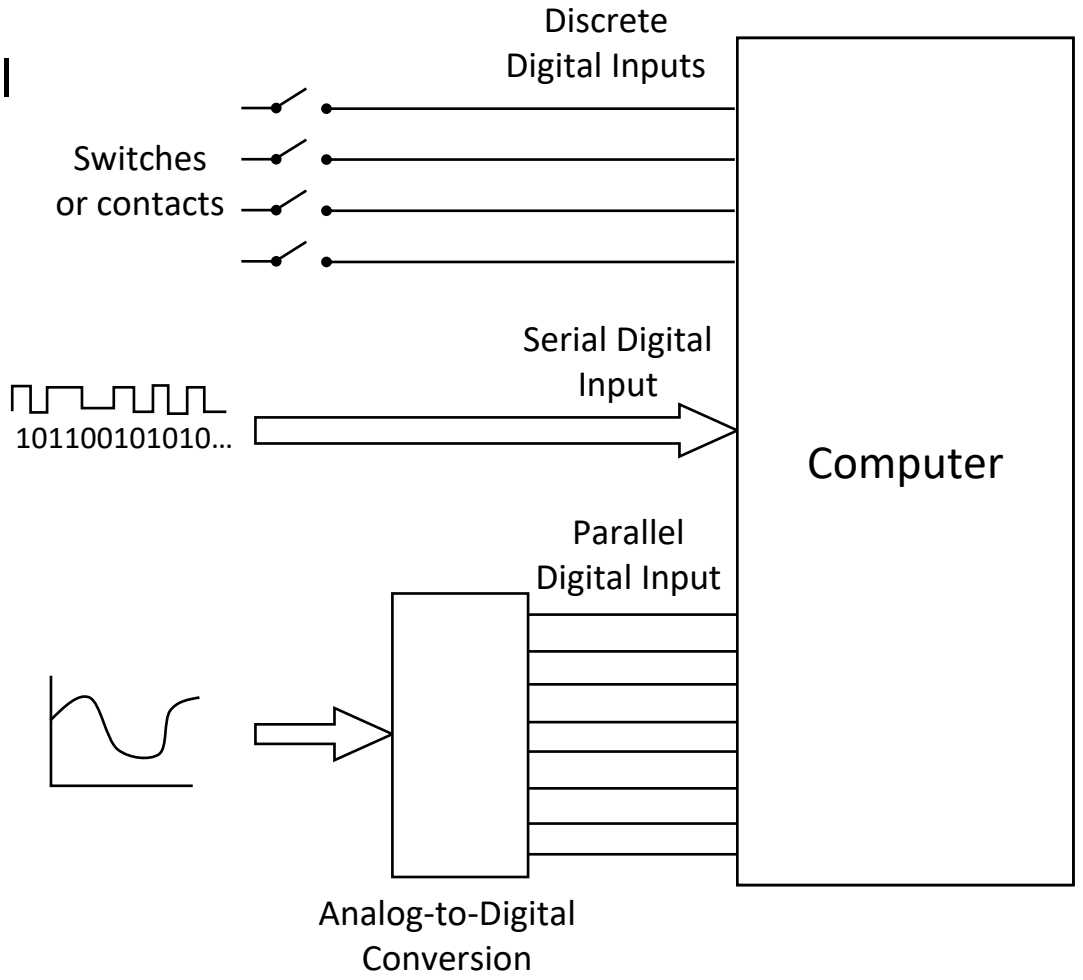
# 시스템 프로그래밍을 위한 C언어

## - Bit Manipulation to Control Hardware -

현대자동차 입문교육
박대진 교수

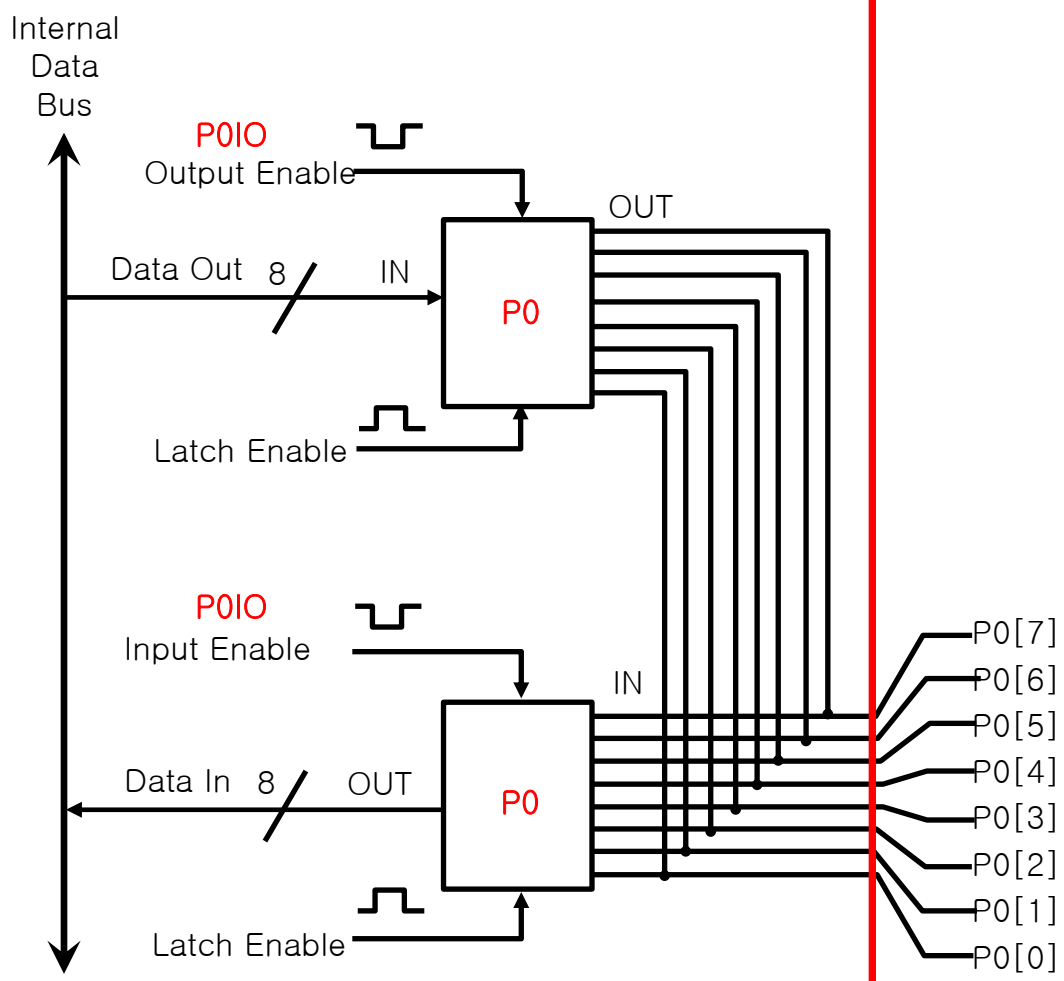
# Real World Interfacing by using MCU

- Analog or Digital
- Bit or Bus



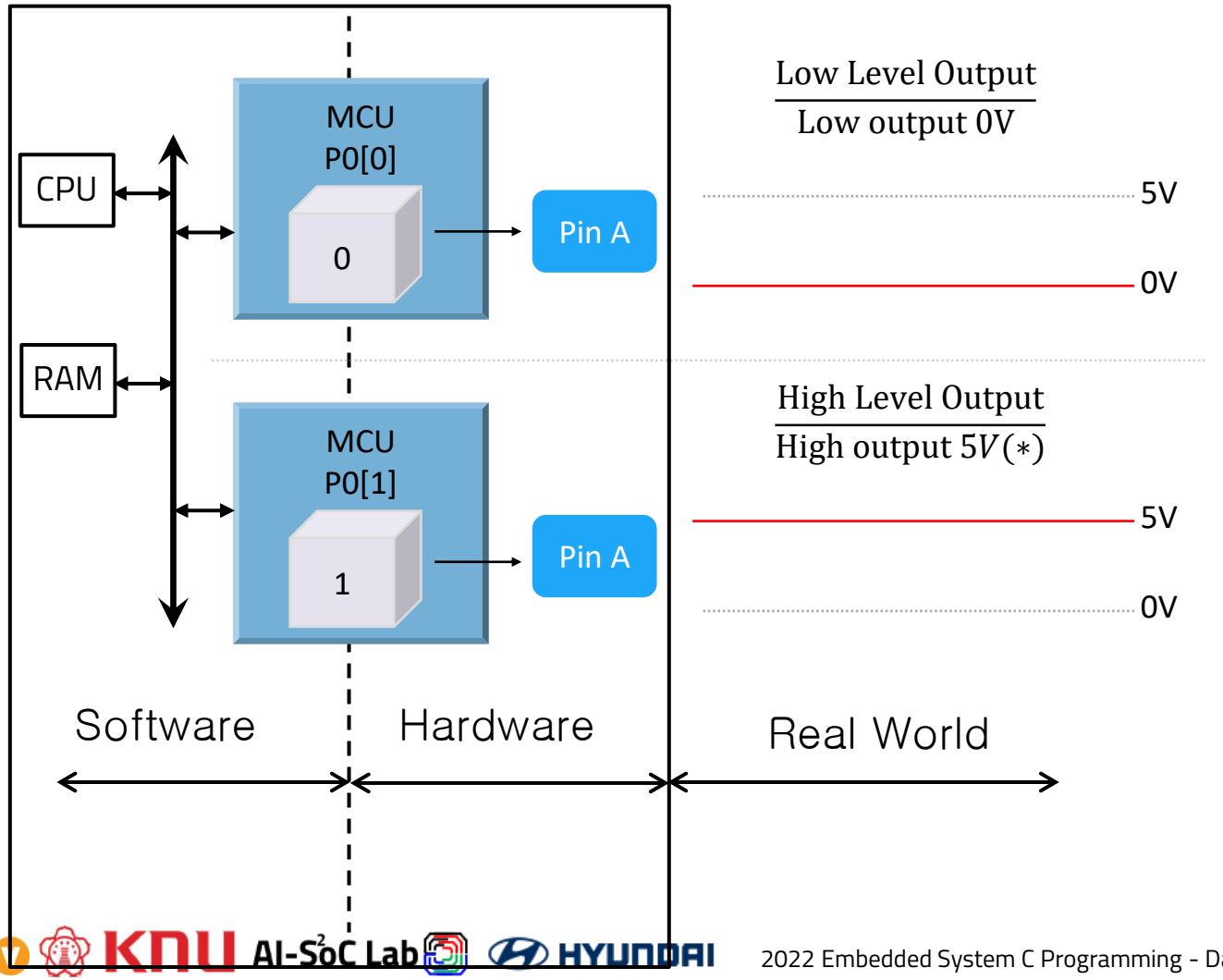
# GPIO and Internal Bus

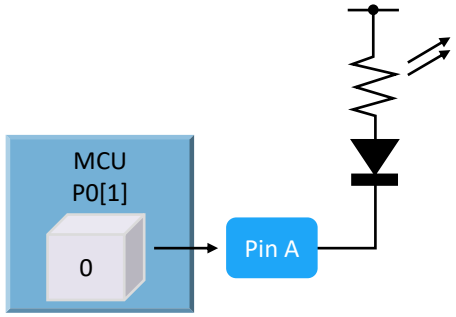
- General Purpose Input/Output
  - Interface Between Internal Bus and Outside world
  - Time-multiplexed Data Path (Input, output)
  - GPIO Port is mapped to registers in Memory Map



# Interfacing via PORT Register on Memory Map : Write Mode

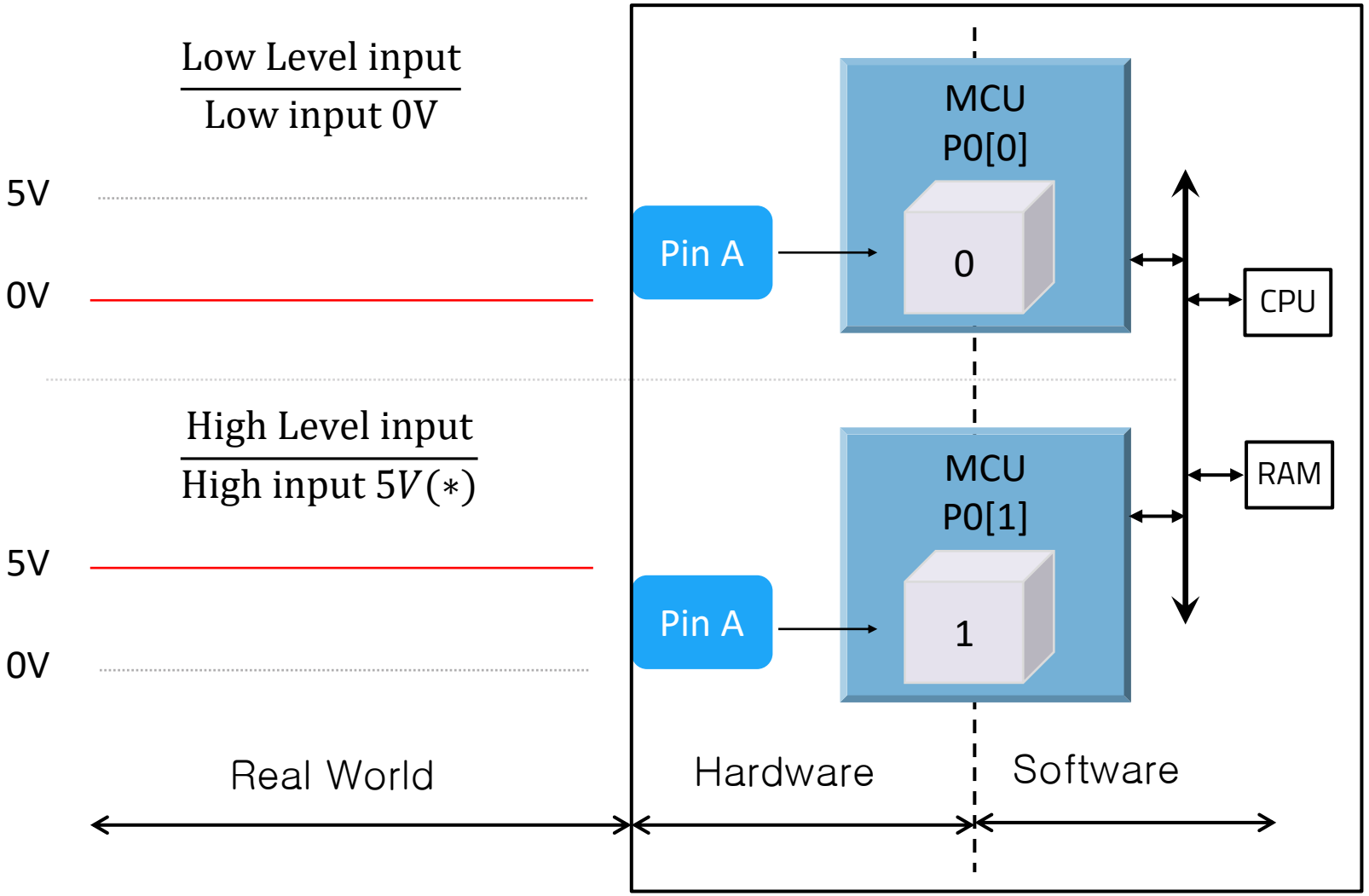
- Write value on Register → Control the output voltage



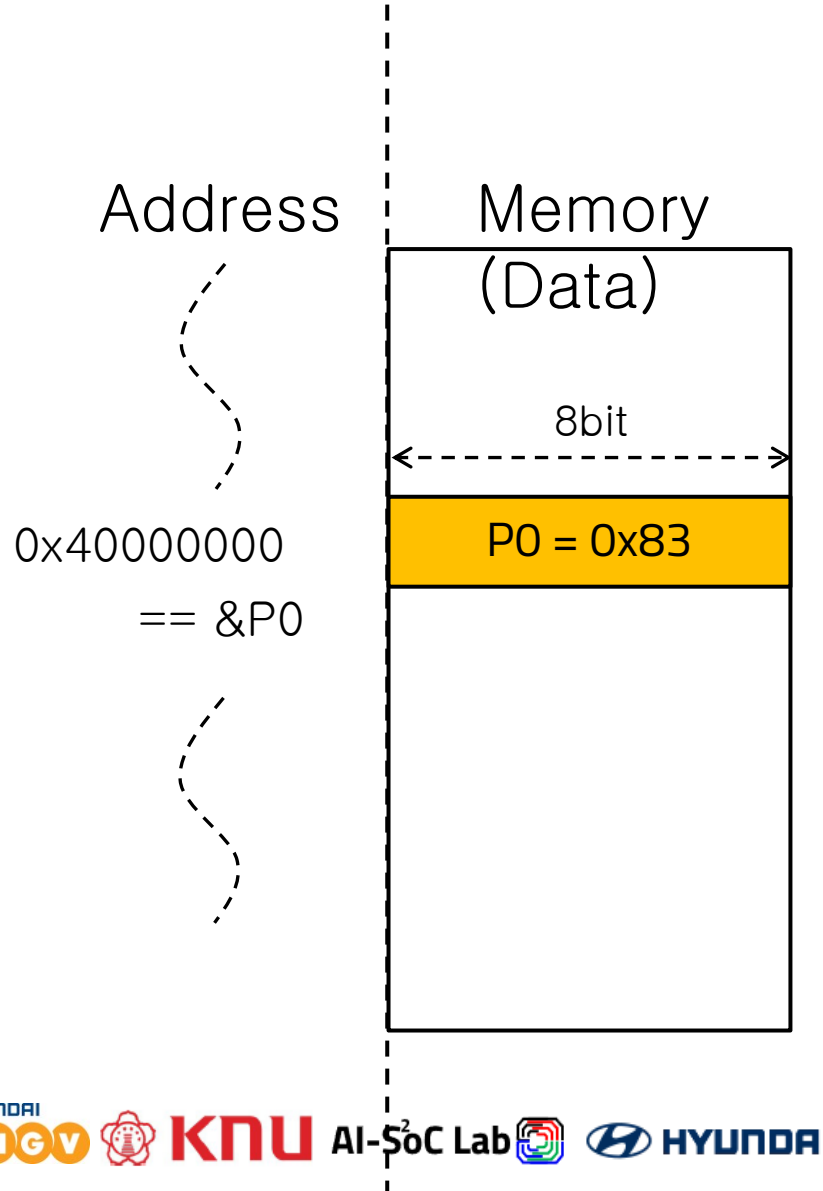



# Interfacing via PORT Register on Memory Map: Read Mode

- Reading Register Value → Can Identify the input voltage



# Pointer: Method to Access Registers in C Language




Region on Memory vs. its location

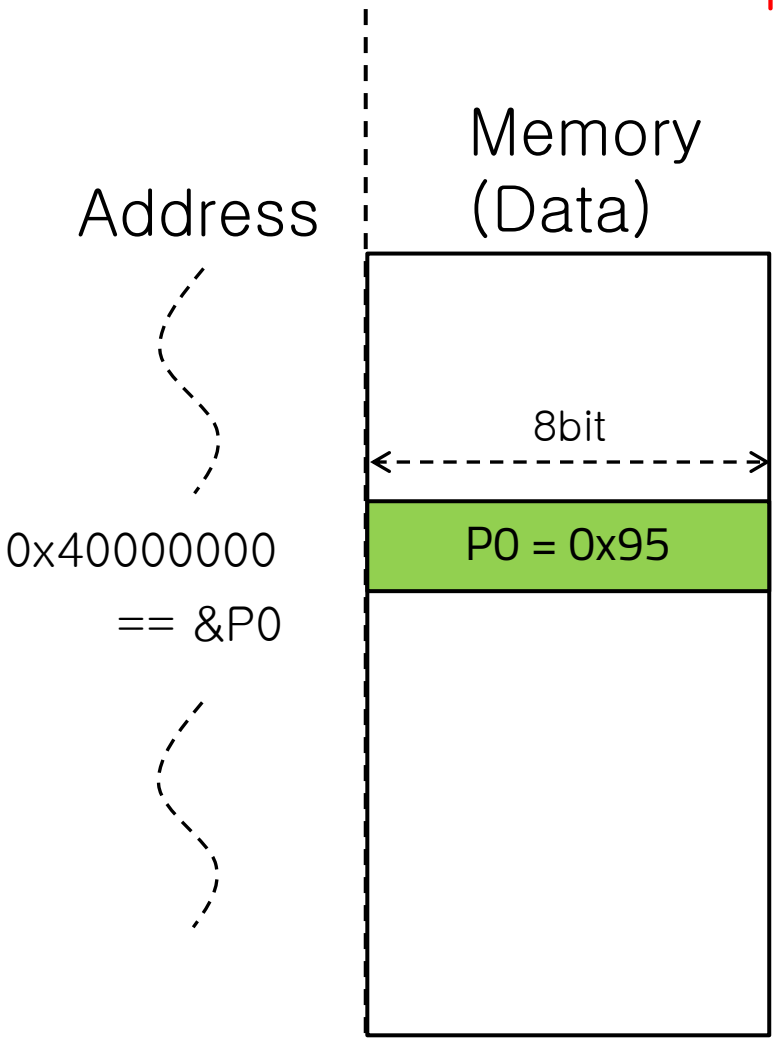
printf("%X", P0) ⟶ 0x83

printf("%X", &P0) ⟶ 0x40000000

Writing a value into memory region

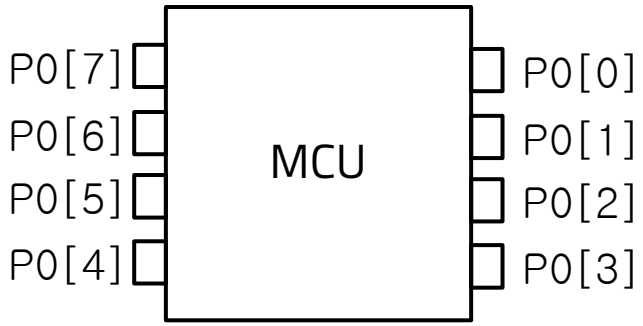
```
P0 = 0x83;

*(&P0) = 0x83;

(*(unsigned char*)0x40000000) = 0x83;
```

# Toggling Individual Bits in Register Data




P0 at 0x40000000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P0[7] | P0[6] | P0[5] | P0[4] | P0[3] | P0[2] | P0[1] | P0[0] |




Setting P0[7],P[4],P[2],P[0] with 1

```
#include “header.h”
#define P0 (*(volatile unsigned char*)0x40000000)
P0 = 0x95;
```

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

# Bit Manipulation for Bit-Clear

We want to clear 4th bit, still don't touch other bits

P0 = 0x95;

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

P0 = P0 & 0xEF;

Bit-by-bit 'and' logical operation

P0 → 1 0 0 1 0 1 0 1

& 0xEF → b1110_1111 → 1 1 1 0 1 1 1 1

→ 1 0 0 0 0 1 0 1

Three alternatives to clear specific bit of register data

- P0 = P0 & ~(0x10);
- P0 = P0 & ~(1<<4)
- P0 &= ~(1<<4)

# Bit Manipulation for Bit-Set

We want to set 3rd bit, still don't touch other bits

P0 = 0x85;

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

P0 = P0 | 0x08;

Bit-by-bit 'or' logical operation

P0

or

0x08 → b0000_1000

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

Three alternatives to set specific bit of register data

P0 = P0 | (0x08);

P0 = P0 | (1<<3)

P0 |= (1<<3)

## ^ (XOR)

The bitwise ^ operator returns 1 if both bits are equal and 0 if not.