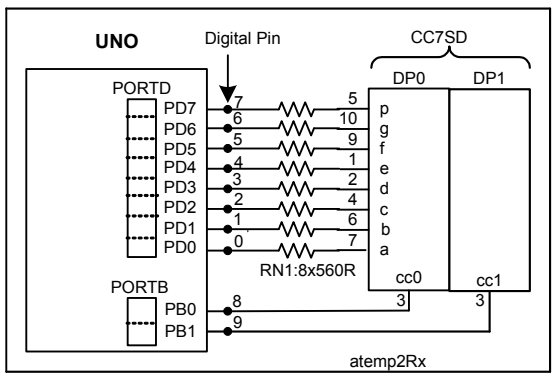


Figure-3.7: *Connection diagram between UNO and two CC7SD devices*

(1) The cc-code of 2 and 5 are to be constructed using Fig-3.4, 3.5. The cc-codes are: 5B and 6D (Only once)
(2) The directions of the IO (input/output) of Port-D are to set as output lines. (Only once)
(3) The direction of PB0 and PB1 lines of PORTB are to be set as output lines. (Only once)
(4) Write cc-code of 2 on PORTD for position DP0. (Again and again with 5 ms time gap)
(5) Write logic-low (LOW) on cc0-pin so that 2 appears on DP0-position. Write logic-high at cc1-pin so that DP1 remains OFF; as a result, nothing will appear over there. (Again and again with 5 ms time gap)
(6) Let us place the codes/instructions for above tasks (Step-1 to 5) in the following sketch.

```
void setup()
{
     Serial.begin(9600); //command to establish connection/communication among UNO, PC, IDE, SM

     DDRD = 0b11111111;  //PD0 to PD7 are output line; equivalent to writing 7 lines using pinMode()

     pinMode(8, OUTPUT);      //PB0 = Dpin-8 is output line
     pinMode(9, OUTPUT);      //PB1 = DPin-9 is output line.
     //-------------------------------------
}

void loop()
{
     PORTB = 0x5B;  //cc-code of 2 goes to PORTD; 2 will appear at DP0 if PB0 = LOW and PB1 = HIGH
     //-------------------------------------
     digitalWrite(8, LOW);    //cc0-pin of DP0 is at logic-low state; 2 appears at DP0-position
     digitalWrite(9, HIGH);   //DP1 display is OFF; nothing will appear on it; it is blank
     delay(5);                //wait for 5 ms
     //-------------------------------------

     PORTB = 0x6D;  //cc-code of 5 goes to PORTD; 5 will appear at DP1 if PB0 = HIGH and PB1 = LOW
     //-------------------------------------
     digitalWrite(8, HIGH);   //cc0-pin of DP0 is at HIGH state; it is OFF; nothing will appear on DP0
     digitalWrite(9, LOW);    //cc1-pin of DP0 is LOW state; it is ON; 3 appears at DP1-position
     delay(5);                //wait for 5 ms
     //-------------------------------------
}    //goto at the beginning and show 2 again
```

(7) Upload the sketch of Step-6 into UNO.
(8) After uploading, press the RESET button of the UNO.
(9) Check that 2 have appeared on DP0-position and 5 has appeared on DP1-position. If not, try again by adjusting the time delay.

3.9 Problems

1 Declare a 3-element array where the value each element ranges from 0 to 255.
2 Declare a 3-elemnt array where the value of each element ranges from 0 to 65535 (0x0000 to 0xFFFF). We use this keyword: ***unsigned int*** to refer to a number of the range 0 to 65535 in the positive domain.
3 Declare a 2-elemnt array where the value of each element ranges from -32768 to +32767 (0x8000 to 0x7FFF). We use this keyword: ***signed int*** or simply ***int*** to refer to a number of the range -32768 to +32767 (0x8000 to 0x7FFF).
4 Write codes to find the cc-code for the 1st digit (digit from left) of this variable: ***byte x;***. Save the cc-code in the variable ***y***. The cc-code of the said digit can be found from the following array.

```
bytelupTable[16] = //contains cc-codes for the digits: 0 to 9 and A to F
{0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
```

5 Given that: *byte y = 0x5B*. Write codes to put HIGH (1) at y_7 position of *y*.
6 Look into the sketch of Step-6 of Section-1.3.4; write one line code to replace all the 8-line *pinMode()* codes.

7 Write one line code to set the directions of all the IO lines of Port-B Register as output lines.

8 Write code using *for()* loop and *pinMode()* to set the directions of all the IO lines of Port-D Register as output lines.

9 Create a sketch to show 0, 1, 2, …, F, 0, 1, … on DP0-position of Fig-3.6.

10 Create a sketch to show 7 at DP0-postion of the following display unit.

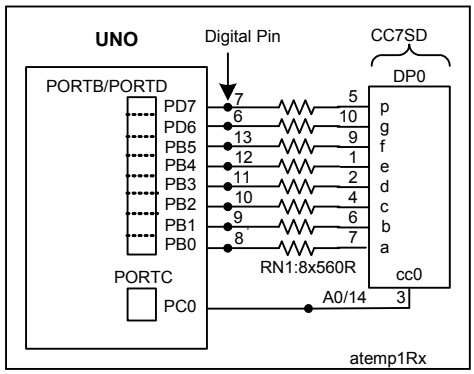


Figure-3.8: *Connection diagram between UNO and a CC7SD devices*

11 Write codes to add the decimal numbers 34 and 37 and show the result (71) on the display unit of Fig-3.7.

12 Modify program of Q10 so that the display shows 7.1.

13 When we execute *digitalWrite(3, HIGH)* instruction, what is the minimum level of the voltage that will be prevailing at the PD3 line? What is the symbolic name of this voltage level? How much current the PD3 line will be able to deliver to an external load?