Microprocessor systems EMISY

Integrated and external peripherals in microprocessor systems, part 1
Lecture 5

Semester 20L – Summer 2020

© Maciej Urbanski, MSc

email: M.Urbanski@elka.pw.edu.pl

WUT

ise

Important remark

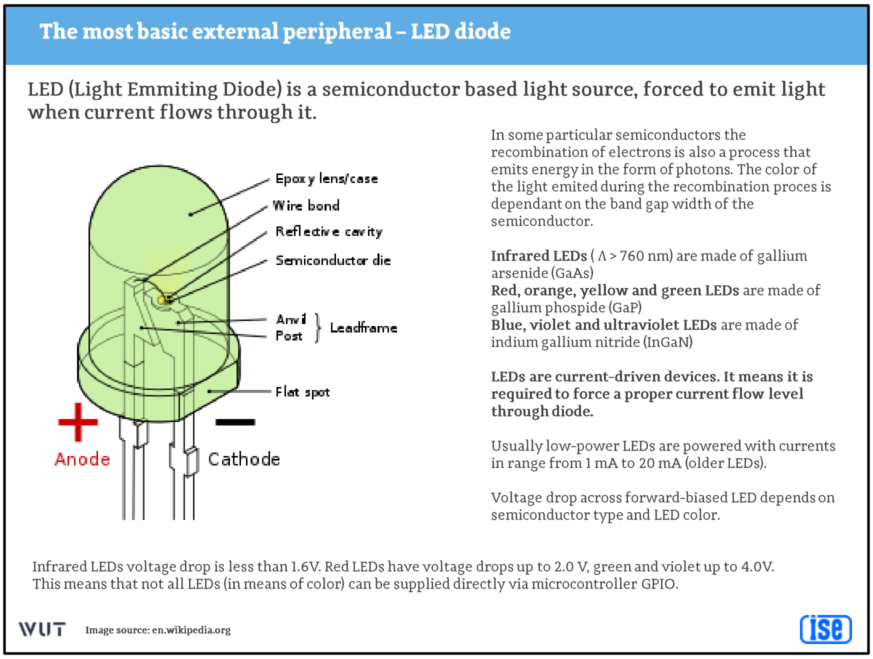
This material is intended to be used by the students during the Microprocessor Systems course for educational purposes only. The course is conducted in the Faculty of Electronics, Warsaw University of Technology.

The use of this material in any other purpose than education is prohibited.

This material has been prepared based on many sources, considered by the author as valueable, however it is possible that the material contains errors and misstatements.

The author takes no responsibility for the usage of this material and any potential losses this usage can lead to. Furthermore the author will be very grateful for pointing out any errors found and also for any other useful remarks on the course material and potential upgrades.

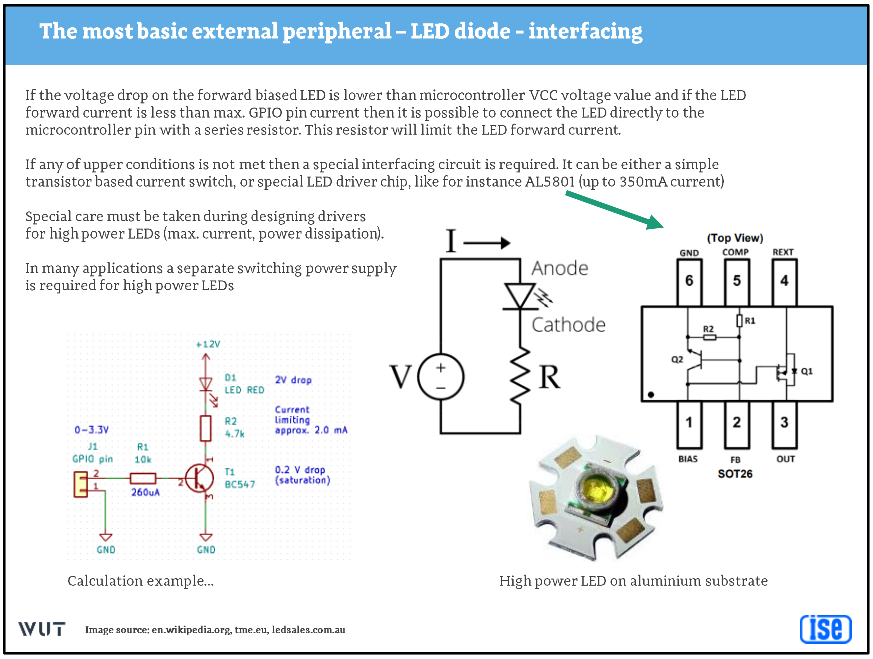
WUT



Further reading about the electron recombination in LEDs (extra material):
https://electronics.howstuffworks.com/led.htm

Basic information about diodes:
https://en.wikipedia.org/wiki/Diode

The most important part to remember here: LEDs are **current driven** devices. You may supply them even with 1000V as long as you maintain safe current flowing through them. In most cases LEDs consume between 1 to 20mA (standard LEDs), but there are so called power LEDs that can consume even more than 1A of current. These power LEDs require special driving circuits (current switches).
You must not forget about the fact that each diode (LED too) creates a voltage drop across its anode and cathode when in forward bias. This voltage drop is usually in range from 0.5 to 1.5V for standard diodes, and from 1.5 to 4 for LEDs.



Let’s describe the examples. To the right there is a LED driver as IC chip. Not a bad idea for modern circuits, but still, the concept requires understanding of basic circuit to the left (most commonly used).

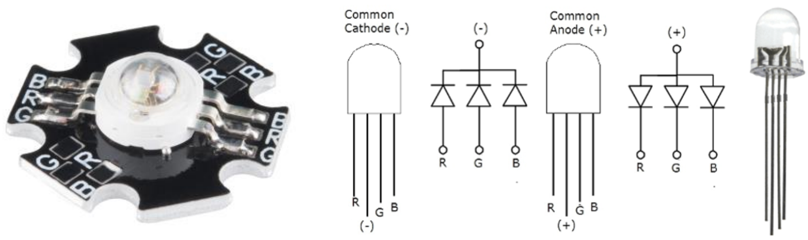
The middle drawing, with LED, resistor and voltage source – the position of the resistor is not important, it can be connected in series with the anode as well.
The choice is forced by the application – does the LED require to be tied to ground – does it have a heatsink that should be connected to ground?

The schematic to the left. We assume that the GPIO pin is capable of driving 0 and 3.3V voltage levels. We need to remember that the bipolar transistor, when in active or saturated mode creates a voltage drop of 0.7V across it’s base and emitter (base is node 2, emitter node 3, collector node 1). So this means that in logic one state the voltage drop across R1 is 3.3V-0.7V = 2.6V. We want to limit the base current to around 200uA (can be 100uA, 300uA, not critical), we just want to make sure that the transistor is saturated. We select R1 = 10kOhms, so using Ohm’s law the current through R1 is 260uA. Now, we assume that the transistor is saturated (and it should be for a big-enough base current – 200-300uA for a typical LED is fine), so the voltage drop across the emitter and collector is close to 0V (0.2V to be exact).

We have then to use the 2nd Kirchoff's law and draw something similar to the middle drawing – Our voltage source is 12V – 0.2V = 12V (almost), the voltage drop caused by the diode (let's assume red one) is 2V, so the voltage across the resistor is around 10V. Diode current should be between 1 and 10mA, so we select 2mA and calculate the resistor value to be 5kOhms – we select 4.7 kOhms because that's the value you can buy.

The most basic external peripheral – RGB diodes

By connecting three LEDs of different colors (red, green, blue) in parallel it is possible to create RGB LED. The final color will be a result of mixing red, green and blue. The intensity of each color can be adjusted with proper driving circuit, for instance PWM.



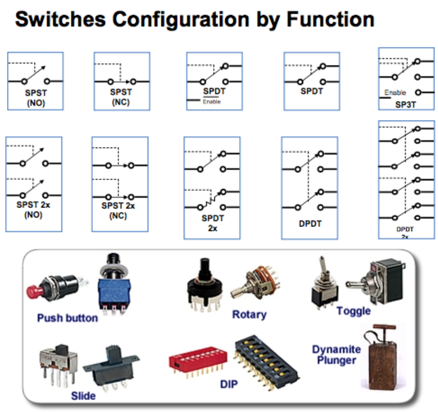
WUT Image source: hackster.io

In order to drive an RGB LED properly you need to use three GPIO pins (and optionally three independent current switches).

Let’s discuss the diode in the left picture. The black PCB it is placed on is made of aluminum and works as a heatsink – needs to be mounted to the external heatsink. Such diode will consume a lot of current and in order to drive it you need a decent current switch, like the MOSFET based one.

Connecting tact switches – introduction to keyboards

Switch is an electronic device that causes a short or break between two or more points in circuit…



There are different types of switches:

SPST – Single Pole Single Throw – the most basic turn on/off switch, available in normally open and close versions
SPDT – Single Pole Dual Throw – three terminal switch, there are two positions available

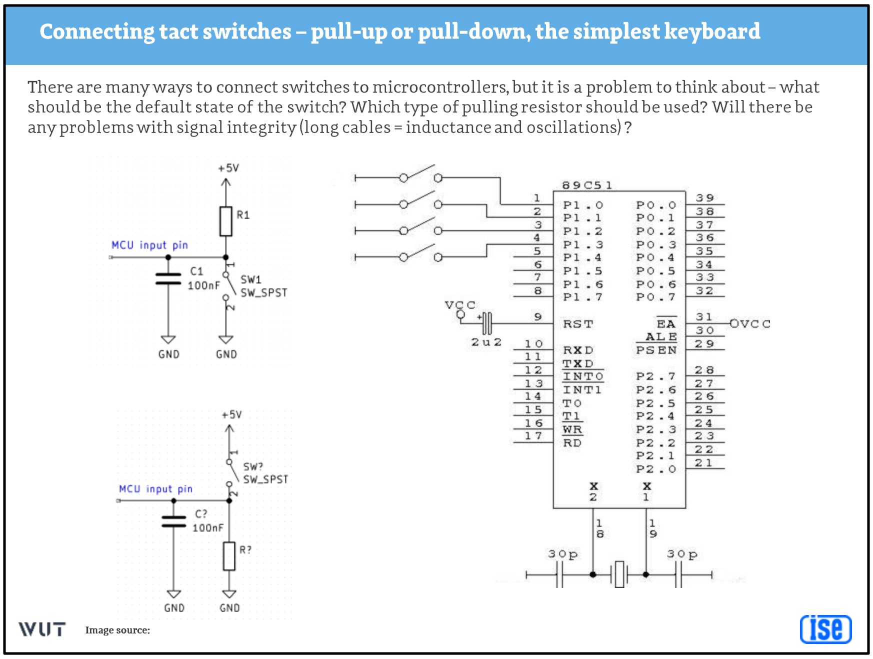
SP3T, SP4T – SPDT variations with multiple ports

DPDT – Dual Pole Dual Throw – dual SPDT, can be used to turn on/off devices connected to 230V AC mains

Push button switches – tact switches – they have one stable state (normally open or normally closed), they are used to create keyboards in microprocessor systems

WUT Image source: https://www.electronicshub.org/switches/ ise

What is important to remember here – what is the difference between NC and NO switches, and how to decode the abbreviations SPST, SPDT, SP4T.



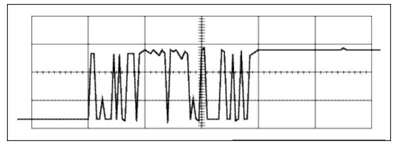
It is important to decide what should be the default state of the input pin when the tact switch is not used.
The upper left schematic is for the default high state – the resistor R1 is a pull-up and the SW1 is normally-open, so the resistor in default state forms a weak high state (weak means that it’s current is significantly limited by a high R1 resistance). When SW1 is pressed we create a strong low state (strong, because the current is limited only be the close to zero resistance of the shorted switch). Capacitor C1, together with R1 forms a low-pass circuit which is used as a hardware debouncing circuit.

If the MCU has internal pull-up resistors you may connect switches as in the right drawing.

The bottom left drawing is similar to the upper left, but the default state is weak low state.

Connecting tact switches – debouncing

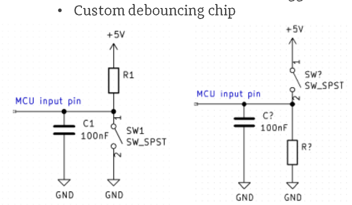
Problem with switches is that they can bounce several times during the switching process before they are locked in desired state. This may lead to software errors, when microcontroller counts more than one level change on its input pin.

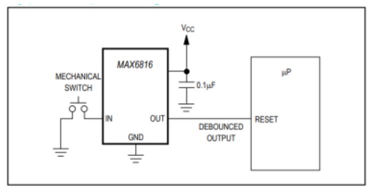


Time plot of switch voltage, showing bouncing effect

There are many ways to deal with the problem:

- Software way
  - Check the state of the pin after some time, like 50 ms, if still in the same state then the button/switch is pressed
- Hardware way
  - RC delay circuit
    - With/without Schmitt trigger buffer
  - Custom debouncing chip





MAX6816 debouncer chip typical application.
It includes up to 15 kV ESD protection.

Debouncing solution must be applied to all types of keyboards and switches connected to microcontrollers. The reliability of the entire system depends on it.

WUT Image source: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/
Maxim Integrated Application Note 287

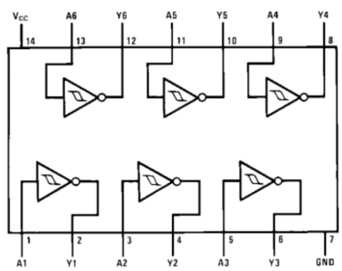
ise

Debouncing is caused by worn out switches. In order to extend their lifetime and improve the robustness of your application you must not forget about the debouncing techniques.

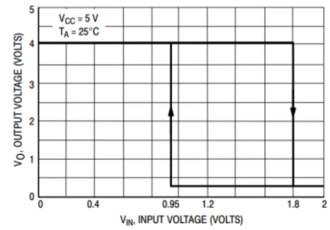
Important to remember – designing the hardware debouncer, describing how software debouncer works.

Connecting tact switches – debouncing – Schmitt trigger buffers

It is also possible to connect switches via Schmitt trigger buffers. Schmitt trigger logic gates have hysteresis, which means that threshold voltage for rising and falling edges of the input signal are different. This results in significant invulnerability to noise in circuit.



Gate diagram of classic 7414 Schmitt inverters chip.



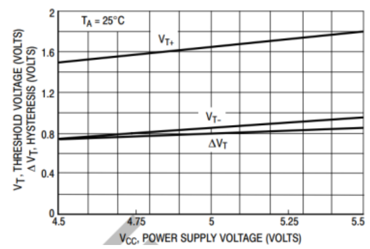


Image source: https://www.onsemi.com/pub/Collateral/SN74LS14-D.PDF

Connecting tact switches – matrix keyboard

There is a problem. How to connect many switches to the microcontroller?
Driving each one with separate GPIO pin is very ineffective.

Let's say we want to connect 16 button keyboard to the system we design. In standard approach we would need 16 GPIO pins, which is full occupation of two ports of 8051 microcontroller.

Instead, we can use only 8 GPIO pins and save the rest for other purposes.

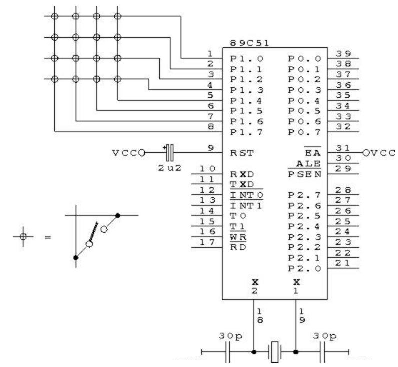
What we need is a matrix keyboard. To determine the state of each pin it is required to read the state of proper row pin and column pin.

For instance by reading pins P1.0 and P1.7 we can determine what is the state of upper left button.

All of the buttons should be fitted with pull-up resistors (internal or external).

What about debouncing?

There are also other ways to connect multi-button keyboards, for instance with using shift registers.

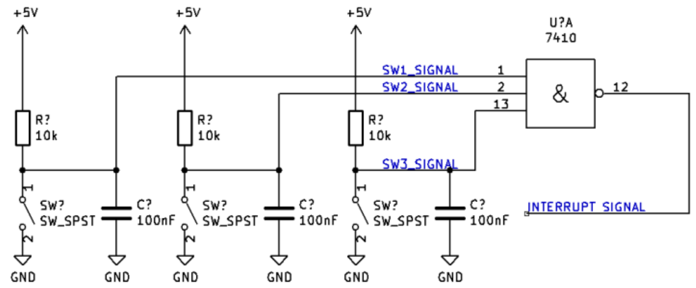


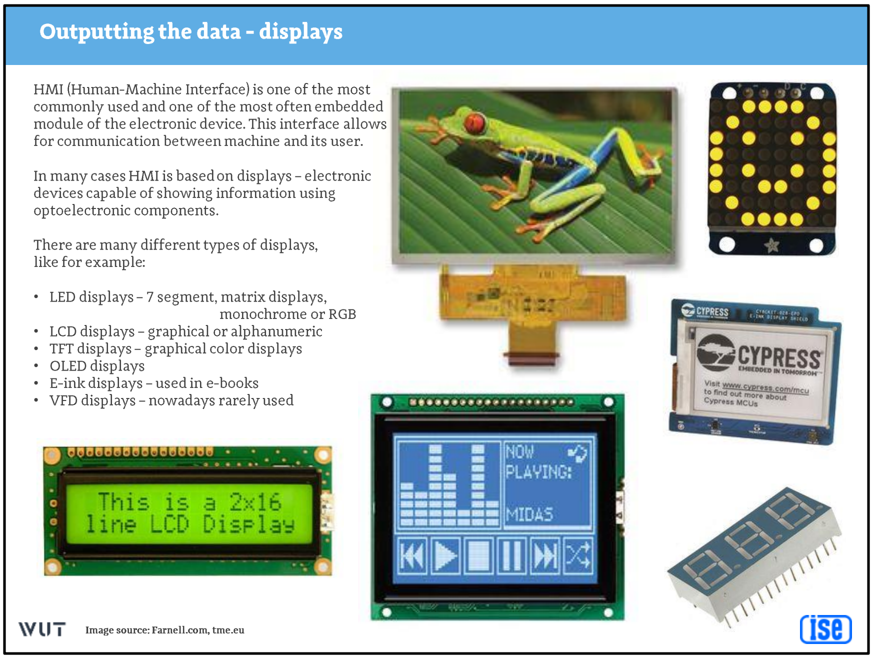
Connecting tact switches – interrupt based keyboard

To be able to determine if the button is pressed, microcontroller needs to constantly monitor the keyboard status. This is very ineffective and requires lots of resources of the MCU.

It is possible to connect keyboards using interrupts (coming up in next lectures). Microcontrollers usually have pins reserved as external interrupt source (for instance P3.2 and P3.3 in AT89S4051). These pins may be used to tell microcontroller that some button is pressed and it should immediately scan the keyboard. This approach allows to free the resources when buttons are not used.

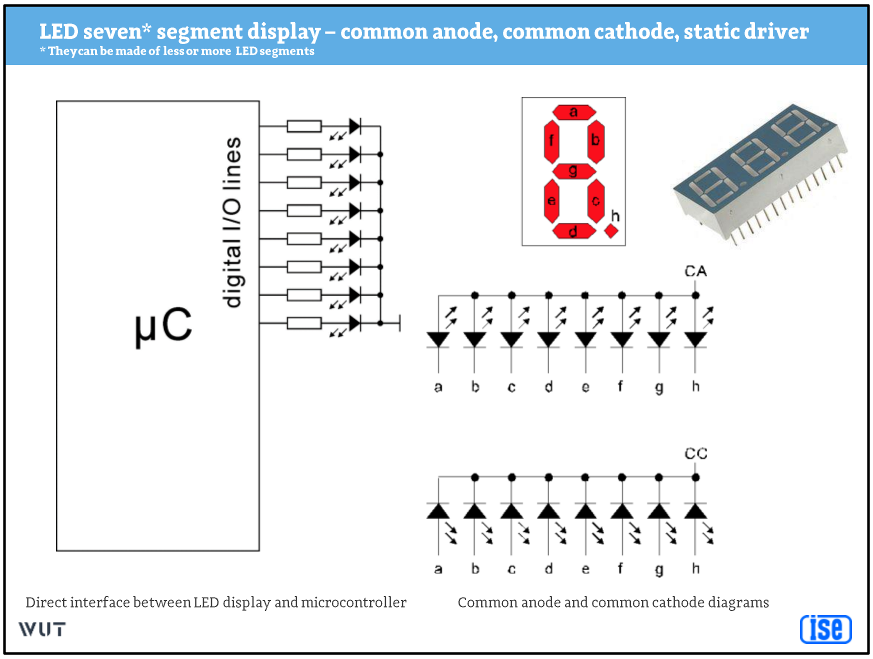
7410 is triple input NAND gate. Default state of the NAND input is 111, so the output is 0. If any of the buttons is pressed, then the interrupt signal goes logic high, telling the MCU that it has to scan the keyboard.
All signals (SW1, SW2, SW3, INTERRUPT) have to be connected to microcontroller.





Presented displays:

- The one with the frog – TFT color display
- The one with the smile – so called LED matrix, used in public transport infotables
- The one with Cypress logo – so called e-ink display – most commonly used in portable ebook readers
- Bottom displays, starting with the left one – the 2x16 alphanumeric HD44780 display (used in labkits), the LCD mono graphic display and 7 segment LED with three digits

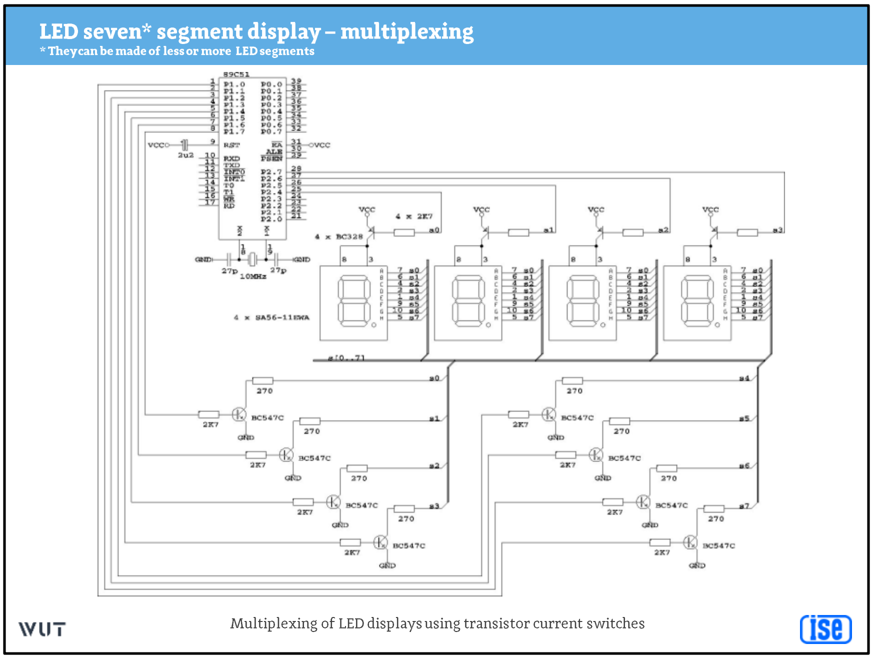


The seven segment LED display is used to display numbers and some other signs. There are variations of these that allow to display all the letters (they simply have more segments and different arrangement).

They are available in two variants – so called common anode and common cathode display and the difference is shown in the slides.

LEDs are current driven devices, so each segment require a current limiter – in most cases an ordinary resistor is enough.

The simples example showing how to connect a single 7 segment display to the MCU is shown to the left. Each segment is connected via limiting resistor to the MCU. By driving pins logic high we can turn on selected segments.
It is important to highlight the problem of GPIO current efficiency and output mode. If GPIO pins are capable of driving LED (current efficiency high enough for the LED, not the power LED – they need a current switch anyways) then the LEDs can be connected directly, otherwise via current switches.

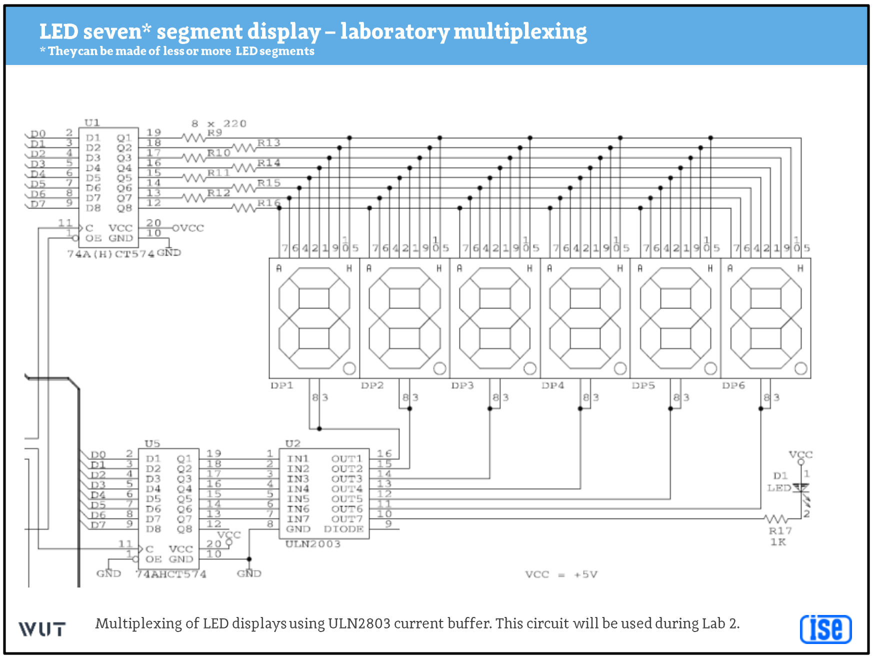


This is a typical way of driving multi 7 segment LED displays – in this example we have the multiplexed connection of 4 independent 7segment LED displays.

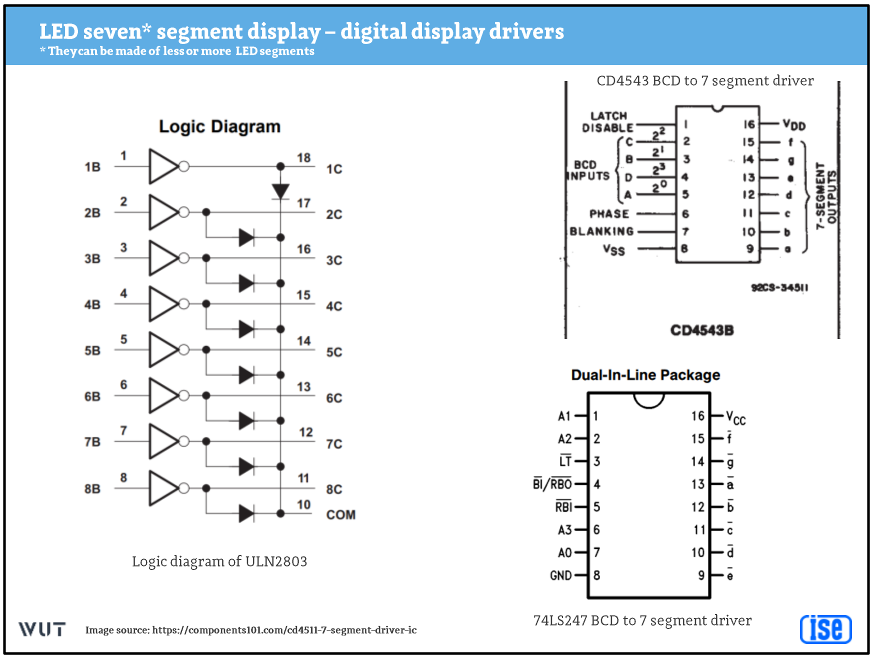
This is an example of common anode displays – notice that all the LED segment pins are connected to the bus and driven via bottom current switches (shorted to ground) and the common pins (the upper ones) are driven via current switches to VCC.

In order to display something using the multiplexing method you have to turn on one display at a time, present the value, switch to next one, display another value, and so on. The switching time between the next displays must be fast enough so the human eye inertion can create an impression that all the displays are turned on in the same time.

In this example the anodes are connected via upper current PNP switches, the LED current is limited by 270 Ohm resistors and the cathodes are shorted to ground via bottom NPN switches. In total 12 transistors are used in this example.



Laboratory multiplexing example. LED segment pins are driven via U1 and displays are driven via U2 and U5. U5 and U1 are connected to a single data bus. In order to drive segments you must first address the U1, send data, switch to U5, send display selection data and so on.



The ULN2803 chip is simply an 8 channel current buffer that improves the efficiency of GPIO pins.

In order to reduce the GPIO pin number needed to drive the LEDs you may also used something called BCD to 7 segment driver. These are commonly used chips, like 74LS247 or CD4543 or CD4511. They convert information given in BCD code (4 bits, means 4 pins per digit) to 7 segment code. In most cases they have output current buffers so they can be connected directly to the LED display.

Alphanumeric LCD display – HD44780 driver based and clones

HD44780 LCD driver chip from Hitachi is currently one of the most popular driver for alphanumeric LCD displays. There are lots of different types of LCD displays based on this driver or its clones.

The way it is connected with the microprocessor system (via 8 bit Motorola 6800 based parallel or 4 bit interface) is a standard in electronics.

HD447800 is compatible with following clones: HD66712, KS0066, KS0073, KS0076, LC7985, NT3881, S6A0069, SED1278, ST7066

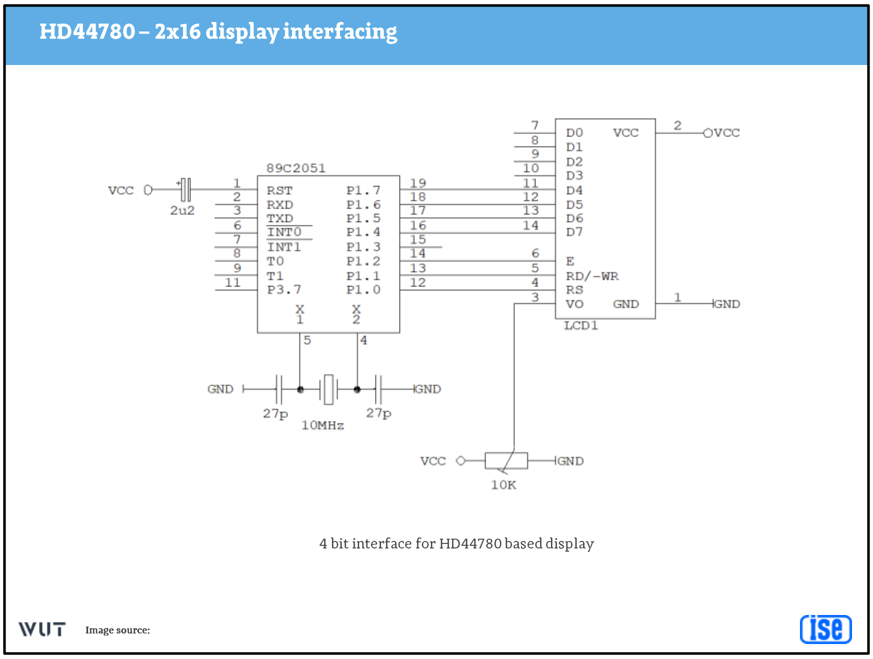
Typically these type of displays are connected with 16 pin connector. Pin 1 is GND. Pin 2 is driver VCC, pin 3 is for contrast regulatio

Pin 4 is RS (command or data pin), then pin 5 is R/W (read or write – in many applications shorted to GND for write-only). In most cases HD44780 should be supplied with +5V DC, thus direct interfacing it with 3.3V logic is not always possible.

Pin 6 is EN (enable) pin. Pins 7 to 14 are for 8 bit data bus. The remaining two pins are usually used for backlight.

HD44780 has two types of internal memories:

- DDRAM (Display Data Ram) – it stores currently displayed signs, this memory should be 80 bytes long (in standard HD44780)
- CGROM (Character Generator ROM) – this memory stores the information about how to display signs on the display. First 64 bytes of CGROM can be written (RAM block of CGROM), the re is read-only.



There are two ways to connect HD44780 based LCD displays – so called 4-bit and 8-bit interfacing. In the slide there is a circuit for the 4 bit variant.
Each command is given to the display as an 8 bit word that is transmitted in two 4 bit nibbles. For the communication the upper 4 bits of the data/command bus are used.

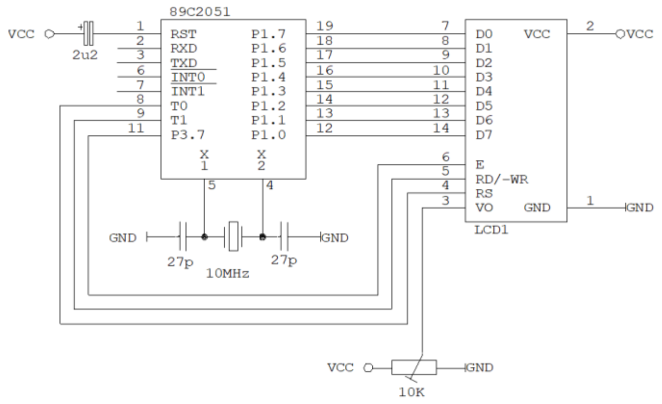
This allows to save 4 GPIO pins in the application, but the display needs to be configured correctly in order to communicate using 4-bit data bus.

Pin E (Enable) is used to generate pulses during communication, telling the display that it can read the command/data.
Pin RS defines wheter the information given is data or command.
Pin RW defines the direction of communication. When shorted to GND the communication is only „to the display” – we cannot read the display. This is very handy when you use 3.3V MCU with a 5V display – some MCUs don’t have proper 5V tolerant GPIO pins and reading is not safe.

The bottom potentiometer is used for LCD contrast adjustment. Usually the voltage and VO pin should be around 3.7V
This means that if you want to use the display in your application (project) you must

supply it with 5V.

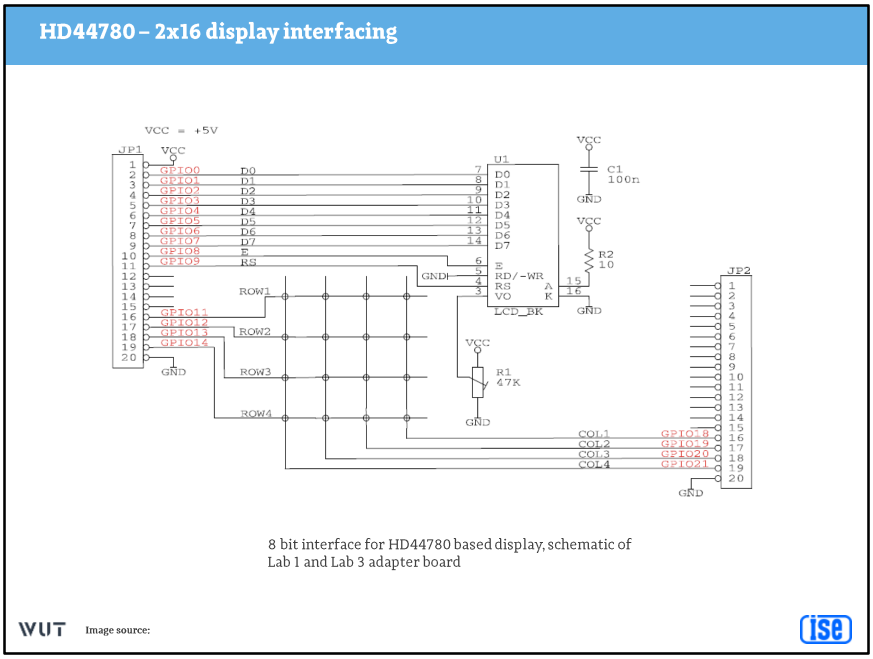
HD44780 – 2x16 display interfacing



8 bit interface for HD44780 based display

WUT Image source:

The 8-bit mode is even simpler that the 4 bit mode. It is the default mode of communication.



The schematic presents the laboratory kit shield with LCD display and matrix keyboard. During laboratory classes this keyboard will not be used.
The display is configured in 8 bit mode and since the MCU in the laboratory kit is the old AT89S52, supplied with +5V, then reading the display is safe.

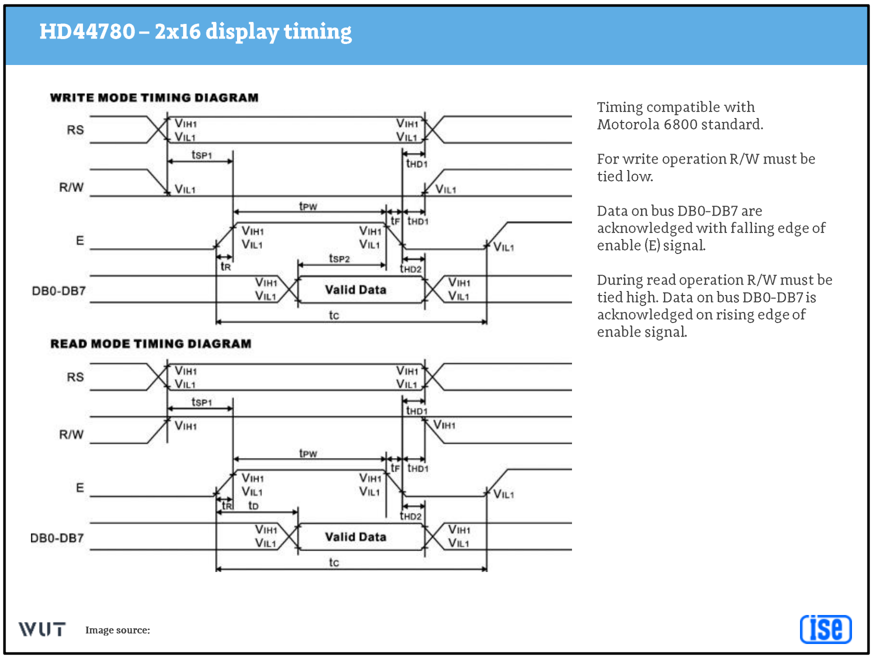
HD44780 – 2x16 display timing

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	ns
	E Pulse Width (High, Low)	tw	230	-	-	ns
	R/W and RS Setup Time	tsu1	40	-	-	ns
	R/W and RS Hold Time	t_{H1}	10	-	-	ns
	Data Setup Time	tsu2	80	-	-	ns
	Data Hold Time	t_{H2}	10	-	-	ns
Read Mode (Refer to Fig-7)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	ns
	E Pulse Width (High, Low)	tw	230	-	-	ns
	R/W and RS Setup Time	tsu	40	-	-	ns
	R/W and RS Hold Time	t_H	10	-	-	ns
	Data Output Delay Time	t_D	-	-	120	ns
	Data Hold Time	t_{DH}	5	-	-	ns

WUT Image source:

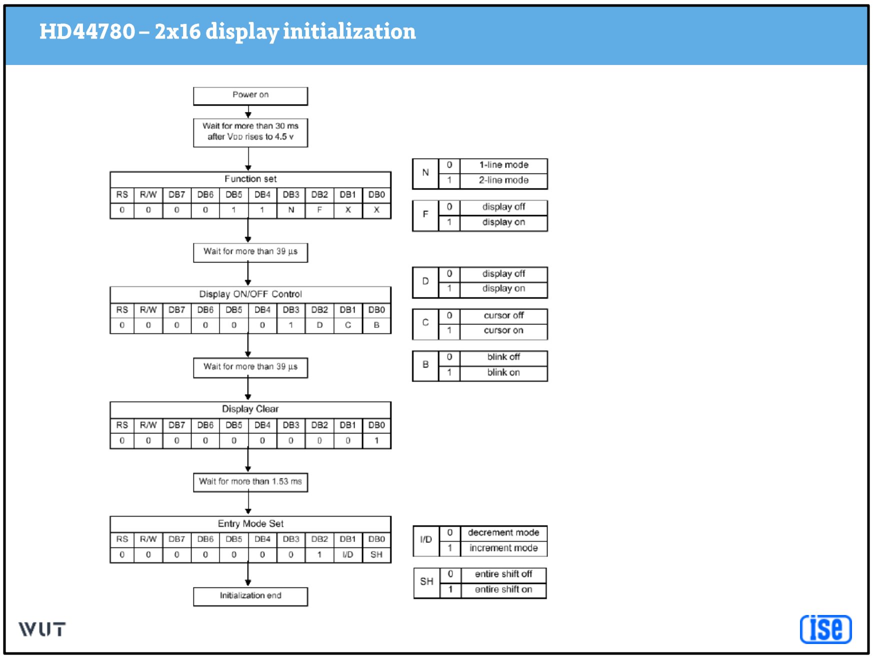
ise

This is important. When communicating with the display some specific time delays must be preserved. Please note that in most cases (when writing to the display) the timing constraints are given as minimum – this means that they can be longer in your application. In order to calculate proper time intervals you should use timer internal peripherals, or, like in very simple and not precise examples, methods using clock cycle calculations (I call them time waste methods) – to use them you need to know the clock frequency for the MCU and the cycle lenght (12 clock cycles for the 8051).



The slide present timing during communication with HD44780 display, 2x16.

When writing (what is important for us) the RW pin should be low (can be tied to GND). Then the RS should define the data/command information.
Then you should send the information and when it’s done you should create a pulse with E line.



This diagram shows a typical initialization scheme for 8 bit mode. It is used in laboratory kits.

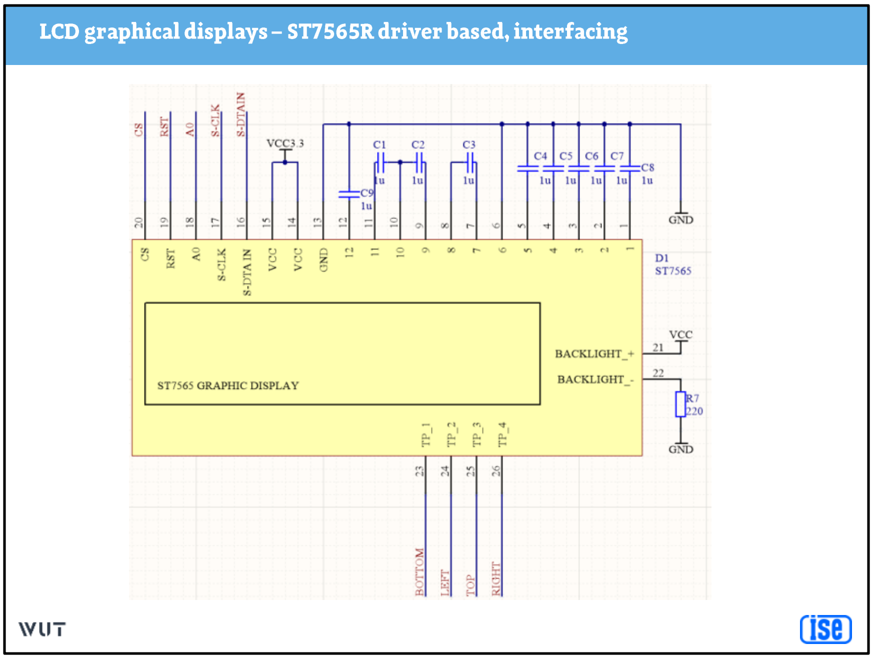
LCD graphical displays

Graphical display drivers are usually working with pixel matrices and allow to display any graphics (usually in monochrome). More sophisticated drivers may have embedded sign generator that can be very useful when displaying text.

Among many popular display drivers the following should be mentioned:

- KS0108/7B - one of the most popular display drivers by Samsung. It drives up to 64x64 displays and is driven via 8 bit Motorola 6800 parallel interface. It does not have sign generator
- T6963C – one of the most popular display drivers with sign generator, manufactured by Toshiba. It works with up to 128x256 pixel display. Internal sign generator can store up to 256 user custom signs. It is controlled via 8 bit Intel 8080 interface
- ST7565R – quite popular display driver, used in COG (Chip on Glass) and DOGM series displays. These displays are working with 3.3V logic and communicate with microprocessor system via SPI write-only interface – only microcontroller sends data ,there is no reading from the display. For proper operation it requires additional capacitors for internal charge pump regulators. It does not have sign generator.
- UC1601 – very similar to ST7565R

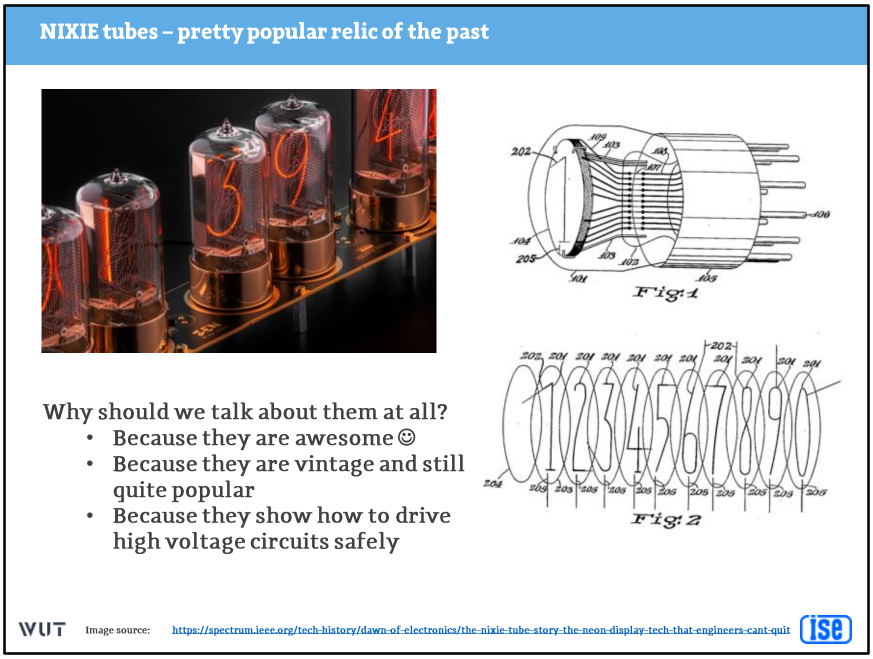
WUT Image source: amazon.com, EP 04/2010



Here we have an example of graphical display schematic. The upper capacitors are defined in the datasheet and are used for internal switching voltage regulator that creates 12V out of 3.3V (boost converter) for contrast of the display. In this example contrast can be adjusted in software.

The display is supplied with 3.3V, only the backlight is supplied with +5V.

The communication interface is SPI, plus two additional lines – RST and A0, defined in the datasheet. There is no Data Out line for the SPI, which means that we can only write to the display.

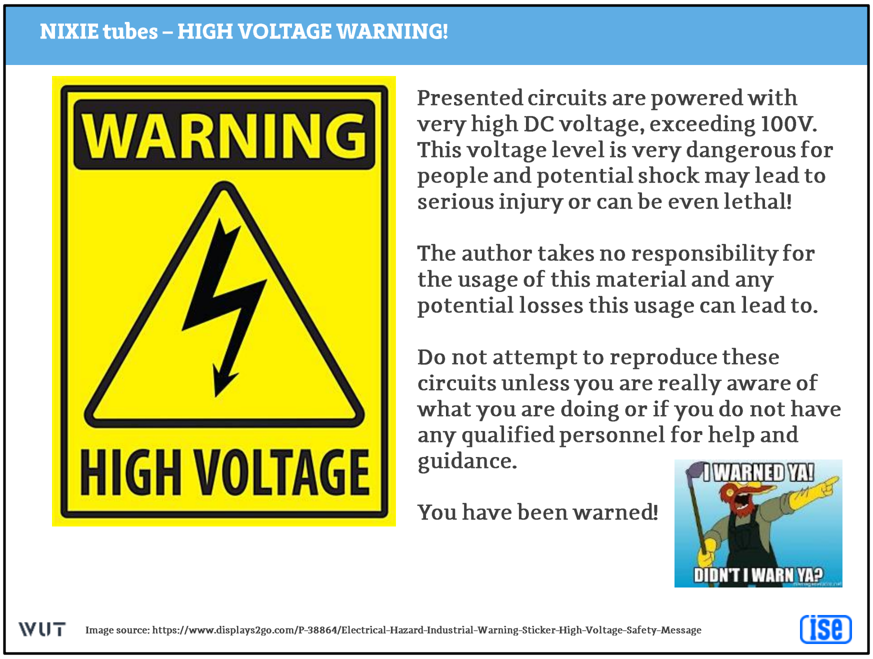


Nixie tube is a device for displaying symbols using glow discharge. The glass tube contains a wire-mesh anode and multiple cathodes, shaped like desired symbols.

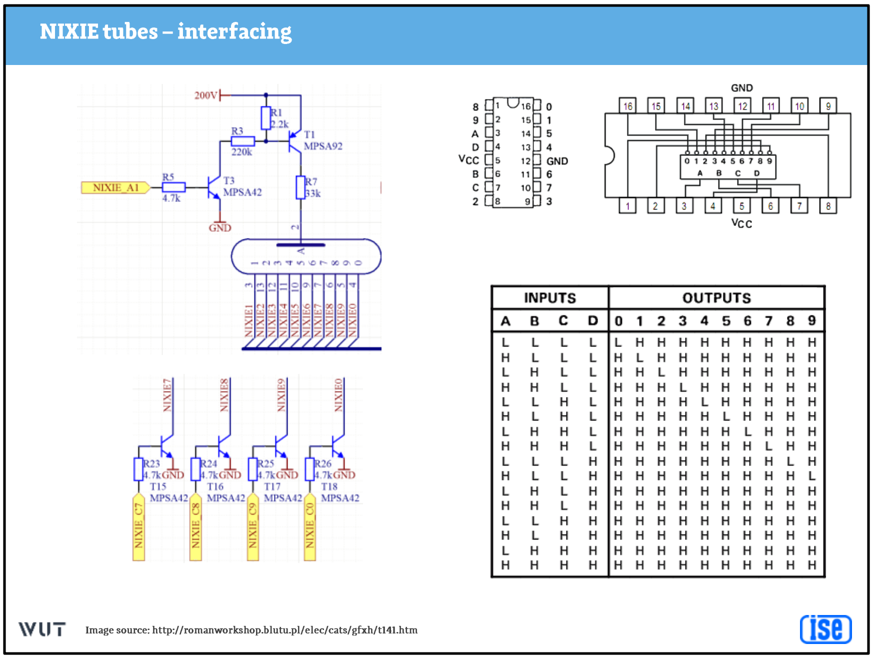
Here is an example for a custom Nixie tube: https://www.youtube.com/watch?v=1nHkhJ52iA4

The tube is filled with a gas at low pressure, usually neon, mercury and argon, the so called Penning mixture.

To operate they require very high voltage, therefore…



They should be considered dangerous and operated with special care!

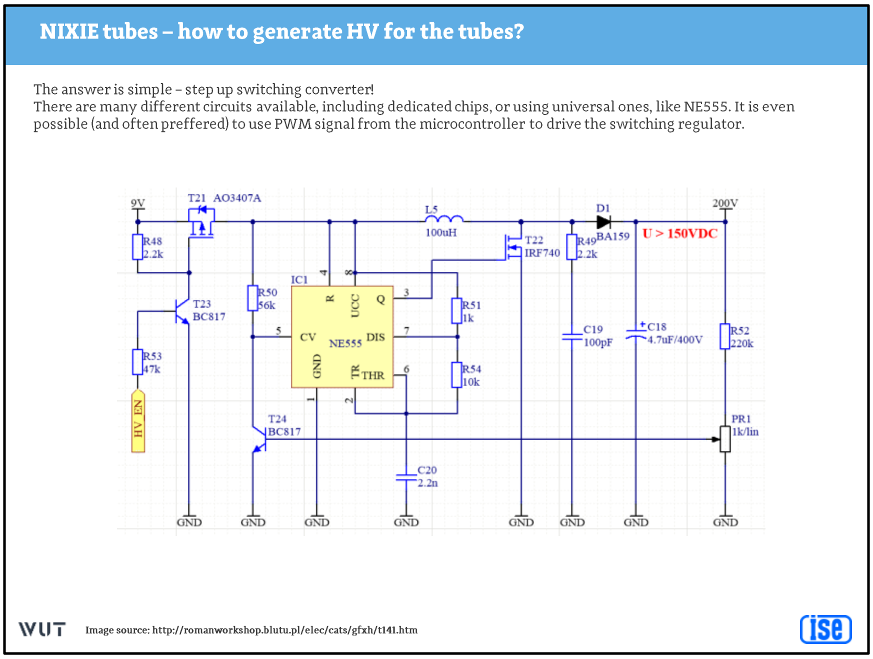


INPUTS				OUTPUTS									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
L	L	L	L	L	H	H	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
H	H	H	L	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
L	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

The slide presents a way to drive the Nixie tube. There are two ways to do it. First (old one) uses special high voltage BCD to 1 out of 10 drivers – the 74141 chips. They are no longer manufactured and very difficult to obtain – mostly as chips salvaged from some old instruments.

The more modern approach uses high voltage transistors – cheap and easily available.

The tube is driven similarly like the LED display. The upper transistors form a high voltage switch (described last week) to drive the anode.
Then the symbol cathodes are driven using high voltage bottom switches. They can be driven directly via the MCU, because they are only shorting the high voltage to ground.
In such applications never forget about the base resistors.



The circuit presented in the slide is a simple boost switching voltage regulator that created high voltage for the NIXIE tubes. The T21 and T23 transistors are the power switch. Then the IC1 is a timer switch that drives the T22 (high voltage switch) to charge the L5 coil and boost the output voltage. A fraction of the output voltage is sampled by the R52 and PR1 and given back to the IC1 via T24 (feedback loop that stabilizes the output voltage).

NIXIE tubes – sample projects, mostly clocks

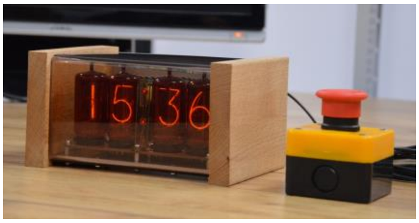


Image source: https://www.etsy.com/market/nixie_tube_clock
http://www.dudeiwantthat.com/household/decor/zen-nixie-tube-clock.asp