ARDUINO GPIO PROGRAMMING

Ravi Suppiah, SoC NUS

LEARNING OBJECTIVES

- By the end of this lecture, you will be able to:
  - Understand the general architecture of a simple microcontroller
  - Read and Write to devices using the General Purpose I/O (GPIO) pins.

- Before that, we first need to get familiarized with the 1's and 0's…

BINARY

- In all digital systems, data is represented as 1's and 0's, namely called the Binary Number System.
- The 1's and 0's are logical representations.
- In actual hardware, they are still represented as voltage signals. Logic '0' is always 0 volts. The voltage of logic '1' depends on the microcontroller's supply voltage. If it is powered through 5V, then logic '1' is also 5V. If it powered through 3.3V, then logic '1' is also 3.3V.

BINARY REPRESENTATION

- Bit (*Bi*nary digi*t*)
  - 0 and 1
  - Represent *false* and *true* in logic
  - Represent the *low* and *high* states in electronic devices
- Other units
  - Byte: 8 bits
  - Nibble: 4 bits (seldom used)
  - Word: Multiples of byte (eg: 1 byte, 2 bytes, 4 bytes, 8 bytes, etc.), depending on the architecture of the computer system

BINARY REPRESENTATION

- *N* bits can represent up to 2^N values.
  - Examples:
    - 2 bits → represent up to 4 values (00, 01, 10, 11)
    - 3 bits → rep. up to 8 values (000, 001, 010, …, 110, 111)
    - 4 bits → rep. up to 16 values (0000, 0001, 0010, …., 1111)
- To represent *M* values, $\log_2 M$ bits are required.
  - Examples:
    - 32 values → requires 5 bits
    - 64 values → requires 6 bits
    - 1024 values → requires 10 bits
    - 40 values → how many bits?
    - 100 values → how many bits?

$$\lceil \log_2 M \rceil$$

WEIGHTED POSITIONAL NUMBER SYSTEM

- A **weighted-positional number** system
  - When the **Base** or **radix** is 10, it refers to the total number of symbols/digits allowed in the system
  - Symbols/digits = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
  - Position is important, as the value of each symbol/digit is dependent on its **type** and its **position** in the number
  - Example, the 9 in the two numbers below has different values:
    - $(7594)_{10} = (7 \times 10^3) + (5 \times 10^2) + (9 \times 10^1) + (4 \times 10^0)$
    - $(912)_{10} = (9 \times 10^2) + (1 \times 10^1) + (2 \times 10^0)$
- Same principal is applied to Binary Systems with Base-2.

BINARY-2-DECIMAL

- For binary number with n digits: $d_{n-1} \ldots d_3\ d_2\ d_1\ d_0$
- decimal = $d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$

- **Example**

  Find the decimal value of 111001_2:

  $111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 57_{10}$

HEXADECIMAL-2-BINARY

- **Hex**, or *hexadecimal*, is a number system of base 16.
- As hex system has 16 digits, the extra needed 6 digits are represented by the first 6 letters of English alphabet.
- Hence, **hex** digits are 0,1,2,3,4,5,6,7,8,9 together with A, B, C, D, E, F.
- To convert Hex to Bin, split it into groups of 4 starting from right.
- Hex to binary conversion examples
  - $(1E3)_{16} = (0001\ 1110\ 0011)_2$
  - $(0A2B)_{16} = (0000\ 1010\ 0010\ 1011)_2$
  - $(7E0C)_{16} = (0111\ 1110\ 0000\ 1100)_2$

BINARY-2-HEX

- For conversion from Binary to Hex, group the Binary bits into groups of 4 starting from the Least-Significant-Bit (LSB).
- Binary to hex conversion examples
  - $(1110)_2$ = 0Eh
  - $(111001)_2$ = 39h
  - $(10011100)_2$ = 9Ch

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

NAMING CONVENTIONS

- There are a variety of naming conventions for the different representations.
- Binary:
  - 1001_2
  - 1001b
  - 0b1001
- Hex:
  - $5A_{16}$
  - 5Ah
  - 0h5A
  - 0x5A

BITWISE OPERATORS IN C

- C has the following bitwise operators:
  - AND (&), OR(|) and NOT (~)
  - The table below shows how these work.

A	B	A&B	A|B	~A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

MASKING

- **The bit-wise operators are very useful for setting or clearing a particular bit.**
  - To clear a bit, create a "mask" where that bit is 0 and all others are 1, and do a bit-wise AND.
  - To set a bit, create a "mask" where that bit is 1 and all others are 0, and do a bit-wise OR.
- **We will consider a variable "value" set to 0b1101 1110 (0xDE).**

  We want to clear bit 3 and set bit 5.
  - Bits are numbered 0 to 7 from right to left

MASKING

- Clearing Bit 3

Value	1	1	0	1	1	1	1	0
	&	&	&	&	&	&	&	&
Mask	1	1	1	1	0	1	1	1
Result	1	1	0	1	0	1	1	0

Our mask here is 0b1111 0111 or 0xF7

- Setting Bit 5

Value	1	1	0	1	0	1	1	0
	|	|	|	|	|	|	|	|
Mask	0	0	1	0	0	0	0	0
Result	1	1	1	1	0	1	1	0

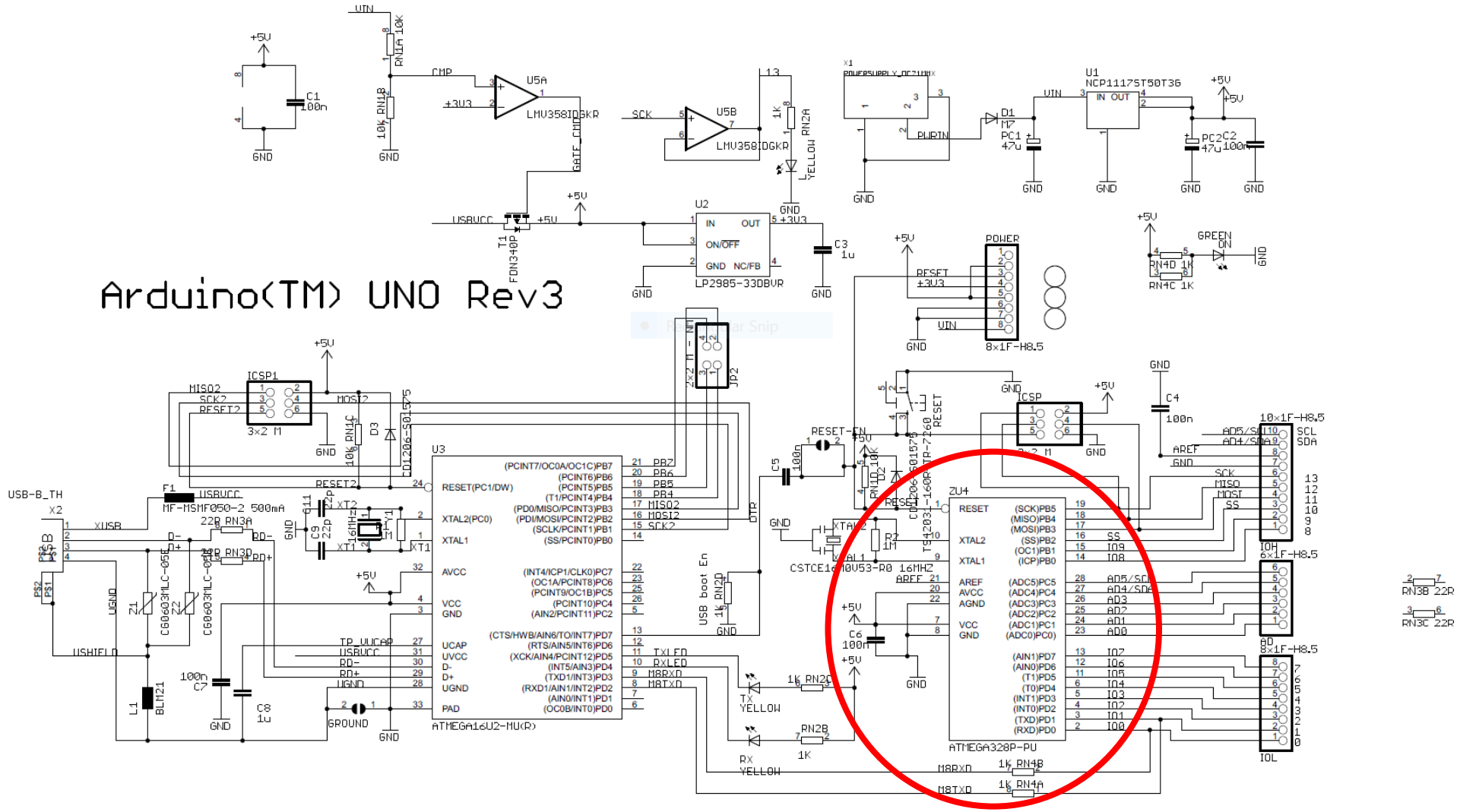
Our mask here is 0b00100000 or 0x20

MASKING

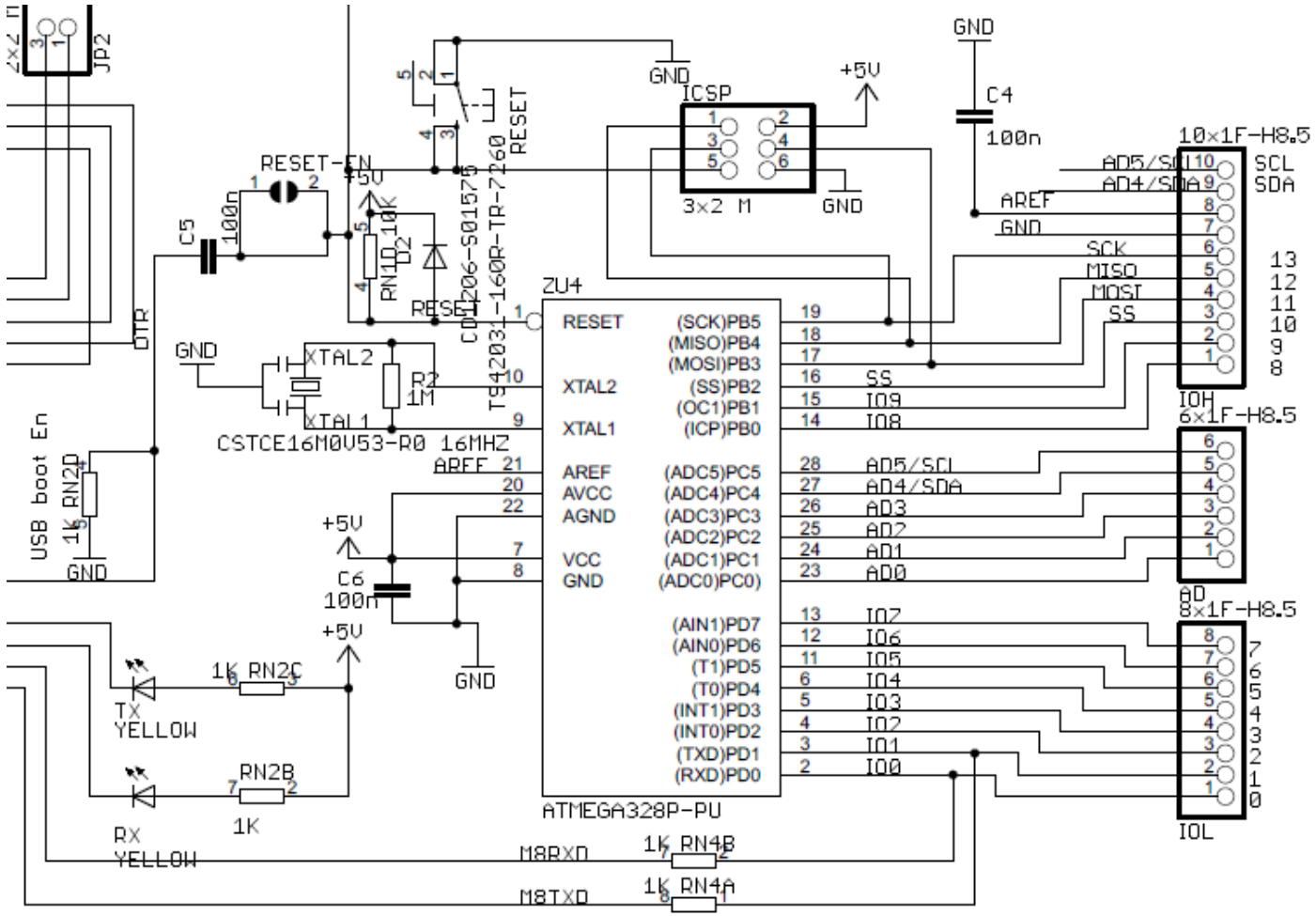
- The C Code to accomplish this is shown below.

```
int value=0xde;
...
// Clear bit 3
value &= 0xF7;

// Set bit 5
value |= 0x20;
```

ARDUINO UNO EMBEDDED SYSTEMS BOARD

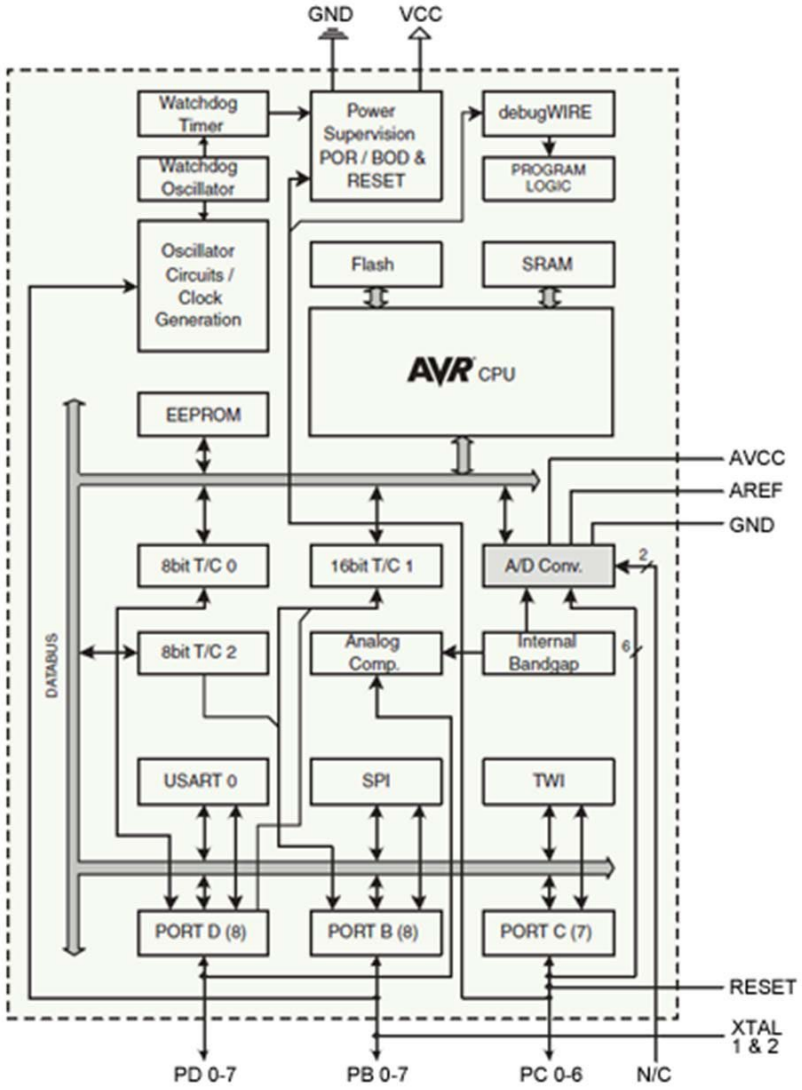


LETS TAKE A CLOSER LOOK

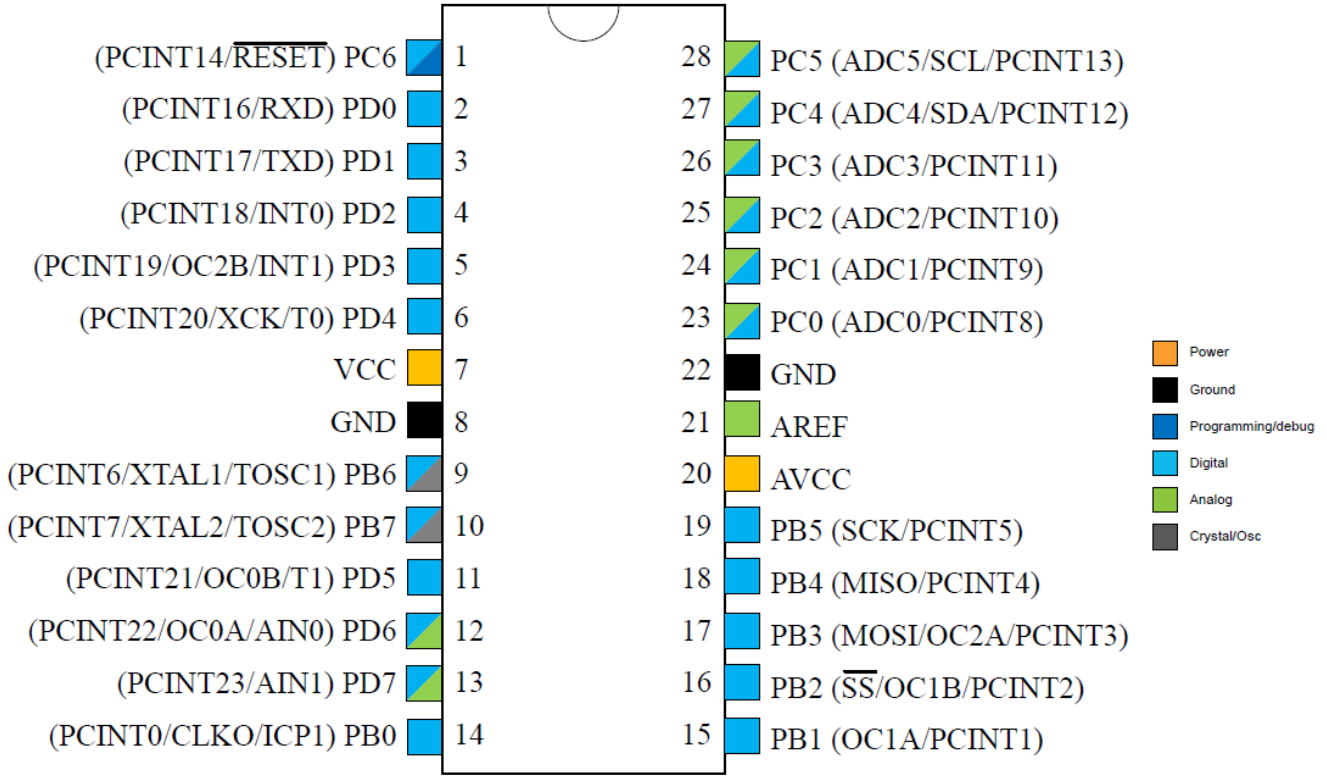


ATMEL ATMEGA ARCHITECTURE

- **Specifications:**
- 20 MHz 8-bit microcontroller
- 32 single-byte registers.
- I/O Options:
  - 23 general purpose I/O (GPIO) lines.
  - 6 analog-digital converter channels (analog in).
  - 3 PWM channels (analog out).
  - 2 8-bit counters and 1 16-bit counter.
  - USART (RS232-like interface)
  - Two-wire Interface (TWI).
  - Serial Peripheral Interface (SPI).

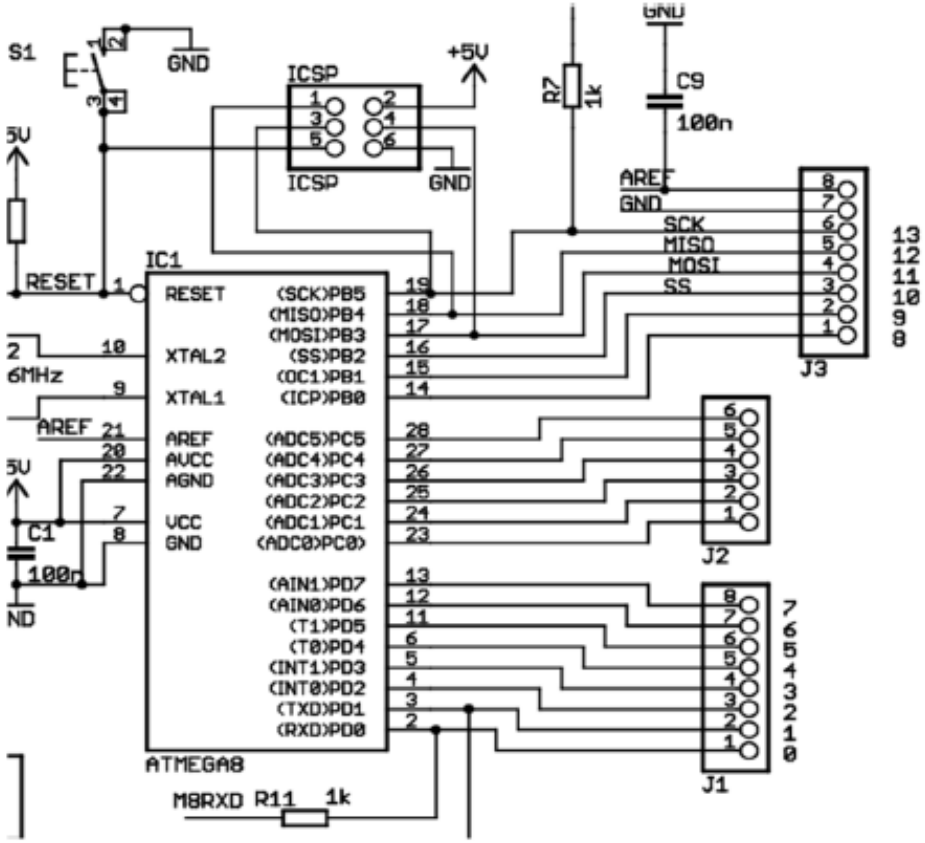


ATMEL AT328P



MAPPING

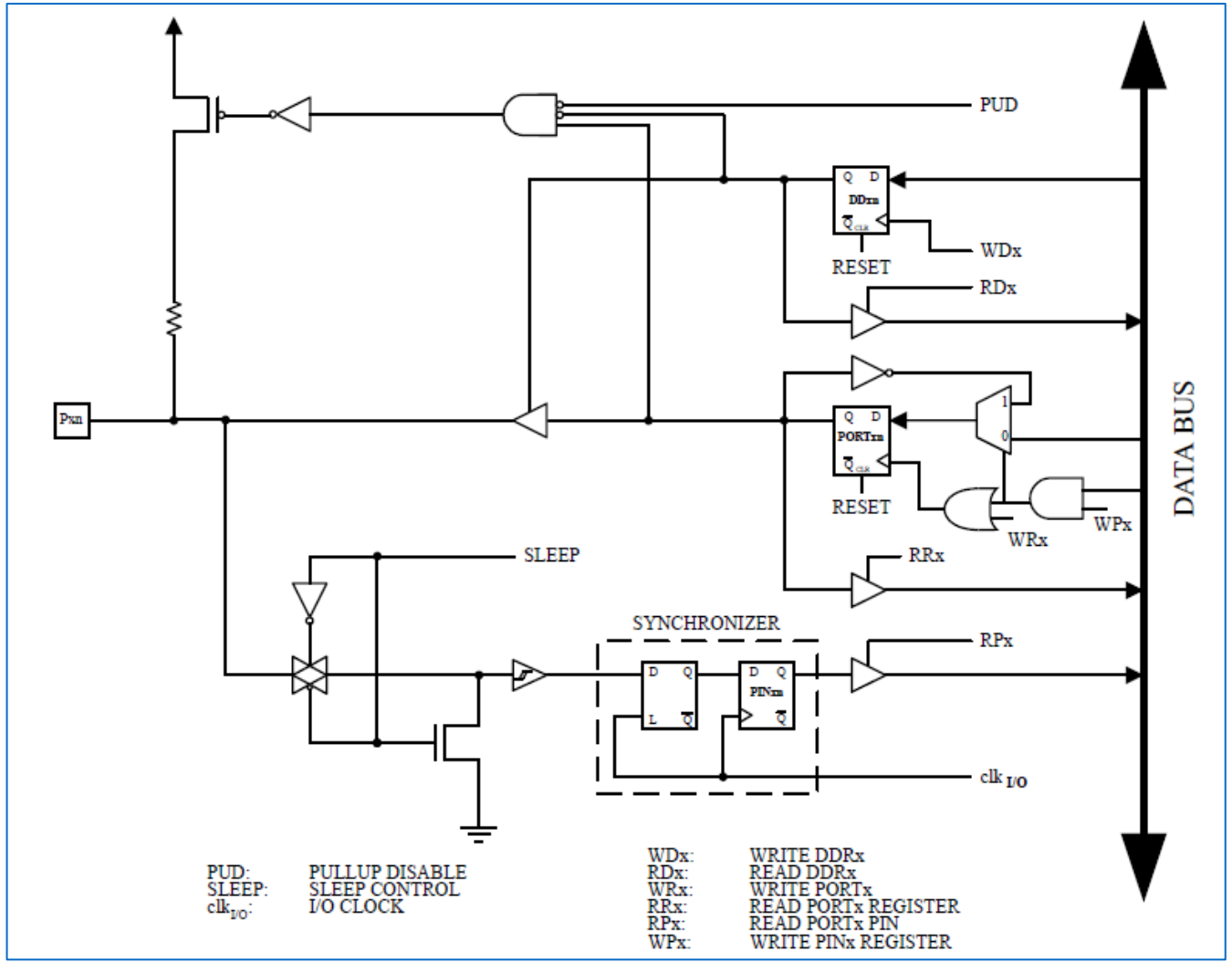
Arduino Pin	Atmega 328 port and pin number
0	Port D, pin 0
1	Port D, pin 1
2	Port D, pin 2
3	Port D, pin 3
4	Port D, pin 4
5	Port D, pin 5
6	Port D, pin 6
7	Port D, pin 7
8	Port B, pin 0
9	Port B, pin 1
10	Port B, pin 2
11	Port B, pin 3
12	Port B, pin 4
13	Port B, pin 5



GPIO PROGRAMMING

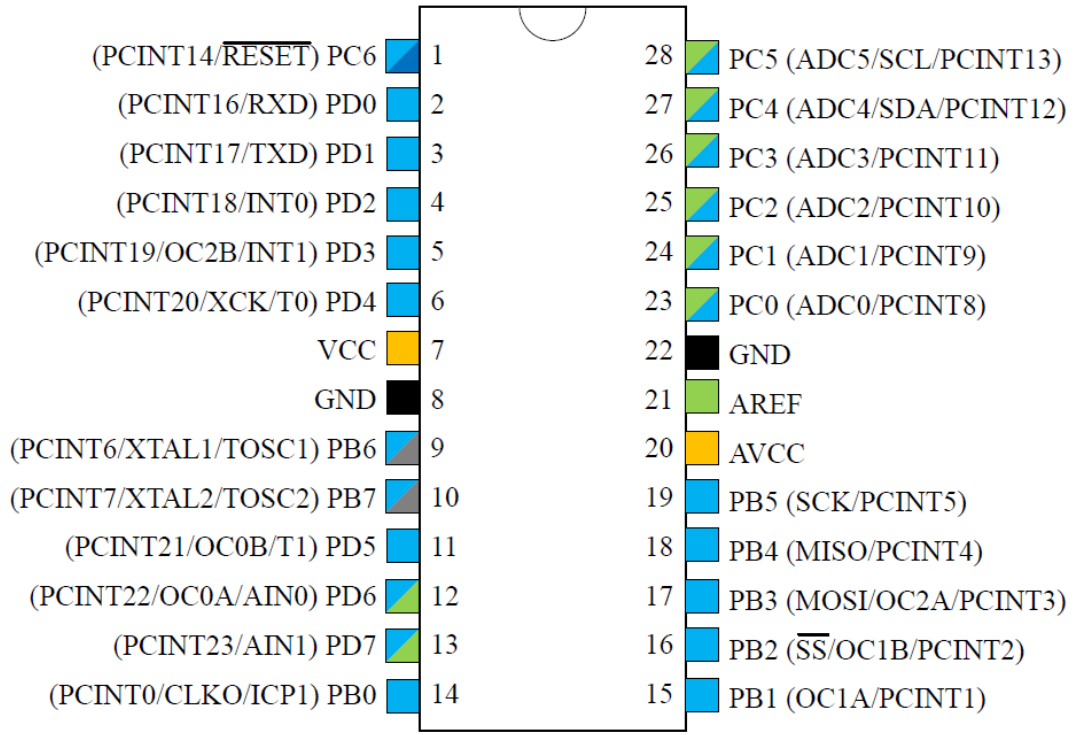
- **The AVR General Purpose Input Output (GPIO) ports are**
- **used for communicating with digital inputs and outputs:**
  - Switches.
  - LEDs.
  - Digital circuits.
  - Etc. etc.

THE INTERNALS OF A GPIO PIN



GPIO PORTS

- **There are three GPIO ports labeled**
- **PORTB, PORTC, and PORTD,**
- **corresponding to pins on the AVR.**
  - Pins are labeled PB0-7 (8 lines), PC0-6 (7 lines) and PD0-7 (8 lines), totaling 23 pins.
  - These pins are also shared with other functions:
- **E.g. PD0 and PD1 are also used by the receive (RXD) and transmit (TXD) lines for the USART.**



GPIO PROGRAMMING

- **To program a GPIO port, we must first set the direction of the individual pins.**

  This is done using the DDRx registers:

  - Example shown is for PORTB:

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- A "1" in bit 6 will set DB6 to be output, a "0" will set it to be input.

GPIO PROGRAMMING

- Once the pin directions have been configured, you can use the PORTx registers to write the pins.

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- If bit 3 in DDRB was set to 1, and if your program wrote a "1" to bit 3 in PORTB, then pin PB3 will be asserted HIGH.

GPIO PROGRAMMING

- **Use PINx to read the pins.**
  - Use bit-masking to test the status of a particular pin. E.g. to test pin PB3

```
result=PINB & 0b00001000;
```

- If PB3 contains a "1"

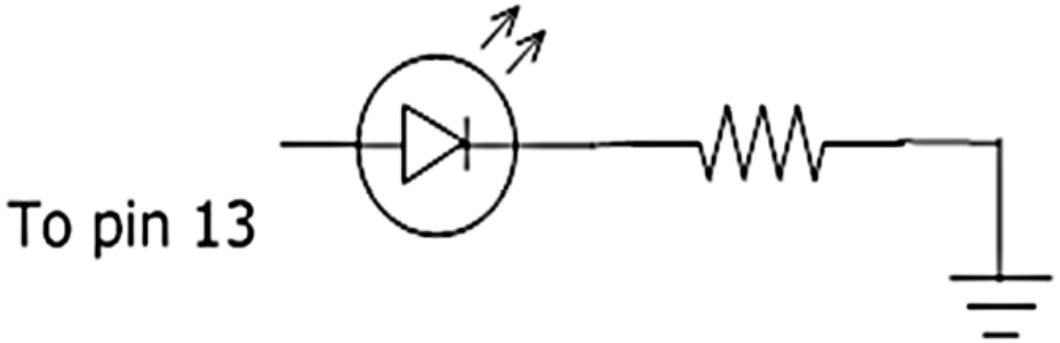
Pin	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Value	1	1	0	1	1	1	0	0
Mask	0	0	0	0	1	0	0	0
Result	0	0	0	0	1	0	0	0

- If PB3 contains a "0"

Pin	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Value	1	1	0	1	0	1	0	0
Mask	0	0	0	0	1	0	0	0
Result	0	0	0	0	0	0	0	0

GPIO PROGRAMMING EXAMPLE 1

- In this example, we will assume that an LED has been connected to Arduino Pin 13 based on the following circuit.



PROGRAMMING BLINKY

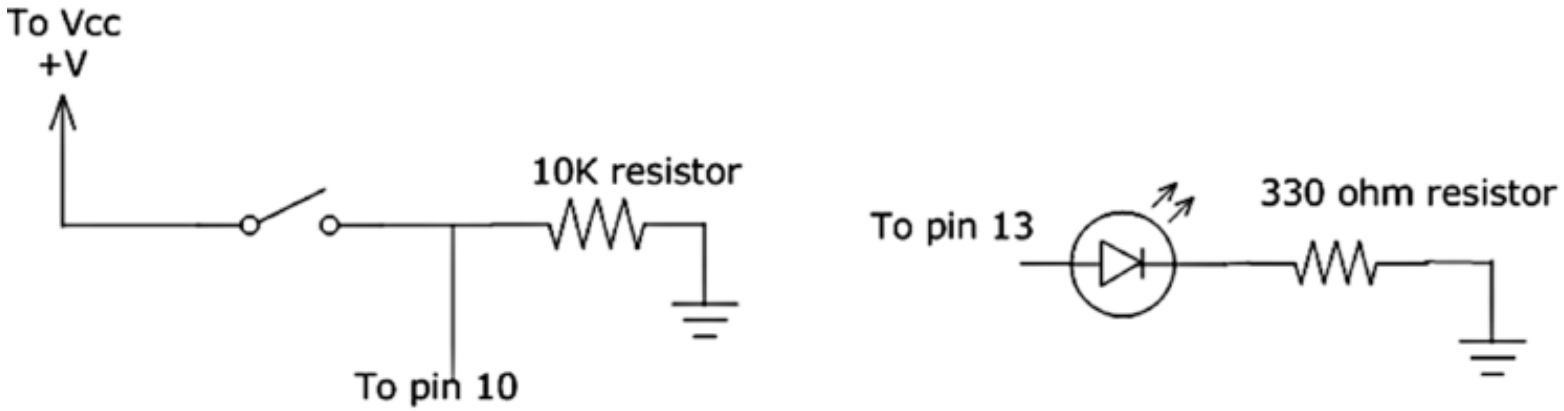
```
#include "Arduino.h"

void setup() {
  // setting DDRB as output
  DDRB = B00110000; // Set PB5 & PB4 as output

}

void loop() {
  // toggle both LED's one at a time
  PORTB = B00100000; //Green LED: ON, Red LED: OFF
  delay(1000);
  PORTB = B00010000; //Green LED: OFF, Red LED: ON
  delay(1000);

}
```

PROGRAMMING BLINKY

```
#include "Arduino.h"

#define PIN5  (1 << 5)
#define PIN4  (1 << 4)

void setup() {
  // setting DDRB as output
  DDRB = ((PIN5) | (PIN4));

}
void loop() {
  // toggle both LED's one at a time
  PORTB = PIN5; //Green LED: ON, Red LED: OFF
  delay(1000);
  PORTB = PIN4; //Green LED: OFF, Red LED: ON
  delay(1000);

}
```

GPIO PROGRAMMING EXAMPLE 2

- Same program as before, except that this time we toggle the LED on and off using a push button switch connected to pin 10 using this circuit:



PROGRAMMING SWITCH & LED

```
#include "Arduino.h"

#define PIN5  (1 << 5)
#define PIN2  (1 << 2)

void setup() {
  // setting PIN5 as output and PIN2 as input
  DDRB = ((PIN5) & ~(PIN2));

}
void loop() {
    if(PINB & PIN2)
        PORTB |= PIN5;
    else
        PORTB &= ~PIN5;
}
```

SUMMARY

- In this set of slides you have
  - Been introduced to Number Systems and Conversions
  - Understood some basic Microcontroller Concepts
  - Learnt how to program GPIO's at the register level

- The process of extracting useful information from the datasheet is critical when you want to start developing your own projects using any Microcontroller.