Chapter 2: Concepts of Microcontrollers

In this chapter the reader will learn general information about:

- Connecting LEDs
- Connecting motors and coils
- Connecting switches and buttons
- Connecting keypads
- Analog to digital conversion
- Digital to analog conversion
- Power supplies for embedded systems
- Clock generation options
- Reset circuitry

Need for Digital Interfacing

- Embedded systems consist of computers embedded in larger systems.
- The processor needs to sense the state of the larger system and environment, and control output devices.
- We use General purpose Digital I/O ports to read from/ write to the larger system.

RX63N I/O ports

- Eighteen I/O ports

- Each port has
  - Port Direction Register (PDR),
  - Port Output Data Register (PODR),
  - Port Mode Register (PMR).

Port Direction Register (PDR)

- This register, as the name suggests, is used to set the data direction (input or output) of a pin.
- Each bit of the **Port Direction Register** represents a pin.
- Set the corresponding bit to 0 to make it input and set the bit to 1 to make it an output pin

Example: Setting Port 4 bit 1 as output

PORT4.PDR.BIT.B1 = 1;

	B7	B6	B5	B4	B3	B2	B1	B0
Value after reset:	0	0	0	0	0	0	0	0

Figure 2.1 Port Direction Register [1], page 661.

Port Output Data Register (PODR)

- Port Output Data Register is used as a buffer for the ports set as output ports.

Example: Sending a 0 on port 4 bit 1 which was already configured as an output pin

```
PORT4.PODR.BIT.B1 = 0;
```

	B7	B6	B5	B4	B3	B2	B1	B0
Value after reset:	0	0	0	0	0	0	0	0

Figure 2.2 Port Output Data Register [1], page 662.

Port Input Register (PIDR)

Port Input Register (PIDR) is used to read the state of a pin

Example: Reading the state of port 4, bit 1and, if it is high, make port D bit 0 high

```
if(PORT4.PIDR.BIT.B1 == 1)
    PORTD.PODR.BIT.B0 = 1;
```

Other Important Registers

Open Drain Control Registers (ODR0/ODR1)

Selects the output pin as either CMOS or NMOS open drain

Pull-Up Resistor Control Register (PCR)

Used to enable pull-up MOS for a particular pin of a port

Drive Capacity Control Register (DSCR)

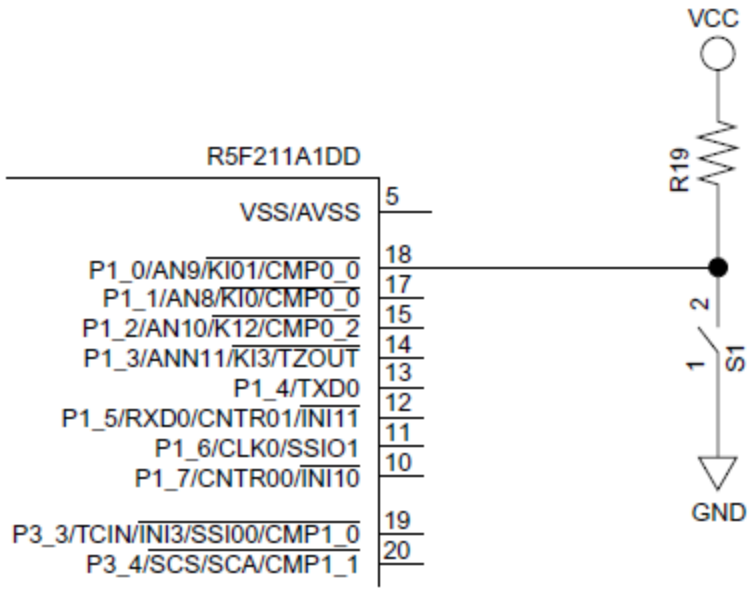
This register is used to switch the drive capacity of a port

Using Switches as Inputs

- The resistor will pull the voltage on that pin to logic HIGH until the button connects to ground.
- When the button is pressed, the voltage at that pin will drop to zero (logic LOW).

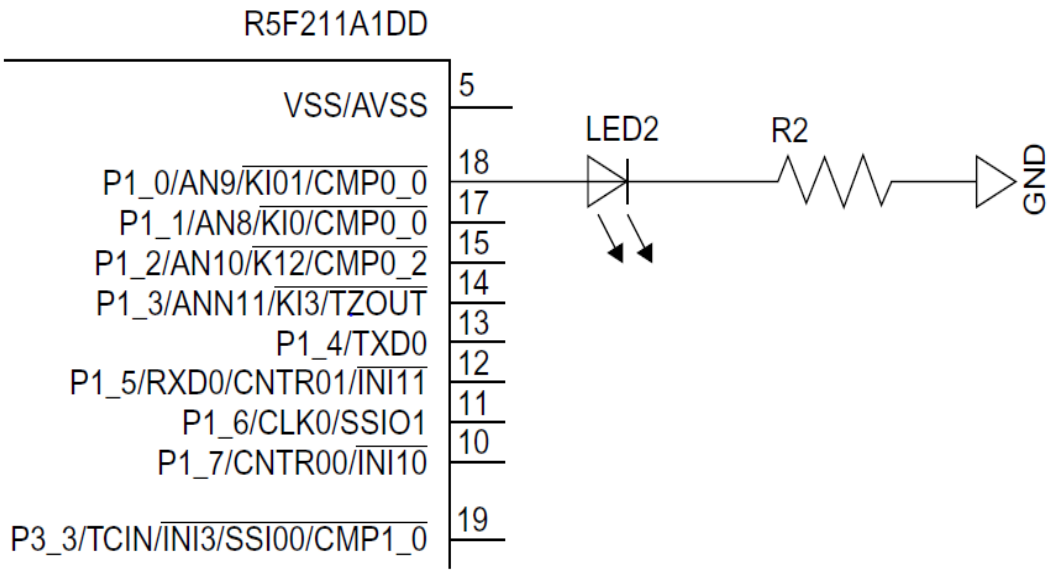
Code Snippet: Code to check a switch connected to Port 1 Bit 0

```
if(PORT1.PIDR.BIT.B0 == 0){
// code
}
```



Using LED as an Output

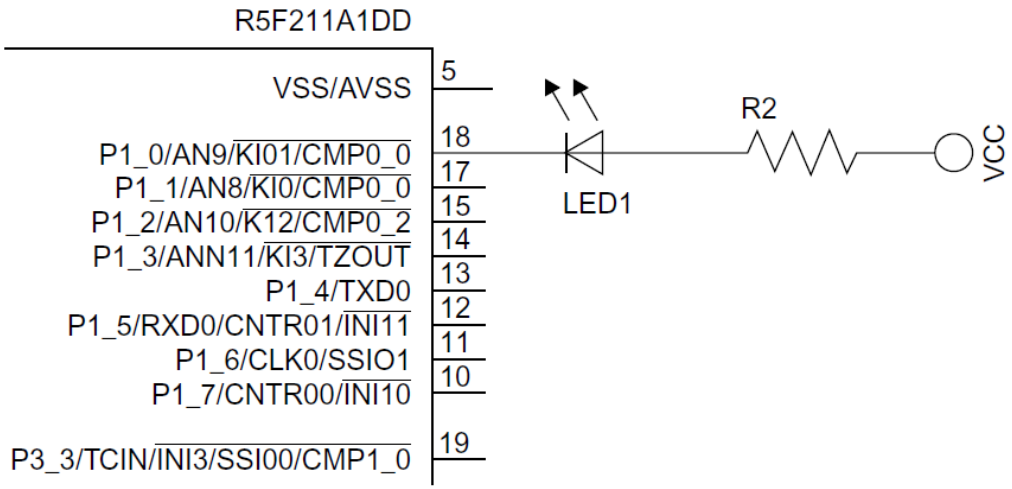
There are two ways in which you can interface an LED with a microcontroller:



The microprocessor is sourcing current to an LED.

Turning on this LED requires a logical HIGH signal on the output pin.

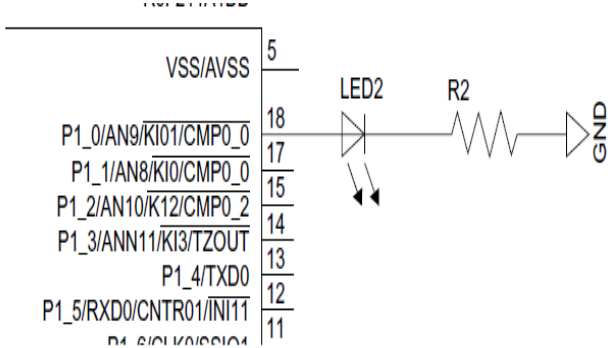
Using LED as an output

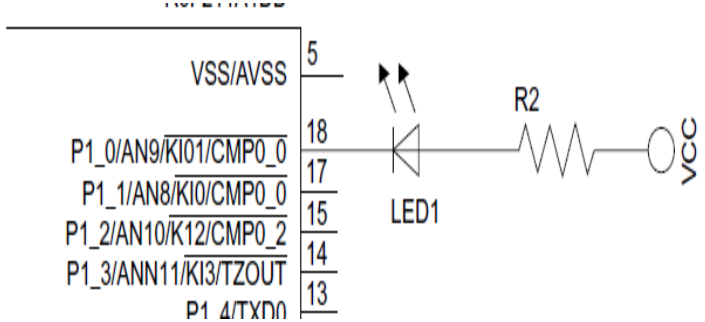


The microprocessor is sinking current from the LED.

Turning on the LED requires a logical LOW on the output pin.

Using LED as output

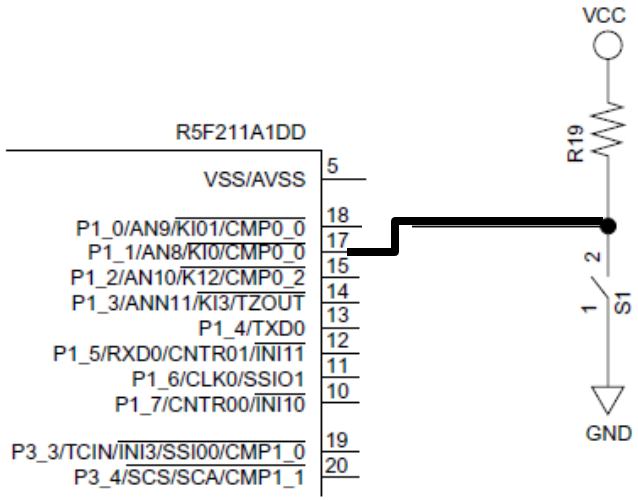




Of the two depicted ways of using LED as output which one is more efficient? Why?

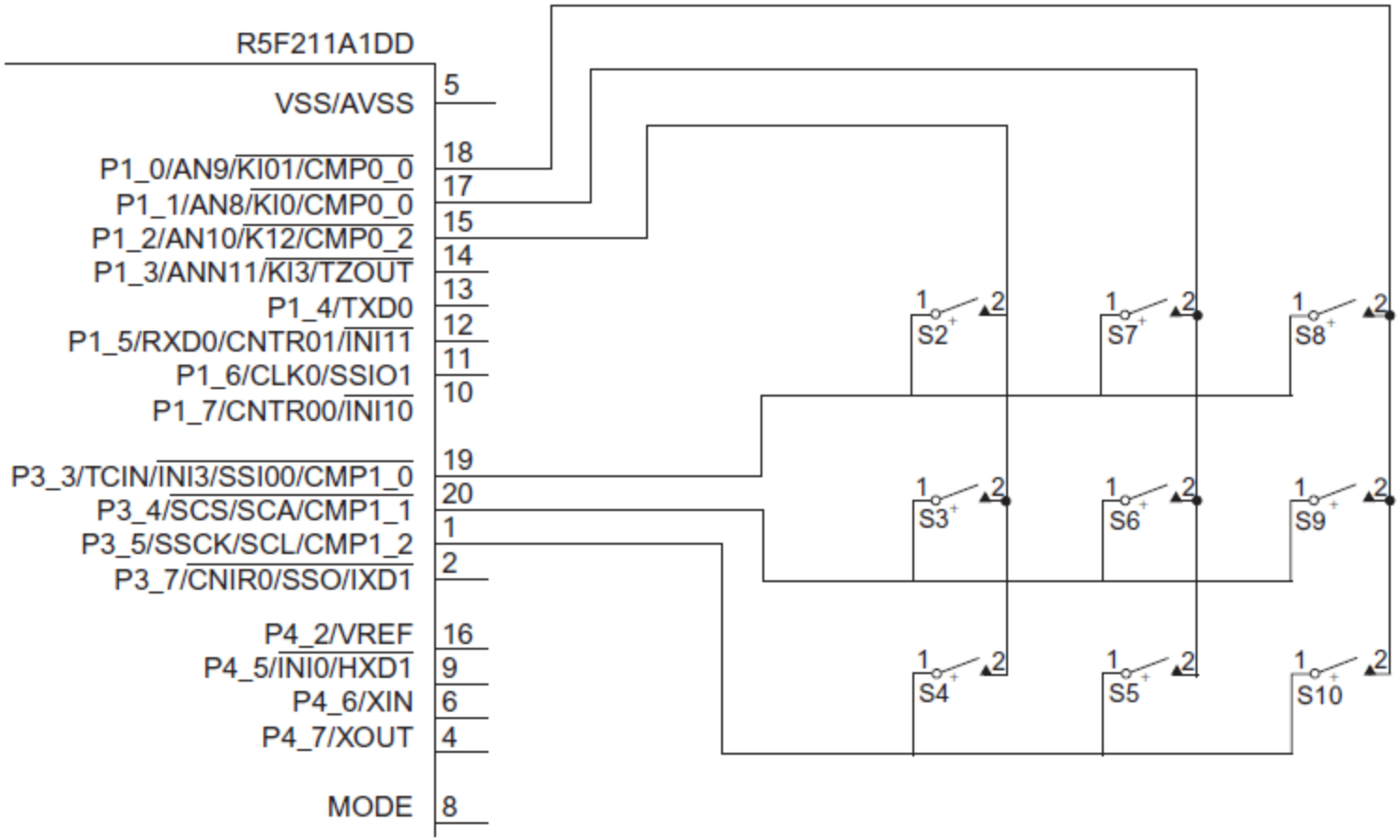
Why are the resistors necessary?

Write code to turn on the LED if S1 is pressed.



Interfacing a matrix keypad

We often find the need for using a Keypad in various applications,
The easiest way to use it is using a matrix of switches.



Interfacing a matrix keypad

Keypads work as a matrix of buttons.

In the previous example:

The top three output pins are used as the columns of the matrix

A button is placed in the circuit at each intersection of a column and a row

One at a time, the output pins turn on and each of the input pins is read. If an input is detected, the microprocessor can determine which button on the keypad had been pressed.

Interfacing a matrix keypad

Code Snippet to read a matrix keypad : If the columns in Figure are connected to Port D bits 0, 1, and 2 respectively, and the rows are connected to Port E bits 0, 1, and 2 respectively; code to read the keypad values of the first column would be:

```
PORTD.PODR.BIT.B0 = 1; //Set the output of the first column to 1
PORTD.PODR.BIT.B1 = 0; //Set the output of the second column to 0
PORTD.PODR.BIT.B2 = 0; //Set the output of the third column to 0
if(PORTE.PIDR.BIT.B0 == 0){
 //Conditional code for Row 0
}

if(PORTE.PIDR.BIT.B0 == 0){
//Conditional code for Row 1
}

if(PORTE.PIDR.BIT.B0 == 0){
//Conditional code for Row 2
}
```

Interfacing with Analog Signals

Analog Signals must be converted to digital values so that the microcontrollers can process them.

Analog to Digital converters are used to convert analog signals to digital values and Digital to Analog converters are used to convert Digital signals to appropriate analog values.

ADC

ADC is a device that converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude.

The number of bits binary number helps determine the resolution or precision of the reading

Types of ADC:

Flash

Successive approximation ADC

Ramp

Resolution of an ADC

If V_{ref} = 5 V, and there is an 8-bit resolution, the volts per ADC raw value step = 5 V/256 = 0.01953 V/step.

To determine the value of the conversion, Formula 2.3 can be used:

$$\text{ADC output code} = \text{int}\left(\left(\left(2^{N\text{bits}} - 1\right) * \left(V_{in}/V_{ref}\right)\right) + 1/2\right) \qquad \textbf{Formula 2.3}$$

For example:

V_{ref} = 5 V, V_{in} = 2.26295 V, with a 10-bit resolution;
int ((1023 * (2.26295 V/5 V)) + 1/2) = 463 ADC raw value steps.

DAC

In systems that control audio, speed, or even light levels; digital output from a microprocessor must be converted into an analog signal.

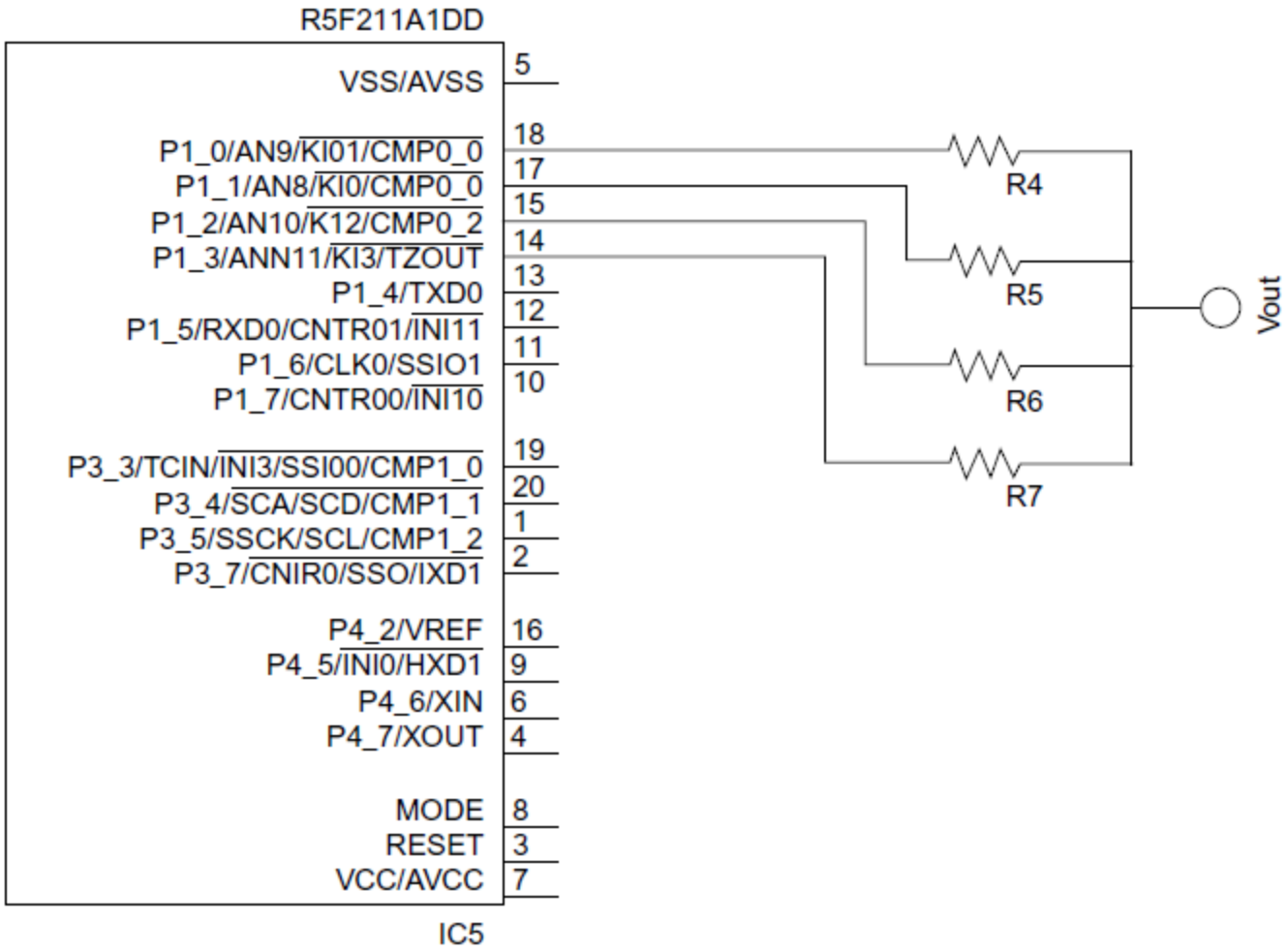
There are several different methods for converting digital signals to analog.

Resistor Networks

Binary Weighted DACs

R-2R DAC

Binary Weighted DAC



Binary Weighted DAC

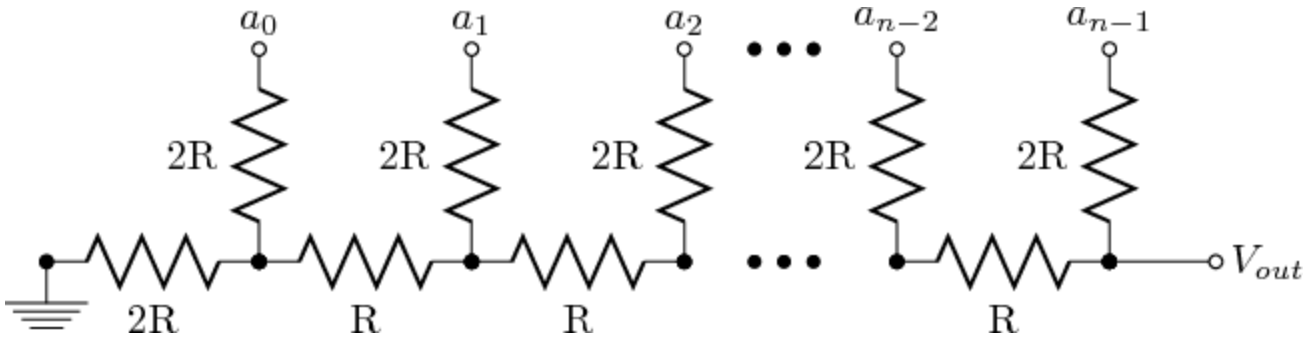
Resistor networks is a simple method for Digital to Analog Conversion.

In this design, resistors are connected together on one end (the analog output), and the other ends are connected directly to the pins of a microcontroller port.

The values of the resistors are based on binary bit place-values (R, 2R, 4R, 8R, etc.; with R being the least significant bit or LSB).

When the pins of the microcontroller are set to HIGH, then the sum voltage of the analog output are related to the value on that port

R-2R Ladder:



The R-2R network causes the digital bits to be weighted in their contribution to the output voltage V_{out}

For a digital value VAL, of a R-2R DAC of N bits of 0 V/V_{ref}, the output voltage V_{out} is:

$V_{out} = V_{ref} \times VAL / 2^N$

R-2R

Another method of using resistor networks to design a DAC is called the *R-2R* or *R/2-R Network.*

It consists of resistors of only two values. One value is twice the value of the other.

For instance, if the R resistors are one kilo-ohm, then the 2R resistors must equal two kiloohms.

Output voltage corresponds to the binary value on the port.

Reset Circuits

Resetting a microprocessor starts the processor off in a predictable state

When a reset is detected, the microprocessor loads predefined values into the system control registers.

All microprocessors designate a pin that is solely usedfor resetting the chip.

These registers are useful in case of a catastrophic event, such as if the power supply were to drop below a certain threshold.