Exercise 6: DCF77 Time Signal

Overview

The time signal transmitter DCF77 is a long wave sender located in Mainflingen, Germany (near Frankfurt am Main). It was established in 1973 and distributes a signal that is used to set clocks and watches to display the correct time and date automatically. The distributed time is actually the officially valid time in Germany. The sender has a working frequency of 77.5 kHz and a power rating of 50 kW.

In this lab course exercise, we will try to decode the DCF77 time signal using a microcontroller and display the current date and time in our debug console. For this purpose, we use a special DCF77 receiver which already provides us with a "clean" digital signal that we can directly input into the microcontroller. Since the signal quality may vary depending on the weather conditions, we have also prepared a DCF77 signal simulator that transmits a well-known date and time value.

DCF77 Time Signal Structure

The signal amplitude $a(t)$ of DCF77 is modulated in intervals of one second. At the beginning of every second (except for the last second of each minute), the signal becomes low for either 100 ms or 200 ms (see also figure 1[1]). The different lengths of the low signal interval are used to binary encode information like the current minute, hour as well as the date:

- A low signal for 100 ms corresponds to a binary *zero* while a low signal for 200 ms corresponds to a binary *one*.
- The fact that the last (59th) second of a minute does not show a low signal can be used to detect the beginning of a new minute (i.e., the next time the signal is low a new minute has just started).
- The transmitted binary information correspond to the minute, hour, day, day of the week, month and year (only the last two digits) of the *upcoming* minute. Numbers are encoded as BCD (Binary Coded Digits), which means that unlike normal binary encoding, where the ith bit has weight 2^i ($i \geq 0$), the weights of the bits are: 1, 2, 4, 8, 10, 20, 40, 80, ...

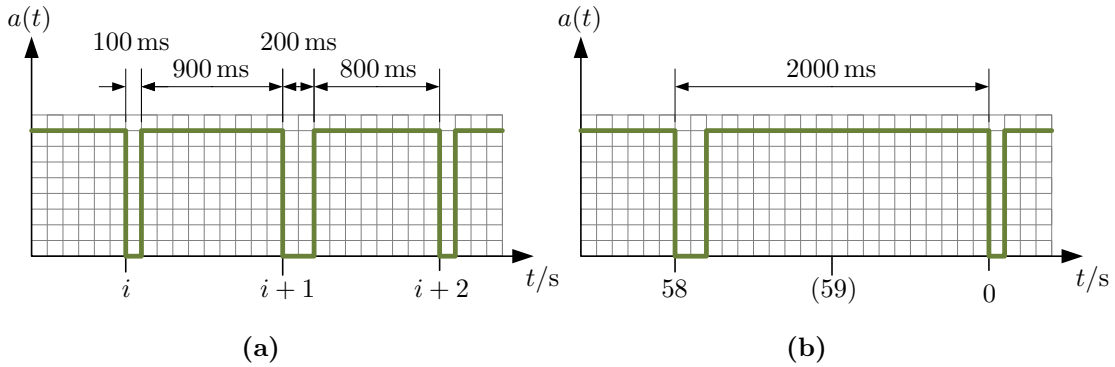

Figure 1: DFC77 Signal Plot: **(a)** Second mark: binary *zero* on the left side (100 ms low followed by 900 ms high) and binary *one* in the middle (200 ms low followed by 800 ms high); **(b)** Minute mark: there is no low signal in the 59th second.

[1]The signal shown in this figure is the one output by the receiver we use. The original terrestrial DCF77 signal actually looks somewhat different, but in this exercise we will only process the digital signal shown here.

Exercise 6.1

a) Can you imagine why binary coded digits are used instead of binary coded numbers?

b) How are the following decimal numbers represented in binary and BCD notation: 7, 12, 16, 59?

c) Besides the synchronization marks for the beginning of a second and a minute, the DCF77 signal has 59 bits data payload per minute. Browse the Internet to find out what the different bit values are used for.

d) Which information and/or bits (including parity bits) do we need to process if we want to obtain the following values: second s, minute n, hour h, day of the month d, day of the week w, month m and year y?

e) Imagine we have stored all data bits in a vector (array) $\vec{b} = \{b_0, b_1, \ldots, b_{59}\}$. How are the actual values for n, h, d, w, m and y calculated? Write down the formulas. Which days of the week do the values $w = 1$ and $w = 6$ correspond to?

f) Why are there parity bits in the signal? What kind of parity is used (odd or even)? Write down formulas involving the parity bits that must hold for the signal to be considered valid.

g) Why can't we be sure that the received data is valid if the parity check does not show any problems? Give an example where the parities are correct, but the received data are not.

h) Which checks besides the parity check can we implement to ensure that only valid data is decoded? Take all available information into account.

Microcontroller-based Analysis of the DCF77 Signal

Exercise 6.2

a) Connect the DCF77 signal cable (yellow) to `PD6` and the ground cable (brown) to `GND`. Write a program that directly outputs the signal received at that pin on LED0. Verify that you can see the different low signal intervals and the minute mark.

b) Collect the data bits of the DCF77 signal and detect the minute mark. Note that when the signal quality is bad, edge detection using pin change interrupts might become unreliable. It is recommended that you configure Timer0 to raise an interrupt every 10 ms to sample the signal instead.

Print debugging information to the console. Once all the information is available, perform parity checks and other checks as specified in exercise 6.1 **h)** and, on success, format the data (e.g., `Thursday, 24.11.2011 15:45:00`). Update the time every second.

Exercise 6.3

- Optional: Implement a button that will, when pressed, trigger another synchronization while the date and time is already known and being output to the debug console. The program should continue to output the date and time to the debug console and switch to the new date and time when it has been re-detected. This is the common behavior of DCF77-based clocks, which will typically synchronize to the time signal regularly during the night.