Mechatronics Engineering

Tutorial #2

Switching Devices, AVR Digital I/O Ports and Timers

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

- Important Switching Devices
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- Digital Input-Output Ports in ATmega328P
- Code example on interfacing
- Timers in Atmega328P
- Code example utilizing TCNT0 as a counter

Tutorial Contents

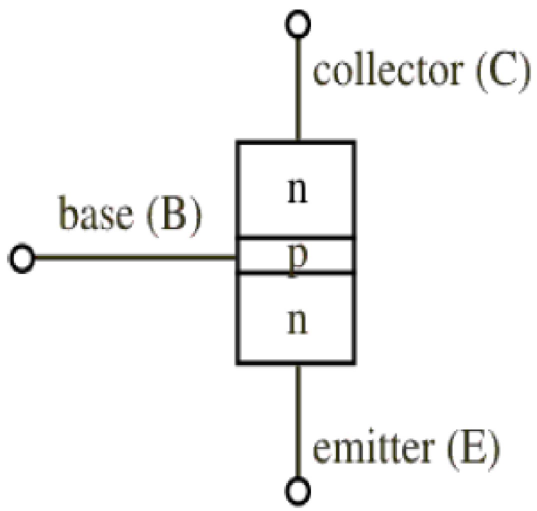
- **Important Switching Devices**
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- Digital Input-Output Ports in ATmega328P
- Code example on interfacing
- Timers in Atmega328P
- Code example utilizing TCNT0 as a counter

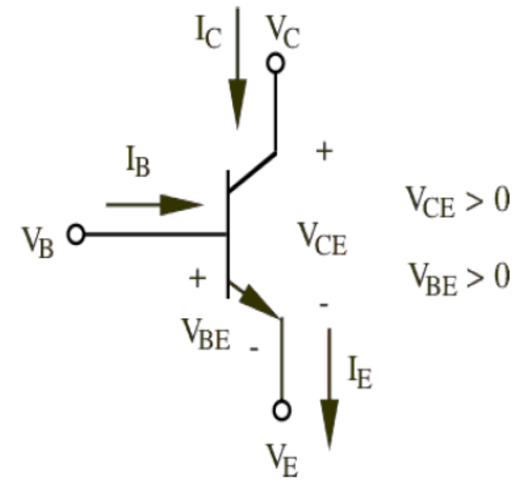
BJT (Bipolar Junction Transistor)

- A semiconductor device that is commonly used to amplify or **switch** electronic signals.

- Can be used as a switch to drive a load using a microcontroller.

Internal Structure and Symbol of **npn BJT**:





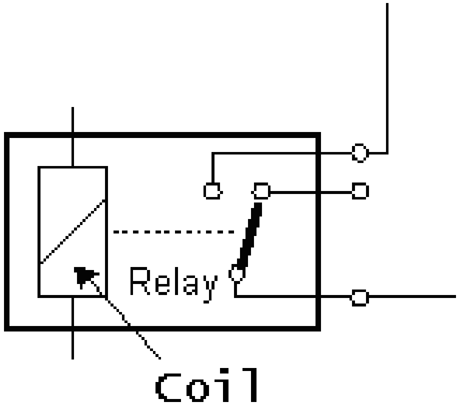
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Relay

- The relay is an electromechanical device, which transforms an electrical signal into mechanical movement.

- Main Components of a Relay:

  - ❖ A coil of insulated wire on a metal core
  - ❖ A metal armature with one or more contacts.



Reed Relay

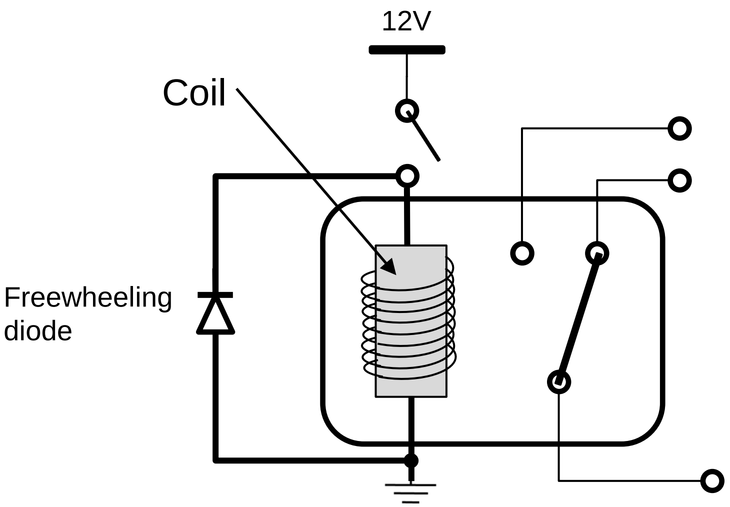
Relay

- Operation of a Relay:
  - ❖ When a supply voltage was delivered to the coil, current would flow and a magnetic field would be produced.
  - ❖ This moves the armature to close one set of contacts and/or open another set.
  - ❖ When the supply is removed and the magnetic field is weakened, the armature returns to its initial position.

•If the coil is charged and the supply is cut suddenly, the magnetic flux in the coil collapses and produces a fairly high voltage in the opposite direction.

$$V = l\frac{di}{dt}$$

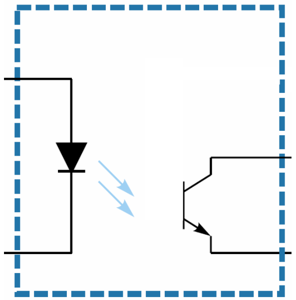
•A Free-wheeling diode should be used to allow the coil to discharge gradually.

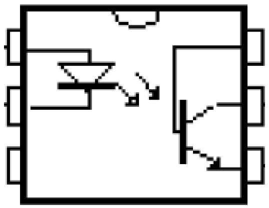


Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Optocoupler

Optocoupler combines a LED and photo-transistor in the same case. The purpose of an optocoupler is to separate two parts of a circuit.





CNY17

Optocouplers can be used as input or output device. They can have additional functions such as Schmitt triggering (the output of a Schmitt trigger is either 0 or 1; it changes slow rising and falling waveforms into definite low or high values). Optocouplers are packaged as a single unit or in groups of two or more in one housing.

They are also called PHOTO INTERRUPTERS where a spoked wheel is inserted in a slot between the LED and phototransistor and each time the light is interrupted, the transistor produces a pulse. Each optocoupler needs two supplies in order to function. They can be used with one supply, but the voltage isolation feature is lost.

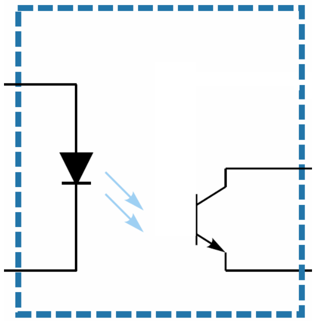
This is done for a number of reasons:

1. **Prevent Interference.**

- One part of a circuit may be in a location where it picks up a lot of interference (such as from electric motors, welding equipment, petrol motors etc.)

- If the output of this circuit goes through an optocoupler to another circuit, only the intended signals will be able to activate the LED.

- Noise signals are not strong enough to active the LED and thus are eliminated.

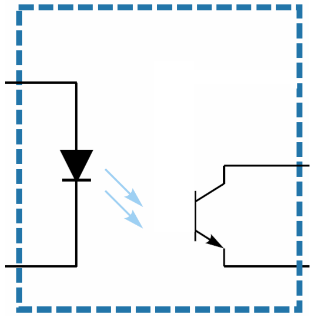


This is done for a number of reasons:

2. Simultaneous separation and intensification of a signal.

- A signal as low as 3v is able to activate an optocoupler (LED) and the output of the optocoupler (phototransistor) can be connected to a different voltage level.
- For example, If a microcontroller requires an input swing of 5v and in this case the 3v signal is amplified to 5v.
- It can also be used to amplify the current of a signal.

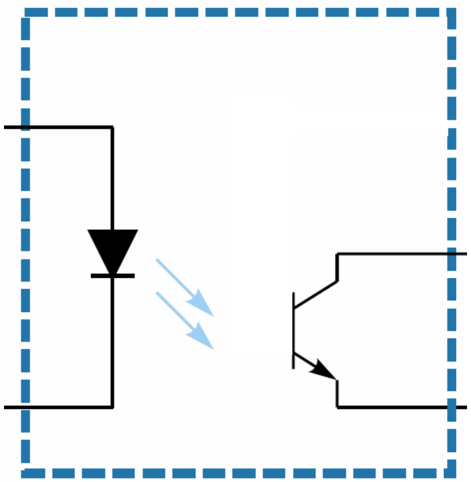
3. High Voltage Separation. Since the LED is completely separate from the photo-transistor, optocouplers can exhibit voltage isolation of 3kV or higher.



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Problem 1:

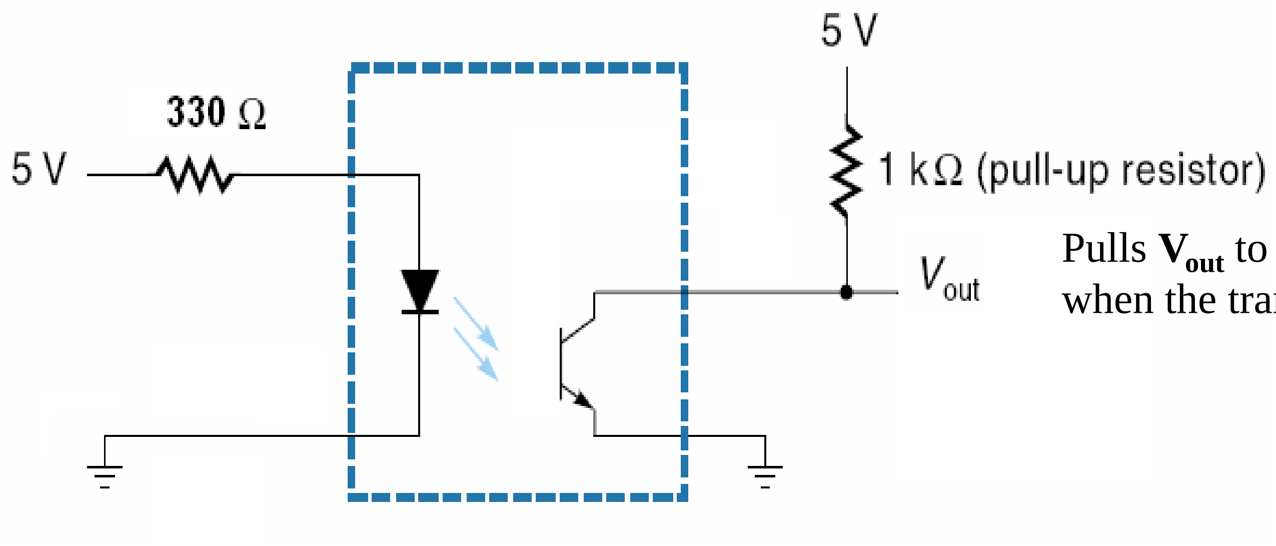
A photo-interrupter comes in a manufactured package that includes the phototransistor and corresponding LED as shown in the following schematic. What external circuitry must be added to obtain a functioning photo-interrupter? Sketch the resulting schematic.



Here, when the LED is **ON** the phototransistor becomes **ON** and allows current to flow

Solution

A voltage source (e.g., 5V) and current limiting series resistor (e.g., 330 Ω) is required on the LED side. On the phototransistor side, a pull-up resistor (e.g, 1k) and a voltage source (e.g., 5V) is required on the collector lead and ground is required on the emitter lead.



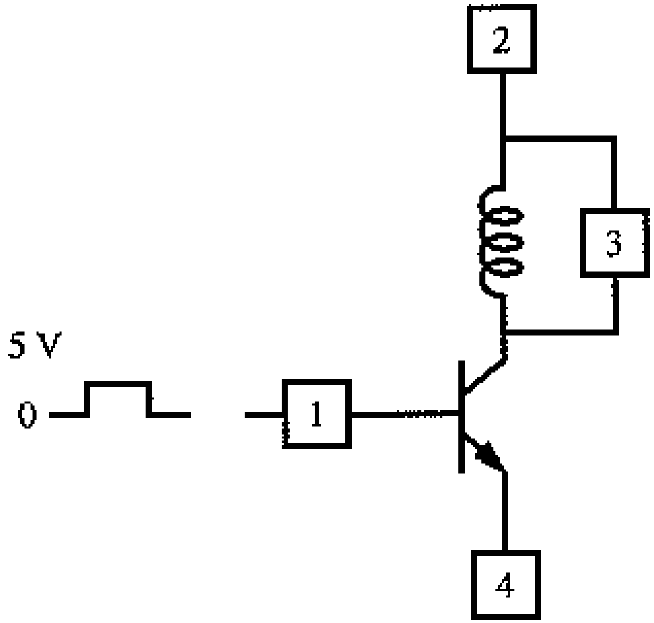
Pulls $\mathbf{V_{out}}$ to 5V (logic High) when the transistor is **OFF**

Tutorial Contents

- Important Switching Devices
  - BJT
  - Relay
  - Optocoupler
- **Interfacing an inductive load**
- Digital Input-Output Ports in ATmega328P
- Code example on interfacing
- Timers in Atmega328P
- Code example utilizing TCNT0 as a counter

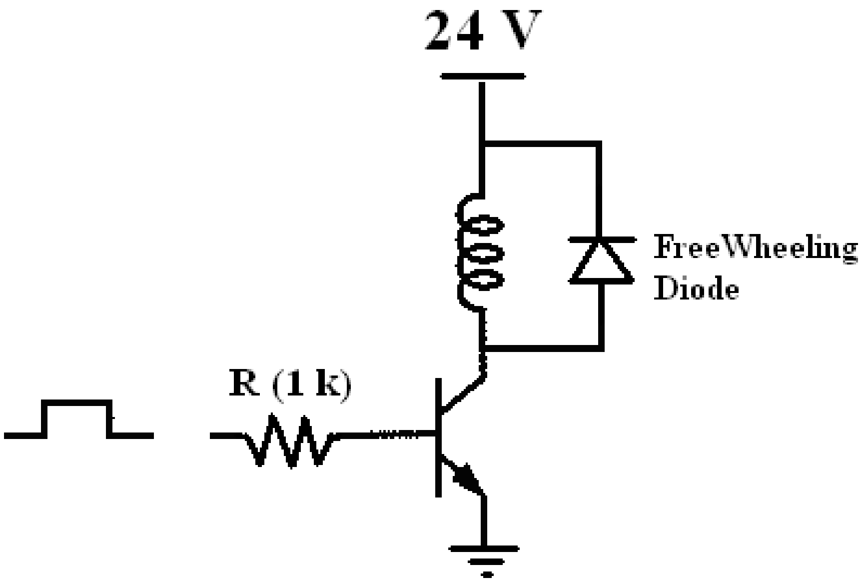
Problem 2:

Consider the design of a solid-state switch using an NPN power transistor that you plan to control with a digital signal (0 V=OFF, 5 V=ON). Start with the following schematic where components that you must select are labeled with numbers. The inductor represents a DC motor that requires 1 A of current at 24 V DC. Replace each of the labeled boxes shown in the figure with the appropriate schematic symbol and then specify the component as completely as you can.



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Solution:



N.B. When the voltage across an inductive load (e.g. DC motor) is reduced or removed a sudden spike voltage is seen across the load.

1. A resistor (e.g., 1k) to limit the base current while ensuring the transistor is in full saturation
2. 24 V DC capable of at least 1A of current
3. Power diode capable of carrying at least 1A for flyback protection (free wheeling diode) used whenever inductive loads are switched off by silicon components
4. Ground

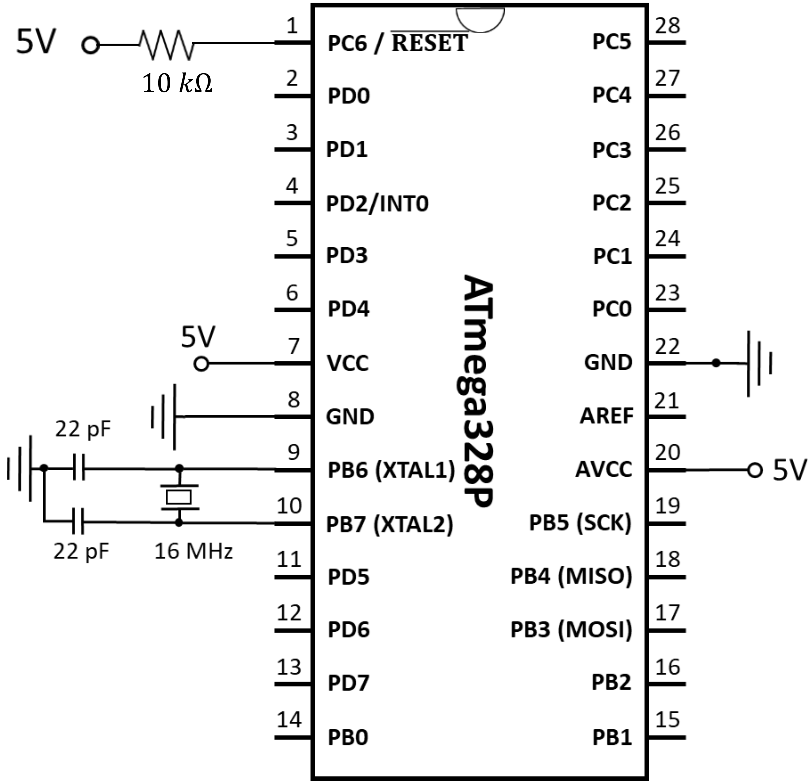
Tutorial Contents

- Important Switching Devices
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- **Digital Input-Output Ports in ATmega328P**
- Code example on interfacing
- Timers in Atmega328P
- Code example utilizing TCNT0 as a counter

GUC
German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Input-Output Configuration Registers in ATmega328P

- General connection of an ATmega328P:



- It is optional to use the external crystal oscillator, there is an 8 MHz internal oscillator that can be used.

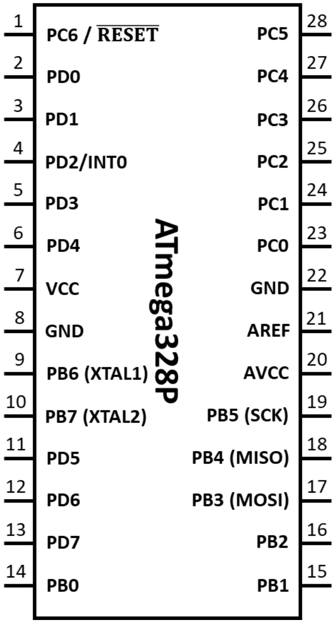
Input-Output Configuration Registers in ATmega328P

ATmega328P has 3 configurable bi-directional input-output ports. They have internal pull-up resistors which can be configured for each bit.

- PORTB (PB7:0)
- PORTC (PC6:0)
- PORTD (PD7:0)

Each Input-Output port has 3 registers associated with it in the I/O section of the data memory. They are designated as:

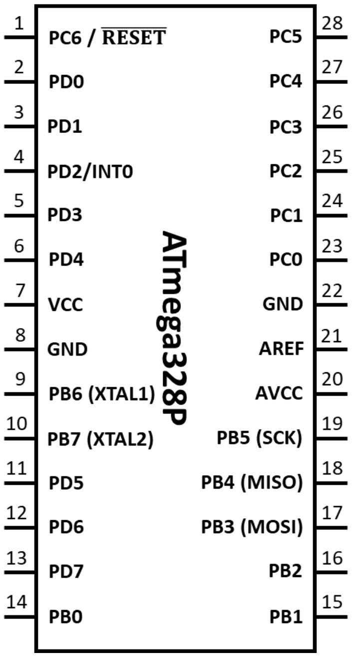
- PORTx (PORTx Data Register)
- DDRx (PORTx Data Direction Register)
- PINx (PORTx Input Pins Register)



GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Input-Output Configuration Registers in ATmega328P

- For example, for PORTD:



PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

DDRDx	PORTDx	Configuration
0	0	Input
0	1	Input with internal pull-up resistor enabled
1	0	Output (LOW)
1	1	Output (HIGH)

If pin is configured as input, PINx register holds the digital input value.

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

C Programming for the AVR MCU

- Some of the useful data types used by C Compilers:

- The C Standard Specifies the minimum size of each datatype as in table.

- The ATmega328p is an 8-bit MCU. Each register is 8-bits wide.

- Choice of datatype can affect the efficiency of the program. E.g., An int will take atleast 2 registers in the RAM while a char will take only one register.

Data Type	Size in Bits	Data Range
unsigned char	8	0 to 255
char	8	-128 to 127
unsigned int	16	0 to 65,535
Int	16	-32,768 to 32,767
unsigned long	32	0 to 4,294,967,295
long	32	-2,147,483,648 to 2,147,483,647
float	32	$\pm 1.175e^{-38}$ to $\pm 3.402e^{38}$

- For example, why use int for a variable day_of_month that varies from 1-31 ? A char variable will be sufficient.

C Programming for the AVR MCU

- Not all C compilers provide access to a single bit in a given register.
- To ensure compatibility, it is recommended to access specific bits using the **AND** and the **OR** bit-wise operations.
- Some useful bit-wise operations:
  - Given a bit in state $\bar{X}$, these truth tables hold.
  - OR can be used to set certain bits in a register without altering the remainder of the register.
  - AND can be used to clear certain bits in a register without altering the remainder of the register.

Bit-wise OR (|) Operation			
Bit	**Operand 2**	**New Value**	**Result**
$\bar{X}$	0	$\bar{X}$	No change
$\bar{X}$	1	1	Set

Bit-wise AND (&) Operation			
Bit	**Operand 2**	**New Value**	**Result**
$\bar{X}$	0	0	Clear
$\bar{X}$	1	$\bar{X}$	No change

Bit-wise XOR (^) Operation			
Bit	**Operand 2**	**New Value**	**Result**
$\bar{X}$	0	$\bar{X}$	No change
$\bar{X}$	1	$\sim\bar{X}$	Inverse

- **Some useful operations:**

- Example of bit-wise AND (&) operation to clear two specific bits in a register:

```
// Clear bit No. 1 and 3 in register PORTB.
PORTB &= 0b11110101; // Equivalent to PORTB = PORTB & 0b11110101;
```

- Bit-wise AND (&) operation can be used to extract a specific bit:

```
// Check if bit No. 1 in port b has a logic high level
if( PINB & 0b00000010);
```

- Example of bit-wise OR operation (|) to set specific bits in register PORTB:

```
// Set bit No. 4 and 5 in register PORTB.
PORTB |= 0b00110000; // Equivalent to PORTB = PORTB | 0b00110000;
```

- Example of bit-wise XOR operation (^) to toggle bit No. 1 in register PORTB:

```
// Invert bit No. 1 in register PORTB.
PORTB ^= 0b00000010; // Equivalent to PORTB = PORTB ^ 0b00000010;
```

- Bit-wise Shift Operations:

  - Shift left: (data) << (No. of shifts to the left)
  - Example:

```
Data = 0b00000001 << 3; // Data contains ‘0b00001000’ now;
Data = 1 << 3; // More compact equivalent to first line
```

  - Shift right: (data) >> (No. of shifts to the right)
  - Example:

```
Data = 0b01010000 >> 2; // Data contains ‘0b00010100’ now;
```

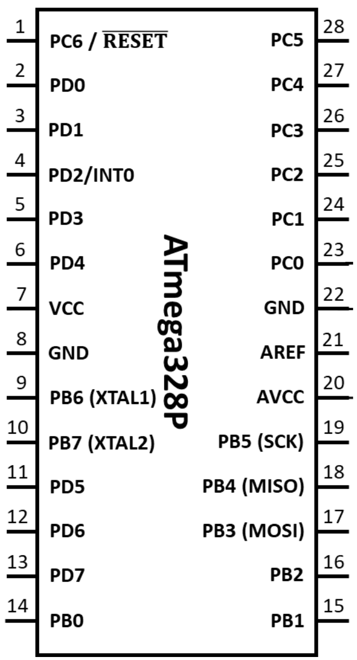
Tutorial Contents

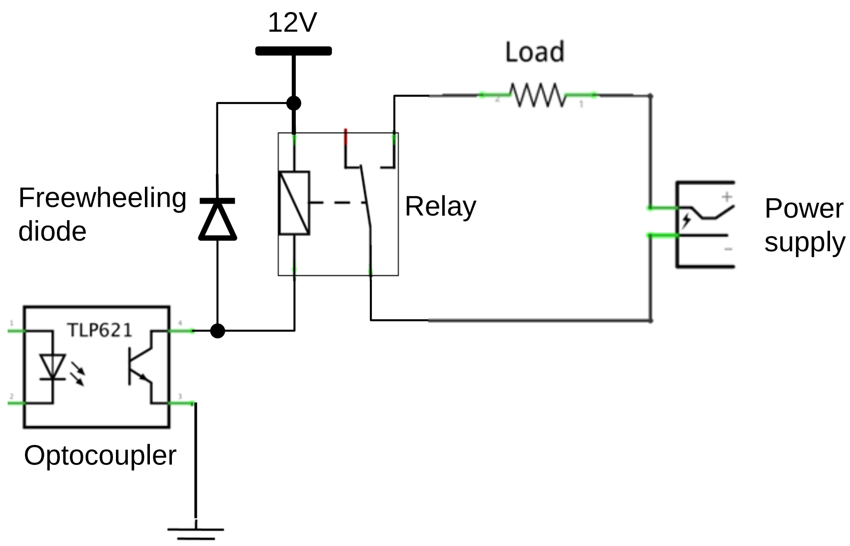
- Important Switching Devices
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- Digital Input-Output Ports in ATmega328P
- **Code example on interfacing**
- Timers in Atmega328P
- Code example utilizing TCNT0 as a counter

Problem 3:

Required: Write C code for the ATmega328P to toggle a relay when a button connected to PD0 is pressed then released (on falling edge of signal). You must perform switch debouncing in your software.

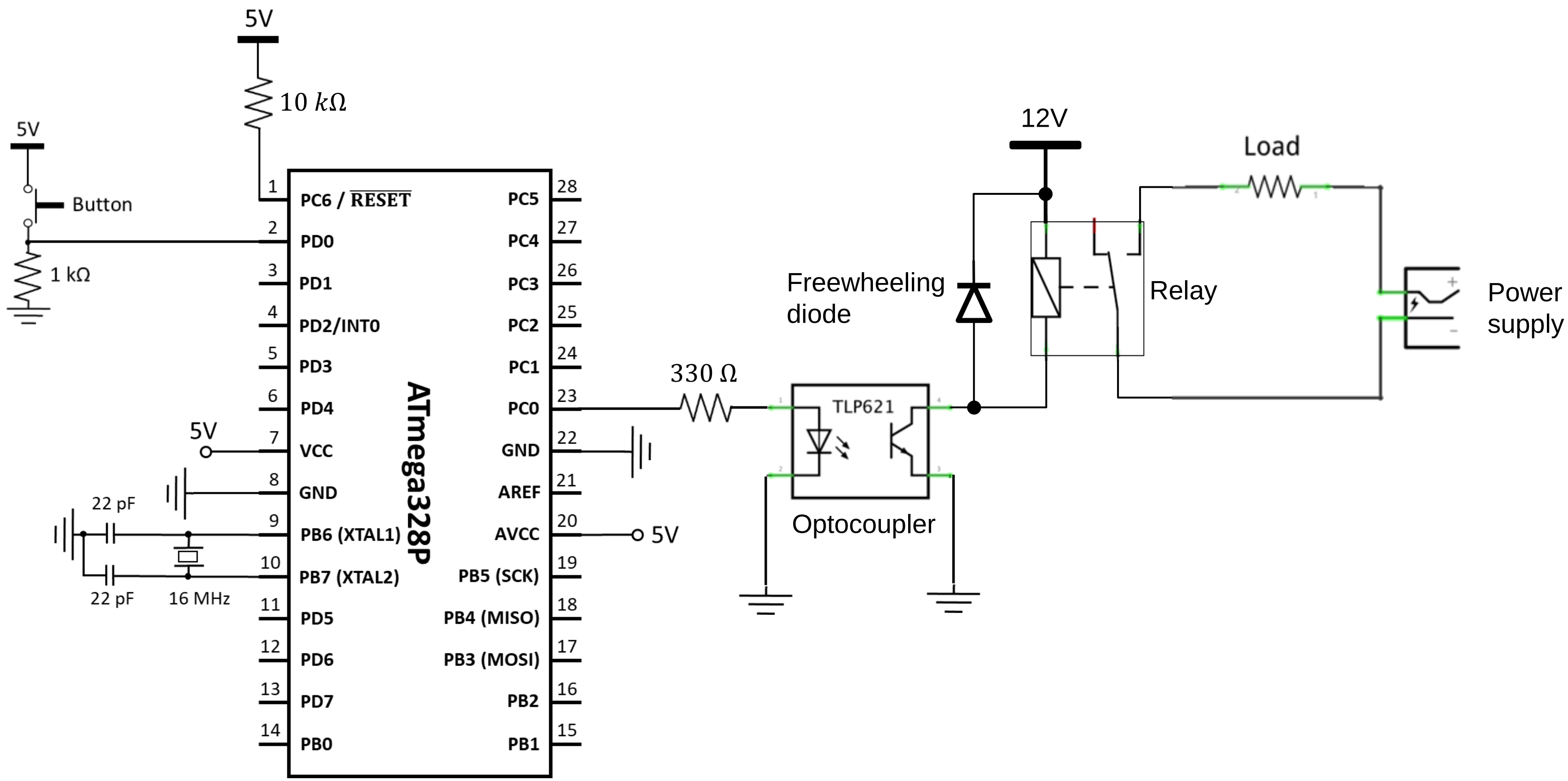
The ATmega328P should use an external oscillator of 16 MHz. **Complete the circuit diagram**, such that the ATmega328P controls the optocoupler through pin PC0.





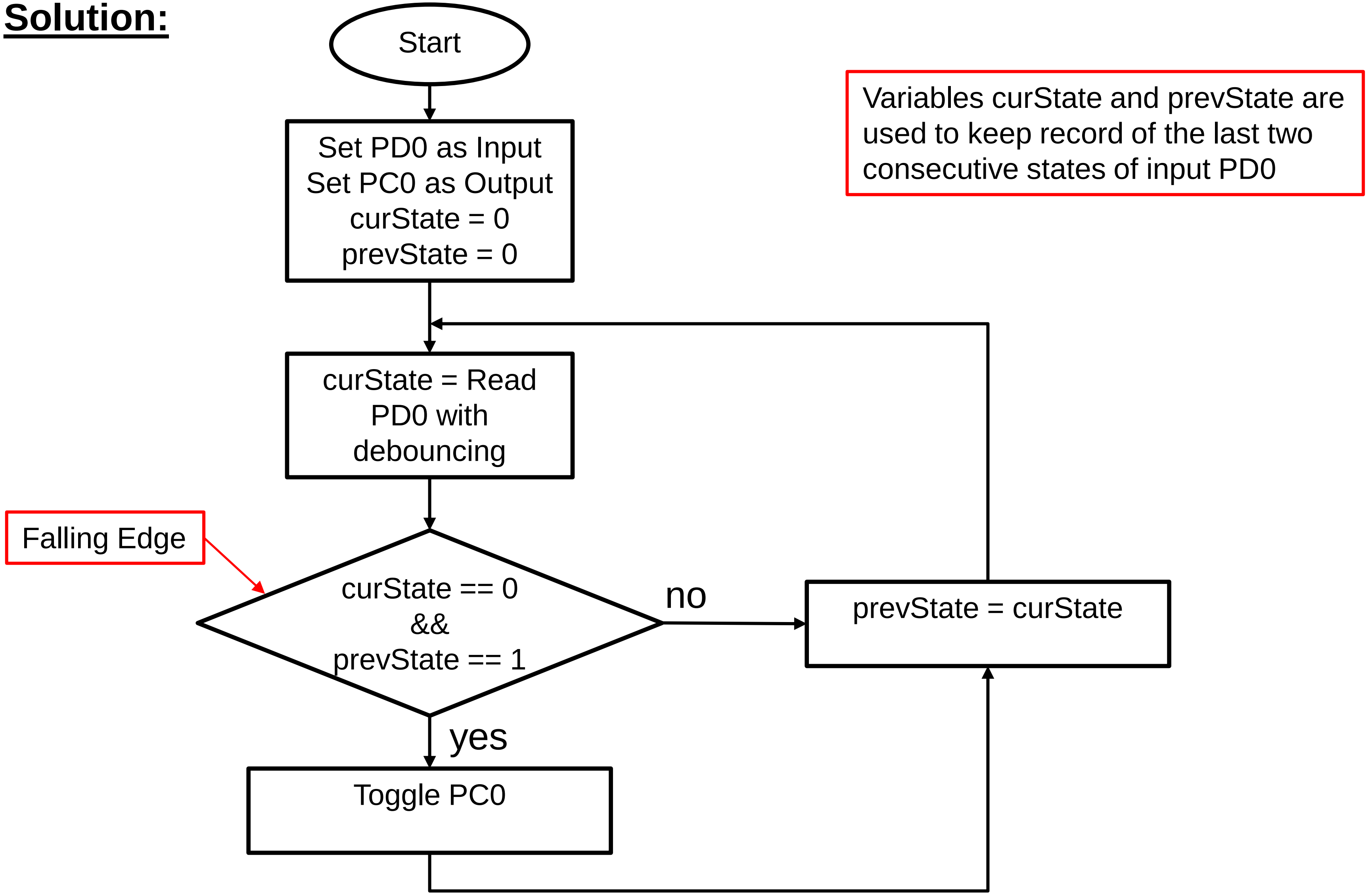
GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Solution:



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Solution:


Start
Set PD0 as Input
Set PC0 as Output
curState = 0
prevState = 0
curState = Read
PD0 with
debouncing
Falling Edge
curState == 0
&&
prevState == 1
no
prevState = curState
yes
Toggle PC0
Variables curState and prevState are used to keep record of the last two consecutive states of input PD0

CODE:

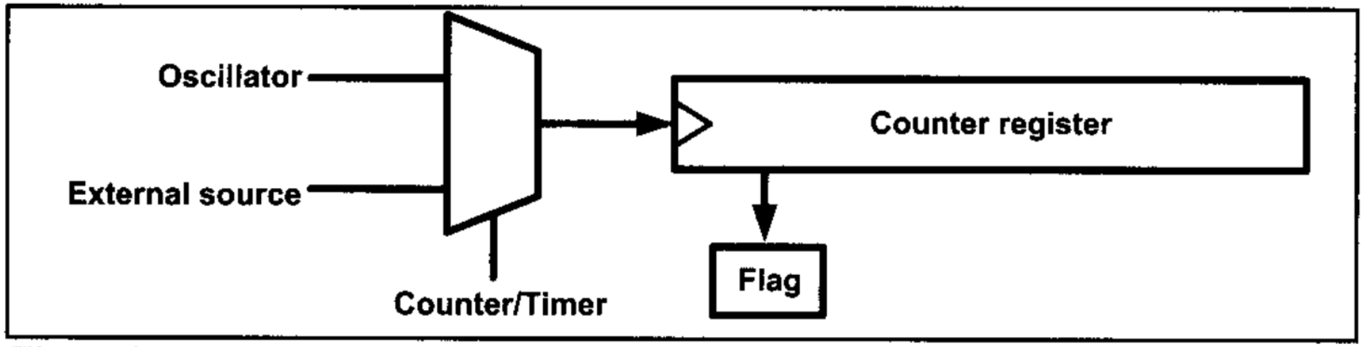
```
#define F_CPU 16000000 // define frequency of CPU clock in Hz to be 16e6 Hz

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    unsigned char currState, prevState;
    prevState = 0;
    currState = 0;

    DDRD &= 0b11111110;// set pin PD0 as input
    DDRC |= 0b00000001;// set pin PC0 as output
    PORTC &= 0b11111110;// clear output PC0 initially

    while (1)
    {
        // read button current status
        if(PIND & 0b00000001)
        {// PD0 is logic high
            _delay_ms(50); // delay and recheck
            if(PIND & 0b00000001)
                currState = 1;
        }else
        { // PD0 is logic low
            _delay_ms(50); // delay and recheck
            if(!(PIND & 0b00000001))
                currState = 0;
        }

        // check for falling edge on PD0
        if(!currState && prevState)
            PORTC ^= 0b00000001; // toggle output PC0

        prevState = currState;
    }
}
```

Tutorial Contents

- Important Switching Devices
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- Digital Input-Output Ports in ATmega328P
- Code example on interfacing
- **Timers in Atmega328P**
- Code example utilizing TCNT0 as a counter

Timers in ATmega328P

- The ATmega328P has 3 timers referred **Timer 0**,**Timer 1** and **Timer 2**. Timer 0 and Timer 2 are **8-bit timers** while Timer 1 is **16-bit**. These Timers can be used as counters or timers.



Example:

- We can use Timer 0 as **counter** that counts specific events on an external input signal received on pin **T0**.

- Timer 0 can also be used as a **timer** and is incremented on every timer **clock cycle.** It counts from 0 to 255. The TCNT0 register holds the timer count. If the timer is turned on it ticks from 0 to 255 and overflows. When it overflows, a Timer Overflow Flag(TOV) is set. You can as well load a count value in TCNT0 and start the timer from a specific count.

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Timers in ATmega328P

- Registers Associated with Timer 0:
  - TCNT0: Timer/Counter 0 Register

  - TCCR0A: Timer/Counter 0 Control Register A
  - TCCR0B: Timer/Counter 0 Control Register B

  - OCR0A: Output Compare Register A
  - OCR0B: Output Compare Register B

  - TIMSK0: Timer/Counter 0 Interrupt Mask Register
  - TIFR0: Timer/Counter 0 Interrupt Flag Register

Timers in ATmega328P

- **TCCR0A/B (Timer/Counter 0 Control Register A/B):**

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[(1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Timer overflow flag on overflow

Timers in ATmega328P

- **TCCR0B (Timer/Counter 0 Control Register B):**

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 14-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Timers in ATmega328P

- **TCCR0B (Timer/Counter 0 Control Register B):**

15.9.7 TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare flag for match between OCR0B and TNTC0

Output Compare flag for match between OCR0A and TNTC0

Timer 0 Overflow Flag

- To clear a flag set it to 1. Example:

```
TIFR0 |= 0b00000001; // Clear TOV0
```

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

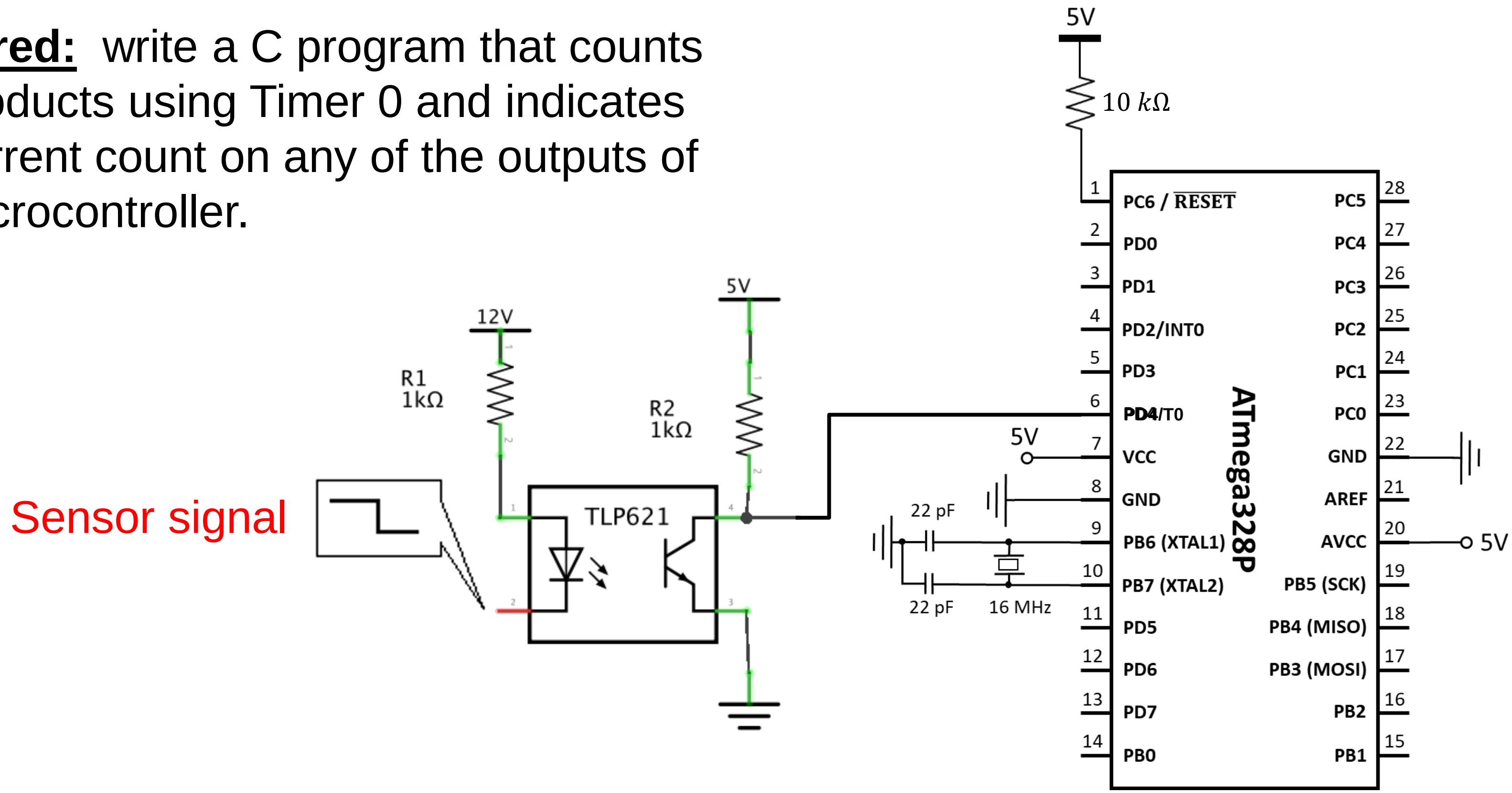
- Important Switching Devices
  - ❖ BJT
  - ❖ Relay
  - ❖ Optocoupler
- Interfacing an inductive load
- Digital Input-Output Ports in ATmega328P
- Code example on interfacing
- Timers in Atmega328P
- **Code example utilizing TCNT0 as a counter**

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Problem 4:

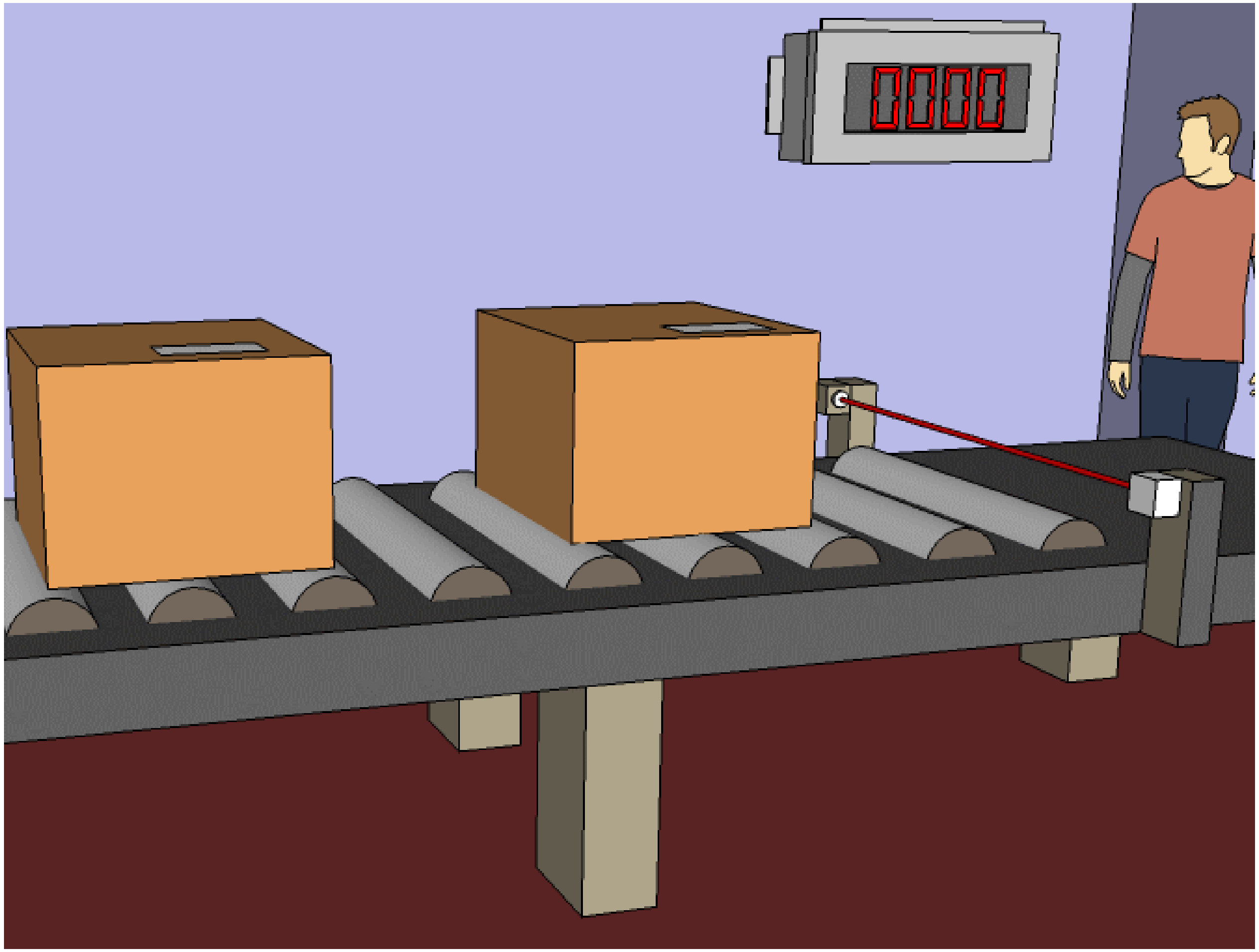
This example is a counter, used for counting products on production line. Let the sensor be a micro-switch. Each time the switch is closed, the LED is illuminated. The LED transfers the signal to the photo-transistor and the operation of the photo-transistor delivers a LOW to input T0 of a microcontroller.

Required: write a C program that counts the products using Timer 0 and indicates the current count on any of the outputs of the microcontroller.



GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Problem 4: Application



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Solution

- **Choose Waveform Generation Mode:**

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	**WGM01**	**WGM00**	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Solution

- **Choose Clock source:**

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 14-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Solution:

This circuit has 1 input and PORTB as an output

Input **T0** | Output 8 **LEDs**

D4/T0 | (PB4:PB0)(PD7:PD5)

Each negative edge ↓

Increase by 1

CODE:

```c
#include <avr/io.h>

int main (void) {

  // char out works also
  uint8_t out = 0;

  //configure Timer 0 as counter for external signal on Pin T0 PD4
  TCCR0B |= 0b00000110;

  // configure PD4 as input
  DDRD &= 0b11101111;

  // configure pins PD7:PD5 as outputs
  DDRD |= 0b11100000;
   // configure pins PB4:PB0 as outputs
   DDRB |= 0b11111111;
  // start timer 0 from 0x00
  TCNT0 = 0x00;
  while(1) {
   out = TCNT0;
   PORTB = out>>3;
   PORTD = out<<5;
  }
}
```