PIC16F628 exercise, Hans Sundgren

DCF77 transmitter simulator

Function

The simulator should create a binary pulse train equal to the DCF77 transmitter in Germany. The date, time, daylight savings time and day of the week can be set.

By connecting the simulator to a DCF77 radio controlled clock, the function can be verified.

User interface

LCD diplay (2x16), ASCII input

[requires 6 pins]

```
2011-04-23 DST=1
23:32:56  day=Mo
```

Controls

START/STOP, push-button [requires 1 pin]
Start/stop of the internal "real-time" clock and start of pulse output

SET/SAVE, push-button [requires 1 pin]
To step through the settings. For each press, the display steps forward and highlights the digit (or area) to be set:

year, month, day, daylight savings time,
hour, minute, second, day of the week.

The values are saved automatically at the end of the setting.

INCREASE, push-button [requires 1 pin]
Increase value

Output

Simulated DCF77 signal according to set values, binary signal **[requires 1 pin]**
Logical zero = 0.1 s inverted pulse every second
Logical one = 0.2 s inverted pulse every second
Marker pulse before a new minute = no pulse

Remark not significant for the simulator:
The DCF transmitter uses an AM modulated signal where the normal state is 100% power and the power is decreased to 75% during the pulse.

Components

PIC16F628 with standard peripheral components
Crystal 32,768 kHz with capacitors for external clock oscillator
LCD display
Pushbuttons

Program parts

Interrrupt, 1 Hz from Timer1

Steps up the time of the internal transmitter clock
Calls function for output pulse

Every 59:th second:
Updating the 60-bit-array.
Conversion from decimal to BCD for different time data
Inserting parity bits 3 times in a 60-bit-array

Main program

Call function Read_time during startup

Loop:
Checking START/STOP button
If START/STOP button is pressed
call function for set time and store time

Functions

Output_pulse:	Reads 60-bit-array and generates corresponding output pulse.
Set_time:	Sets internal time on display with pushbuttons
Store_time:	Reads the set time and stores it in EEPROM
Read_time:	Reads the time from EEPROM and set the clock accordingly

Verification of function

Preparations

1. Disassamble a commercial DCF77 radio controlled clock, for example from www.kjell.com.
2. Identify the pulse signal between the AM radio cirucit and the decoding & display circuit. This position can be found by visual inspection of the PC board and by measurement with a voltmeter.
3. Attach one wire to the pulse position and one wire to the ground.
4. Connect the wires to the output of the DCF77 transmitter simulator. Perhaps some intermediate electronic interface is required,
5. Remove the antenna from the radio receiver cirucit or cut the PC board wire to the radio circuit.

Verification

1. Start the simulator and set a future time on the display.

2. Wait some minutes for the radio clock to decode and synchronise the bits.
3. Check that the time of the radio clock matches the DCF77 simulator.

Optional exercises

More sophisticated setting of time
- using up/down, left/right buttons (perhaps joystick ?)
- using step frequency dependant on time of press

Instruction messages on display
- "I am a DCF77 transmitter"
- "Set the time."
- "Time is stored."
- "Stop"

Audible feedback of "zeros" and "ones" pulse output
- bip, bip, bip, beeeep, bit, beeeep ...

Very optional exercise

Use the optional 15 first bits in each minute to simulate the weather forecast used by the radio controlled weather stations.
But... the complete protocol is unknown, only parts of it.

Bit	**Name**	**Description**	**Comment**
0...14	-	Reserved	Meteo weather data
15	R	Send antenna	0 = Standard antenna / 1 = **R**eserve antenna
16	A1	**A**nnouncement **1**	1 = Next hour a Daylight Saving Time (DST) change occurs
17	Z1	Time **Z**one bit **1**	0 = Winter / 1 = Summer (DST)
18	Z2	Time **Z**one bit **2**	0 = Summer / 1 = Winter (usually the opposite of Z1)
19	A2	**A**nnouncement **2**	1 = Next hour an extra second is inserted (leap-second)
20	S	**S**tartbit	Always 1, startbit coded time information
21...27		Minutes	7 bit, BCD, LSB first (00...59)
28	P1	**P**arity bit **1**	Even parity for all received bits from the minutes
29...34		Hours	6 bit, BCD, LSB first (00...23)
35	P2	**P**arity bit **2**	Even parity for all received bits from the hours
36...41		Day of the month	6 bit, BCD, LSB first (01...31)
42...44		Day of the week	3 bit, BCD, LSB first (1 = Monday...7 = Sunday)
45...49		Month	5 bit, BCD, LSB first (01...12)
50...57		Year	8 bit, BCD, LSB first (00...99)

58	P3	**P**arity bit **3**	Even parity for all received bits from the date

Time coding scheme

The bits are coded as follows:

bit	symbol	bcd code	description
0	M		Minute mark
1-14			Encoded ;-) weather information
15	R		1 if backup antenna is used
16	A1		announcement day light saving time
17-18	Z1, Z2		01 = winter, 10 = summer
19	A2		announcement leap second
20	S		start time information (allways 1)
21-27		-xxx.xxxx	minutes (7 bits, lsb first)
28	P1		paritycheck 21-27, even parity
29-34		--xx.xxxx	hours (6 bits, lsb first)
35	P2		paritycheck 29-34, even parity
36-41		--xx.xxxx	date (6 bits, lsb first)
42-44		----.-xxx	weekday (3 bits, lsb first, 1 = Monday)
45-49		---x.xxxx	month (5 bits, lsb first)
50-57		xxxx.xxxx	year (8 bits, lsb first)
58	P3		paritycheck 36-57, even parity
(59)	-		Minute mark