ENEL260 CA8: Switching Logic

Ciaran Moore

> *"Logic takes care of itself; all we have to do is to look and see how it does it."*
> – Ludwig Wittgenstein

In the previous lecture we began to look at General Purpose Input/Output (GPIO), which allows a microcontroller to interact with the outside world. But GPIO pins on a microcontroller can only handle a limited amount of current. So for many real-world applications the signal from the GPIO pin is used to drive a transistor-based switch that can handle larger currents. In the lecture we'll look at how those switches work. Along the way, we'll learn a little more about how logic gates could be implemented using transistors.

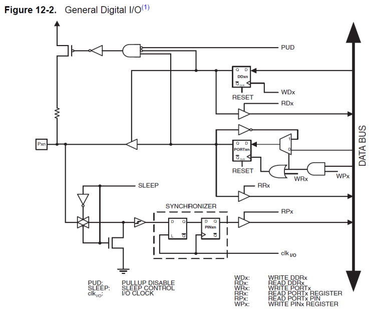


Figure 1: The logical circuit for a single GPIO pin on the ATmega32U2 microcontroller.

Towards the end of the last lecture we started to look at how to connect the computer to the outside world. Those connections are made using General Purpose Input/Output (GPIO) pins, like the one in Fig. 1. By writing 1s or 0s to the D flip-flops in the GPIO circuit we can make those signals available to the outside world (or read them from the outside world). But *how* are those 1s and 0s actually communicated? And what about components connected to the GPIO pins that aren't digital (such as the example in Fig. 2)? What do they see?

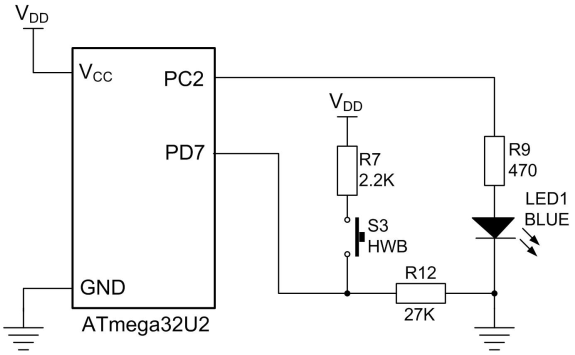


Figure 2: Example of interfacing to GPIO pins on the microcontroller - part of the UCFK.

Logical Values as Voltages

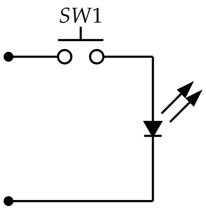


Figure 3: A simple switch controlling an LED.

Let's ignore the microcontroller for a moment, and just think about how we might control a simple LED. If we connect the LED to a switch (Fig. 3), we can turn the LED on and off by closing and opening the switch. When the switch is closed, a voltage is applied to the LED, current flows, and the LED illuminates. When the switch is open, there is not voltage across the LED, no current flows, and the LED is dark.

By using more than one switch we can create more complex behaviour, as Fig. 4 (a) and (b) illustrate.

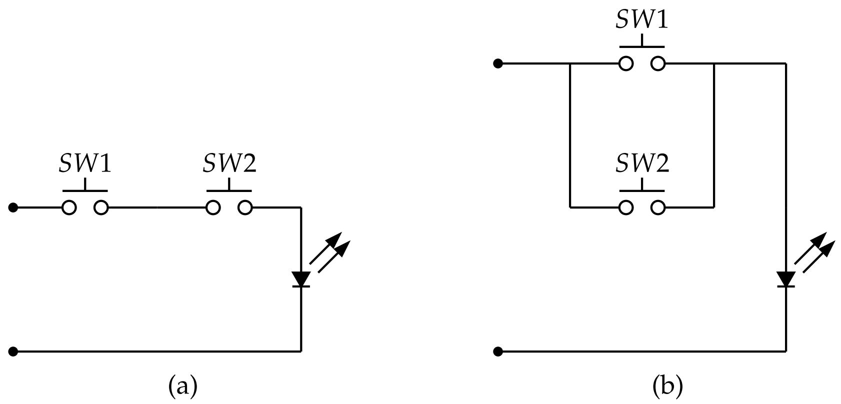


Figure 4: (a) The LED will turn on when SW1 *AND* SW2 are closed; (b) The LED will turn on when SW1 *OR* SW2 are closed

The switching behaviour in Fig. 4 gives us a clue as to how we might create logic circuits. If we treat a "high" voltage (one that will turn the LED on) as a logical 1, and a "low" voltage (no current through the LED) as a logical 0, then the switch configurations in Fig. 4 allow us to express the logical functions AND and OR of the switch states. This is, in fact, exactly how *digital logic* works: logical values are represented by different *voltages* ("low" and "high"). Provided that we can come up with a way to set switch states electronically (instead of having to manually press buttons), we can build complex logic circuits.

The Transistor Switch

So, how can we switch between different voltages under some kind of electronic[1] control? One way to do so–the way that's most common in modern computing–is to use a semiconductor device called a Field Effect Transistor (FET). A MOSFET (Metal Oxide Semiconductor FET) is constructed from semiconductor[2] materials with differing properties that interact to produce a controllable conductive channel, as shown in Fig. 5.

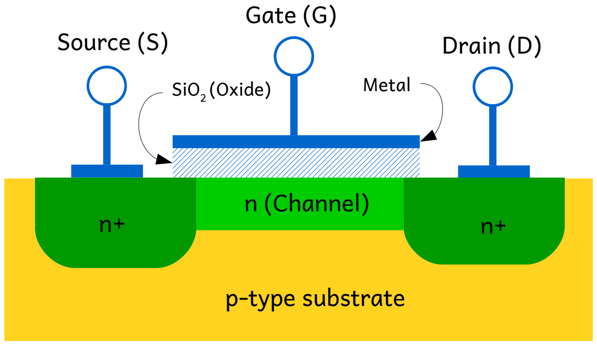


Figure 5: Basic structure of the n-channel MOSFET (based on a diagram from Sedra and Smith).

In a FET the channel runs between the "drain" and the "source". The *"conductive width"* of the channel is controlled by the *gate* voltage with respect to the *source*. That voltage has the symbol V_{GS}.

[1] That is, without manually pushing a button.

[2] We won't get into the details of how semiconductors work in this class. ENEL270 includes an introduction to semiconductors.

You can think of the FET as a bit like a hose-pipe:

- In an *n-channel* device the hose-pipe is squeezed closed unless V_{GS} is *positive* (the gate has a higher voltage than the source). As V_{GS} is made *more positive* the channel opens up and allows more current to flow from the *drain* to the *source*.
- In a *p-channel* device the hose-pipe is squeezed closed unless V_{GS} is *negative* (the gate voltage is lower than the source). As V_{GS} is made more *negative* the channel opens up and allows current to flow from the *source* to the *drain*.

Small changes in the gate voltage can lead to much larger changes in the drain-source current[3]. This is a desirable *analog* characteristic of transistors. However in *digital* or switching circuits we are only interested in operating the transistor as fully conductive or not — hose-pipe fully open or fully squeezed closed — so the transistors are usually wired up (biased) to ensure that "high" or "low" gate voltages result in a fully open or closed channel.

As you may have noticed, the n-channel and p-channel MOSFETs work in a *complementary* way: the n-channel will conduct when the gate is at a "high" voltage relative to the source, while the p-channel will conduct when the gate is at a "low" voltage relative to the source (see Fig. 6). The combination of the two kinds of FETs is known as **CMOS** (complementary MOSFETs), and can be used to build logic circuits.

You can implement logic gates using almost anything that allows you to create switching behaviour. Here are logic gates as patterns of dominoes: https://en.wikipedia.org/wiki/File:Domino_Logic_Gates.svg

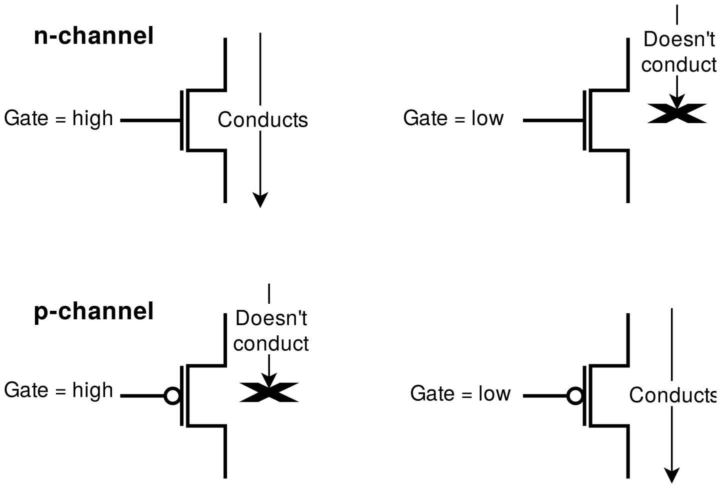


Figure 6: Behaviour of n-channel and p-channel MOSFETs

Example: The circuit in Fig. 7 acts as an *inverter* (NOT). When the input is high (logical 1), the n-channel FET connects the output to the "ground" (or 0V). When the input is low (logical 0), the p-channel FET connects the output to the V_{CC}. The relationship of output to input can be represented by the following *truth table*:

v_{in}	v_{out}
High	Low
Low	High

[3] In other words, variations in gate voltage are *amplified* by the transistor.

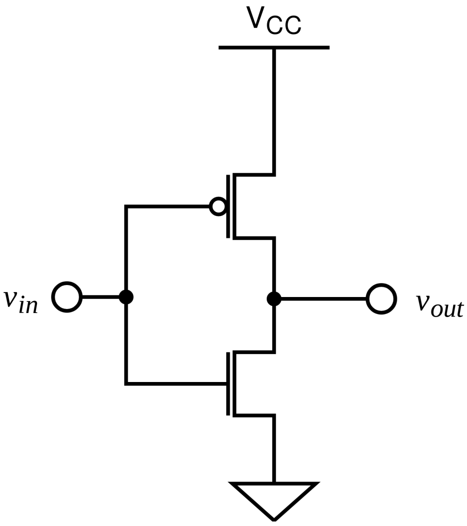


Figure 7: CMOS inverter

Example: The circuit in Fig. 8 acts as an *NAND*. The relationship of output to inputs can be represented by the following *truth table*:

v_A	v_B	v_{out}
Low	Low	High
Low	High	High
High	Low	High
High	High	Low

You should work through the different states of the transistors for different input combinations to make sure you udnerstand why the NAND circuit works.

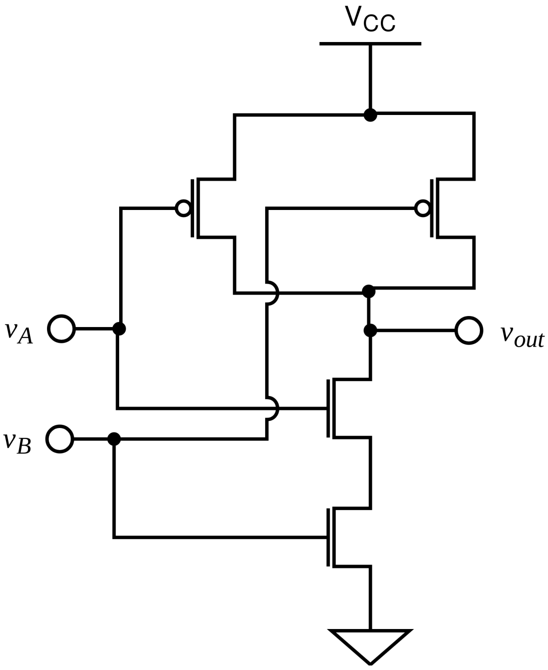


Figure 8: CMOS NAND gate

It's important to note that the discussion of circuit behaviour above makes some simplifying assumptions. We've assumed steady state voltages[4], which allows us to abstract away the analog behaviour of the transistors and think in digital terms. But there are some important electrical properties of transistors that mean that a computer is fundamentally an **analog** device, even if we mostly consider it from a digital perspective.

Important Properties of Electronic Switches

THERE ARE THREE INTERESTING ISSUES we need to be familiar with in regard to electronic switches:

- Propagation delay
- Fan-out and fan-in
- Noise margin

Propagation Delay

Real switches cannot change state in zero time. It takes a small, but finite, time for charge carriers to cross semiconductor junctions and for recombination to occur within the transistor switch; this is a *capacitative*[5] effect and means that dV/dt must be finite.

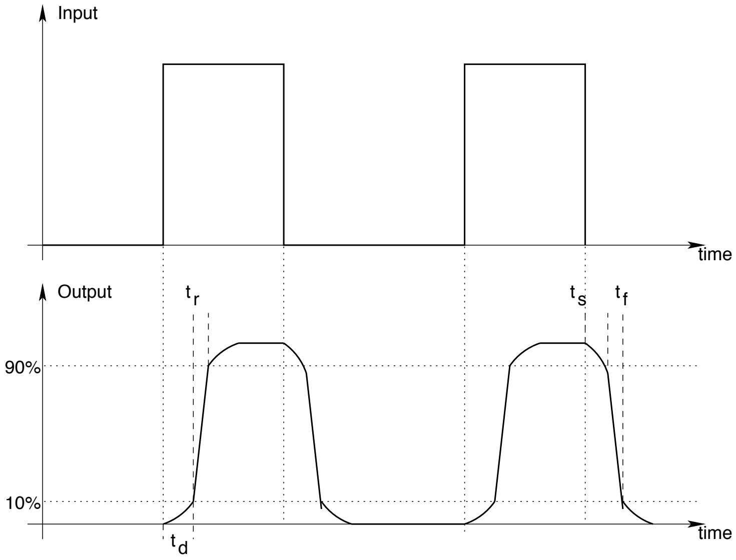


Figure 9: Propagation delay of a change in switch state due to instantaneous input changes

The important design parameters, shown in Fig. 9, are:

- t_d is the turn-on delay; time to change from 0% to 10% of final value.
- t_r is the rise time; the time to change from 10% to 90% of final value.
- t_s is the turn-off delay[6]; the time to change from 100% to 90% of initial value. Normally $t_s > t_d$.

[4] That is, we haven't thought about what happens when the voltage *transitions* from low to high or high to low.

[5] Recall that the current through a capacitor is $i(t) = Cdv(t)/dt$

[6] sometimes called "storage delay"

- t_f is the fall time; the time to change from 90% to 10% of initial value. t_f is similar to t_r, both are greatly affected by transistor capacitances.

All of the above times are important in digital design. They collectively determine the switching speed of a device, and the delay that occurs when signals propagate through digital circuits. The quoted "propagation delay" found on datasheets for logic circuits is often specified as the time for a signal to transition to 50% of its final level, and thus depends on both the turn-on (or turn-off) delay and the rise (or fall) time. So, if we want a clock speed of (say) 100 MHz (i.e., a clock period of 10 ns), we need devices with t_d, t_r, etc. $<<$ 10 ns.

Fan-Out and Fan-In

Fan-out refers to the maximum number of switch inputs that can be driven by the output of another switch, as depicted in Fig. 10(a). The driver switch needs to supply or "source" current to the other switches (although the current may flow in either direction) to make sure that they switch reliably.

Fan-in refers to the number of switch outputs that can safely be connected to the input of another switch, Fig. 10(b). The switch input needs to accept or "sink" current from the other switches (again the current may flow in either direction).

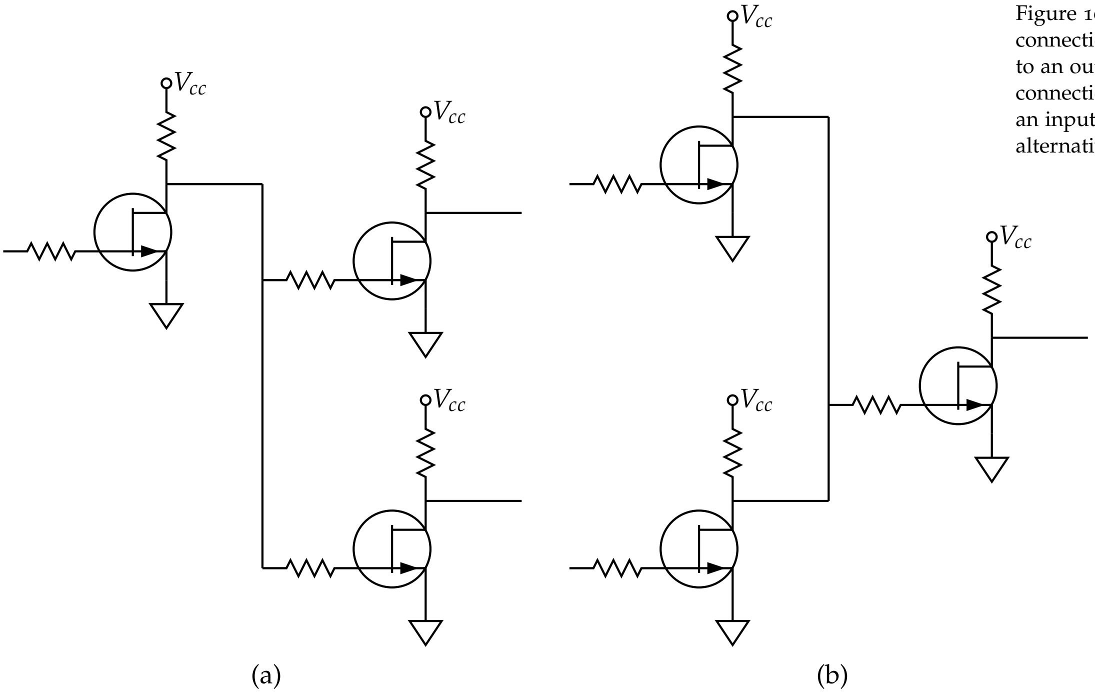


Figure 10: (a) Fan-out refers to the connection of down-stream inputs to an output, (b) Fan-in refers the connection of up-stream outputs to an input. The symbols used here are alternative symbols for n-channel FETs.

Noise Margins

While it is nice to think of our switches acting in an on/off or digital sort of way, they are in fact *analog* devices. Logic levels are therefore never defined precisely but occupy a range of possible values.

A manufacturer, as shown in Fig. 11, will ensure that:

1. A logic high output will be between V_{OH} and V_{CC}.
2. An input will be recognised as a high between V_{IH} and V_{CC}. Note that $V_{\mathrm{IH}} < V_{\mathrm{OH}}$.
3. A logic low out will be between V_{sat} and V_{OL}.
4. An input will be recognised as a low between 0 and V_{IL}. Note that $V_{\mathrm{OL}} < V_{\mathrm{IL}}$.

V_{CC} is the power supply voltage, and represents the highest possible voltage in the circuit. You'll sometimes see it labelled V_{DD} or V_{SS} instead.

V_{sat} is the voltage at the output of a switch that is fully on.

The differences,

$$\begin{aligned}\Delta 1 &= V_{\mathrm{OH}} - V_{\mathrm{IH}}\\ \Delta 0 &= V_{\mathrm{IL}} - V_{\mathrm{OL}}\end{aligned}$$

are called the *noise margins*.

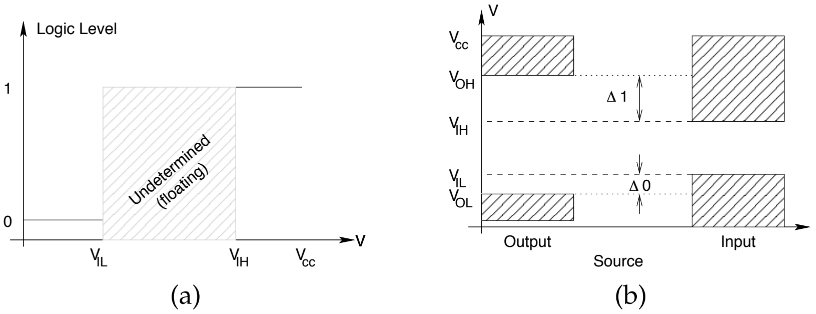


Figure 11: In real systems everything is analog; we just interpret voltage regions to have a digital meaning.

Connecting to a microcontroller

The issues that we have looked at above need to be considered when we connect other electronics to a microcontroller. And we need to take care that the voltage levels that are present are compatible at both ends of each such connection. Fortunately manufacturers provide good information to help and there are a relatively small range of voltages that are in use. For digital circuits, you are unlikely to encounter components which operate outside the range of 1.8 V to 5 V (HIGH level and Vcc) or lower than 0 V (LOW level and Ground).

For a connection to transfer meaningful information, there has to be some flow of current, albeit in some cases a very small current. In choosing components, a computer engineer generally wants to set the current as low as possible to guarantee the correct transfer. However designing to keep currents too low may make the circuit susceptible to interference from stray magnetic fields. There always needs to be a healthy margin for noise and to allow for the variation which occurs in the manufacture of components.

Example: Let's return to the example that we started with (Fig. 12), which is based on actual connections on the UCFK.

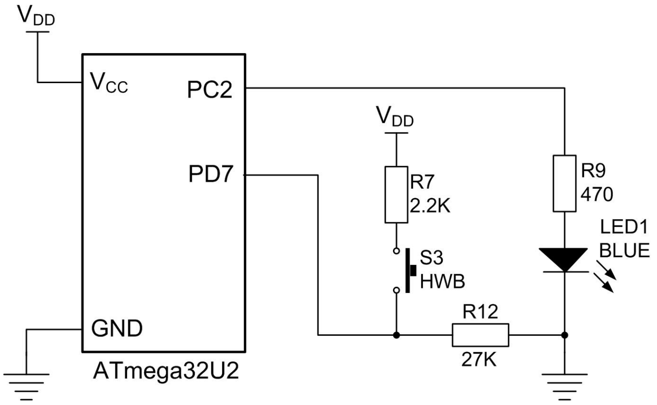


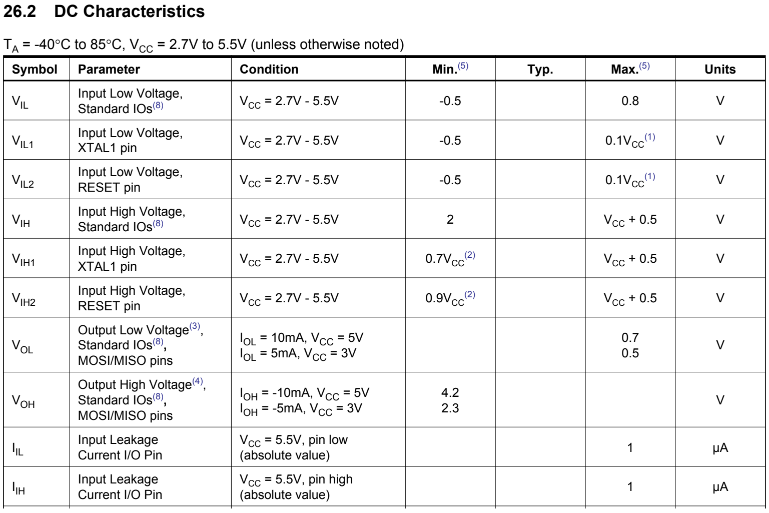
Figure 12: Example of interfacing to GPIO pins on the microcontroller - part of the UCFK. The supply voltage for the UCFK V_{DD} is 5 V.

A switch "**HWB**" is connected to pin **PD7** (bit 7 of Port D) which must be configured for input. A blue LED ("**LED1**") is connected to pin **PC2** (bit 2 of Port C) which must be configured for output. Both have resistors attached to control the voltage and the current which flows.

For the switch, when it is open (not pushed), PD7 is connected to ground by a 27 K ohm resistor. The specifications for the ATmega32U2 show that I_{IL} is 1 μA, so the voltage on PD7 is 0.027 V, well below the maximum value for V_{IL}, shown in Fig. 13.

When the switch is closed (pushed), a "voltage divider" is formed by R7 and R12, so that the voltage on PD7 is then $27/(27+2.2) \times 5 =$ 4.62 V, well above the minimum value for V_{IH} when V_{CC} is 5 V. Note that in this case the input leakage current for the input (I_{IH} = 1 μA) has negligible effect since 5 / (27 K + 2.2 K) = 171 μA flows through the resistors (using Ohm's Law).

For LED1, when LOW is output on PC2, the output voltage on the pin is no higher than 0.7 V (V_{OL}), not enough to forward bias the diode and generate light. However, when HIGH is output on PC2, the output voltage on the pin is at least 4.2 V (V_{OH}), enough to forward bias the diode. The diode has a forward drop of 1.2 V, which means that there is at least 3 V across the resistor R9 and (by Ohm's Law) a current of at least 6.4 mA will flow to generate blue light.



26.2 DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.[5]	Typ.	Max.[5]	Units
V_{IL}	Input Low Voltage, Standard IOs[8]	V_{CC} = 2.7V - 5.5V	-0.5		0.8	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.1V_{CC}$[1]	V
V_{IL2}	Input Low Voltage, RESET pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.1V_{CC}$[1]	V
V_{IH}	Input High Voltage, Standard IOs[8]	V_{CC} = 2.7V - 5.5V	2		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 2.7V - 5.5V	$0.7V_{CC}$[2]		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage, RESET pin	V_{CC} = 2.7V - 5.5V	$0.9V_{CC}$[2]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[3], Standard IOs[8], MOSI/MISO pins	I_{OL} = 10mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V			0.7 0.5	V
V_{OH}	Output High Voltage[4], Standard IOs[8], MOSI/MISO pins	I_{OH} = -10mA, V_{CC} = 5V I_{OH} = -5mA, V_{CC} = 3V	4.2 2.3			V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA

Figure 13: DC characteristics of GPIO pins from the ATmega32U2 microcontroller datasheet.