Binary DCF-77 Clock Reference Manual

Generated by Doxygen 1.5.1

Wed Jan 3 22:22:35 2007

Contents

Chapter 1

Binary DCF-77 Clock

1.1 Introduction

In Germany, the official time is transmitted in a signal called DCF-77. You can find many descriptions of the signal format on the internet.

The Binary DCF-77 Clock is a simple device to receive and decode the signal and display the current date and time in binary form. The signal is received in a stock DCF-77 receiver module, decoded with an ATmega8 microcontroller and displayed in binary form on an array of LEDs. This array consists of for lines with eight LEDs each. The ATmega8 is not able to control 32 LEDs at once, so an SAA1064 module is used which is connected via I2C-bus.

The time should be displayed in several different binary formats, the format can be selected with a simple button. The formats will be described later.

The distribution contains the firmware for the controller, the schematics, the documentation and a copy of the GPL license.

1.2 Building and installing

The firmware for this project requires avr-gcc and avr-libc (a C-library for the AVR controller). Please read the instructions at `http://www.nongnu.org/avr-libc/user-manual/install_-tools.html` for how to install the GNU toolchain (avr-gcc, assembler, linker etc.) and avr-libc.

Once you have the GNU toolchain for AVR microcontrollers installed, you can run "make" in the subdirectory "firmware". You may need to customize the makefile. Also, you might have to edit the array byte[] in main.c, which describes the order of the output LEDs. The current order works for me because I soldered the LEDs as compact as possible, it's slightly different from the layout shown in the circuit.

Also, you may have to edit the Makefile to use your preferred downloader with "make program". The current version is built for avrdude with a USB connection to an avr109-compatible programmer.

No external crystal is needed, so you don't have to struggle with setting any fuse-bits.

After making your changes, you can compile and flash to the device:

```
make program
```

1.3 Usage

Connect the device to a DC power source with 9V. As long as no time has been decoded, a running light is shown on the output LED array. The single DCF indicator LED should start flashing to indicate that a signal is received. It is set to on when the input signal is high, and switched off if the signal is low. So you should see it flashing with one flash per second, each flash being 100ms or 200ms long.

If the signal is received correctly, after about two minutes the clock should be able to tell the correct time.

1.3.1 Reading the time

The time and date are displayed in seven different styles. You can select the style by pressing the button for a while. A pattern of lights indicate which mode is selected, you can read it as a binary value.

1.3.1.1 Mode 1: Time as binary

This simply displays the hours, minutes and seconds as bytes, one after each other. The fourth line of the display stays blank.

1.3.1.2 Mode 2: Date as binary

This is like the previous, with the difference that it displays the day of the month, the month and the year in the first three lines. The last line shows the day of the week, monday being a 1, tuesday a 2 and so on.

1.3.1.3 Mode 3: Time as BCD

This shows the time as binary coded digits (BCD). The first line displays the hours. The left four LEDs indicate the 10-hours, the right four LEDs indicate the 1-hours.

In the same way, the second and third line display the minutes and the seconds.

1.3.1.4 Mode 4: Date as BCD

This is like the previous mode, but the date is displayed.

1.3.1.5 Mode 5: Time as BCD (vertically)

This shows the time in a BCD-form as described in mode 3, but the BCD-values are put vertically next to each other. So in the first two colums you can read the hours, the third column is empty, the fourth and fifth columns show the minutes, the sixth is empty and the seventh and eighths indicate the seconds.

1.3.1.6 Mode 6: Date as BCD (vertically)

This is like mode 5, but it displays the date.

1.3.1.7 Mode 7: Unix timestamp

This is probably the least human readable format. It shows a 32-bit value of the seconds since january 1st, 1970. :-)

1.3.2 Demo mode

If you connect the clock in a place with a poor DCF-reception, but want to demonstrate the functions, you can use the demo mode. To toggle this, you can touch and hold the button for about five seconds. Afterwards, you can switch through the different display modes. The time displayed will stand still, so this can be used to explain the display modes without a hurry.

Switching to demo mode is indicated by all LEDs flashing for a short moment. Leaving demo mode shows an empty rectangle for a short moment.

1.4 Drawbacks

I didn't expect the DCF-signal to be so easily disturbed. In my case sometimes there is no usable signal left when I put my notebook with WLAN next to the clock. Fortunately, the time will be counted further until the next 'correct minute' is received.

1.5 Files in the distribution

- *Readme.txt:* Documentation, created from the htmldoc-directory.
- *firmware:* Source code of the controller firmware.
- *circuit:* Circuit diagrams in PDF and EAGLE 4 format. A free version of EAGLE is available for Linux, Mac OS X and Windows from `http://www.cadsoft.de/.`
- *License.txt:* Public license for all contents of this project.
- *Changelog.txt:* Logfile documenting changes in firm- and hardware.
- *refman.pdf:* Full documentation of the software.

1.6 Thanks!

I'd like to thank **Michael Meier**, who developed and published a much more sophisticated clock on his site. The SAA1064-stuff and the routine to calculate the Unix timestamp are based on his project. You can find it under `http://www.mulder.franken.de/ntpdcfledclock/.`

And once again I'd like to give special credits to **Thomas Stegemann** for help with the C language.

1.7 About the license

This project is licensed under the GNU General Public License (GPL). A copy of the GPL is included in License.txt.

(c) 2006 by Ronald Schaten - `http://www.schatenseite.de`

Chapter 2

Binary DCF-77 Clock Data Structure Index

2.1 Binary DCF-77 Clock Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Binary DCF-77 Clock File Index

3.1 Binary DCF-77 Clock File List

Here is a list of all files with brief descriptions:

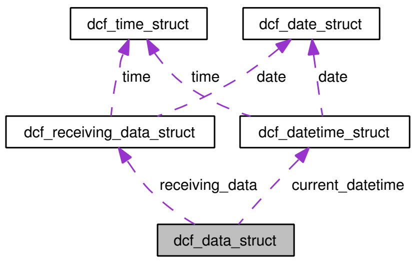
Chapter 4

Binary DCF-77 Clock Data Structure Documentation

4.1 dcf_data_struct Struct Reference

format of the DCF data.

Collaboration diagram for dcf_data_struct:



Data Fields

- dcf_datetime current_datetime [2]

  two full datasets

- boolean use_first_current_datetime

  flag if the first or the second dataset is used

- dcf_sample current_datetime_sample

  number of the current sample

- dcf_receiving_data receiving_data

  data being filled

4.1.1 Detailed Description

format of the DCF data.

dcf_current_datetime() and dcf_sample() may be called from different contexts. To avoid changing the current_datetime while it is read: if use_first_current_datetime is true: dcf_current_datetime reads current_datetime[0] and dcf_sample changes current_datetime[1] if use_first_current_datetime is false: vice versa

Definition at line 62 of file dcftime.c.

4.1.2 Field Documentation

4.1.2.1 dcf_datetime dcf_data_struct::current_datetime[2]

two full datasets

Definition at line 63 of file dcftime.c.

Referenced by dcf_current_datetime(), and dcf_init().

4.1.2.2 boolean dcf_data_struct::use_first_current_datetime

flag if the first or the second dataset is used

Definition at line 64 of file dcftime.c.

Referenced by dcf_current_datetime(), and dcf_init().

4.1.2.3 dcf_sample dcf_data_struct::current_datetime_sample

number of the current sample

Definition at line 65 of file dcftime.c.

Referenced by dcf_init(), and dcf_signal().

4.1.2.4 dcf_receiving_data dcf_data_struct::receiving_data

data being filled

Definition at line 66 of file dcftime.c.

Referenced by dcf_current_datetime(), dcf_init(), and dcf_signal().

The documentation for this struct was generated from the following file:

- firmware/dcftime.c

4.2 dcf_date_struct Struct Reference

format of the dcf_date

```
#include <dcftime.h>
```

Data Fields

- dcf_dayofweek dayofweek

  day of week
- dcf_dayofmonth dayofmonth

  day of month
- dcf_month month

  month
- dcf_year year

  year

4.2.1 Detailed Description

format of the dcf_date

Definition at line 74 of file dcftime.h.

4.2.2 Field Documentation

4.2.2.1 dcf_dayofweek dcf_date_struct::dayofweek

day of week

Definition at line 75 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.2.2.2 dcf_dayofmonth dcf_date_struct::dayofmonth

day of month

Definition at line 76 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.2.2.3 dcf_month dcf_date_struct::month

month

Definition at line 77 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.2.2.4 dcf_year dcf_date_struct::year

year

Definition at line 78 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

The documentation for this struct was generated from the following file:

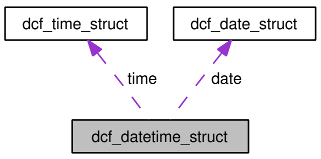
- firmware/dcftime.h

4.3 dcf_datetime_struct Struct Reference

format of the dcf_datetime

`#include <dcftime.h>`

Collaboration diagram for dcf_datetime_struct:



Data Fields

- dcf_time time

  the time
- dcf_date date

  the time
- boolean is_valid

  if is_valid is False: no complete signal received, do not use date and times
- boolean has_signal

  if has_signal is True: currently receiving signal

4.3.1 Detailed Description

format of the dcf_datetime

Definition at line 84 of file dcftime.h.

4.3.2 Field Documentation

4.3.2.1 dcf_time dcf_datetime_struct::time

the time

Definition at line 85 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.3.2.2 dcf_date dcf_datetime_struct::date

the time

Definition at line 86 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.3.2.3 boolean dcf_datetime_struct::is_valid

if is_valid is False: no complete signal received, do not use date and times

Definition at line 87 of file dcftime.h.

Referenced by timerInterrupt().

4.3.2.4 boolean dcf_datetime_struct::has_signal

if has_signal is True: currently receiving signal

Definition at line 88 of file dcftime.h.

Referenced by dcf_current_datetime().

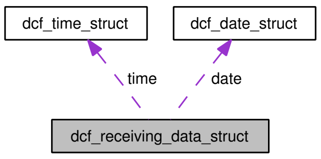
The documentation for this struct was generated from the following file:

- firmware/dcftime.h

4.4 dcf_receiving_data_struct Struct Reference

format of the received data, filled during reception

Collaboration diagram for dcf_receiving_data_struct:



Data Fields

- dcf_date date

  date

- dcf_time time

  time

- boolean parity

  parity of the received data

- boolean is_valid

  data is valid

- dcf_logic_signal current_signal

  logical state of the received data

- dcf_sample low_samples

  counts low signal samples per second

- dcf_sample high_samples

  counts high signal samples per second

4.4.1 Detailed Description

format of the received data, filled during reception

Definition at line 45 of file dcftime.c.

4.4.2 Field Documentation

4.4.2.1 dcf_date dcf_receiving_data_struct::date

date

Definition at line 46 of file dcftime.c.

4.4.2.2 dcf_time dcf_receiving_data_struct::time

time

Definition at line 47 of file dcftime.c.

Referenced by dcf_signal().

4.4.2.3 boolean dcf_receiving_data_struct::parity

parity of the received data

Definition at line 48 of file dcftime.c.

4.4.2.4 boolean dcf_receiving_data_struct::is_valid

data is valid

Definition at line 49 of file dcftime.c.

Referenced by dcf_current_datetime(), and dcf_signal().

4.4.2.5 dcf_logic_signal dcf_receiving_data_struct::current_signal

logical state of the received data

Definition at line 50 of file dcftime.c.

Referenced by dcf_signal().

4.4.2.6 dcf_sample dcf_receiving_data_struct::low_samples

counts low signal samples per second

Definition at line 51 of file dcftime.c.

Referenced by dcf_signal().

4.4.2.7 dcf_sample dcf_receiving_data_struct::high_samples

counts high signal samples per second

Definition at line 52 of file dcftime.c.

Referenced by dcf_signal().

The documentation for this struct was generated from the following file:

- firmware/dcftime.c

4.5 dcf_time_struct Struct Reference

format of the dcf_time

```
#include <dcftime.h>
```

Data Fields

- dcf_second second

  seconds

- dcf_minute minute

  minutes

- dcf_hour hour

  hours

- dcf_is_dst is_dst

  daylight saving time

4.5.1 Detailed Description

format of the dcf_time

Definition at line 64 of file dcftime.h.

4.5.2 Field Documentation

4.5.2.1 dcf_second dcf_time_struct::second

seconds

Definition at line 65 of file dcftime.h.

Referenced by dcf_signal(), setOutput(), and timerInterrupt().

4.5.2.2 dcf_minute dcf_time_struct::minute

minutes

Definition at line 66 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.5.2.3 dcf_hour dcf_time_struct::hour

hours

Definition at line 67 of file dcftime.h.

Referenced by setOutput(), and timerInterrupt().

4.5.2.4 dcf_is_dst dcf_time_struct::is_dst

daylight saving time

Definition at line 68 of file dcftime.h.

The documentation for this struct was generated from the following file:

- firmware/dcftime.h

Chapter 5

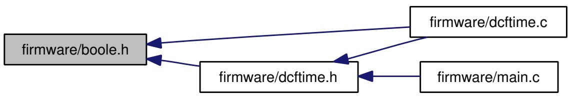
Binary DCF-77 Clock File Documentation

5.1 binarydcf77clock.dox File Reference

5.2 firmware/boole.h File Reference

Simple boolean variables.

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef enum boolean_enum boolean

Enumerations

- enum boolean_enum { False = 0, True = 1 }

5.2.1 Detailed Description

Simple boolean variables.

Author:

Thomas Stegemann

VersIdn:

boole.h,v 1.1 2007/01/02 21:30:40 rschaten Exp

License: See documentation.

Definition in file boole.h.

5.2.2 Typedef Documentation

5.2.2.1 typedef enum boolean_enum boolean

Definition at line 15 of file boole.h.

5.2.3 Enumeration Type Documentation

5.2.3.1 enum boolean_enum

Enumerator:

False

True

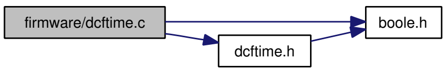
Definition at line 13 of file boole.h.

5.3 firmware/dcftime.c File Reference

Decoder for DCF-77 time signals.

```
#include "boole.h"
#include "dcftime.h"
```

Include dependency graph for dcftime.c:



Data Structures

- struct dcf_receiving_data_struct

  format of the received data, filled during reception

- struct dcf_data_struct

  format of the DCF data.

Typedefs

- typedef unsigned int dcf_sample

  number of the current sample

- typedef unsigned int dcf_sizetype

  used for the size of a month

- typedef enum dcf_logic_signal_enum dcf_logic_signal

  definition of logical signal states

- typedef dcf_receiving_data_struct dcf_receiving_data

  definition of the received data, filled during reception

Enumerations

- enum dcf_logic_signal_enum { dcf_signal_no, dcf_signal_false, dcf_signal_true, dcf_signal_invalid }

  definition of logical signal states

Functions

- void dcf_init (void)

  Initialize the DCF-module.

- void dcf_signal (boolean signal)
  Tell the DCF-module if the signal is high or low.
- dcf_datetime dcf_current_datetime (void)
  Fetch the current date and time.
- const char ∗ dcf_dayofweek_name (dcf_dayofweek dow)
  Get the name of the current weekday.
- const char ∗ dcf_is_dst_name (dcf_is_dst dst)
  Get the name of the current daylight saving time (summertime, wintertime).

Variables

- const dcf_sample dcf_second_samples = (DCF_RATE)
  number of samples per second
- const dcf_sample dcf_logic_false_min = (DCF_RATE)∗3/100
  dcf signal between 30ms and 130ms => dcf logic false (lower value)
- const dcf_sample dcf_logic_false_max = (DCF_RATE)∗13/100
  dcf signal between 30ms and 130ms => dcf logic false (upper value)
- const dcf_sample dcf_logic_true_min = (DCF_RATE)∗14/100
  dcf signal between 140ms and 230ms => dcf logic true (lower value)
- const dcf_sample dcf_logic_true_max = (DCF_RATE)∗23/100
  dcf signal between 140ms and 230ms => dcf logic true (upper value)
- const dcf_sample dcf_second_tolerance_min = (DCF_RATE) - (DCF_RATE)∗3/100
  duration between begin of dcf second (== begin of signal), should be 1 ∗ second +/- 3% (lower value)
- const dcf_sample dcf_second_tolerance_max = (DCF_RATE) + (DCF_RATE)∗3/100
  duration between begin of dcf second (== begin of signal), should be 1 ∗ second +/- 3% (upper value)

5.3.1 Detailed Description

Decoder for DCF-77 time signals.

Author:

Ronald Schaten & Thomas Stegemann

VersIdn:

dcftime.c,v 1.2 2007/01/03 12:38:55 rschaten Exp

License: See documentation.

Definition in file dcftime.c.

5.3.2 Typedef Documentation

5.3.2.1 typedef enum dcf_logic_signal_enum dcf_logic_signal

definition of logical signal states

Definition at line 42 of file dcftime.c.

5.3.2.2 typedef struct dcf_receiving_data_struct dcf_receiving_data

definition of the received data, filled during reception

Definition at line 55 of file dcftime.c.

5.3.2.3 typedef unsigned int dcf_sample

number of the current sample

Definition at line 17 of file dcftime.c.

5.3.2.4 typedef unsigned int dcf_sizetype

used for the size of a month

Definition at line 18 of file dcftime.c.

5.3.3 Enumeration Type Documentation

5.3.3.1 enum dcf_logic_signal_enum

definition of logical signal states

Enumerator:

dcf_signal_no	no signal
dcf_signal_false	'false' signal
dcf_signal_true	'true' signal
dcf_signal_invalid	invalid signal

Definition at line 35 of file dcftime.c.

5.3.4 Function Documentation

5.3.4.1 dcf_datetime dcf_current_datetime (void)

Fetch the current date and time.

Returns:

The current date and time in a dcf_datetime structure

Definition at line 407 of file dcftime.c.

References dcf_data_struct::current_datetime, dcf_datetime_struct::has_signal, dcf_receiving_data_-struct::is_valid, dcf_data_struct::receiving_data, and dcf_data_struct::use_first_current_datetime.

Referenced by timerInterrupt().

5.3.4.2 const char∗ dcf_dayofweek_name (dcf_dayofweek *dow*)

Get the name of the current weekday.

Parameters:

dow Day of the current week. Monday = 1, tuesday = 2...

Returns:

Pointer to the name

Definition at line 417 of file dcftime.c.

5.3.4.3 void dcf_init (void)

Initialize the DCF-module.

Call dcf_init before any other DCF function.

Definition at line 350 of file dcftime.c.

References dcf_data_struct::current_datetime, dcf_data_struct::current_datetime_sample, dcf_data_-struct::receiving_data, True, and dcf_data_struct::use_first_current_datetime.

Referenced by main().

5.3.4.4 const char∗ dcf_is_dst_name (dcf_is_dst *dst*)

Get the name of the current daylight saving time (summertime, wintertime).

Parameters:

dst daylight saving time bit from the time signal

Returns:

Pointer to the name

Definition at line 438 of file dcftime.c.

5.3.4.5 void dcf_signal (boolean *signal*)

Tell the DCF-module if the signal is high or low.

This function decides if the received bit is a long or a short one, and if it is usable at all. It should be called regularly, the number of calls per second is defined in DCF_RATE.

Parameters:

signal True if the input signal is high, False if it is low.

Definition at line 358 of file dcftime.c.

References dcf_data_struct::current_datetime_sample, dcf_receiving_data_struct::current_signal, dcf_-logic_false_max, dcf_logic_false_min, dcf_logic_true_max, dcf_logic_true_min, dcf_second_samples, dcf_signal_false, dcf_signal_invalid, dcf_signal_true, False, dcf_receiving_data_struct::high_samples, dcf_receiving_data_struct::is_valid, dcf_receiving_data_struct::low_samples, dcf_data_struct::receiving_-data, dcf_time_struct::second, dcf_receiving_data_struct::time, and True.

Referenced by timerInterrupt().

5.3.5 Variable Documentation

5.3.5.1 const dcf_sample dcf_logic_false_max = (DCF_RATE)*13/100

dcf signal between 30ms and 130ms => dcf logic false (upper value)

Definition at line 24 of file dcftime.c.

Referenced by dcf_signal().

5.3.5.2 const dcf_sample dcf_logic_false_min = (DCF_RATE)*3/100

dcf signal between 30ms and 130ms => dcf logic false (lower value)

Definition at line 22 of file dcftime.c.

Referenced by dcf_signal().

5.3.5.3 const dcf_sample dcf_logic_true_max = (DCF_RATE)*23/100

dcf signal between 140ms and 230ms => dcf logic true (upper value)

Definition at line 28 of file dcftime.c.

Referenced by dcf_signal().

5.3.5.4 const dcf_sample dcf_logic_true_min = (DCF_RATE)*14/100

dcf signal between 140ms and 230ms => dcf logic true (lower value)

Definition at line 26 of file dcftime.c.

Referenced by dcf_signal().

5.3.5.5 const dcf_sample dcf_second_samples = (DCF_RATE)

number of samples per second

Definition at line 20 of file dcftime.c.

Referenced by dcf_signal().

5.3.5.6 const dcf_sample dcf_second_tolerance_max = (DCF_RATE) + (DCF_RATE)∗3/100

duration between begin of dcf second (== begin of signal), should be 1 ∗ second +/- 3% (upper value)

Definition at line 32 of file dcftime.c.

5.3.5.7 const dcf_sample dcf_second_tolerance_min = (DCF_RATE) - (DCF_RATE)∗3/100

duration between begin of dcf second (== begin of signal), should be 1 ∗ second +/- 3% (lower value)

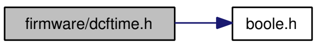
Definition at line 30 of file dcftime.c.

5.4 firmware/dcftime.h File Reference

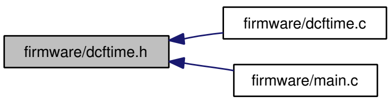
Decoder for DCF-77 time signals.

```
#include "boole.h"
```

Include dependency graph for dcftime.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct dcf_time_struct

  format of the dcf_time

- struct dcf_date_struct

  format of the dcf_date

- struct dcf_datetime_struct

  format of the dcf_datetime

Defines

- #define DCF_RATE 244

  number of samples per second.

Typedefs

- typedef unsigned int dcf_second

  seconds (0-59)

- typedef unsigned int dcf_minute

  minutes (0-59)

- typedef unsigned int dcf_hour

  hours (0-24)

- typedef unsigned int dcf_dayofmonth

  day of month (1-31)

- typedef unsigned int dcf_year

  year (0-99)

- typedef boolean dcf_is_dst

  daylight saving: True: MESZ, False: MEZ

- typedef enum dcf_dayofweek_enum dcf_dayofweek

  definition of weekdays

- typedef enum dcf_month_enum dcf_month

  definition of months

- typedef dcf_time_struct dcf_time

  definition of dcf_time

- typedef dcf_date_struct dcf_date

  definition of dcf_date

- typedef dcf_datetime_struct dcf_datetime

  definition of dcf_datetime

Enumerations

- enum dcf_dayofweek_enum {
  dcf_monday = 1, dcf_tuesday, dcf_wednesday, dcf_thursday,
  dcf_friday, dcf_saturday, dcf_sunday }

  definition of weekdays

- enum dcf_month_enum {
  dcf_january = 1, dcf_february, dcf_march, dcf_april,
  dcf_may, dcf_june, dcf_july, dcf_august,
  dcf_september, dcf_october, dcf_november, dcf_december }

  definition of months

Functions

- void dcf_init (void)

  Initialize the DCF-module.

- void dcf_signal (boolean signal)

  Tell the DCF-module if the signal is high or low.

- dcf_datetime dcf_current_datetime (void)

  Fetch the current date and time.

- const char ∗ dcf_dayofweek_name (dcf_dayofweek dow)

  Get the name of the current weekday.

- const char ∗ dcf_is_dst_name (dcf_is_dst dst)

  Get the name of the current daylight saving time (summertime, wintertime).

5.4.1 Detailed Description

Decoder for DCF-77 time signals.

Author:

Ronald Schaten & Thomas Stegemann

VersIdn:

dcftime.h,v 1.2 2007/01/03 12:38:55 rschaten Exp

License: See documentation.

Definition in file dcftime.h.

5.4.2 Define Documentation

5.4.2.1 #define DCF_RATE 244

number of samples per second.

dcf_signal() should be called this often

Definition at line 19 of file dcftime.h.

5.4.3 Typedef Documentation

5.4.3.1 typedef struct dcf_date_struct dcf_date

definition of dcf_date

Definition at line 81 of file dcftime.h.

5.4.3.2 typedef struct dcf_datetime_struct dcf_datetime

definition of dcf_datetime

Definition at line 91 of file dcftime.h.

5.4.3.3 typedef unsigned int dcf_dayofmonth

day of month (1-31)

Definition at line 28 of file dcftime.h.

5.4.3.4 typedef enum dcf_dayofweek_enum dcf_dayofweek

definition of weekdays

Definition at line 43 of file dcftime.h.

5.4.3.5 typedef unsigned int dcf_hour

hours (0-24)

Definition at line 27 of file dcftime.h.

5.4.3.6 typedef boolean dcf_is_dst

daylight saving: True: MESZ, False: MEZ

Definition at line 30 of file dcftime.h.

5.4.3.7 typedef unsigned int dcf_minute

minutes (0-59)

Definition at line 26 of file dcftime.h.

5.4.3.8 typedef enum dcf_month_enum dcf_month

definition of months

Definition at line 61 of file dcftime.h.

5.4.3.9 typedef unsigned int dcf_second

seconds (0-59)

Definition at line 25 of file dcftime.h.

5.4.3.10 typedef struct dcf_time_struct dcf_time

definition of dcf_time

Definition at line 71 of file dcftime.h.

5.4.3.11 typedef unsigned int dcf_year

year (0-99)

Definition at line 29 of file dcftime.h.

5.4.4 Enumeration Type Documentation

5.4.4.1 enum dcf_dayofweek_enum

definition of weekdays

Enumerator:

dcf_monday	monday = 1
dcf_tuesday	tuesday
dcf_wednesday	wednesday
dcf_thursday	thursday
dcf_friday	friday
dcf_saturday	saturday
dcf_sunday	sunday = 7

Definition at line 33 of file dcftime.h.

5.4.4.2 enum dcf_month_enum

definition of months

Enumerator:

dcf_january	january = 1
dcf_february	february
dcf_march	march
dcf_april	april
dcf_may	may
dcf_june	june
dcf_july	july
dcf_august	august
dcf_september	september
dcf_october	october
dcf_november	november
dcf_december	december = 12

Definition at line 46 of file dcftime.h.

5.4.5 Function Documentation

5.4.5.1 dcf_datetime dcf_current_datetime (void)

Fetch the current date and time.

Returns:

The current date and time in a dcf_datetime structure

Definition at line 407 of file dcftime.c.

References dcf_data_struct::current_datetime, dcf_datetime_struct::has_signal, dcf_receiving_data_-struct::is_valid, dcf_data_struct::receiving_data, and dcf_data_struct::use_first_current_datetime.

Referenced by timerInterrupt().

5.4.5.2 const char∗ dcf_dayofweek_name (dcf_dayofweek *dow*)

Get the name of the current weekday.

Parameters:

> ***dow*** Day of the current week. Monday = 1, tuesday = 2...

Returns:

> Pointer to the name

Definition at line 417 of file dcftime.c.

5.4.5.3 void dcf_init (void)

Initialize the DCF-module.

Call dcf_init before any other DCF function.

Definition at line 350 of file dcftime.c.

References dcf_data_struct::current_datetime, dcf_data_struct::current_datetime_sample, dcf_data_-struct::receiving_data, True, and dcf_data_struct::use_first_current_datetime.

Referenced by main().

5.4.5.4 const char∗ dcf_is_dst_name (dcf_is_dst *dst*)

Get the name of the current daylight saving time (summertime, wintertime).

Parameters:

> ***dst*** daylight saving time bit from the time signal

Returns:

> Pointer to the name

Definition at line 438 of file dcftime.c.

5.4.5.5 void dcf_signal (boolean *signal*)

Tell the DCF-module if the signal is high or low.

This function decides if the received bit is a long or a short one, and if it is usable at all. It should be called regularly, the number of calls per second is defined in DCF_RATE.

Parameters:

signal True if the input signal is high, False if it is low.

Definition at line 358 of file dcftime.c.

References dcf_data_struct::current_datetime_sample, dcf_receiving_data_struct::current_signal, dcf_-logic_false_max, dcf_logic_false_min, dcf_logic_true_max, dcf_logic_true_min, dcf_second_samples, dcf_signal_false, dcf_signal_invalid, dcf_signal_true, False, dcf_receiving_data_struct::high_samples, dcf_receiving_data_struct::is_valid, dcf_receiving_data_struct::low_samples, dcf_data_struct::receiving_-data, dcf_time_struct::second, dcf_receiving_data_struct::time, and True.

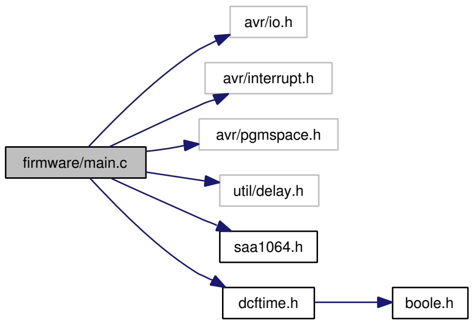
Referenced by timerInterrupt().

5.5 firmware/main.c File Reference

Firmware for the binary DCF-77 clock.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include "saa1064.h"
#include "dcftime.h"
```

Include dependency graph for main.c:



Enumerations

- enum modes {
  timeasbinary, dateasbinary, timeasbcdhorizontal, dateasbcdhorizontal,
  timeasbcdvertical, dateasbcdvertical, timestamp }

  the display-modes

Functions

- void setLeds (void)

  sends the current content of output[ ] to the LEDs if it has changed.

- void setOutput (dcf_datetime datetime)

  Takes the current time and converts it into different output-formats.

- void setWaiting (void)

  Sets the output to a running light.

- void timerInterrupt (void)

  Timer interrupt function.

- int main (void)

  Main-function.

Variables

- uint8_t byte [4] = { 2, 3, 1, 0 }

  the order of the connected output-LED-rows

- uint8_t output [4]

  current content of the LEDs

- uint8_t outputOld [4]

  old content of the LEDs

- enum modes mode

  the current display-mode

- uint8_t demomode = 0

  demo mode active

5.5.1 Detailed Description

Firmware for the binary DCF-77 clock.

Author:

Ronald Schaten

VersIdn:

main.c,v 1.2 2007/01/03 12:38:55 rschaten Exp

License: See documentation.

Definition in file main.c.

5.5.2 Enumeration Type Documentation

5.5.2.1 enum modes

the display-modes

Enumerator:

timeasbinary display hours, minutes and seconds, one byte per row

dateasbinary display day of month, month, year and day of week, one byte per row

timeasbcdhorizontal display hours, minutes and seconds, two BCDs per row

dateasbcdhorizontal display day of month, month, year and day of week, two BCDs per row

timeasbcdvertical display hours, minutes and seconds, one BCD per column

dateasbcdvertical display day of month, month and year, one BCD per column

timestamp display unix timestamp, one byte per row

Definition at line 23 of file main.c.

5.5.3 Function Documentation

5.5.3.1 int main (void)

Main-function.

Initializes the hardware and starts the main loop of the application.

Returns:

An integer. Whatever... :-)

Definition at line 323 of file main.c.

References dcf_init(), led_init(), mode, set_led_brightness(), set_led_digit(), timeasbinary, and timer-Interrupt().

5.5.3.2 void setLeds (void)

sends the current content of output[] to the LEDs if it has changed.

Definition at line 42 of file main.c.

References byte, output, outputOld, and set_led_digit().

Referenced by timerInterrupt().

5.5.3.3 void setOutput (dcf_datetime *datetime*)

Takes the current time and converts it into different output-formats.

Parameters:

datetime the current time

Definition at line 56 of file main.c.

References dcf_datetime_struct::date, dateasbcdhorizontal, dateasbcdvertical, dateasbinary, dcf_date_-struct::dayofmonth, dcf_date_struct::dayofweek, dcf_time_struct::hour, dcf_time_struct::minute, mode, dcf_date_struct::month, output, dcf_time_struct::second, dcf_datetime_struct::time, timeasbcdhorizontal, timeasbcdvertical, timeasbinary, timestamp, and dcf_date_struct::year.

Referenced by timerInterrupt().

5.5.3.4 void setWaiting (void)

Sets the output to a running light.

This is used when no valid time can be displayed.

Definition at line 191 of file main.c.

References output.

Referenced by timerInterrupt().

5.5.3.5 void timerInterrupt (void)

Timer interrupt function.

This is called on every timer-interrupt (which happens 488 times per second.

takes the current time and date

internal tick, is incremented with every timer-loop

used to defeat bouncing buttons

used to switch to demo mode

set when the mode has been switched, displays bars to indicate the new mode.

Definition at line 220 of file main.c.

References dcf_datetime_struct::date, dcf_date_struct::dayofmonth, dcf_date_struct::dayofweek, dcf_-current_datetime(), dcf_signal(), demomode, False, dcf_time_struct::hour, dcf_datetime_struct::is_-valid, dcf_time_struct::minute, mode, dcf_date_struct::month, output, dcf_time_struct::second, setLeds(), setOutput(), setWaiting(), dcf_datetime_struct::time, timeasbinary, timestamp, True, and dcf_date_-struct::year.

Referenced by main().

5.5.4 Variable Documentation

5.5.4.1 uint8_t byte[4] = { 2, 3, 1, 0 }

the order of the connected output-LED-rows

Definition at line 18 of file main.c.

Referenced by setLeds().

5.5.4.2 uint8_t demomode = 0

demo mode active

Definition at line 36 of file main.c.

Referenced by timerInterrupt().

5.5.4.3 enum modes mode

the current display-mode

Definition at line 33 of file main.c.

Referenced by main(), setOutput(), and timerInterrupt().

5.5.4.4 uint8_t output[4]

current content of the LEDs

Definition at line 19 of file main.c.

Referenced by setLeds(), setOutput(), setWaiting(), and timerInterrupt().

5.5.4.5 uint8_t outputOld[4]

old content of the LEDs

Definition at line 20 of file main.c.

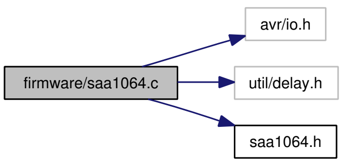
Referenced by setLeds().

5.6 firmware/saa1064.c File Reference

I2C-connection to the SAA1064 LED-driver.

```
#include <avr/io.h>
#include <util/delay.h>
#include "saa1064.h"
```

Include dependency graph for saa1064.c:



Defines

- #define LEDPORT PORTC

  the Port used for the connection

- #define LEDPIN PINC

  the Port used for the connection

- #define LEDDDR DDRC

  the Port used for the connection

- #define SDAPIN PC4

  which pins of the port

- #define SCLPIN PC5

  which pins of the port

- #define SAA_ADR 0x70

  the I2C addresses of the SAA 1064 LED drivers

- #define I2C_READ 0x01

  command used to read from I2C

- #define I2C_WRITE 0x00

  command used to write to I2C

- #define DELAYVAL 3

  pause between certain actions on the bus.

Functions

- void led_init (void)

  Initialize the LED module.

- void set_led_digit (uint8_t digit, uint8_t val)

  This sets one digit on the LED module.

- void set_led_brightness (uint8_t led_brightness)

  Configures the brightness of the LEDs.

5.6.1 Detailed Description

I2C-connection to the SAA1064 LED-driver.

Author:

Ronald Schaten

Version:

saa1064.c,v 1.2 2007/01/03 12:38:55 rschaten Exp

License: See documentation.

Definition in file saa1064.c.

5.6.2 Define Documentation

5.6.2.1 #define DELAYVAL 3

pause between certain actions on the bus.

Should be at least (10 ∗ freq) / 3, so we set 3 at 1 MHz

Definition at line 28 of file saa1064.c.

5.6.2.2 #define I2C_READ 0x01

command used to read from I2C

Definition at line 25 of file saa1064.c.

5.6.2.3 #define I2C_WRITE 0x00

command used to write to I2C

Definition at line 26 of file saa1064.c.

Referenced by set_led_brightness(), and set_led_digit().

5.6.2.4 #define LEDDDR DDRC

the Port used for the connection

Definition at line 18 of file saa1064.c.

5.6.2.5 #define LEDPIN PINC

the Port used for the connection

Definition at line 17 of file saa1064.c.

5.6.2.6 #define LEDPORT PORTC

the Port used for the connection

Definition at line 16 of file saa1064.c.

5.6.2.7 #define SAA_ADR 0x70

the I2C addresses of the SAA 1064 LED drivers

Definition at line 23 of file saa1064.c.

Referenced by set_led_brightness(), and set_led_digit().

5.6.2.8 #define SCLPIN PC5

which pins of the port

Definition at line 21 of file saa1064.c.

5.6.2.9 #define SDAPIN PC4

which pins of the port

Definition at line 20 of file saa1064.c.

5.6.3 Function Documentation

5.6.3.1 void led_init (void)

Initialize the LED module.

This basically enables the pullups on the I2C Bus pins.

Definition at line 30 of file saa1064.c.

Referenced by main().

5.6.3.2 void set_led_brightness (uint8_t *led_brightness*)

Configures the brightness of the LEDs.

Or rather: the current the driver allows through them.

Parameters:

led_brightness The values 0 through 7 can be used, corresponding to 0 through 21 mA

Definition at line 126 of file saa1064.c.

References I2C_WRITE, and SAA_ADR.

Referenced by main().

5.6.3.3 void set_led_digit (uint8_t *digit*, uint8_t *val*)

This sets one digit on the LED module.

Parameters:

digit the number of the digit (0 - 3)

val a bitfield that contains the values to set

Definition at line 117 of file saa1064.c.

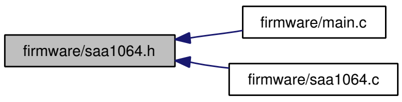
References I2C_WRITE, and SAA_ADR.

Referenced by main(), and setLeds().

5.7 firmware/saa1064.h File Reference

I2C-connection to the SAA1064 LED-driver.

This graph shows which files directly or indirectly include this file:



Functions

- void set_led_digit (uint8_t digit, uint8_t val)

  This sets one digit on the LED module.

- void set_led_brightness (uint8_t led_brightness)

  Configures the brightness of the LEDs.

- void led_init (void)

  Initialize the LED module.

5.7.1 Detailed Description

I2C-connection to the SAA1064 LED-driver.

Author:

Ronald Schaten

VersIdn:

saa1064.h,v 1.2 2007/01/03 12:38:55 rschaten Exp

License: See documentation.

Definition in file saa1064.h.

5.7.2 Function Documentation

5.7.2.1 void led_init (void)

Initialize the LED module.

This basically enables the pullups on the I2C Bus pins.

Definition at line 30 of file saa1064.c.

Referenced by main().

5.7.2.2 void set_led_brightness (uint8_t *led_brightness*)

Configures the brightness of the LEDs.

Or rather: the current the driver allows through them.

Parameters:

led_brightness The values 0 through 7 can be used, corresponding to 0 through 21 mA

Definition at line 126 of file saa1064.c.

References I2C_WRITE, and SAA_ADR.

Referenced by main().

5.7.2.3 void set_led_digit (uint8_t *digit*, uint8_t *val*)

This sets one digit on the LED module.

Parameters:

digit the number of the digit (0 - 3)

val a bitfield that contains the values to set

Definition at line 117 of file saa1064.c.

References I2C_WRITE, and SAA_ADR.

Referenced by main(), and setLeds().

Index