```
/* Copyright 2004 - KOCH Engineering                                    */
/* Code to create a DCF77 watch in a standard 2x16 character LCD display  */
/* Input is from a radio receiver module using a ferite antenna.        */

#include <pic.h>
#include "bluebird.h"   // BYTE
#include "dcf77.h"
#include <stdio.h>  // printf, sprintf
#include <string.h> // string


/*
---------------------------------------------------------------------------
  dcf77_settime()
  ~~~~~~~~~~~~~~~
  This is function is called from the timer interrupt when a valid time
  has been received from DCF77.
---------------------------------------------------------------------------
*/

void dcf77_settime (BYTE hour,      /* 0-23 */
                    BYTE minute,    /* 0-59 */
                    BYTE date,      /* 1-31 */
                    BYTE month,     /* 1-12 */
                    BYTE year,      /* 0-99 */
                    BYTE weekday)   /* 1-7, 1=Monday */
{
  watch_second  = 0;
  watch_hour    = hour;
  watch_minute  = minute;
  watch_date    = date;
  watch_month   = month;
  watch_year    = year;
  watch_weekday = weekday;

  NewTelegramFlag = TRUE;
  RC0 = 1;      // tænder telegram modtaget lampe når denne funktion køres
}


/*
---------------------------------------------------------------------------
  DCF77 Outer state machine
  ~~~~~~~~~~~~~~~~~~~~~~~~~
  The outer state machine receives '0's, '1's, and minute synchronisation
  events from the inner state machine and assembles this information to a
  valid time and date.  When a valid time and date has been received, the
  function dcf77_settime() is called with the received time and date.
---------------------------------------------------------------------------
*/
void dcf77_osm (BYTE event)
{
    static BYTE parity;     // skal huskes fra gang til gang derfor static

    event_copy = event;

    switch (osm_state)
    {
        case OSM_START:
        case OSM_FUTURE01:
```

```
        case OSM_FUTURE02:
        case OSM_FUTURE03:
        case OSM_FUTURE04:
        case OSM_FUTURE05:
        case OSM_FUTURE06:
        case OSM_FUTURE07:
        case OSM_FUTURE08:
        case OSM_FUTURE09:
        case OSM_FUTURE10:
        case OSM_FUTURE11:
        case OSM_FUTURE12:
        case OSM_FUTURE13:
        case OSM_FUTURE14:
        // These bits aren't used
                break;
        case OSM_ANTENNA:       //  antenne status
                dcf77_telegram.antenna = event;
                break;
        case OSM_BEFORE_DST:    //  tidszoneændring
                dcf77_telegram.time_zone_change_announce = event;
                break;
        case OSM_DST:           // Z1 modtages
                dcf77_telegram.time_zone = event;       // Z1 gemmes
                break;
        case OSM_NO_DST:        // Z2 modtages
                dcf77_telegram.time_zone += (event *2); // Z2 gemmes
                break;
        case OSM_LEAP_SECOND:
                dcf77_telegram.leap_second_announce = event;
                break;

        case OSM_TIMESTART:
            // This bit must be '1'
            dcf77_telegram.minute       = 0;
            dcf77_telegram.hour         = 0;
            dcf77_telegram.day          = 0;
            dcf77_telegram.day_of_week  = 0;
            dcf77_telegram.month        = 0;
            dcf77_telegram.year         = 0;
            if (event == EVENT_ONEBIT)  // check for startbit
                osm_state = OSM_MIN01;      // startbit OK, continue
            else
            {                               // missing startbit - something
wrong
                if (event == EVENT_MINUTESYNC)
                    osm_state = OSM_START;      // unexpected minute sync
                else
                    osm_state = OSM_RESYNC;     // startbit bad
            }
           return;

        case OSM_MIN01: parity = 0;             // parity nulstilles
                        dcf77_telegram.minute = event;
                        break;
        case OSM_MIN02: dcf77_telegram.minute += (event * 2); break;
        case OSM_MIN04: dcf77_telegram.minute += (event * 4); break;
        case OSM_MIN08: dcf77_telegram.minute += (event * 8); break;
        case OSM_MIN10: dcf77_telegram.minute += (event * 10); break;
        case OSM_MIN20: dcf77_telegram.minute += (event * 20); break;
        case OSM_MIN40: dcf77_telegram.minute += (event * 40); break;
```

```
        case OSM_PARITY1:
            dcf77_telegram.parity1 = (event);

            if (event == parity)
                osm_state = OSM_HOUR01;     // parity OK, continue
            else
            {
                if (event == EVENT_MINUTESYNC)
                    osm_state = OSM_START;      // unexpected minute sync
                else
                    osm_state = OSM_RESYNC;     // parity bad
            }
            return;

        case OSM_HOUR01:  parity = 0;           // parity nulstilles
                          dcf77_telegram.hour = event;          break;
        case OSM_HOUR02:  dcf77_telegram.hour += (event * 2);   break;
        case OSM_HOUR04:  dcf77_telegram.hour += (event * 4);   break;
        case OSM_HOUR08:  dcf77_telegram.hour += (event * 8);   break;
        case OSM_HOUR10:  dcf77_telegram.hour += (event * 10);  break;
        case OSM_HOUR20:  dcf77_telegram.hour += (event * 20);  break;

        case OSM_PARITY2:
            dcf77_telegram.parity2 = event;

            if (event == parity)
                osm_state = OSM_DATE01;     // parity OK, continue

            else
            {
                if (event == EVENT_MINUTESYNC)
                    osm_state = OSM_START;      // unexpected minute sync
                else
                    osm_state = OSM_RESYNC;     // parity bad
            }
            return;

        // day of month
        case OSM_DATE01:    parity = 0;
                            dcf77_telegram.day = event;         break;
        case OSM_DATE02:    dcf77_telegram.day += (event * 2);  break;
        case OSM_DATE04:    dcf77_telegram.day += (event * 4);  break;
        case OSM_DATE08:    dcf77_telegram.day += (event * 8);  break;
        case OSM_DATE10:    dcf77_telegram.day += (event * 10); break;
        case OSM_DATE20:    dcf77_telegram.day += (event * 20); break;

        // day of week
        case OSM_WEEKDAY01: dcf77_telegram.day_of_week = event;
break;
        case OSM_WEEKDAY02: dcf77_telegram.day_of_week += (event * 2);
break;
        case OSM_WEEKDAY04: dcf77_telegram.day_of_week += (event * 4);
break;

        // month
        case OSM_MONTH01:   dcf77_telegram.month = event;           break;
        case OSM_MONTH02:   dcf77_telegram.month += (event * 2);    break;
        case OSM_MONTH04:   dcf77_telegram.month += (event * 4);    break;
        case OSM_MONTH08:   dcf77_telegram.month += (event * 8);    break;
        case OSM_MONTH10:   dcf77_telegram.month += (event * 10);   break;
```

```
        // year (only tens and ones)
        case OSM_YEAR01:    dcf77_telegram.year = event;            break;
        case OSM_YEAR02:    dcf77_telegram.year += (event * 2);     break;
        case OSM_YEAR04:    dcf77_telegram.year += (event * 4);     break;
        case OSM_YEAR08:    dcf77_telegram.year += (event * 8);     break;
        case OSM_YEAR10:    dcf77_telegram.year += (event * 10);    break;
        case OSM_YEAR20:    dcf77_telegram.year += (event * 20);    break;
        case OSM_YEAR40:    dcf77_telegram.year += (event * 40// y  break;
        case OSM_YEAR80:    dcf77_telegram.year += (event * 80);    break;

        case OSM_PARITY3:
            dcf77_telegram.parity3 = event;

            if (event == parity)
                osm_state = OSM_MINUTESYNC;     // parity OK, continue

            else
            {
                if (event == EVENT_MINUTESYNC)
                    osm_state = OSM_START;      // unexpected minute sync
                else
                    osm_state = OSM_RESYNC;     // parity bad
            }
            return;

        case OSM_MINUTESYNC:
            if (event == EVENT_MINUTESYNC)
            {
                // Hooray, correct time followed by a minute sync
                dcf77_settime (dcf77_telegram.hour,
                               dcf77_telegram.minute,
                               dcf77_telegram.day,
                               dcf77_telegram.month,
                               dcf77_telegram.year,
                               dcf77_telegram.day_of_week);
                osm_state = OSM_START;
            }
            else
            {
                // Too bad.  What seemed to be a valid time was received,
but
                //we didn't get the two-second pause after it.
                osm_state = OSM_RESYNC;
                // print out paritybits
            }
            return;

        case OSM_RESYNC:
            // Error condition, try to find the start of a new frame
            if (event == EVENT_MINUTESYNC)
                osm_state = OSM_START;
            return;

    } // switch

  /* This default processing is performed for all cases that don't return.
     If a minute sync unexpectedly happens in the middle of a frame, the
     state machine is reset to await the start of a new frame.  Otherwise
     (a zerobit or onebit is received), the state machine is advanced to
```

```
     await the next bit. */

        parity ^= event;   // exclusive OR  ???????? virker ikke helt

        if (event == EVENT_MINUTESYNC)
            osm_state = OSM_START;
        else
            osm_state++;
}


/*
---------------------------------------------------------------------------

  DCF77 Inner state machine
  ~~~~~~~~~~~~~~~~~~~~~~~~~
  The inner state machine picks bits out of the received DCF77 signal and
  sends them to the outer state machine.
  dcf77_ism() is called from the timer interrupt at 200 Hz.  The parameter
  RC2 is the value of the received DCF77 signal (zero or non-zero).

  An incoming pulse of 10-150 msec sends a '0' to the
  outer state machine.  A pulse of 150-300 msec sends a
  '1'.  Pulses shorter than 10 ms. or longer than 300 ms. cause an
  inner state machine resynchronisation.

  Ideally, a pulse comes either 1 second (normal bit) or 2 seconds (minute
  synchronisation) after the previous pulse.

  If the pause between the start of a pulse and the start of the next one
  is 800-1200 msec, nothing special happens.  If the pause
  is 1800-2200 msec, a minute synch is sent to the outer
  state machine.  If the pause isn't within these intervals, it is an error,
  and the inner state machine is resynchronized.

  Resynchronizing the inner state machine means waiting for a 0-to-1
  transition of the DCF77 signal, i.e. waiting for the signal being 0,
  and then waiting for it to be 1.

---------------------------------------------------------------------------
*/
void dcf77_ism (void)
{
    switch (ism_state)
    {
        // wait for pause (signal off)
        case ISM_RESYNC0:       // Resynching - waiting for '0'     [0]
            if (RC2)        // HUSK omvendt logik - active low
                ism_state = ISM_RESYNC1;
            break;

        // wait for active signal
        case ISM_RESYNC1:       // Resynching - waiting for '1'     [1]
           if (!RC2)
           {
                // 0-to-1 edge detected
                ism_ticks = 0;
                ism_state = ISM_ACTIVE;
           }
           break;
```

```
        // still active signal received
        case ISM_ACTIVE:        // '1' is being received            [2]
            ism_ticks++;
            if (!RC2)
            {
                // Still '1' coming in
                if (ism_ticks > 300)
                {
                    // Incoming '1' for more than 300 ms!  Something is
wrong.
                    ism_state = ISM_RESYNC0;
                }
            }
            else
            {
                // 0-to-1 edge detected
                if (ism_ticks < 10)
                {
                    // Pulse < 10 ms!  Something is wrong.
                    ism_state = ISM_RESYNC0;
                }
                else
                {
                    if (ism_ticks <= 142)
                    {
                        // Ordinary short pulse.  This is a '0'.
                        LastReceived = EVENT_ZEROBIT;
                        dcf77_osm (EVENT_ZEROBIT);
                        RA0 = 1;
                        RA1 = 0;
                        led_on_cnt = 500;           // timer værdi sættes
til 500ms
                        LedTimerOn = TRUE;          // LED timer startes
                        DCF_bit_received = TRUE;    //
                        ism_state = ISM_INACTIVE;   // vent på næste flanke
                    }
                    else
                    {
                        // Ordinary long pulse.  This is a '1'.
                        LastReceived = EVENT_ONEBIT;
                        dcf77_osm (EVENT_ONEBIT);
                        RA0 = 0;
                        RA1 = 1;
                        led_on_cnt = 500;
                        LedTimerOn = TRUE;
                        DCF_bit_received = TRUE;
                        ism_state = ISM_INACTIVE;   // vent på næste flanke
                    } // else
                } // else
            } // else
            break;

        case ISM_INACTIVE:      // '0' is being received            [3]
            ism_ticks++;        // add one millisecond
            if (!RC2)
            {
                /* Beginning of a new pulse (1-to-0 edge) detected.
ism_ticks
                is the number of ticks between the beginning of the old
```

```
                pulse and the beginning of the new one. */
                if (ism_ticks < 750)    // 800
                {
                    // 1-799 ms, this is too close
                    ism_state = ISM_RESYNC0;
                }
                else
                {
                    if (ism_ticks <= 1200)
                    {
                        // 800-1200 ms, normal interval
                        ism_ticks = 0;
                        ism_state = ISM_ACTIVE;
                    }
                    else
                    {
                        if (ism_ticks < 1750)
                        {
                            // 1201-1799 ms, this is illegal
                            ism_state = ISM_RESYNC0;
                        }
                        else
                        {
                            // 1800-2200 ms, end of minute sync interval
                            ism_ticks = 0;
                            ism_state = ISM_ACTIVE;
                            LastReceived = EVENT_MINUTESYNC;
                            dcf77_osm (EVENT_MINUTESYNC);
                            RA0 = 1;
                            RA1 = 1;
                            led_on_cnt = 500;
                            LedTimerOn = TRUE;
                        }
                    }
                }
            }
            else
            {
                // Still '0' coming in
                if (ism_ticks >= 2200)
                {
                    // 2200 ms have passed without seeing a new pulse, the
                    // receiver is probably dead
                    ism_state = ISM_RESYNC0;
                }
            } // if
            break;
    } // switch
} // function

void dcf77_init (void)
{
  ism_state = ISM_RESYNC0;
  osm_state = OSM_RESYNC;
}
```