IAR Application Note AVR - 011
Radio controlled clock (DCF77) using AVR

SUMMARY

A radio signal sent out near Frankfurt gives the time with a very high degree of accuracy. This time signal is decoded by an Atmel AVR (STK200 equipped with an AT90S8515) and sent to a standard LCD display.

KEYWORDS

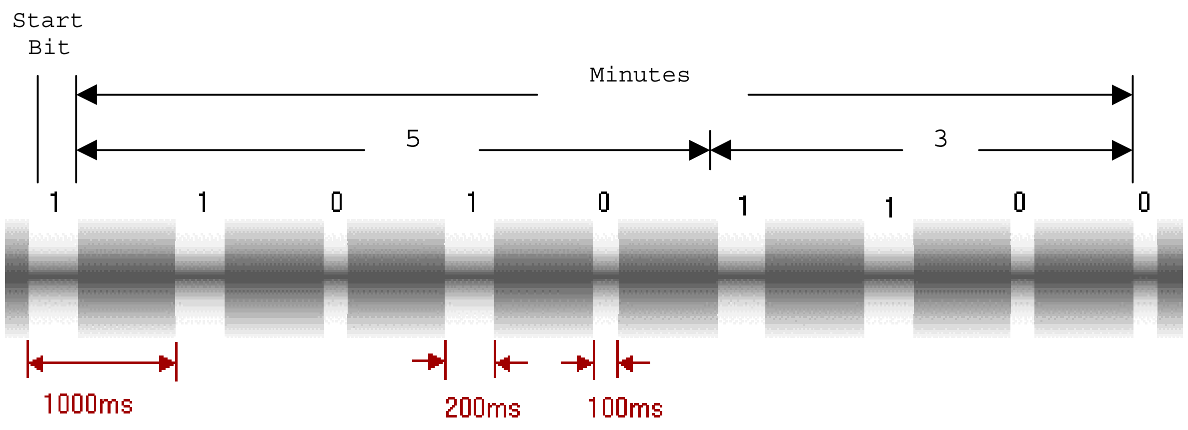
STK200, interrupt, timer, LCD

The problem to be solved

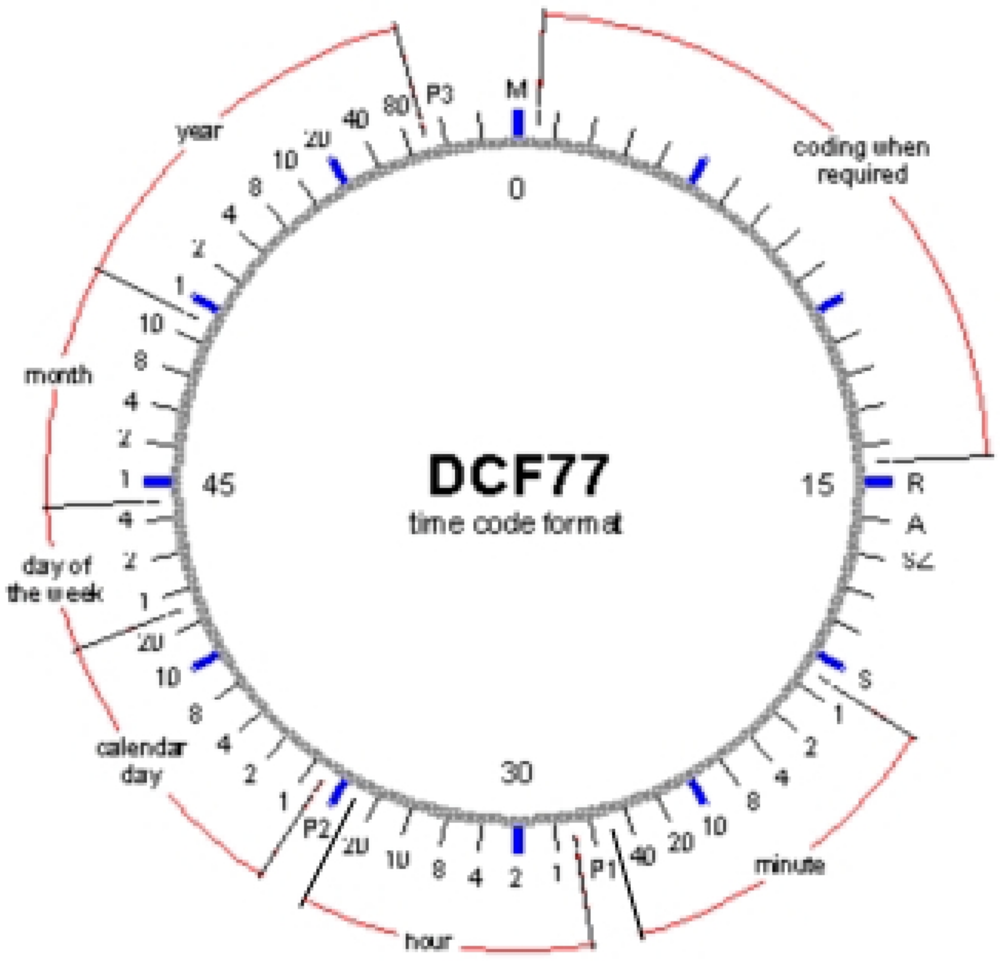
The signal received by a small DCF antenna has to be decoded to get the actual time and date. This information is to be displayed on a LCD with a HD44780 compatible controller.

The difficulties involved

The DCF77 is sending with a carrier frequency of 77.5kHz.
Time signal: The carrier is amplitude modulated with second marks. At the beginning of each second (with the exception of the 59th second of each minute), the carrier amplitude is reduced to 25% for the duration of either 0.1 or 0.2 seconds. The start of the carrier reduction marks the precise beginning of the second. The minute is marked by the absence of the previous second mark. The second marks are phase-synchronous with the carrier. There is a relatively large uncertainty possible in the time of the second mark, which depends on the position of the receiver. There are several causes for this; the relatively low bandwidth of the antenna, space waves, and other sources of interference. Despite this, it is possible to achieve an accuracy that exceeds 1ms at distances of several hundred kilometres.



Time code:
The transmission of the numerical values for minute, hour, calendar day, day of the week, month, and year are BCD-encoded by the pulse duration modulation of the second marks. A second mark with a duration of 0.1s encodes a binary “0” and a duration of 0.2s encodes a binary “1”. The order of encoding is shown in the following diagram:



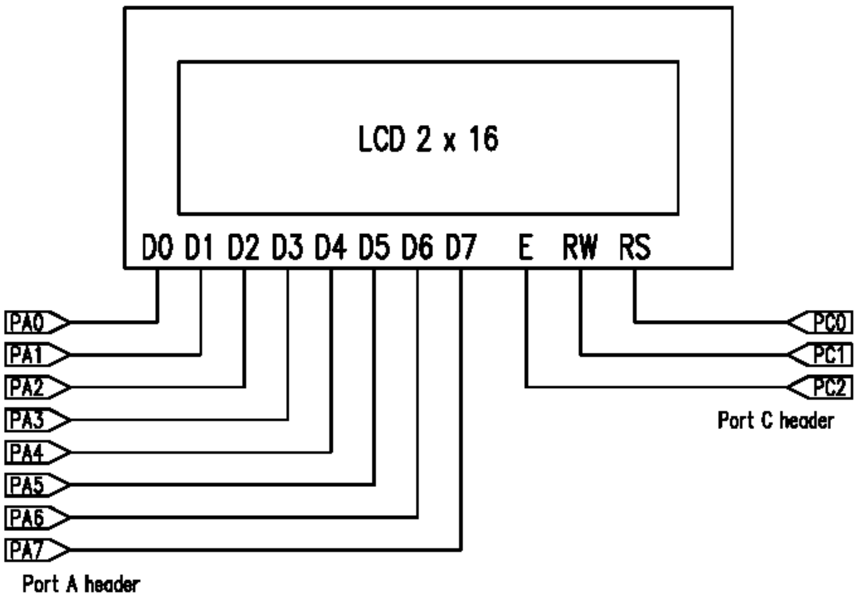
The three test bits P1, P2 and P3 extend the 3 major sections of the time code (7 bits for minutes, 6 bits for the hour and 22 bits for the date, including the week day number) to maintain an even count of 1's. Second marks 17 and 18 indicate the time system for the transmitted time codes. In the case of transmission of CET (Central European Time), mark 18 has a duration of 0.2s and mark 17 a duration of 0.1s. If CEST (Central European Summer Time) is being transmitted, this is reversed. Furthermore, an approaching transition from CET to CEST or back is announced by extending mark 16 from 0.1s to 0.2s for one hour prior to the changeover.

The solution

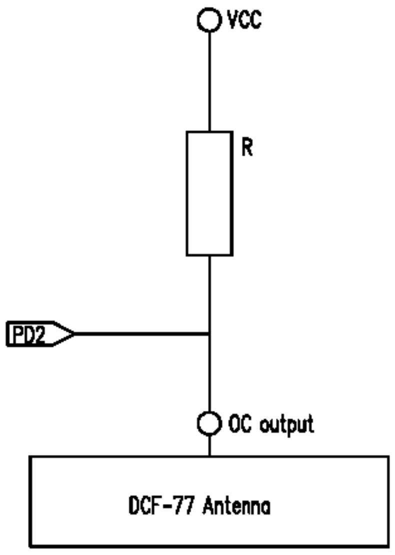
1. Hardware description:
   We use the Atmel STK 200 Starter Kit with an AT90S8515 without external SRAM, a standard 2 x 16 character LCD display for the time

display and an active antenna from HKW Elektronik to receive the DCF-77 signal.

Connecting the display :

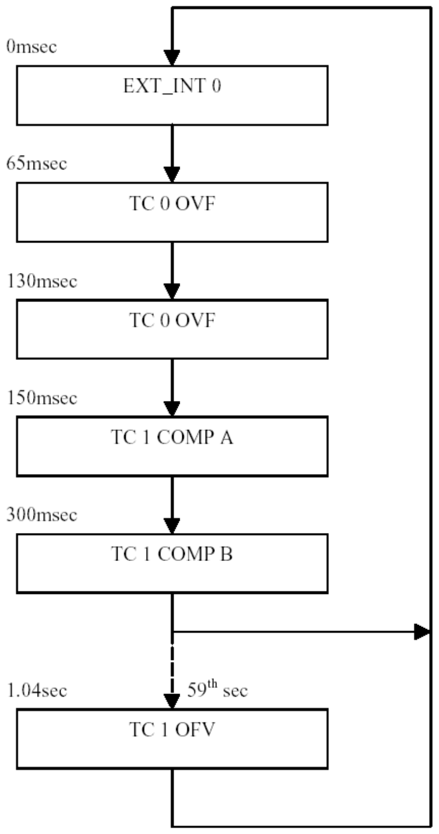


The open collector output from the DCF-77 antenna is connected to Port D pin 2, the input pin for the external interrupt 0.



2. Implementation

The program decodes the DCF-77 time telegram, checks the parity bits, and displays the time on the LCD display.
The program flow is controlled by the external interrupt 0 and the timer 1 interrupts. The external interrupt routine initializes the timers, refreshes the display, and disables the external interrupt. Then the program takes two samples from PIND 2, 65ms and 130ms after the external interrupt, to detect a logical 0 or 1 from the DCF-77 signal. About 20ms later the timer 1 compare A interrupt occurs and the information is stored. Another 150ms later external interrupt 0 is enabled, and the program waits for the next second mark. In the 59th second there is no mark to indicate the beginning of the next minute. This overflows timer 1 and the raw time information is decoded by the interrupt routine. The following second the new time is displayed.



With this information the C code is easy to understand. In addition, it is generously commented.

User benefits

This example is meant to show more than just how to implement the radio decoder. It is also intended to demonstrate some basic programming rules. In the C code you can find examples how to use extended keywords to optimize pointer access, and how to put variables into registers. You can also see how to define bit variables on the AVR, which does not have good support for bits at all. (Bit operations are possible only in registers and in the SFR area.)

Conclusions

Anybody who wants to program an AVR should at least have a look at the C code as there are many good examples of how to solve several tasks in your daily work with an AVR.

References

This application note is based on the work of two students at FH Furthwangen. The code has been ported to the new AVR compiler and has been slightly optimized.

Digital Systems Project –
WS 99/00 Willi Draxler, Roger Koller, Jürgen Müller, Manfred Pannewitz
SS 00 Christian Rudel, Christian Bär, Philipp Röttele