mmculib Reference Manual

1.0.3

Generated by Doxygen 1.4.6

Fri Aug 24 14:44:40 2007

Contents

Chapter 1

mmculib Hierarchical Index

1.1 mmculib Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

mmculib Data Structure Index

2.1 mmculib Data Structures

Here are the data structures with brief descriptions:

Chapter 3

mmculib File Index

3.1 mmculib File List

Here is a list of all files with brief descriptions:

Chapter 4

mmculib Data Structure Documentation

4.1 biseq_obj_t Struct Reference

`#include <biseq.h>`

Data Fields

- char * str
- uint8_t step
- int8_t dir
- biseq_mode_t mode
- int8_t(* callback)(void *data, char *str)
- void * callback_data

4.1.1 Field Documentation

4.1.1.1 int8_t(* biseq_obj_t::callback)(void *data, char *str)

4.1.1.2 void* biseq_obj_t::callback_data

4.1.1.3 int8_t biseq_obj_t::dir

4.1.1.4 biseq_mode_t biseq_obj_t::mode

4.1.1.5 uint8_t biseq_obj_t::step

4.1.1.6 char* biseq_obj_t::str

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

- biseq.h

4.2 busart_dev_struct Struct Reference

Data Fields

- void(∗ tx_irq_enable)(void)
- void(∗ rx_irq_enable)(void)
- bool(∗ tx_finished_p)(void)
- ring_t tx_ring
- ring_t rx_ring

4.2.1 Field Documentation

4.2.1.1 void(∗ busart_dev_struct::rx_irq_enable)(void)

4.2.1.2 ring_t busart_dev_struct::rx_ring

4.2.1.3 bool(∗ busart_dev_struct::tx_finished_p)(void)

4.2.1.4 void(∗ busart_dev_struct::tx_irq_enable)(void)

4.2.1.5 ring_t busart_dev_struct::tx_ring

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

- busart.c

4.3 button_cfg_t Struct Reference

`#include <button.h>`

Data Fields

- port_t port
- port_mask_t bitmask

4.3.1 Field Documentation

4.3.1.1 port_mask_t button_cfg_t::bitmask

4.3.1.2 port_t button_cfg_t::port

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

- button.h

4.4 button_private_t Struct Reference

`#include <button.h>`

Data Fields

- button_state_t state
- const button_cfg_t * cfg
- uint8_t count
- uint8_t hold_count

4.4.1 Field Documentation

4.4.1.1 const button_cfg_t* button_private_t::cfg

4.4.1.2 uint8_t button_private_t::count

4.4.1.3 uint8_t button_private_t::hold_count

4.4.1.4 button_state_t button_private_t::state

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

- button.h

4.5 buttons_private_t Struct Reference

`#include <buttons.h>`

Data Fields

- button_t buttons
- uint8_t num

4.5.1 Detailed Description

Private multiple button structure.

Note:
These elements should be considered private so do not access them.

4.5.2 Field Documentation

4.5.2.1 button_t buttons_private_t::buttons

4.5.2.2 uint8_t buttons_private_t::num

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

- buttons.h

4.6 chaser_private_t Struct Reference

`#include <chaser.h>`

Data Fields

- uint8_t flasher_num
- flasher_t * flashers
- chaser_sequence_t seq
- font_t * font
- uint8_t step
- int8_t dir
- chaser_mode_t mode
- flasher_pattern_t * on_pattern
- flasher_pattern_t * off_pattern

4.6.1 Field Documentation

4.6.1.1 int8_t chaser_private_t::dir

4.6.1.2 uint8_t chaser_private_t::flasher_num

4.6.1.3 flasher_t* chaser_private_t::flashers

4.6.1.4 font_t* chaser_private_t::font

4.6.1.5 chaser_mode_t chaser_private_t::mode

4.6.1.6 flasher_pattern_t* chaser_private_t::off_pattern

4.6.1.7 flasher_pattern_t* chaser_private_t::on_pattern

4.6.1.8 chaser_sequence_t chaser_private_t::seq

4.6.1.9 uint8_t chaser_private_t::step

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

- chaser.h

4.7 cleds_private_t Struct Reference

`#include <cleds.h>`

Data Fields

- const led_cfg_t * leds
- const led_cfg_t * row_config
- uint8_t rows_num
- uint8_t cols_num
- uint8_t row

4.7.1 Field Documentation

4.7.1.1 uint8_t cleds_private_t::cols_num

4.7.1.2 const led_cfg_t* cleds_private_t::leds

4.7.1.3 uint8_t cleds_private_t::row

4.7.1.4 const led_cfg_t* cleds_private_t::row_config

4.7.1.5 uint8_t cleds_private_t::rows_num

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

- cleds.h

4.8 flasher_obj_t Struct Reference

`#include <flasher.h>`

Data Fields

- char dummy [sizeof(flasher_private_t)]

4.8.1 Field Documentation

4.8.1.1 char flasher_obj_t::dummy[sizeof(flasher_private_t)]

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

- flasher.h

4.9 flasher_pattern_t Struct Reference

`#include <flasher.h>`

Data Fields

- uint8_t mod_period
- uint8_t mod_duty
- uint8_t flasher_period
- uint8_t flasher_duty
- uint8_t flashes
- uint8_t period

4.9.1 Field Documentation

4.9.1.1 uint8_t flasher_pattern_t::flasher_duty

4.9.1.2 uint8_t flasher_pattern_t::flasher_period

4.9.1.3 uint8_t flasher_pattern_t::flashes

4.9.1.4 uint8_t flasher_pattern_t::mod_duty

4.9.1.5 uint8_t flasher_pattern_t::mod_period

4.9.1.6 uint8_t flasher_pattern_t::period

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

- flasher.h

4.10 flasher_private_t Struct Reference

`#include <flasher.h>`

Data Fields

- flasher_pattern_t * pattern
- uint8_t mod_count
- uint8_t flasher_count
- uint8_t flashes_count
- uint8_t flasher_prescale

4.10.1 Field Documentation

4.10.1.1 uint8_t flasher_private_t::flasher_count

4.10.1.2 uint8_t flasher_private_t::flasher_prescale

4.10.1.3 uint8_t flasher_private_t::flashes_count

4.10.1.4 uint8_t flasher_private_t::mod_count

4.10.1.5 flasher_pattern_t* flasher_private_t::pattern

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

- flasher.h

4.11 font_t Struct Reference

`#include <font.h>`

Data Fields

- uint8_t pixels
- uint8_t offset
- uint8_t size
- uint8_t data [ ]

4.11.1 Field Documentation

4.11.1.1 uint8_t font_t::data[ ]

4.11.1.2 uint8_t font_t::offset

4.11.1.3 uint8_t font_t::pixels

4.11.1.4 uint8_t font_t::size

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

- font.h

4.12 lcd_cfg_t Struct Reference

`#include <lcd.h>`

Data Fields

- port_t data_port
- port_bit_t d_bit
- port_t e_port
- port_bit_t e_bit
- port_t rs_port
- port_bit_t rs_bit

4.12.1 Field Documentation

4.12.1.1 port_bit_t lcd_cfg_t::d_bit

4.12.1.2 port_t lcd_cfg_t::data_port

4.12.1.3 port_bit_t lcd_cfg_t::e_bit

4.12.1.4 port_t lcd_cfg_t::e_port

4.12.1.5 port_bit_t lcd_cfg_t::rs_bit

4.12.1.6 port_t lcd_cfg_t::rs_port

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

- lcd.h

4.13 lcd_obj_t Struct Reference

`#include <lcd.h>`

Data Fields

- const lcd_cfg_t * cfg
- port_mask_t e_mask
- port_mask_t rs_mask
- uint8_t data

4.13.1 Field Documentation

4.13.1.1 const lcd_cfg_t* lcd_obj_t::cfg

4.13.1.2 uint8_t lcd_obj_t::data

4.13.1.3 port_mask_t lcd_obj_t::e_mask

4.13.1.4 port_mask_t lcd_obj_t::rs_mask

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

- lcd.h

4.14 led_cfg_t Struct Reference

`#include <led.h>`

Data Fields

- port_t port
- port_mask_t bitmask

4.14.1 Field Documentation

4.14.1.1 port_mask_t led_cfg_t::bitmask

4.14.1.2 port_t led_cfg_t::port

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

- led.h

4.15 lmatrix_port_t Struct Reference

`#include <lmatrix.h>`

Data Fields

- port_t port
- port_mask_t bitmask

4.15.1 Field Documentation

4.15.1.1 port_mask_t lmatrix_port_t::bitmask

4.15.1.2 port_t lmatrix_port_t::port

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

- lmatrix.h

4.16 lmatrix_private_t Struct Reference

`#include <lmatrix.h>`

Data Fields

- lmatrix_port_t * col_port
- uint8_t col
- lmatrix_row_state_t state [LMATRIX_COLS]

4.16.1 Field Documentation

4.16.1.1 uint8_t lmatrix_private_t::col

4.16.1.2 lmatrix_port_t* lmatrix_private_t::col_port

4.16.1.3 lmatrix_row_state_t lmatrix_private_t::state[LMATRIX_COLS]

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

- lmatrix.h

4.17 mbuttons_private_t Struct Reference

`#include <mbuttons.h>`

Data Fields

- button_t buttons
- const button_cfg_t const * row_config
- uint8_t rows_num
- uint8_t cols_num

4.17.1 Field Documentation

4.17.1.1 button_t mbuttons_private_t::buttons

4.17.1.2 uint8_t mbuttons_private_t::cols_num

4.17.1.3 const button_cfg_t const* mbuttons_private_t::row_config

4.17.1.4 uint8_t mbuttons_private_t::rows_num

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

- mbuttons.h

4.18 mcleds_private_t Struct Reference

`#include <mcleds.h>`

Data Fields

- cleds_obj_t cleds
- colourmap_t * colourmap
- uint8_t colourmap_size
- ticker8_t primary_ticker
- mcleds_state_t * state

4.18.1 Field Documentation

4.18.1.1 cleds_obj_t mcleds_private_t::cleds

4.18.1.2 colourmap_t* mcleds_private_t::colourmap

4.18.1.3 uint8_t mcleds_private_t::colourmap_size

4.18.1.4 ticker8_t mcleds_private_t::primary_ticker

4.18.1.5 mcleds_state_t* mcleds_private_t::state

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

- mcleds.h

4.19 mcleds_state_t Struct Reference

`#include <mcleds.h>`

Data Fields

- uint8_t duty

4.19.1 Field Documentation

4.19.1.1 uint8_t mcleds_state_t::duty

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

- mcleds.h

4.20 mmelody_private_t Struct Reference

`#include <mmelody.h>`

Data Fields

- ticker_t ticker
- const char * cur
- const char * start
- const char * loop_start
- int8_t loop_count
- uint8_t note_fraction
- mmelody_speed_t speed
- mmelody_volume_t volume
- uint8_t octave
- void(* play_callback)(void *data, uint8_t note, uint8_t volume)
- void * play_callback_data
- uint16_t poll_rate

4.20.1 Field Documentation

4.20.1.1 const char* mmelody_private_t::cur

4.20.1.2 int8_t mmelody_private_t::loop_count

4.20.1.3 const char* mmelody_private_t::loop_start

4.20.1.4 uint8_t mmelody_private_t::note_fraction

4.20.1.5 uint8_t mmelody_private_t::octave

4.20.1.6 void(* mmelody_private_t::play_callback)(void *data, uint8_t note, uint8_t volume)

4.20.1.7 void* mmelody_private_t::play_callback_data

4.20.1.8 uint16_t mmelody_private_t::poll_rate

4.20.1.9 mmelody_speed_t mmelody_private_t::speed

4.20.1.10 const char* mmelody_private_t::start

4.20.1.11 ticker_t mmelody_private_t::ticker

4.20.1.12 mmelody_volume_t mmelody_private_t::volume

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

- mmelody.h

4.21 mpwm_channel_t Struct Reference

`#include <mpwm.h>`

Data Fields

- uint8_t duty

4.21.1 Field Documentation

4.21.1.1 uint8_t mpwm_channel_t::duty

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

- mpwm.h

4.22 mpwm_obj_t Struct Reference

`#include <mpwm.h>`

Data Fields

- uint8_t period
- uint8_t count
- mpwm_channel_t * channels
- uint8_t num_channels

4.22.1 Field Documentation

4.22.1.1 mpwm_channel_t* mpwm_obj_t::channels

4.22.1.2 uint8_t mpwm_obj_t::count

4.22.1.3 uint8_t mpwm_obj_t::num_channels

4.22.1.4 uint8_t mpwm_obj_t::period

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

- mpwm.h

4.23 mtext_obj_t Struct Reference

`#include <mtext.h>`

Data Fields

- const char * cur
- const char * start
- font_t * font
- uint8_t pixels
- uint8_t * image
- uint8_t * screen
- uint16_t poll_rate
- mtext_mode_t mode
- scroller_obj_t scroller
- uint8_t speed
- ticker_t ticker

4.23.1 Field Documentation

4.23.1.1 const char* mtext_obj_t::cur

4.23.1.2 font_t* mtext_obj_t::font

4.23.1.3 uint8_t* mtext_obj_t::image

4.23.1.4 mtext_mode_t mtext_obj_t::mode

4.23.1.5 uint8_t mtext_obj_t::pixels

4.23.1.6 uint16_t mtext_obj_t::poll_rate

4.23.1.7 uint8_t* mtext_obj_t::screen

4.23.1.8 scroller_obj_t mtext_obj_t::scroller

4.23.1.9 uint8_t mtext_obj_t::speed

4.23.1.10 const char* mtext_obj_t::start

4.23.1.11 ticker_t mtext_obj_t::ticker

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

- mtext.h

4.24 mu1wire_obj_t Struct Reference

`#include <mu1wire.h>`

Data Fields

- mu1wire_rom_code_t rom_code

4.24.1 Field Documentation

4.24.1.1 mu1wire_rom_code_t mu1wire_obj_t::rom_code

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

- mu1wire.h

4.25 mu1wire_rom_code_t Union Reference

`#include <mu1wire.h>`

Data Fields

- struct {
  uint8_t family
  uint8_t serial [6]
  uint8_t crc
  } fields

- uint8_t bytes [8]

4.25.1 Field Documentation

4.25.1.1 uint8_t mu1wire_rom_code_t::bytes[8]

4.25.1.2 uint8_t mu1wire_rom_code_t::crc

4.25.1.3 uint8_t mu1wire_rom_code_t::family

4.25.1.4 struct { ... } mu1wire_rom_code_t::fields

4.25.1.5 uint8_t mu1wire_rom_code_t::serial[6]

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

- mu1wire.h

4.26 muxleds_cfg_t Struct Reference

`#include <muxleds.h>`

Data Fields

- port_t port
- uint8_t bitmask

4.26.1 Field Documentation

4.26.1.1 uint8_t muxleds_cfg_t::bitmask

4.26.1.2 port_t muxleds_cfg_t::port

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

- muxleds.h

4.27 muxleds_col_t Struct Reference

`#include <muxleds.h>`

Data Fields

- port_t port
- uint8_t bitmask
- uint8_t row_state

4.27.1 Field Documentation

4.27.1.1 uint8_t muxleds_col_t::bitmask

4.27.1.2 port_t muxleds_col_t::port

4.27.1.3 uint8_t muxleds_col_t::row_state

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

- muxleds.h

4.28 muxleds_obj_t Struct Reference

`#include <muxleds.h>`

Data Fields

- muxleds_row_t rows [MUXLEDS_ROWS_NUM]
- muxleds_col_t cols [MUXLEDS_COLS_NUM]
- uint8_t col
- uint8_t row_on
- uint8_t rows_num
- uint8_t cols_num

4.28.1 Field Documentation

4.28.1.1 uint8_t muxleds_obj_t::col

4.28.1.2 muxleds_col_t muxleds_obj_t::cols[MUXLEDS_COLS_NUM]

4.28.1.3 uint8_t muxleds_obj_t::cols_num

4.28.1.4 uint8_t muxleds_obj_t::row_on

4.28.1.5 muxleds_row_t muxleds_obj_t::rows[MUXLEDS_ROWS_NUM]

4.28.1.6 uint8_t muxleds_obj_t::rows_num

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

- muxleds.h

4.29 muxleds_row_t Struct Reference

`#include <muxleds.h>`

Data Fields

- port_t port
- uint8_t bitmask

4.29.1 Field Documentation

4.29.1.1 uint8_t muxleds_row_t::bitmask

4.29.1.2 port_t muxleds_row_t::port

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

- muxleds.h

4.30 nrf_cfg_t Struct Reference

`#include <nrf2401.h>`

Data Fields

- port_t cs_port
- port_bit_t cs_bitno
- port_t ce_port
- port_bit_t ce_bitno
- port_t dr_port
- port_bit_t dr_bitno

4.30.1 Field Documentation

4.30.1.1 port_bit_t nrf_cfg_t::ce_bitno

4.30.1.2 port_t nrf_cfg_t::ce_port

4.30.1.3 port_bit_t nrf_cfg_t::cs_bitno

4.30.1.4 port_t nrf_cfg_t::cs_port

4.30.1.5 port_bit_t nrf_cfg_t::dr_bitno

4.30.1.6 port_t nrf_cfg_t::dr_port

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

- nrf2401.h

4.31 nrf_config_bits_t Struct Reference

`#include <nrf2401.h>`

Data Fields

- uint8_t rxen:1
- uint8_t rx_ch_num:7
- uint8_t rf_pwr:2
- uint8_t xo_f:3
- uint8_t rfdr_sb:1
- uint8_t cm:1
- uint8_t rx2_en:1
- uint8_t crc_en:1
- uint8_t crc_l:1
- uint8_t addr_w:6
- rf_address_t addr_1
- rf_address_t addr_2
- uint8_t data1_w
- uint8_t data2_w

4.31.1 Field Documentation

4.31.1.1 rf_address_t nrf_config_bits_t::addr_1

4.31.1.2 rf_address_t nrf_config_bits_t::addr_2

4.31.1.3 uint8_t nrf_config_bits_t::addr_w

4.31.1.4 uint8_t nrf_config_bits_t::cm

4.31.1.5 uint8_t nrf_config_bits_t::crc_en

4.31.1.6 uint8_t nrf_config_bits_t::crc_l

4.31.1.7 uint8_t nrf_config_bits_t::data1_w

4.31.1.8 uint8_t nrf_config_bits_t::data2_w

4.31.1.9 uint8_t nrf_config_bits_t::rf_pwr

4.31.1.10 uint8_t nrf_config_bits_t::rfdr_sb

4.31.1.11 uint8_t nrf_config_bits_t::rx2_en

4.31.1.12 uint8_t nrf_config_bits_t::rx_ch_num

4.31.1.13 uint8_t nrf_config_bits_t::rxen

4.31.1.14 uint8_t nrf_config_bits_t::xo_f

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

- nrf2401.h

4.32 nrf_config_t Union Reference

`#include <nrf2401.h>`

Data Fields

- uint8_t bytes [15]
- nrf_config_bits_t bits

4.32.1 Field Documentation

4.32.1.1 nrf_config_bits_t nrf_config_t::bits

4.32.1.2 uint8_t nrf_config_t::bytes[15]

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

- nrf2401.h

4.33 nrf_obj_t Struct Reference

`#include <nrf2401.h>`

Data Fields

- nrf_pins_t pins
- nrf_config_t config

4.33.1 Field Documentation

4.33.1.1 nrf_config_t nrf_obj_t::config

4.33.1.2 nrf_pins_t nrf_obj_t::pins

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

- nrf2401.h

4.34 nrf_pins_t Struct Reference

`#include <nrf2401.h>`

Data Fields

- port_t cs_port
- uint8_t cs_bitmask
- port_t ce_port
- uint8_t ce_bitmask
- port_t dr_port
- uint8_t dr_bitmask

4.34.1 Field Documentation

4.34.1.1 uint8_t nrf_pins_t::ce_bitmask

4.34.1.2 port_t nrf_pins_t::ce_port

4.34.1.3 uint8_t nrf_pins_t::cs_bitmask

4.34.1.4 port_t nrf_pins_t::cs_port

4.34.1.5 uint8_t nrf_pins_t::dr_bitmask

4.34.1.6 port_t nrf_pins_t::dr_port

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

- nrf2401.h

4.35 pga_cfg_t Struct Reference

`#include <pga.h>`

Data Fields

- port_t cs_port
- port_mask_t cs_bitmask

4.35.1 Field Documentation

4.35.1.1 port_mask_t pga_cfg_t::cs_bitmask

4.35.1.2 port_t pga_cfg_t::cs_port

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

- pga.h

4.36 pga_private_t Struct Reference

`#include <pga.h>`

Data Fields

- port_t cs_port
- port_mask_t cs_bitmask

4.36.1 Field Documentation

4.36.1.1 port_mask_t pga_private_t::cs_bitmask

4.36.1.2 port_t pga_private_t::cs_port

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

- pga.h

4.37 piezo_cfg_t Struct Reference

`#include <piezo.h>`

Data Fields

- port_t port
- port_mask_t bitmask

4.37.1 Field Documentation

4.37.1.1 port_mask_t piezo_cfg_t::bitmask

4.37.1.2 port_t piezo_cfg_t::port

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

- piezo.h

4.38 rf_address_t Struct Reference

`#include <nrf2401.h>`

Data Fields

- uint8_t bytes [5]

4.38.1 Field Documentation

4.38.1.1 uint8_t rf_address_t::bytes[5]

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

- nrf2401.h

4.39 rf_node_t Struct Reference

`#include <rf.h>`

Data Fields

- rf_id_t id
- rf_channel_t channel

4.39.1 Field Documentation

4.39.1.1 rf_channel_t rf_node_t::channel

4.39.1.2 rf_id_t rf_node_t::id

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

- rf.h

4.40 rf_probe_t Struct Reference

`#include <rf.h>`

Data Fields

- rf_node_t node
- uint8_t device_id [RF_DEVICE_ID_SIZE]

4.40.1 Field Documentation

4.40.1.1 uint8_t rf_probe_t::device_id[RF_DEVICE_ID_SIZE]

4.40.1.2 rf_node_t rf_probe_t::node

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

- rf.h

4.41 ring_struct Struct Reference

`#include <ring.h>`

Data Fields

- char * in
- char * out
- char * top
- char * end

4.41.1 Field Documentation

4.41.1.1 char* ring_struct::end

4.41.1.2 char* ring_struct::in

4.41.1.3 char* ring_struct::out

4.41.1.4 char* ring_struct::top

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

- ring.h

4.42 scroller_obj_t Struct Reference

`#include <scroller.h>`

Data Fields

- uint8_t rows
- uint8_t cols
- uint8_t index
- scroller_dir_t dir
- bool running

4.42.1 Field Documentation

4.42.1.1 uint8_t scroller_obj_t::cols

4.42.1.2 scroller_dir_t scroller_obj_t::dir

4.42.1.3 uint8_t scroller_obj_t::index

4.42.1.4 uint8_t scroller_obj_t::rows

4.42.1.5 bool scroller_obj_t::running

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

- scroller.h

4.43 seq_obj_t Struct Reference

`#include <seq.h>`

Data Fields

- const char * str
- const char * cur
- const char *(* callback)(void *data, const char *str)
- void * callback_data

4.43.1 Field Documentation

4.43.1.1 const char*(* seq_obj_t::callback)(void *data, const char *str)

4.43.1.2 void* seq_obj_t::callback_data

4.43.1.3 const char* seq_obj_t::cur

4.43.1.4 const char* seq_obj_t::str

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

- seq.h

4.44 sflash_obj_t Struct Reference

`#include <sflash.h>`

Data Fields

- sflash_pattern_t pattern
- sflash_pattern_t current

4.44.1 Field Documentation

4.44.1.1 sflash_pattern_t sflash_obj_t::current

4.44.1.2 sflash_pattern_t sflash_obj_t::pattern

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

- sflash.h

4.45 spi_eeprom_cfg_t Struct Reference

`#include <s_eeprom.h>`

Data Fields

- port_t cs_port
- port_bit_t cs_bitno

4.45.1 Field Documentation

4.45.1.1 port_bit_t spi_eeprom_cfg_t::cs_bitno

4.45.1.2 port_t spi_eeprom_cfg_t::cs_port

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

- s_eeprom.h

4.46 spi_eeprom_private_t Struct Reference

`#include <s_eeprom.h>`

Data Fields

- port_t cs_port
- port_mask_t cs_bitmask

4.46.1 Field Documentation

4.46.1.1 port_mask_t spi_eeprom_private_t::cs_bitmask

4.46.1.2 port_t spi_eeprom_private_t::cs_port

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

- s_eeprom.h

4.47 spwm_obj_t Struct Reference

`#include <spwm.h>`

Data Fields

- uint8_t period
- uint8_t duty
- uint8_t count

4.47.1 Field Documentation

4.47.1.1 uint8_t spwm_obj_t::count

4.47.1.2 uint8_t spwm_obj_t::duty

4.47.1.3 uint8_t spwm_obj_t::period

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

- spwm.h

4.48 squeaker_private_t Struct Reference

`#include <squeaker.h>`

Data Fields

- uint8_t note_clock
- uint8_t note_period
- uint8_t note_duty
- uint8_t note_holdoff
- const char * start
- const char * cur
- uint8_t holdoff
- uint16_t poll_rate
- const char * loop_start
- int8_t loop_count
- uint8_t prescaler
- uint8_t note_fraction
- squeaker_speed_t speed
- squeaker_volume_t volume
- ticker8_t ticker
- squeaker_scale_t * scale_table
- uint8_t octave

4.48.1 Field Documentation

4.48.1.1 const char* squeaker_private_t::cur

4.48.1.2 uint8_t squeaker_private_t::holdoff

4.48.1.3 int8_t squeaker_private_t::loop_count

4.48.1.4 const char* squeaker_private_t::loop_start

4.48.1.5 uint8_t squeaker_private_t::note_clock

4.48.1.6 uint8_t squeaker_private_t::note_duty

4.48.1.7 uint8_t squeaker_private_t::note_fraction

4.48.1.8 uint8_t squeaker_private_t::note_holdoff

4.48.1.9 uint8_t squeaker_private_t::note_period

4.48.1.10 uint8_t squeaker_private_t::octave

4.48.1.11 uint16_t squeaker_private_t::poll_rate

4.48.1.12 uint8_t squeaker_private_t::prescaler

4.48.1.13 squeaker_scale_t* squeaker_private_t::scale_table

4.48.1.14 squeaker_speed_t squeaker_private_t::speed

4.48.1.15 const char* squeaker_private_t::start

4.48.1.16 ticker8_t squeaker_private_t::ticker

4.48.1.17 squeaker_volume_t squeaker_private_t::volume

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

- squeaker.h

4.49 stext_obj_t Struct Reference

`#include <stext.h>`

Data Fields

- seq_t seq
- font_t * font
- void(* callback)(void *data, uint8_t pixel, bool val)
- void * callback_data
- seq_obj_t seq_info

4.49.1 Field Documentation

4.49.1.1 void(* stext_obj_t::callback)(void *data, uint8_t pixel, bool val)

4.49.1.2 void* stext_obj_t::callback_data

4.49.1.3 font_t* stext_obj_t::font

4.49.1.4 seq_t stext_obj_t::seq

4.49.1.5 seq_obj_t stext_obj_t::seq_info

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

- stext.h

4.50 ticker16_t Struct Reference

`#include <ticker.h>`

Data Fields

- uint16_t period
- uint16_t clock

4.50.1 Field Documentation

4.50.1.1 uint16_t ticker16_t::clock

4.50.1.2 uint16_t ticker16_t::period

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

- ticker.h

4.51 ticker8_t Struct Reference

`#include <ticker.h>`

Data Fields

- uint8_t period
- uint8_t clock

4.51.1 Field Documentation

4.51.1.1 uint8_t ticker8_t::clock

4.51.1.2 uint8_t ticker8_t::period

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

- ticker.h

4.52 ticker_t Struct Reference

`#include <ticker.h>`

Data Fields

- uint16_t period
- uint16_t clock

4.52.1 Field Documentation

4.52.1.1 uint16_t ticker_t::clock

4.52.1.2 uint16_t ticker_t::period

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

- ticker.h

4.53 time Struct Reference

`#include <time.h>`

Data Fields

- uint16_t us_ticks
- uint16_t ms_ticks

4.53.1 Field Documentation

4.53.1.1 uint16_t time::ms_ticks

4.53.1.2 uint16_t time::us_ticks

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

- time.h

4.54 tweeter_private_t Struct Reference

`#include <tweeter.h>`

Data Fields

- uint8_t note_clock
- uint8_t note_period
- uint8_t note_duty
- uint16_t note_holdoff
- uint16_t poll_rate
- tweeter_scale_t * scale_table

4.54.1 Field Documentation

4.54.1.1 uint8_t tweeter_private_t::note_clock

4.54.1.2 uint8_t tweeter_private_t::note_duty

4.54.1.3 uint16_t tweeter_private_t::note_holdoff

4.54.1.4 uint8_t tweeter_private_t::note_period

4.54.1.5 uint16_t tweeter_private_t::poll_rate

4.54.1.6 tweeter_scale_t* tweeter_private_t::scale_table

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

- tweeter.h

4.55 u1wire_enumerate_t Struct Reference

`#include <u1wire_enumerate.h>`

Data Fields

- u1wire_obj_t device
- u1wire_state_t state

4.55.1 Field Documentation

4.55.1.1 u1wire_obj_t u1wire_enumerate_t::device

4.55.1.2 u1wire_state_t u1wire_enumerate_t::state

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

- u1wire_enumerate.h

4.56 u1wire_obj_t Struct Reference

`#include <u1wire.h>`

Data Fields

- u1wire_rom_code_t rom_code

4.56.1 Field Documentation

4.56.1.1 u1wire_rom_code_t u1wire_obj_t::rom_code

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

- u1wire.h

4.57 u1wire_rom_code_t Union Reference

`#include <u1wire.h>`

Data Fields

- struct {
  uint8_t family
  uint8_t serial [6]
  uint8_t crc
  } fields

- uint8_t bytes [8]

4.57.1 Field Documentation

4.57.1.1 uint8_t u1wire_rom_code_t::bytes[8]

4.57.1.2 uint8_t u1wire_rom_code_t::crc

4.57.1.3 uint8_t u1wire_rom_code_t::family

4.57.1.4 struct { ... } u1wire_rom_code_t::fields

4.57.1.5 uint8_t u1wire_rom_code_t::serial[6]

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

- u1wire.h

4.58 u1wire_state_t Struct Reference

`#include <u1wire_enumerate.h>`

Data Fields

- int8_t last_discrepancy
- int8_t last_device
- int8_t last_family_discrepancy

4.58.1 Field Documentation

4.58.1.1 int8_t u1wire_state_t::last_device

4.58.1.2 int8_t u1wire_state_t::last_discrepancy

4.58.1.3 int8_t u1wire_state_t::last_family_discrepancy

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

- u1wire_discover.c
- u1wire_enumerate.h

4.59 usart_dev_struct Struct Reference

Data Fields

- int8_t(∗ putc)(char ch)
- int8_t(∗ getc)(void)
- bool(∗ read_ready_p)(void)
- bool(∗ write_ready_p)(void)
- bool(∗ write_finished_p)(void)

4.59.1 Field Documentation

4.59.1.1 int8_t(∗ usart_dev_struct::getc)(void)

4.59.1.2 int8_t(∗ usart_dev_struct::putc)(char ch)

4.59.1.3 bool(∗ usart_dev_struct::read_ready_p)(void)

4.59.1.4 bool(∗ usart_dev_struct::write_finished_p)(void)

4.59.1.5 bool(∗ usart_dev_struct::write_ready_p)(void)

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

- usart.c

Chapter 5

mmculib File Documentation

5.1 ads7870.c File Reference

```
#include "config.h"
#include "port.h"
#include "spi.h"
#include "spi_adc.h"
```

Enumerations

- enum { ADS7870_CONVERT = 0x80, ADS7870_REG_READ = 0x40, ADS7870_REG_WRITE = 0x00, ADS7870_REG_16BIT = 0x20 }
- enum ads7870_register_t {
  ADS7870_RESULTLO, ADS7870_RESULTHI, ADS7870_PGAVALID, ADS7870_ADCTRL,
  ADS7870_GAINMUX, ADS7870_DIGIOSTATE, ADS7870_DIGIOCTRL, ADS7870_REFOSC }
- enum ads7870_pga_gain_t {
  ADS7870_PGA_GAIN_1 = 0, ADS7870_PGA_GAIN_2 = 1, ADS7870_PGA_GAIN_4 = 2, ADS7870_PGA_GAIN_5 = 3,
  ADS7870_PGA_GAIN_8 = 4, ADS7870_PGA_GAIN_10 = 5, ADS7870_PGA_GAIN_16 = 6, ADS7870_PGA_GAIN_20 = 7 }
- enum {
  ADS7870_REFOSC_OSCR = BIT (5), ADS7870_REFOSC_OSCE = BIT (4), ADS7870_-REFOSC_REFE = BIT (3), ADS7870_REFOSC_BUFE = BIT (2),
  ADS7870_REFOSC_R2V = BIT (1), ADS7870_REFOSC_RBE = BIT (0) }
- enum { ADS7870_GAINMUX_CNVBSY = BIT (7) }

Functions

- static void ads7870_chip_select (void)
- static void ads7870_chip_deselect (void)
- static void ads7870_reg_write (ads7870_register_t reg, uint8_t val)
- static uint8_t ads7870_reg_read (ads7870_register_t reg)

- int16_t ads7870_read (void)
- static int8_t ads7870_read_ready_p (void)
- void ads7870_channel_start (uint8_t channel, spi_adc_mode_t mode)
- int16_t ads7870_channel_convert (uint8_t channel, spi_adc_mode_t mode)
- void ads7870_init (void)

5.1.1 Enumeration Type Documentation

5.1.1.1 anonymous enum

Enumerator:

ADS7870_CONVERT

ADS7870_REG_READ

ADS7870_REG_WRITE

ADS7870_REG_16BIT

5.1.1.2 anonymous enum

Enumerator:

ADS7870_REFOSC_OSCR

ADS7870_REFOSC_OSCE

ADS7870_REFOSC_REFE

ADS7870_REFOSC_BUFE

ADS7870_REFOSC_R2V

ADS7870_REFOSC_RBE

5.1.1.3 anonymous enum

Enumerator:

ADS7870_GAINMUX_CNVBSY

5.1.1.4 enum ads7870_pga_gain_t

Enumerator:

ADS7870_PGA_GAIN_1

ADS7870_PGA_GAIN_2

ADS7870_PGA_GAIN_4

ADS7870_PGA_GAIN_5

ADS7870_PGA_GAIN_8

ADS7870_PGA_GAIN_10

ADS7870_PGA_GAIN_16

ADS7870_PGA_GAIN_20

5.1.1.5 enum ads7870_register_t

Enumerator:

ADS7870_RESULTLO

ADS7870_RESULTHI

ADS7870_PGAVALID

ADS7870_ADCTRL

ADS7870_GAINMUX

ADS7870_DIGIOSTATE

ADS7870_DIGIOCTRL

ADS7870_REFOSC

5.1.2 Function Documentation

5.1.2.1 int16_t ads7870_channel_convert (uint8_t *channel*, spi_adc_mode_t *mode*)

5.1.2.2 void ads7870_channel_start (uint8_t *channel*, spi_adc_mode_t *mode*)

5.1.2.3 static void ads7870_chip_deselect (void) [static]

5.1.2.4 static void ads7870_chip_select (void) [static]

5.1.2.5 void ads7870_init (void)

5.1.2.6 int16_t ads7870_read (void)

5.1.2.7 static int8_t ads7870_read_ready_p (void) [static]

5.1.2.8 static uint8_t ads7870_reg_read (ads7870_register_t *reg*) [static]

5.1.2.9 static void ads7870_reg_write (ads7870_register_t *reg*, uint8_t *val*) [static]

5.2 ads7870.h File Reference

ADS7870 SPI ADC.

```
#include "config.h"
```

Functions

- void ads7870_init (void)
- int16_t ads7870_read (void)
- int16_t ads7870_convert (void)
- void ads7870_channel_start (uint8_t channel, spi_adc_mode_t mode)
- int16_t ads7870_channel_convert (uint8_t channel, spi_adc_mode_t mode)

5.2.1 Detailed Description

ADS7870 SPI ADC.

Author:
M. P. Hayes, UCECE

Date:
09 August 2007

5.2.2 Function Documentation

5.2.2.1 int16_t ads7870_channel_convert (uint8_t *channel*, spi_adc_mode_t *mode*)

5.2.2.2 void ads7870_channel_start (uint8_t *channel*, spi_adc_mode_t *mode*)

5.2.2.3 int16_t ads7870_convert (void)

5.2.2.4 void ads7870_init (void)

5.2.2.5 int16_t ads7870_read (void)

5.3 ads8325.c File Reference

```
#include "config.h"
#include "port.h"
#include "spi.h"
```

Functions

- static void ads8325_chip_select (void)
- static void ads8325_chip_deselect (void)
- void ads8325_init (void)
- int16_t ads8325_read (void)

5.3.1 Function Documentation

5.3.1.1 static void ads8325_chip_deselect (void) [static]

5.3.1.2 static void ads8325_chip_select (void) [static]

5.3.1.3 void ads8325_init (void)

5.3.1.4 int16_t ads8325_read (void)

5.4 ads8327.c File Reference

```
#include "config.h"
#include "port.h"
#include "spi.h"
#include "spi_adc.h"
```

Functions

- static void ads8327_chip_select (void)
- static void ads8327_chip_deselect (void)
- static uint16_t ads8327_command (uint16_t val)
- void ads8327_init (void)
- int16_t ads8327_read (void)
- int16_t ads8327_convert (void)

5.4.1 Function Documentation

5.4.1.1 static void ads8327_chip_deselect (void) [static]

5.4.1.2 static void ads8327_chip_select (void) [static]

5.4.1.3 static uint16_t ads8327_command (uint16_t *val*) [static]

5.4.1.4 int16_t ads8327_convert (void)

5.4.1.5 void ads8327_init (void)

5.4.1.6 int16_t ads8327_read (void)

5.5 ads8327.h File Reference

ADS8327 SPI ADC.

```
#include "config.h"
```

Functions

- void ads8327_init (void)
- int16_t ads8327_convert (void)
- int16_t ads8327_read (void)

5.5.1 Detailed Description

ADS8327 SPI ADC.

Author:
M. P. Hayes, UCECE

Date:
09 August 2007

5.5.2 Function Documentation

5.5.2.1 int16_t ads8327_convert (void)

5.5.2.2 void ads8327_init (void)

5.5.2.3 int16_t ads8327_read (void)

5.6 biseq.c File Reference

Bidirectional sequencer.

```
#include <limits.h>
#include "biseq.h"
```

Functions

- biseq_t biseq_init (biseq_obj_t *dev, int8_t(*callback)(void *data, char *str), void *callback_data)
- void biseq_set (biseq_t biseq, char *str)
- char * biseq_get (biseq_t biseq)
- void biseq_mode_set (biseq_t biseq, biseq_mode_t mode)
- biseq_mode_t biseq_mode_get (biseq_t biseq)
- int8_t biseq_update (biseq_t biseq)

5.6.1 Detailed Description

Bidirectional sequencer.

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.6.2 Function Documentation

5.6.2.1 char* biseq_get (biseq_t *biseq*)

5.6.2.2 biseq_t biseq_init (biseq_obj_t * *dev*, int8_t(*)(void *data, char *str) *callback*, void * *callback_data*)

5.6.2.3 biseq_mode_t biseq_mode_get (biseq_t *biseq*)

5.6.2.4 void biseq_mode_set (biseq_t *biseq*, biseq_mode_t *mode*)

5.6.2.5 void biseq_set (biseq_t *biseq*, char * *str*)

5.6.2.6 int8_t biseq_update (biseq_t *biseq*)

5.7 biseq.h File Reference

```
#include "config.h"
```

Data Structures

- struct biseq_obj_t

Typedefs

- typedef biseq_struct * biseq_t

Enumerations

- enum biseq_mode_t { BISEQ_MODE_NORMAL, BISEQ_MODE_CYCLE, BISEQ_MODE_NUM }

Functions

- biseq_t biseq_init (biseq_obj_t *dev, int8_t(*callback)(void *data, char *str), void *callback_data)
- void biseq_set (biseq_t biseq, char *str)
- char * biseq_get (biseq_t biseq)
- void biseq_mode_set (biseq_t biseq, biseq_mode_t mode)
- biseq_mode_t biseq_mode_get (biseq_t biseq)
- int8_t biseq_update (biseq_t biseq)

5.7.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.7.2 Typedef Documentation

5.7.2.1 typedef struct biseq_struct* biseq_t

5.7.3 Enumeration Type Documentation

5.7.3.1 enum biseq_mode_t

Enumerator:
BISEQ_MODE_NORMAL

BISEQ_MODE_CYCLE

BISEQ_MODE_NUM

5.7.4 Function Documentation

5.7.4.1 char* biseq_get (biseq_t *biseq*)

5.7.4.2 biseq_t biseq_init (biseq_obj_t * *dev*, int8_t(*)(void *data, char *str) *callback*, void * *callback_data*)

5.7.4.3 biseq_mode_t biseq_mode_get (biseq_t *biseq*)

5.7.4.4 void biseq_mode_set (biseq_t *biseq*, biseq_mode_t *mode*)

5.7.4.5 void biseq_set (biseq_t *biseq*, char * *str*)

5.7.4.6 int8_t biseq_update (biseq_t *biseq*)

5.8 bits.h File Reference

Defines

- #define BITS_MASK(first, last) ((1 << ((last) + 1)) - (1 << (first)))
- #define BITS_CLR(reg, first, last) ((reg) &= BITS_MASK (first, last))
- #define BITS_SET(reg, first, last) ((reg) |= BITS_MASK (first, last))
- #define BITS_EXTRACT(reg, first, last) (((reg) & BITS_MASK (first, last)) >> (first))
- #define BITS_INSERT(reg, val, first, last)

5.8.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.8.2 Define Documentation

5.8.2.1 #define BITS_CLR(reg, first, last) ((reg) &= BITS_MASK (first, last))

5.8.2.2 #define BITS_EXTRACT(reg, first, last) (((reg) & BITS_MASK (first, last)) >> (first))

5.8.2.3 #define BITS_INSERT(reg, val, first, last)

Value:

```
(reg) = ((reg) & ~BITS_MASK (first, last))  \
        | (((val) & BITS_MASK (0, last - first)) << (first))
```

5.8.2.4 #define BITS_MASK(first, last) ((1 << ((last) + 1)) - (1 << (first)))

5.8.2.5 #define BITS_SET(reg, first, last) ((reg) |= BITS_MASK (first, last))

5.9 busart.c File Reference

Buffered USART implementation.

```
#include "ring.h"
#include "busart.h"
#include "peripherals.h"
#include <string.h>
```

Data Structures

- struct busart_dev_struct

Defines

- #define BUSART0_ENABLE (USART_NUM >= 1)
- #define BUSART1_ENABLE (USART_NUM >= 2)

Functions

- busart_t busart_init (uint8_t channel, uint16_t baud_divisor, char *tx_buffer, ring_size_t tx_size, char *rx_buffer, ring_size_t rx_size)
- ring_size_t busart_write (busart_t busart, const void *data, ring_size_t size)
- ring_size_t busart_write_block (busart_t busart, const void *data, ring_size_t size)
- ring_size_t busart_read (busart_t busart, void *data, ring_size_t size)
- ring_size_t busart_read_block (busart_t busart, void *data, ring_size_t size)
- ring_size_t busart_read_num (busart_t busart)
- ring_size_t busart_write_num (busart_t busart)
- bool busart_read_ready_p (busart_t busart)
- bool busart_write_ready_p (busart_t busart)
- bool busart_write_finished_p (busart_t busart)
- int8_t busart_getc (busart_t busart)
- int8_t busart_putc (busart_t busart, char ch)
- int8_t busart_puts (busart_t busart, const char *str)

5.9.1 Detailed Description

Buffered USART implementation.

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.9.2 Define Documentation

5.9.2.1 #define BUSART0_ENABLE (USART_NUM >= 1)

5.9.2.2 #define BUSART1_ENABLE (USART_NUM >= 2)

5.9.3 Function Documentation

5.9.3.1 int8_t busart_getc (busart_t *busart*)

5.9.3.2 busart_t busart_init (uint8_t *channel*, uint16_t *baud_divisor*, char * *tx_buffer*, ring_size_t *tx_size*, char * *rx_buffer*, ring_size_t *rx_size*)

5.9.3.3 int8_t busart_putc (busart_t *busart*, char *ch*)

5.9.3.4 int8_t busart_puts (busart_t *busart*, const char * *str*)

5.9.3.5 ring_size_t busart_read (busart_t *busart*, void * *data*, ring_size_t *size*)

5.9.3.6 ring_size_t busart_read_block (busart_t *busart*, void * *data*, ring_size_t *size*)

5.9.3.7 ring_size_t busart_read_num (busart_t *busart*)

5.9.3.8 bool busart_read_ready_p (busart_t *busart*)

5.9.3.9 ring_size_t busart_write (busart_t *busart*, const void * *data*, ring_size_t *size*)

5.9.3.10 ring_size_t busart_write_block (busart_t *busart*, const void * *data*, ring_size_t *size*)

5.9.3.11 bool busart_write_finished_p (busart_t *busart*)

5.9.3.12 ring_size_t busart_write_num (busart_t *busart*)

5.9.3.13 bool busart_write_ready_p (busart_t *busart*)

5.10 busart.h File Reference

Buffered USART interface.

```
#include "config.h"
#include "ring.h"
#include "usart0.h"
```

Defines

- #define BUSART_BAUD_DIVISOR(BAUD_RATE) USART0_BAUD_DIVISOR(BAUD_RATE)

Typedefs

- typedef busart_dev_struct busart_dev_t
- typedef busart_dev_t * busart_t

Functions

- busart_t busart_init (uint8_t channel, uint16_t baud_divisor, char *tx_buffer, ring_size_t tx_size, char *rx_buffer, ring_size_t rx_size)
- ring_size_t busart_read (busart_t busart, void *data, ring_size_t size)
- ring_size_t busart_read_block (busart_t busart, void *data, ring_size_t size)
- ring_size_t busart_write (busart_t busart, const void *data, ring_size_t size)
- ring_size_t busart_write_block (busart_t busart, const void *data, ring_size_t size)
- ring_size_t busart_read_num (busart_t busart)
- ring_size_t busart_write_num (busart_t busart)
- bool busart_read_ready_p (busart_t busart)
- bool busart_write_ready_p (busart_t busart)
- bool busart_write_finished_p (busart_t busart)
- int8_t busart_getc (busart_t busart)
- int8_t busart_putc (busart_t busart, char ch)
- int8_t busart_puts (busart_t busart, const char *str)

5.10.1 Detailed Description

Buffered USART interface.

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.10.2 Define Documentation

5.10.2.1 #define BUSART_BAUD_DIVISOR(BAUD_RATE) USART0_BAUD_DIVISOR(BAUD_RATE)

5.10.3 Typedef Documentation

5.10.3.1 typedef struct busart_dev_struct busart_dev_t

5.10.3.2 typedef busart_dev_t* busart_t

5.10.4 Function Documentation

5.10.4.1 int8_t busart_getc (busart_t *busart*)

5.10.4.2 busart_t busart_init (uint8_t *channel*, uint16_t *baud_divisor*, char * *tx_buffer*, ring_size_t *tx_size*, char * *rx_buffer*, ring_size_t *rx_size*)

5.10.4.3 int8_t busart_putc (busart_t *busart*, char *ch*)

5.10.4.4 int8_t busart_puts (busart_t *busart*, const char * *str*)

5.10.4.5 ring_size_t busart_read (busart_t *busart*, void * *data*, ring_size_t *size*)

5.10.4.6 ring_size_t busart_read_block (busart_t *busart*, void * *data*, ring_size_t *size*)

5.10.4.7 ring_size_t busart_read_num (busart_t *busart*)

5.10.4.8 bool busart_read_ready_p (busart_t *busart*)

5.10.4.9 ring_size_t busart_write (busart_t *busart*, const void * *data*, ring_size_t *size*)

5.10.4.10 ring_size_t busart_write_block (busart_t *busart*, const void * *data*, ring_size_t *size*)

5.10.4.11 bool busart_write_finished_p (busart_t *busart*)

5.10.4.12 ring_size_t busart_write_num (busart_t *busart*)

5.10.4.13 bool busart_write_ready_p (busart_t *busart*)

5.11 button.c File Reference

```
#include "button.h"
#include "delay.h"
```

Functions

- void button_poll_count_set (uint8_t poll_count)
- button_t button_init (button_obj_t *info, const button_cfg_t *cfg)
- button_state_t button_debounce (button_t button, bool pressed)
- button_state_t button_poll (button_t button)
- bool button_held_p (button_t button, uint8_t hold_count)
- bool button_hold_released_p (button_t button, uint8_t hold_count)

Variables

- static uint8_t button_poll_count

5.11.1 Detailed Description

Author:
M.P. Hayes

Date:
16 Feb 2003

Description: This debounces pushbuttons and switches using a polled wait-and-see method implemented with a state machine. It assumes that a pushed button gives a logic low on the input port.

5.11.2 Function Documentation

5.11.2.1 button_state_t button_debounce (button_t *button*, bool *pressed*)

Poll the specified button and return its debounced state.

5.11.2.2 bool button_held_p (button_t *button*, uint8_t *hold_count*)

Return true if button held for hold_count.

5.11.2.3 bool button_hold_released_p (button_t *button*, uint8_t *hold_count*)

Return true if button held for hold_count has been released.

5.11.2.4 button_t button_init (button_obj_t * *info*, const button_cfg_t * *cfg*)

Create a new button object.

5.11.2.5 button_state_t button_poll (button_t *button*)

Poll the specified button and return its debounced state.

5.11.2.6 void button_poll_count_set (uint8_t *poll_count*)

Set the number of polls required for the debounce period.

5.11.3 Variable Documentation

5.11.3.1 uint8_t button_poll_count [static]

5.12 button.h File Reference

Button polling and debouncing.

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct button_cfg_t
- struct button_private_t

Defines

- #define BUTTON_DEBOUNCE_RATE (1000 / BUTTON_DEBOUNCE_MS)
- #define BUTTON_POLL_COUNT(POLL_RATE) ((POLL_RATE) / BUTTON_DEBOUNCE_RATE)
- #define BUTTON_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

Typedefs

- typedef button_private_t button_obj_t
- typedef button_obj_t * button_t

Enumerations

- enum { BUTTON_DEBOUNCE_MS = 50 }
- enum button_state_t { BUTTON_STATE_UP, BUTTON_STATE_DOWN, BUTTON_STATE_PUSHED, BUTTON_STATE_RELEASED }

Functions

- void button_poll_count_set (uint8_t poll_count)
- button_t button_init (button_obj_t *info, const button_cfg_t *cfg)
- button_state_t button_poll (button_t button)
- bool button_held_p (button_t button, uint8_t hold_count)
- bool button_hold_released_p (button_t button, uint8_t hold_count)
- static button_state_t button_state_get (button_t button)
- static uint8_t button_hold_count_get (button_t button)
- static bool button_pushed_p (button_t button)
- static bool button_released_p (button_t button)
- static bool button_down_p (button_t button)
- static bool button_pressed_p (button_t button)

5.12.1 Detailed Description

Button polling and debouncing.

Author:
M. P. Hayes, UCECE

Date:
15 Feb 2003

5.12.2 Define Documentation

5.12.2.1 #define BUTTON_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.12.2.2 #define BUTTON_DEBOUNCE_RATE (1000 / BUTTON_DEBOUNCE_MS)

5.12.2.3 #define BUTTON_POLL_COUNT(POLL_RATE) ((POLL_RATE) / BUTTON_DEBOUNCE_RATE)

5.12.3 Typedef Documentation

5.12.3.1 typedef button_private_t button_obj_t

5.12.3.2 typedef button_obj_t* button_t

5.12.4 Enumeration Type Documentation

5.12.4.1 anonymous enum

Debounce period in milliseconds.

Enumerator:
BUTTON_DEBOUNCE_MS

5.12.4.2 enum button_state_t

Button states.

Enumerator:
BUTTON_STATE_UP

BUTTON_STATE_DOWN

BUTTON_STATE_PUSHED

BUTTON_STATE_RELEASED

5.12.5 Function Documentation

5.12.5.1 static bool button_down_p (button_t *button*) [static]

Return true if button down (pressed).

5.12.5.2 bool button_held_p (button_t *button*, uint8_t *hold_count*)

Return true if button held for hold_count.

5.12.5.3 static uint8_t button_hold_count_get (button_t *button*) [static]

Return duration button held (in polling periods).

5.12.5.4 bool button_hold_released_p (button_t *button*, uint8_t *hold_count*)

Return true if button held for hold_count has been released.

5.12.5.5 button_t button_init (button_obj_t * *info*, const button_cfg_t * *cfg*)

Create a new button object.

5.12.5.6 button_state_t button_poll (button_t *button*)

Poll the specified button and return its debounced state.

5.12.5.7 void button_poll_count_set (uint8_t *poll_count*)

Set the number of polls required for the debounce period.

5.12.5.8 static bool button_pressed_p (button_t *button*) [static]

Return true if button possibly pressed. Note this is not debounced. For a debounced version use button_-pushed_p.

5.12.5.9 static bool button_pushed_p (button_t *button*) [static]

Return true if button pushed.

5.12.5.10 static bool button_released_p (button_t *button*) [static]

Return true if button released.

5.12.5.11 static button_state_t button_state_get (button_t *button*) [static]

Return button state.

5.13 buttons.c File Reference

```
#include "buttons.h"
```

Functions

- void buttons_init (buttons_obj_t *buttons, button_obj_t *button_objs, const button_cfg_t *config, uint8_t buttons_num, uint8_t poll_count)
- void buttons_poll (buttons_t buttons)
- bool buttons_any_pushed_p (buttons_t buttons)

5.13.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.13.2 Function Documentation

5.13.2.1 bool buttons_any_pushed_p (buttons_t *buttons*)

Return true if any buttons pushed.

5.13.2.2 void buttons_init (buttons_obj_t * *buttons*, button_obj_t * *buttons_info*, const button_cfg_t * *config*, uint8_t *buttons_num*, uint8_t *poll_count*)

Initialise multiple buttons.

5.13.2.3 void buttons_poll (buttons_t *buttons*)

Poll multiple buttons.

5.14 buttons.h File Reference

Multiple button polling and debouncing.

`#include "button.h"`

Data Structures

- struct buttons_private_t

Typedefs

- typedef buttons_private_t buttons_obj_t
- typedef buttons_obj_t * buttons_t

Functions

- void buttons_init (buttons_obj_t *buttons, button_obj_t *buttons_info, const button_cfg_t *config, uint8_t buttons_num, uint8_t poll_count)
- void buttons_poll (buttons_t buttons)
- bool buttons_any_pushed_p (buttons_t buttons)
- static bool buttons_pushed_p (buttons_t buttons, uint8_t id)
- static bool buttons_released_p (buttons_t buttons, uint8_t id)
- static bool buttons_held_p (buttons_t buttons, uint8_t id, uint8_t hold_count)
- static bool buttons_hold_released_p (buttons_t buttons, uint8_t id, uint8_t hold_count)

5.14.1 Detailed Description

Multiple button polling and debouncing.

Author:
M. P. Hayes, UCECE

Date:
17 Nov 2006

5.14.2 Typedef Documentation

5.14.2.1 typedef buttons_private_t buttons_obj_t

5.14.2.2 typedef buttons_obj_t* buttons_t

5.14.3 Function Documentation

5.14.3.1 bool buttons_any_pushed_p (buttons_t *buttons*)

Return true if any buttons pushed.

5.14.3.2 static bool buttons_held_p (buttons_t *buttons*, uint8_t *id*, uint8_t *hold_count*) [static]

Return true if selected button held for hold_count.

5.14.3.3 static bool buttons_hold_released_p (buttons_t *buttons*, uint8_t *id*, uint8_t *hold_count*) [static]

Return true if selected button held for hold_count is released.

5.14.3.4 void buttons_init (buttons_obj_t * *buttons*, button_obj_t * *buttons_info*, const button_cfg_t * *config*, uint8_t *buttons_num*, uint8_t *poll_count*)

Initialise multiple buttons.

5.14.3.5 void buttons_poll (buttons_t *buttons*)

Poll multiple buttons.

5.14.3.6 static bool buttons_pushed_p (buttons_t *buttons*, uint8_t *id*) [static]

Return true if selected button pushed.

5.14.3.7 static bool buttons_released_p (buttons_t *buttons*, uint8_t *id*) [static]

Return true if selected button released.

5.15 chaser.c File Reference

```
#include <limits.h>
#include "chaser.h"
#include "flasher.h"
#include "font.h"
```

Functions

- chaser_t chaser_init (chaser_obj_t *dev, flasher_t *flashers, uint8_t flasher_num)
- void chaser_sequence_set (chaser_t chaser, chaser_sequence_t seq)
- void chaser_mode_set (chaser_t chaser, chaser_mode_t mode)
- static void chaser_pixel_set (void *data, uint8_t pixel, bool val)
- int8_t chaser_update (chaser_t chaser)

5.15.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.15.2 Function Documentation

5.15.2.1 chaser_t chaser_init (chaser_obj_t * *dev*, flasher_t * *flashers*, uint8_t *flasher_num*)

5.15.2.2 void chaser_mode_set (chaser_t *chaser*, chaser_mode_t *mode*)

5.15.2.3 static void chaser_pixel_set (void * *data*, uint8_t *pixel*, bool *val*) [static]

5.15.2.4 void chaser_sequence_set (chaser_t *chaser*, chaser_sequence_t *seq*)

5.15.2.5 int8_t chaser_update (chaser_t *chaser*)

5.16 chaser.h File Reference

```
#include "config.h"
#include "flasher.h"
#include "font.h"
```

Data Structures

- struct chaser_private_t

Typedefs

- typedef char chaser_font_index_t
- typedef chaser_font_index_t * chaser_sequence_t
- typedef chaser_private_t chaser_obj_t
- typedef chaser_obj_t * chaser_t

Enumerations

- enum chaser_mode_t {

  CHASER_MODE_NORMAL, CHASER_MODE_CYCLE, CHASER_MODE_INVERT, CHASER_MODE_CYCLE_INVERT,

  CHASER_MODE_NUM }

Functions

- chaser_t chaser_init (chaser_obj_t *dev, flasher_t *flashers, uint8_t flasher_num)
- void chaser_sequence_set (chaser_t chaser, chaser_sequence_t seq)
- void chaser_mode_set (chaser_t chaser, chaser_mode_t mode)
- int8_t chaser_update (chaser_t chaser)
- static void chaser_patterns_set (chaser_t chaser, flasher_pattern_t *on_pattern, flasher_pattern_t *off_pattern)
- static chaser_sequence_t chaser_sequence_get (chaser_t chaser)
- static void chaser_font_set (chaser_t chaser, font_t *font)

5.16.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.16.2 Typedef Documentation

5.16.2.1 typedef char chaser_font_index_t

5.16.2.2 typedef chaser_private_t chaser_obj_t

5.16.2.3 typedef chaser_font_index_t* chaser_sequence_t

5.16.2.4 typedef chaser_obj_t* chaser_t

5.16.3 Enumeration Type Documentation

5.16.3.1 enum chaser_mode_t

Enumerator:

CHASER_MODE_NORMAL

CHASER_MODE_CYCLE

CHASER_MODE_INVERT

CHASER_MODE_CYCLE_INVERT

CHASER_MODE_NUM

5.16.4 Function Documentation

5.16.4.1 static void chaser_font_set (chaser_t *chaser*, font_t * *font*) [static]

5.16.4.2 chaser_t chaser_init (chaser_obj_t * *dev*, flasher_t * *flashers*, uint8_t *flasher_num*)

5.16.4.3 void chaser_mode_set (chaser_t *chaser*, chaser_mode_t *mode*)

5.16.4.4 static void chaser_patterns_set (chaser_t *chaser*, flasher_pattern_t * *on_pattern*, flasher_pattern_t * *off_pattern*) [static]

5.16.4.5 static chaser_sequence_t chaser_sequence_get (chaser_t *chaser*) [static]

5.16.4.6 void chaser_sequence_set (chaser_t *chaser*, chaser_sequence_t *seq*)

5.16.4.7 int8_t chaser_update (chaser_t *chaser*)

5.17 cleds.c File Reference

```
#include "cleds.h"
#include "port.h"
```

Functions

- void cleds_init (cleds_obj_t *cleds, const led_cfg_t *row_config, uint8_t rows_num, const led_cfg_t *col_config, uint8_t cols_num)
- uint8_t cleds_common_set (cleds_t cleds, uint8_t row)

5.17.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
29 June 2007

5.17.2 Function Documentation

5.17.2.1 uint8_t cleds_common_set (cleds_t *cleds*, uint8_t *row*)

5.17.2.2 void cleds_init (cleds_obj_t * *cleds*, const led_cfg_t * *row_config*, uint8_t *rows_num*, const led_cfg_t * *col_config*, uint8_t *cols_num*)

5.18 cleds.h File Reference

```
#include "config.h"
#include "port.h"
#include "led.h"
```

Data Structures

- struct cleds_private_t

Typedefs

- typedef cleds_private_t cleds_obj_t
- typedef cleds_obj_t * cleds_t

Functions

- void cleds_init (cleds_obj_t *cleds, const led_cfg_t *row_config, uint8_t rows_num, const led_cfg_t *col_config, uint8_t cols_num)
- uint8_t cleds_common_set (cleds_t cleds, uint8_t row)
- static void cleds_set (cleds_t cleds, uint8_t id, uint8_t val)
- static uint8_t cleds_cols_num_get (cleds_t cleds)
- static uint8_t cleds_rows_num_get (cleds_t cleds)
- static uint8_t cleds_active_row_get (cleds_t cleds)
- static uint8_t cleds_common_cycle (cleds_t cleds)

5.18.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
29 June 2007

5.18.2 Typedef Documentation

5.18.2.1 typedef cleds_private_t cleds_obj_t

5.18.2.2 typedef cleds_obj_t* cleds_t

5.18.3 Function Documentation

5.18.3.1 static uint8_t cleds_active_row_get (cleds_t *cleds*) [static]

5.18.3.2 static uint8_t cleds_cols_num_get (cleds_t *cleds*) [static]

5.18.3.3 static uint8_t cleds_common_cycle (cleds_t *cleds*) [static]

5.18.3.4 uint8_t cleds_common_set (cleds_t *cleds*, uint8_t *row*)

5.18.3.5 void cleds_init (cleds_obj_t * *cleds*, const led_cfg_t * *row_config*, uint8_t *rows_num*, const led_cfg_t * *col_config*, uint8_t *cols_num*)

5.18.3.6 static uint8_t cleds_rows_num_get (cleds_t *cleds*) [static]

5.18.3.7 static void cleds_set (cleds_t *cleds*, uint8_t *id*, uint8_t *val*) [static]

5.19 colourmap.h File Reference

```
#include "config.h"
```

Defines

- #define COLOURMAP_R_WEIGHT 1.0
- #define COLOURMAP_G_WEIGHT 1.0
- #define COLOURMAP_B_WEIGHT 1.0
- #define COLOURMAP_R(X) ((X) * COLOURMAP_R_WEIGHT * COLOURMAP_SCALE + 0.5)
- #define COLOURMAP_G(X) ((X) * COLOURMAP_G_WEIGHT * COLOURMAP_SCALE + 0.5)
- #define COLOURMAP_B(X) ((X) * COLOURMAP_B_WEIGHT * COLOURMAP_SCALE + 0.5)
- #define COLOURMAP_ENTRY(R, G, B) {COLOURMAP_R(R), COLOURMAP_G(G), COLOURMAP_B(B)},

Typedefs

- typedef uint8_t colourmap_elt_t
- typedef colourmap_elt_t colourmap_t [3]

5.19.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
3 July 2007

5.19.2 Define Documentation

5.19.2.1 #define COLOURMAP_B(X) ((X) * COLOURMAP_B_WEIGHT * COLOURMAP_SCALE + 0.5)

5.19.2.2 #define COLOURMAP_B_WEIGHT 1.0

5.19.2.3 #define COLOURMAP_ENTRY(R, G, B) {COLOURMAP_R(R), COLOURMAP_G(G), COLOURMAP_B(B)},

5.19.2.4 #define COLOURMAP_G(X) ((X) * COLOURMAP_G_WEIGHT * COLOURMAP_SCALE + 0.5)

5.19.2.5 #define COLOURMAP_G_WEIGHT 1.0

5.19.2.6 #define COLOURMAP_R(X) ((X) * COLOURMAP_R_WEIGHT * COLOURMAP_SCALE + 0.5)

5.19.2.7 #define COLOURMAP_R_WEIGHT 1.0

5.19.3 Typedef Documentation

5.19.3.1 typedef uint8_t colourmap_elt_t

5.19.3.2 typedef colourmap_elt_t colourmap_t[3]

5.20 crc8541.c File Reference

```
#include "crc8541.h"
```

Functions

- static crc8_t crc8541_bit (crc8_t crc, uint8_t in)
- crc8_t crc8541_byte (crc8_t crc, uint8_t val)
- crc8_t crc8541 (crc8_t crc, uint8_t *bytes, uint8_t size)

5.20.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.20.2 Function Documentation

5.20.2.1 crc8_t crc8541 (crc8_t *crc*, uint8_t * *bytes*, uint8_t *size*)

5.20.2.2 static crc8_t crc8541_bit (crc8_t *crc*, uint8_t *in*) [static]

5.20.2.3 crc8_t crc8541_byte (crc8_t *crc*, uint8_t *val*)

5.21 crc8541.h File Reference

```
#include "config.h"
```

Typedefs

- typedef uint8_t crc8_t

Functions

- crc8_t crc8541_byte (crc8_t crc, uint8_t val)
- crc8_t crc8541 (crc8_t crc, uint8_t *bytes, uint8_t size)

5.21.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.21.2 Typedef Documentation

5.21.2.1 typedef uint8_t crc8_t

5.21.3 Function Documentation

5.21.3.1 crc8_t crc8541 (crc8_t *crc*, uint8_t * *bytes*, uint8_t *size*)

5.21.3.2 crc8_t crc8541_byte (crc8_t *crc*, uint8_t *val*)

5.22 crc8541_test.c File Reference

```
#include <stdlib.h>
#include "crc8541.h"
```

Functions

- int main (int argc, char **argv)

5.22.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.22.2 Function Documentation

5.22.2.1 int main (int *argc*, char ** *argv*)

5.23 ds18b20.c File Reference

```
#include <stdio.h>
#include "ds18b20.h"
#include "u1wire.h"
#include "delay.h"
#include "dscrc8.h"
```

Defines

- #define DS18B20_DEBUG 0
- #define DS18B20_CRC_CHECK 0

Enumerations

- enum { DS18B20_FAMILY_CODE = 0x28, DS1820_FAMILY_CODE = 0x10, DS18S20_FAMILY_CODE = 0x10 }
- enum { DS18B20_SCRATCHPAD_BYTES = 9 }
- enum { DS18B20_CONVERT_T = 0x44, DS18B20_READ_SCRATCHPAD = 0xbe, DS18B20_WRITE_SCRATCHPAD = 0x4e }

Functions

- int8_t ds18b20_temp_conversion_start (u1wire_t dev)
- bool ds18b20_temp_ready_p (u1wire_t dev)
- int8_t ds18b20_temp_read (u1wire_t dev, ds18b20_temp_t *ptemp)
- bool ds18b20_device_p (u1wire_obj_t *dev)
- u1wire_t ds18b20_init (u1wire_obj_t *dev)

Variables

- static uint8_t ds18b20_data [DS18B20_SCRATCHPAD_BYTES]

5.23.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.23.2 Define Documentation

5.23.2.1 #define DS18B20_CRC_CHECK 0

5.23.2.2 #define DS18B20_DEBUG 0

5.23.3 Enumeration Type Documentation

5.23.3.1 anonymous enum

Enumerator:

DS18B20_FAMILY_CODE

DS1820_FAMILY_CODE

DS18S20_FAMILY_CODE

5.23.3.2 anonymous enum

Enumerator:

DS18B20_SCRATCHPAD_BYTES

5.23.3.3 anonymous enum

Enumerator:

DS18B20_CONVERT_T

DS18B20_READ_SCRATCHPAD

DS18B20_WRITE_SCRATCHPAD

5.23.4 Function Documentation

5.23.4.1 bool ds18b20_device_p (u1wire_obj_t * *dev*)

5.23.4.2 u1wire_t ds18b20_init (u1wire_obj_t * *dev*)

5.23.4.3 int8_t ds18b20_temp_conversion_start (u1wire_t *dev*)

5.23.4.4 int8_t ds18b20_temp_read (u1wire_t *dev*, ds18b20_temp_t * *ptemp*)

5.23.4.5 bool ds18b20_temp_ready_p (u1wire_t *dev*)

5.23.5 Variable Documentation

5.23.5.1 uint8_t ds18b20_data[DS18B20_SCRATCHPAD_BYTES] [static]

5.24 ds18b20.h File Reference

```
#include "config.h"
#include "u1wire.h"
```

Defines

- #define DS18B20_TEMP_INT(TEMP) (((TEMP) + DS18B20_COUNTS_PER_DEGREE / 2) >> DS18B20_BITS_PER_DEGREE)
- #define DS18B20_TEMP_DOUBLE(TEMP) ((TEMP) / (double)DS18B20_COUNTS_PER_-DEGREE + 0.5)

Typedefs

- typedef int16_t ds18b20_temp_t

Enumerations

- enum { DS18B20_BITS_PER_DEGREE = 8 }
- enum { DS18B20_COUNTS_PER_DEGREE = (1 << 8) }
- enum { DS18B20_TEMP_BITS = 12, DS1820_TEMP_BITS = 9 }

Functions

- int8_t ds18b20_temp_conversion_start (u1wire_t dev)
- bool ds18b20_temp_ready_p (u1wire_t dev)
- int8_t ds18b20_temp_read (u1wire_t dev, ds18b20_temp_t *temp)
- bool ds18b20_device_p (u1wire_obj_t *dev)
- u1wire_t ds18b20_init (u1wire_obj_t *dev)

5.24.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.24.2 Define Documentation

5.24.2.1 #define DS18B20_TEMP_DOUBLE(TEMP) ((TEMP) / (double)DS18B20_COUNTS_PER_DEGREE + 0.5)

5.24.2.2 #define DS18B20_TEMP_INT(TEMP) (((TEMP) + DS18B20_COUNTS_PER_DEGREE / 2) >> DS18B20_BITS_PER_DEGREE)

5.24.3 Typedef Documentation

5.24.3.1 typedef int16_t ds18b20_temp_t

5.24.4 Enumeration Type Documentation

5.24.4.1 anonymous enum

Enumerator:

DS18B20_BITS_PER_DEGREE

5.24.4.2 anonymous enum

Enumerator:

DS18B20_COUNTS_PER_DEGREE

5.24.4.3 anonymous enum

Enumerator:

DS18B20_TEMP_BITS

DS1820_TEMP_BITS

5.24.5 Function Documentation

5.24.5.1 bool ds18b20_device_p (u1wire_obj_t * *dev*)

5.24.5.2 u1wire_t ds18b20_init (u1wire_obj_t * *dev*)

5.24.5.3 int8_t ds18b20_temp_conversion_start (u1wire_t *dev*)

5.24.5.4 int8_t ds18b20_temp_read (u1wire_t *dev*, ds18b20_temp_t * *temp*)

5.24.5.5 bool ds18b20_temp_ready_p (u1wire_t *dev*)

5.25 ds2450.c File Reference

```
#include <stdio.h>
#include "ds2450.h"
#include "dscrc16.h"
#include "u1wire.h"
#include "delay.h"
```

Defines

- #define DS2450_DEBUG 1
- #define DS2450_CRC_CHECK 1

Enumerations

- enum { DS2450_FAMILY_CODE = 0x20 }
- enum { DS2450_MEMORY_BYTES = 24 }
- enum { DS2450_CONVERT = 0x3c, DS2450_READ_MEMORY = 0xaa, DS2450_WRITE_MEMORY = 0x55 }

Functions

- int8_t ds2450_adc_conversion_start (u1wire_t dev, uint8_t channel_mask)
- bool ds2450_adc_ready_p (u1wire_t dev)
- static int8_t ds2450_memory_read (u1wire_t dev, uint16_t addr, uint8_t *data, uint8_t size)
- static int8_t ds2450_memory_write (u1wire_t dev, uint16_t addr, uint8_t *data, uint8_t size)
- int8_t ds2450_adc_read (u1wire_t dev, uint8_t channel_mask, uint16_t *adc)
- bool ds2450_device_p (u1wire_obj_t *dev)
- void ds2450_debug (u1wire_t dev)
- u1wire_t ds2450_init (u1wire_obj_t *dev)

5.25.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.25.2 Define Documentation

5.25.2.1 #define DS2450_CRC_CHECK 1

5.25.2.2 #define DS2450_DEBUG 1

5.25.3 Enumeration Type Documentation

5.25.3.1 anonymous enum

Enumerator:

DS2450_FAMILY_CODE

5.25.3.2 anonymous enum

Enumerator:

DS2450_MEMORY_BYTES

5.25.3.3 anonymous enum

Enumerator:

DS2450_CONVERT

DS2450_READ_MEMORY

DS2450_WRITE_MEMORY

5.25.4 Function Documentation

5.25.4.1 int8_t ds2450_adc_conversion_start (u1wire_t *dev*, uint8_t *channel_mask*)

5.25.4.2 int8_t ds2450_adc_read (u1wire_t *dev*, uint8_t *channel_mask*, uint16_t * *adc*)

5.25.4.3 bool ds2450_adc_ready_p (u1wire_t *dev*)

5.25.4.4 void ds2450_debug (u1wire_t *dev*)

5.25.4.5 bool ds2450_device_p (u1wire_obj_t * *dev*)

5.25.4.6 u1wire_t ds2450_init (u1wire_obj_t * *dev*)

5.25.4.7 static int8_t ds2450_memory_read (u1wire_t *dev*, uint16_t *addr*, uint8_t * *data*, uint8_t *size*) [static]

5.25.4.8 static int8_t ds2450_memory_write (u1wire_t *dev*, uint16_t *addr*, uint8_t * *data*, uint8_t *size*) [static]

5.26 ds2450.h File Reference

```
#include "config.h"
#include "u1wire.h"
```

Enumerations

- enum { DS2450_CHANNELS_NUM = 4 }

Functions

- int8_t ds2450_adc_conversion_start (u1wire_t dev, uint8_t channel_mask)
- bool ds2450_adc_ready_p (u1wire_t dev)
- int8_t ds2450_adc_read (u1wire_t dev, uint8_t channel_mask, uint16_t *adc)
- bool ds2450_device_p (u1wire_obj_t *dev)
- void ds2450_debug (u1wire_t dev)
- u1wire_t ds2450_init (u1wire_obj_t *dev)

5.26.1 Enumeration Type Documentation

5.26.1.1 anonymous enum

Enumerator:
DS2450_CHANNELS_NUM

5.26.2 Function Documentation

5.26.2.1 int8_t ds2450_adc_conversion_start (u1wire_t *dev*, uint8_t *channel_mask*)

5.26.2.2 int8_t ds2450_adc_read (u1wire_t *dev*, uint8_t *channel_mask*, uint16_t * *adc*)

5.26.2.3 bool ds2450_adc_ready_p (u1wire_t *dev*)

5.26.2.4 void ds2450_debug (u1wire_t *dev*)

5.26.2.5 bool ds2450_device_p (u1wire_obj_t * *dev*)

5.26.2.6 u1wire_t ds2450_init (u1wire_obj_t * *dev*)

5.27 dscrc16.c File Reference

```
#include "dscrc16.h"
```

Functions

- static crc16_t dscrc16_bit (crc16_t crc, uint8_t in)
- crc16_t dscrc16_byte (crc16_t crc, uint8_t val)
- crc16_t dscrc16 (crc16_t crc, void *bytes, uint8_t size)

5.27.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.27.2 Function Documentation

5.27.2.1 crc16_t dscrc16 (crc16_t *crc*, void * *bytes*, uint8_t *size*)

5.27.2.2 static crc16_t dscrc16_bit (crc16_t *crc*, uint8_t *in*) [static]

5.27.2.3 crc16_t dscrc16_byte (crc16_t *crc*, uint8_t *val*)

5.28 dscrc16.h File Reference

```
#include "config.h"
```

Typedefs

- typedef uint16_t crc16_t

Functions

- crc16_t dscrc16_byte (crc16_t crc, uint8_t val)
- crc16_t dscrc16 (crc16_t crc, void *bytes, uint8_t size)

5.28.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.28.2 Typedef Documentation

5.28.2.1 typedef uint16_t crc16_t

5.28.3 Function Documentation

5.28.3.1 crc16_t dscrc16 (crc16_t *crc*, void * *bytes*, uint8_t *size*)

5.28.3.2 crc16_t dscrc16_byte (crc16_t *crc*, uint8_t *val*)

5.29 dscrc8.c File Reference

```
#include "dscrc8.h"
```

Functions

- crc8_t dscrc8_byte (crc8_t crc, uint8_t data)
- crc8_t dscrc8 (crc8_t crc, uint8_t *bytes, uint8_t size)

5.29.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.29.2 Function Documentation

5.29.2.1 crc8_t dscrc8 (crc8_t *crc*, uint8_t * *bytes*, uint8_t *size*)

5.29.2.2 crc8_t dscrc8_byte (crc8_t *crc*, uint8_t *data*)

5.30 dscrc8.h File Reference

```
#include "config.h"
```

Typedefs

- typedef uint8_t crc8_t

Functions

- crc8_t dscrc8_byte (crc8_t crc, uint8_t val)
- crc8_t dscrc8 (crc8_t crc, uint8_t *bytes, uint8_t size)

5.30.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.30.2 Typedef Documentation

5.30.2.1 typedef uint8_t crc8_t

5.30.3 Function Documentation

5.30.3.1 crc8_t dscrc8 (crc8_t *crc*, uint8_t * *bytes*, uint8_t *size*)

5.30.3.2 crc8_t dscrc8_byte (crc8_t *crc*, uint8_t *val*)

5.31 flasher.c File Reference

```
#include "flasher.h"
```

Defines

- #define FLASHER_TRANSPARENT

Functions

- int8_t flasher_pattern_set (flasher_t flasher, flasher_pattern_t *pattern)
- flasher_pattern_t * flasher_pattern_get (flasher_t flasher)
- int8_t flasher_phase_set (flasher_t flasher, uint8_t phase)
- bool flasher_update (flasher_t flasher)
- flasher_t flasher_init (flasher_obj_t *flasher)

5.31.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
13 March 2005

5.31.2 Define Documentation

5.31.2.1 #define FLASHER_TRANSPARENT

5.31.3 Function Documentation

5.31.3.1 flasher_t flasher_init (flasher_obj_t * *flasher*)

5.31.3.2 flasher_pattern_t* flasher_pattern_get (flasher_t *flasher*)

5.31.3.3 int8_t flasher_pattern_set (flasher_t *flasher*, flasher_pattern_t * *pattern*)

5.31.3.4 int8_t flasher_phase_set (flasher_t *flasher*, uint8_t *phase*)

5.31.3.5 bool flasher_update (flasher_t *flasher*)

5.32 flasher.h File Reference

```
#include "config.h"
```

Data Structures

- struct flasher_pattern_t
- struct flasher_private_t
- struct flasher_obj_t

Defines

- #define FLASHER_PRESCALE 8
- #define FLASHER_PATTERN(POLL_RATE, MOD_FREQ, MOD_DUTY, FLASHER_PERIOD,FLASHER_DUTY, FLASHES, PERIOD)
- #define FLASHER_ACTIVE_P(FLASHER) (((flasher_obj_t *)(FLASHER)) → pattern != 0)
- #define FLASHER_PATTERN_FLASHES_SET(PATTERN, FLASHES) (PATTERN) → flashes = (FLASHES)

Typedefs

- typedef flasher_obj_t * flasher_t

Functions

- int8_t flasher_pattern_set (flasher_t flasher, flasher_pattern_t *pattern)
- flasher_pattern_t * flasher_pattern_get (flasher_t flasher)
- int8_t flasher_phase_set (flasher_t flasher, uint8_t phase)
- bool flasher_update (flasher_t)
- flasher_t flasher_init (flasher_obj_t *info)

5.32.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
13 March 2005

5.32.2 Define Documentation

5.32.2.1 #define FLASHER_ACTIVE_P(FLASHER) (((flasher_obj_t *)(FLASHER)) → pattern != 0)

5.32.2.2 #define FLASHER_PATTERN(POLL_RATE, MOD_FREQ, MOD_DUTY, FLASHER_PERIOD, FLASHER_DUTY, FLASHES, PERIOD)

Value:

```
(POLL_RATE) / (double)(MOD_FREQ) + 0.5,                          \
    (POLL_RATE) * (double)(MOD_DUTY) / (MOD_FREQ) / 100.0 + 0.5,        \
    (MOD_FREQ) * (FLASHER_PERIOD) / (double)FLASHER_PRESCALE + 0.5,     \
    (MOD_FREQ) * (FLASHER_PERIOD) * (FLASHER_DUTY) / 100.0 / FLASHER_PRESCALE + 0.5, \
    (FLASHES),                                                       \
    (PERIOD) / (double)(FLASHER_PERIOD) + 0.5
```

5.32.2.3 #define FLASHER_PATTERN_FLASHES_SET(PATTERN, FLASHES) (PATTERN) → flashes = (FLASHES)

5.32.2.4 #define FLASHER_PRESCALE 8

5.32.3 Typedef Documentation

5.32.3.1 typedef flasher_obj_t* flasher_t

5.32.4 Function Documentation

5.32.4.1 flasher_t flasher_init (flasher_obj_t * *info*)

5.32.4.2 flasher_pattern_t* flasher_pattern_get (flasher_t *flasher*)

5.32.4.3 int8_t flasher_pattern_set (flasher_t *flasher*, flasher_pattern_t * *pattern*)

5.32.4.4 int8_t flasher_phase_set (flasher_t *flasher*, uint8_t *phase*)

5.32.4.5 bool flasher_update (flasher_t)

5.33 flasher_tweak.c File Reference

```
#include "flasher.h"
```

Functions

- void flasher_tweak_mod_duty (flasher_pattern_t *pattern, uint8_t mod_duty)
- void flasher_tweak_mod_freq (flasher_pattern_t *pattern, uint16_t poll_freq, uint8_t mod_freq)

5.33.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.33.2 Function Documentation

5.33.2.1 void flasher_tweak_mod_duty (flasher_pattern_t * *pattern*, uint8_t *mod_duty*)

5.33.2.2 void flasher_tweak_mod_freq (flasher_pattern_t * *pattern*, uint16_t *poll_freq*, uint8_t *mod_freq*)

5.34 flasher_tweak.h File Reference

```
#include "flasher.h"
```

Functions

- void flasher_tweak_mod_duty (flasher_pattern_t *pattern, uint8_t mod_duty)
- void flasher_tweak_mod_freq (flasher_pattern_t *pattern, uint16_t poll_freq, uint8_t mod_freq)

5.34.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.34.2 Function Documentation

5.34.2.1 void flasher_tweak_mod_duty (flasher_pattern_t * *pattern*, uint8_t *mod_duty*)

5.34.2.2 void flasher_tweak_mod_freq (flasher_pattern_t * *pattern*, uint16_t *poll_freq*, uint8_t *mod_freq*)

5.35 font.c File Reference

```
#include "font.h"
```

Functions

- bool font_display (char ch, font_t *font, void(*display)(void *data, uint8_t pixel, bool val), void *data)

5.35.1 Function Documentation

5.35.1.1 bool font_display (char *ch*, font_t * *font*, void(*)(void *data, uint8_t pixel, bool val) *display*, void * *data*)

5.36 font.h File Reference

```
#include "config.h"
```

Data Structures

- struct font_t

Functions

- bool font_display (char ch, font_t *font, void(*display)(void *data, uint8_t pixel, bool val), void *data)

5.36.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 March 2007

5.36.2 Function Documentation

5.36.2.1 bool font_display (char *ch*, font_t * *font*, void(*)(void *data, uint8_t pixel, bool val) *display*, void * *data*)

5.37 isqrt16.c File Reference

```
#include <limits.h>
#include <stdint.h>
```

Functions

- uint8_t isqrt (uint16_t val)

5.37.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.37.2 Function Documentation

5.37.2.1 uint8_t isqrt (uint16_t *val*)

5.38 isqrt32.c File Reference

```
#include <limits.h>
#include <stdint.h>
```

Functions

- uint16_t isqrt32 (uint32_t val)

5.38.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.38.2 Function Documentation

5.38.2.1 uint16_t isqrt32 (uint32_t *val*)

5.39 lcd.c File Reference

```
#include "config.h"
#include "lcd.h"
#include "delay.h"
```

Defines

- #define LCD_DEBUG HOSTED
- #define lcd_wait(dev) DELAY_US (40 + 20)
- #define lcd_data_set(dev, data)
- #define lcd_mode_control(dev) port_pins_set_low (dev → cfg → rs_port, dev → rs_mask)
- #define lcd_mode_data(dev) port_pins_set_high (dev → cfg → rs_port, dev → rs_mask)
- #define lcd_strobe(dev)
- #define lcd_write(dev, data)

Functions

- void lcd_putc (lcd_t dev, char ch)
- void lcd_puts (lcd_t dev, const char *str)
- void lcd_clear (lcd_t dev)
- void lcd_goto (lcd_t dev, uint8_t row, uint8_t col)
- lcd_t lcd_init (lcd_obj_t *info, const lcd_cfg_t *cfg)

Variables

- static const uint8_t lcd_init_data [ ]

5.39.1 Detailed Description

Author:
M.P. Hayes

Date:
16 Feb 2003

Description: Routines to interface to a Hitachi HD44780/KS0066 LCD controller in 4-bit mode with no readback.

5.39.2 Define Documentation

5.39.2.1 #define lcd_data_set(dev, data)

Value:

```
port_bus_write (dev->cfg->data_port,                    \
                dev->cfg->d_bit, dev->cfg->d_bit + 3, (data))
```

5.39.2.2 #define LCD_DEBUG HOSTED

5.39.2.3 #define lcd_mode_control(dev) port_pins_set_low (dev → cfg → rs_port, dev → rs_mask)

5.39.2.4 #define lcd_mode_data(dev) port_pins_set_high (dev → cfg → rs_port, dev → rs_mask)

5.39.2.5 #define lcd_strobe(dev)

Value:

```
do                                                              \
{                                                               \
    port_pins_set_high (dev->cfg->e_port, dev->e_mask);                 \
    DELAY_US (2);                                               \
    port_pins_set_low (dev->cfg->e_port, dev->e_mask);                  \
}                                                               \
while (0)
```

5.39.2.6 #define lcd_wait(dev) DELAY_US (40 + 20)

5.39.2.7 #define lcd_write(dev, data)

Value:

```
do                                                              \
{                                                               \
    uint8_t _tmp = (data);                                      \
                                                                \
    /* Send MS nibble.  */                                      \
    lcd_data_set (dev, _tmp >> 4);                              \
    lcd_strobe (dev);                                           \
    /* Send LS nibble.  */                                      \
    lcd_data_set (dev, _tmp);                                   \
    lcd_strobe (dev);                                           \
    lcd_wait (dev);                                             \
}                                                               \
while (0)
```

5.39.3 Function Documentation

5.39.3.1 void lcd_clear (lcd_t *dev*)

5.39.3.2 void lcd_goto (lcd_t *dev*, uint8_t *row*, uint8_t *col*)

5.39.3.3 lcd_t lcd_init (lcd_obj_t * *info*, const lcd_cfg_t * *cfg*)

5.39.3.4 void lcd_putc (lcd_t *dev*, char *ch*)

5.39.3.5 void lcd_puts (lcd_t *dev*, const char * *str*)

5.39.4 Variable Documentation

5.39.4.1 const uint8_t lcd_init_data[ ] [static]

Initial value:

```
{
    LCD_FUNCTION | BIT (3) | BIT (2),
    LCD_DISPLAY | BIT (2),
    LCD_ENTRY_MODE | BIT (1),
    LCD_CLEAR
}
```

5.40 lcd.h File Reference

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct lcd_cfg_t
- struct lcd_obj_t

Typedefs

- typedef lcd_obj_t * lcd_t

Enumerations

- enum {
  LCD_CLEAR = BIT (0), LCD_HOME = BIT (1), LCD_ENTRY_MODE = BIT (2), LCD_DISPLAY = BIT (3),
  LCD_SHIFT = BIT (4), LCD_FUNCTION = BIT (5), LCD_CG_RAM_ADDRESS = BIT (6), LCD_DD_RAM_ADDRESS = BIT (7) }

Functions

- void lcd_putc (lcd_t dev, char)
- void lcd_puts (lcd_t dev, const char *)
- void lcd_clear (lcd_t dev)
- void lcd_goto (lcd_t dev, uint8_t row, uint8_t col)
- lcd_t lcd_init (lcd_obj_t *info, const lcd_cfg_t *cfg)

5.40.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 Feb 2003

5.40.2 Typedef Documentation

5.40.2.1 typedef lcd_obj_t* lcd_t

5.40.3 Enumeration Type Documentation

5.40.3.1 anonymous enum

Enumerator:
LCD_CLEAR

LCD_HOME
LCD_ENTRY_MODE
LCD_DISPLAY
LCD_SHIFT
LCD_FUNCTION
LCD_CG_RAM_ADDRESS
LCD_DD_RAM_ADDRESS

5.40.4 Function Documentation

5.40.4.1 void lcd_clear (lcd_t *dev*)

5.40.4.2 void lcd_goto (lcd_t *dev*, uint8_t *row*, uint8_t *col*)

5.40.4.3 lcd_t lcd_init (lcd_obj_t * *info*, const lcd_cfg_t * *cfg*)

5.40.4.4 void lcd_putc (lcd_t *dev*, char)

5.40.4.5 void lcd_puts (lcd_t *dev*, const char *)

5.41 led.c File Reference

```
#include "led.h"
```

Functions

- led_t led_init (const led_cfg_t *cfg)

5.41.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.41.2 Function Documentation

5.41.2.1 led_t led_init (const led_cfg_t * *cfg*)

5.42 led.h File Reference

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct led_cfg_t

Defines

- #define LED_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

Typedefs

- typedef const led_cfg_t led_obj_t
- typedef led_obj_t * led_t

Functions

- static void led_set (led_t led, uint8_t state)
- static void led_toggle (led_t led)
- led_t led_init (const led_cfg_t *cfg)

5.42.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.42.2 Define Documentation

5.42.2.1 #define LED_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.42.3 Typedef Documentation

5.42.3.1 typedef const led_cfg_t led_obj_t

5.42.3.2 typedef led_obj_t* led_t

5.42.4 Function Documentation

5.42.4.1 led_t led_init (const led_cfg_t * *cfg*)

5.42.4.2 static void led_set (led_t *led*, uint8_t *state*) [static]

5.42.4.3 static void led_toggle (led_t *led*) [static]

5.43 led_flash.c File Reference

```
#include "config.h"
#include "delay.h"
#include "led.h"
```

Functions

- void led_flash (led_t led, uint8_t blinks, uint8_t delayms)

5.43.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
10 Jan 2006

5.43.2 Function Documentation

5.43.2.1 void led_flash (led_t *led*, uint8_t *blinks*, uint8_t *delayms*)

5.44 led_flash.h File Reference

LED flashing routine.

```
#include "config.h"
#include "led.h"
```

Functions

- void led_flash (led_t led, uint8_t blinks, uint8_t delayms)

5.44.1 Detailed Description

LED flashing routine.

Author:
M. P. Hayes, UCECE

Date:
10 Jan 2006

5.44.2 Function Documentation

5.44.2.1 void led_flash (led_t *led*, uint8_t *blinks*, uint8_t *delayms*)

5.45 lmatrix.c File Reference

```
#include "lmatrix.h"
#include "port.h"
```

Defines

- #define LMATRIX_TRANSPARENT
- #define ROWBIT(N) BIT(N)

Functions

- void lmatrix_update (lmatrix_t lmatrix)
- void lmatrix_set (lmatrix_t lmatrix, uint8_t row, uint8_t col, bool val)
- void lmatrix_write (lmatrix_t lmatrix, uint8_t *screen, uint8_t *map)
- lmatrix_t lmatrix_init (lmatrix_obj_t *lmatrix)

Variables

- static lmatrix_port_t col_ports [ ]

5.45.1 Define Documentation

5.45.1.1 #define LMATRIX_TRANSPARENT

5.45.1.2 #define ROWBIT(N) BIT(N)

5.45.2 Function Documentation

5.45.2.1 lmatrix_t lmatrix_init (lmatrix_obj_t * *lmatrix*)

5.45.2.2 void lmatrix_set (lmatrix_t *lmatrix*, uint8_t *row*, uint8_t *col*, bool *val*)

5.45.2.3 void lmatrix_update (lmatrix_t *lmatrix*)

5.45.2.4 void lmatrix_write (lmatrix_t *lmatrix*, uint8_t * *screen*, uint8_t * *map*)

5.45.3 Variable Documentation

5.45.3.1 lmatrix_port_t col_ports[ ] [static]

Initial value:

```
{
    {LMATRIX_COL1_PORT, BIT (LMATRIX_COL1_BIT)},
    {LMATRIX_COL2_PORT, BIT (LMATRIX_COL2_BIT)},
    {LMATRIX_COL3_PORT, BIT (LMATRIX_COL3_BIT)},
    {LMATRIX_COL4_PORT, BIT (LMATRIX_COL4_BIT)},
    {LMATRIX_COL5_PORT, BIT (LMATRIX_COL5_BIT)}
}
```

5.46 lmatrix.h File Reference

This drives a multiplexed LED matrix. It only supports a single instance.

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct lmatrix_port_t
- struct lmatrix_private_t

Typedefs

- typedef uint8_t lmatrix_row_state_t
- typedef lmatrix_private_t lmatrix_obj_t
- typedef lmatrix_obj_t * lmatrix_t

Enumerations

- enum { LMATRIX_PIXELS = LMATRIX_ROWS * LMATRIX_COLS }

Functions

- void lmatrix_set (lmatrix_t lmatrix, uint8_t row, uint8_t col, bool val)
- void lmatrix_write (lmatrix_t lmatrix, uint8_t *screen, uint8_t *map)
- void lmatrix_update (lmatrix_t lmatrix)
- lmatrix_t lmatrix_init (lmatrix_obj_t *info)

5.46.1 Detailed Description

This drives a multiplexed LED matrix. It only supports a single instance.

Author:
M. P. Hayes, UCECE

Date:
28 March 2007

5.46.2 Typedef Documentation

5.46.2.1 typedef lmatrix_private_t lmatrix_obj_t

5.46.2.2 typedef uint8_t lmatrix_row_state_t

5.46.2.3 typedef lmatrix_obj_t* lmatrix_t

5.46.3 Enumeration Type Documentation

5.46.3.1 anonymous enum

Enumerator:
LMATRIX_PIXELS

5.46.4 Function Documentation

5.46.4.1 lmatrix_t lmatrix_init (lmatrix_obj_t * *info*)

5.46.4.2 void lmatrix_set (lmatrix_t *lmatrix*, uint8_t *row*, uint8_t *col*, bool *val*)

5.46.4.3 void lmatrix_update (lmatrix_t *lmatrix*)

5.46.4.4 void lmatrix_write (lmatrix_t *lmatrix*, uint8_t * *screen*, uint8_t * *map*)

5.47 mbuttons.c File Reference

```
#include "mbuttons.h"
#include "port.h"
```

Functions

- void mbuttons_init (mbuttons_obj_t *mbuttons, button_obj_t *mbutton_objs, const button_cfg_t *row_config, uint8_t rows_num, const button_cfg_t *col_config, uint8_t cols_num, uint8_t poll_count)
- void mbuttons_poll (mbuttons_t mbuttons)
- bool mbuttons_any_state_p (mbuttons_t mbuttons, button_state_t state)
- void mbuttons_wakeup_init (mbuttons_t mbuttons)

5.47.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
29 June 2007

5.47.2 Function Documentation

5.47.2.1 bool mbuttons_any_state_p (mbuttons_t *mbuttons*, button_state_t *state*)

5.47.2.2 void mbuttons_init (mbuttons_obj_t * *mbuttons*, button_obj_t * *mbutton_objs*, const button_cfg_t * *row_config*, uint8_t *rows_num*, const button_cfg_t * *col_config*, uint8_t *cols_num*, uint8_t *poll_count*)

5.47.2.3 void mbuttons_poll (mbuttons_t *mbuttons*)

5.47.2.4 void mbuttons_wakeup_init (mbuttons_t *mbuttons*)

5.48 mbuttons.h File Reference

```
#include "button.h"
```

Data Structures

- struct mbuttons_private_t

Typedefs

- typedef mbuttons_private_t mbuttons_obj_t
- typedef mbuttons_obj_t * mbuttons_t

Functions

- void mbuttons_init (mbuttons_obj_t *mbuttons, button_obj_t *mbuttons_info, const button_cfg_t *row_config, uint8_t rows_num, const button_cfg_t *col_config, uint8_t cols_num, uint8_t poll_count)
- void mbuttons_poll (mbuttons_t mbuttons)
- bool mbuttons_any_state_p (mbuttons_t mbuttons, button_state_t state)
- static bool mbuttons_any_pushed_p (mbuttons_t mbuttons)
- bool mbuttons_any_down_p (mbuttons_t mbuttons)
- void mbuttons_wakeup_init (mbuttons_t mbuttons)
- static bool mbuttons_pushed_p (mbuttons_t mbuttons, uint8_t id)
- static bool mbuttons_released_p (mbuttons_t mbuttons, uint8_t id)
- static bool mbuttons_held_p (mbuttons_t mbuttons, uint8_t id, uint8_t hold_time)
- static bool mbuttons_hold_released_p (mbuttons_t mbuttons, uint8_t id, uint8_t hold_time)
- static bool mbuttons_wakeup_p (mbuttons_t mbuttons)

5.48.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
29 June 2007

5.48.2 Typedef Documentation

5.48.2.1 typedef mbuttons_private_t mbuttons_obj_t

5.48.2.2 typedef mbuttons_obj_t* mbuttons_t

5.48.3 Function Documentation

5.48.3.1 bool mbuttons_any_down_p (mbuttons_t *mbuttons*)

5.48.3.2 static bool mbuttons_any_pushed_p (mbuttons_t *mbuttons*) [static]

5.48.3.3 bool mbuttons_any_state_p (mbuttons_t *mbuttons*, button_state_t *state*)

5.48.3.4 static bool mbuttons_held_p (mbuttons_t *mbuttons*, uint8_t *id*, uint8_t *hold_time*) [static]

5.48.3.5 static bool mbuttons_hold_released_p (mbuttons_t *mbuttons*, uint8_t *id*, uint8_t *hold_time*) [static]

5.48.3.6 void mbuttons_init (mbuttons_obj_t * *mbuttons*, button_obj_t * *mbuttons_info*, const button_cfg_t * *row_config*, uint8_t *rows_num*, const button_cfg_t * *col_config*, uint8_t *cols_num*, uint8_t *poll_count*)

5.48.3.7 void mbuttons_poll (mbuttons_t *mbuttons*)

5.48.3.8 static bool mbuttons_pushed_p (mbuttons_t *mbuttons*, uint8_t *id*) [static]

5.48.3.9 static bool mbuttons_released_p (mbuttons_t *mbuttons*, uint8_t *id*) [static]

5.48.3.10 void mbuttons_wakeup_init (mbuttons_t *mbuttons*)

5.48.3.11 static bool mbuttons_wakeup_p (mbuttons_t *mbuttons*) [static]

5.49 mcleds.c File Reference

```
#include "mcleds.h"
```

Functions

- mcleds_t mcleds_init (mcleds_obj_t *mcleds, const led_cfg_t *row_config, uint8_t rows_num, const led_cfg_t *col_config, uint8_t cols_num, colourmap_t *colourmap, uint8_t colourmap_size, mcleds_state_t *state, uint8_t update_rate)
- bool mcleds_update (mcleds_t mcleds, uint8_t *screen)
- void mcleds_enable (mcleds_t mcleds, uint8_t row)
- void mcleds_off (mcleds_t mcleds)

5.49.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
3 July 2007

5.49.2 Function Documentation

5.49.2.1 void mcleds_enable (mcleds_t *mcleds*, uint8_t *row*)

5.49.2.2 mcleds_t mcleds_init (mcleds_obj_t * *mcleds*, const led_cfg_t * *row_config*, uint8_t *rows_num*, const led_cfg_t * *col_config*, uint8_t *cols_num*, colourmap_t * *colourmap*, uint8_t *colourmap_size*, mcleds_state_t * *state*, uint8_t *update_rate*)

5.49.2.3 void mcleds_off (mcleds_t *mcleds*)

5.49.2.4 bool mcleds_update (mcleds_t *mcleds*, uint8_t * *screen*)

5.50 mcleds.h File Reference

```
#include "config.h"
#include "cleds.h"
#include "colourmap.h"
#include "ticker.h"
```

Data Structures

- struct mcleds_state_t
- struct mcleds_private_t

Typedefs

- typedef mcleds_private_t mcleds_obj_t
- typedef mcleds_obj_t * mcleds_t

Functions

- mcleds_t mcleds_init (mcleds_obj_t *mcleds, const led_cfg_t *row_config, uint8_t rows_num, const led_cfg_t *col_config, uint8_t cols_num, colourmap_t *colourmap, uint8_t colourmap_size, mcleds_state_t *state, uint8_t update_rate)
- bool mcleds_update (mcleds_t mcleds, uint8_t *screen)
- void mcleds_off (mcleds_t mcleds)
- void mcleds_enable (mcleds_t mcleds, uint8_t row)
- static uint8_t mcleds_disable (mcleds_t mcleds)
- static void mcleds_colourmap_set (mcleds_t mcleds, colourmap_t *colourmap, uint8_t colourmap_size)

5.50.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
3 July 2007

5.50.2 Typedef Documentation

5.50.2.1 typedef mcleds_private_t mcleds_obj_t

5.50.2.2 typedef mcleds_obj_t* mcleds_t

5.50.3 Function Documentation

5.50.3.1 static void mcleds_colourmap_set (mcleds_t *mcleds*, colourmap_t * *colourmap*, uint8_t *colourmap_size*) [static]

5.50.3.2 static uint8_t mcleds_disable (mcleds_t *mcleds*) [static]

5.50.3.3 void mcleds_enable (mcleds_t *mcleds*, uint8_t *row*)

5.50.3.4 mcleds_t mcleds_init (mcleds_obj_t * *mcleds*, const led_cfg_t * *row_config*, uint8_t *rows_num*, const led_cfg_t * *col_config*, uint8_t *cols_num*, colourmap_t * *colourmap*, uint8_t *colourmap_size*, mcleds_state_t * *state*, uint8_t *update_rate*)

5.50.3.5 void mcleds_off (mcleds_t *mcleds*)

5.50.3.6 bool mcleds_update (mcleds_t *mcleds*, uint8_t * *screen*)

5.51 mmelody.c File Reference

Play simple melodies.

```
#include "mmelody.h"
```

Defines

- #define MMELODY_TRANSPARENT 1

Enumerations

- enum { MMELODY_SCALE_SIZE = 12 }

Functions

- static void mmelody_ticker_set (mmelody_t mmelody)
- static void mmelody_note_play (mmelody_t mmelody, mmelody_note_t note)
- static void mmelody_note_fraction_set (mmelody_t mmelody, uint8_t note_fraction)
- static mmelody_note_t mmelody_char_to_note (uint8_t ch)
- static const char * mmelody_scan (mmelody_t mmelody, const char *str)
- void mmelody_play (mmelody_t mmelody, const char *str)
- void mmelody_speed_set (mmelody_t mmelody, mmelody_speed_t speed)
- void mmelody_volume_set (mmelody_t mmelody, mmelody_volume_t volume)
- void mmelody_update (mmelody_t mmelody)
- mmelody_t mmelody_init (mmelody_obj_t *mmelody, uint16_t poll_rate, mmelody_callback_t play_callback, void *play_callback_data)

5.51.1 Detailed Description

Play simple melodies.

Author:
M. P. Hayes, UCECE

Date:
20 April 2007

5.51.2 Define Documentation

5.51.2.1 #define MMELODY_TRANSPARENT 1

5.51.3 Enumeration Type Documentation

5.51.3.1 anonymous enum

Enumerator:
MMELODY_SCALE_SIZE

5.51.4 Function Documentation

5.51.4.1 **static mmelody_note_t mmelody_char_to_note (uint8_t *ch*)** `[static]`

5.51.4.2 **mmelody_t mmelody_init (mmelody_obj_t * *mmelody*, uint16_t *poll_rate*, mmelody_callback_t *play_callback*, void * *play_callback_data*)**

5.51.4.3 **static void mmelody_note_fraction_set (mmelody_t *mmelody*, uint8_t *note_fraction*)** `[static]`

5.51.4.4 **static void mmelody_note_play (mmelody_t *mmelody*, mmelody_note_t *note*)** `[static]`

5.51.4.5 **void mmelody_play (mmelody_t *mmelody*, const char * *str*)**

5.51.4.6 **static const char* mmelody_scan (mmelody_t *mmelody*, const char * *str*)** `[static]`

5.51.4.7 **void mmelody_speed_set (mmelody_t *mmelody*, mmelody_speed_t *speed*)**

5.51.4.8 **static void mmelody_ticker_set (mmelody_t *mmelody*)** `[static]`

5.51.4.9 **void mmelody_update (mmelody_t *mmelody*)**

5.51.4.10 **void mmelody_volume_set (mmelody_t *mmelody*, mmelody_volume_t *volume*)**

5.52 mmelody.h File Reference

Play simple melodies.

```
#include "config.h"
#include "font.h"
#include "ticker.h"
```

Data Structures

- struct mmelody_private_t

Typedefs

- typedef uint8_t mmelody_speed_t
- typedef uint8_t mmelody_scale_t
- typedef uint8_t mmelody_note_t
- typedef uint8_t mmelody_volume_t
- typedef mmelody_private_t mmelody_obj_t
- typedef mmelody_obj_t * mmelody_t
- typedef void(* mmelody_callback_t)(void *data, uint8_t note, uint8_t volume)

Enumerations

- enum { MMELODY_OCTAVE_DEFAULT = 4 }
- enum { MMELODY_SPEED_DEFAULT = 200 }

Functions

- mmelody_t mmelody_init (mmelody_obj_t *dev, uint16_t poll_rate, mmelody_callback_t play_callback, void *play_callback_data)
- void mmelody_play (mmelody_t mmelody, const char *str)
- void mmelody_update (mmelody_t mmelody)
- void mmelody_speed_set (mmelody_t mmelody, mmelody_speed_t speed)
- void mmelody_volume_set (mmelody_t mmelody, mmelody_volume_t volume)

5.52.1 Detailed Description

Play simple melodies.

Author:
M. P. Hayes, UCECE

Date:
20 April 2007

5.52.2 Typedef Documentation

5.52.2.1 typedef void(∗ mmelody_callback_t)(void ∗data, uint8_t note, uint8_t volume)

5.52.2.2 typedef uint8_t mmelody_note_t

5.52.2.3 typedef mmelody_private_t mmelody_obj_t

5.52.2.4 typedef uint8_t mmelody_scale_t

5.52.2.5 typedef uint8_t mmelody_speed_t

5.52.2.6 typedef mmelody_obj_t∗ mmelody_t

5.52.2.7 typedef uint8_t mmelody_volume_t

5.52.3 Enumeration Type Documentation

5.52.3.1 anonymous enum

Enumerator:
MMELODY_OCTAVE_DEFAULT

5.52.3.2 anonymous enum

Enumerator:
MMELODY_SPEED_DEFAULT

5.52.4 Function Documentation

5.52.4.1 mmelody_t mmelody_init (mmelody_obj_t ∗ *dev*, uint16_t *poll_rate*, mmelody_callback_t *play_callback*, void ∗ *play_callback_data*)

5.52.4.2 void mmelody_play (mmelody_t *mmelody*, const char ∗ *str*)

5.52.4.3 void mmelody_speed_set (mmelody_t *mmelody*, mmelody_speed_t *speed*)

5.52.4.4 void mmelody_update (mmelody_t *mmelody*)

5.52.4.5 void mmelody_volume_set (mmelody_t *mmelody*, mmelody_volume_t *volume*)

5.53 mpwm.c File Reference

```
#include "config.h"
#include "mpwm.h"
```

Functions

- void mpwm_period_set (mpwm_t mpwm, uint16_t period)
- void mpwm_duty_set (mpwm_t mpwm, uint8_t channel, uint16_t duty)
- bool mpwm_update (mpwm_t mpwm)
- mpwm_t mpwm_init (mpwm_obj_t *dev, mpwm_channel_t *channels, uint8_t num_channels)

5.53.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.53.2 Function Documentation

5.53.2.1 void mpwm_duty_set (mpwm_t *mpwm*, uint8_t *channel*, uint16_t *duty*)

5.53.2.2 mpwm_t mpwm_init (mpwm_obj_t * *dev*, mpwm_channel_t * *channels*, uint8_t *num_channels*)

5.53.2.3 void mpwm_period_set (mpwm_t *mpwm*, uint16_t *period*)

5.53.2.4 bool mpwm_update (mpwm_t *mpwm*)

5.54 mpwm.h File Reference

```
#include "config.h"
```

Data Structures

- struct mpwm_channel_t
- struct mpwm_obj_t

Typedefs

- typedef mpwm_struct * mpwm_t

Functions

- void mpwm_period_set (mpwm_t mpwm, uint16_t period)
- void mpwm_duty_set (mpwm_t mpwm, uint8_t channel, uint16_t duty)
- bool mpwm_update (mpwm_t)
- mpwm_t mpwm_init (mpwm_obj_t *info, mpwm_channel_t *channels, uint8_t num_channels)

5.54.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.54.2 Typedef Documentation

5.54.2.1 typedef struct mpwm_struct* mpwm_t

5.54.3 Function Documentation

5.54.3.1 void mpwm_duty_set (mpwm_t *mpwm*, uint8_t *channel*, uint16_t *duty*)

5.54.3.2 mpwm_t mpwm_init (mpwm_obj_t * *info*, mpwm_channel_t * *channels*, uint8_t *num_channels*)

5.54.3.3 void mpwm_period_set (mpwm_t *mpwm*, uint16_t *period*)

5.54.3.4 bool mpwm_update (mpwm_t)

5.55 mtext.c File Reference

Moving text.

```
#include "mtext.h"
#include "font.h"
#include <limits.h>
```

Functions

- static void mtext_pixel_set (void *data, uint8_t pixel, bool val)
- static void mtext_display (mtext_t mtext, char ch)
- static const char * mtext_scan (mtext_t mtext, const char *str)
- mtext_t mtext_init (mtext_obj_t *mtext, uint16_t poll_rate, font_t *font, uint8_t *image, uint8_t font_rows, uint8_t font_cols, uint8_t *screen, uint8_t rows, uint8_t cols)
- void mtext_scroller_dir_set (mtext_t mtext, scroller_dir_t dir)
- scroller_dir_t mtext_scroller_dir_get (mtext_t mtext)
- int8_t mtext_update (mtext_t mtext)
- void mtext_speed_set (mtext_t mtext, uint8_t speed)

5.55.1 Detailed Description

Moving text.

Author:
M. P. Hayes, UCECE

Date:
8 April 2007

5.55.2 Function Documentation

5.55.2.1 static void mtext_display (mtext_t *mtext*, char *ch*) [static]

5.55.2.2 mtext_t mtext_init (mtext_obj_t * *mtext*, uint16_t *poll_rate*, font_t * *font*, uint8_t * *image*, uint8_t *font_rows*, uint8_t *font_cols*, uint8_t * *screen*, uint8_t *rows*, uint8_t *cols*)

5.55.2.3 static void mtext_pixel_set (void * *data*, uint8_t *pixel*, bool *val*) [static]

5.55.2.4 static const char* mtext_scan (mtext_t *mtext*, const char * *str*) [static]

5.55.2.5 scroller_dir_t mtext_scroller_dir_get (mtext_t *mtext*)

5.55.2.6 void mtext_scroller_dir_set (mtext_t *mtext*, scroller_dir_t *dir*)

5.55.2.7 void mtext_speed_set (mtext_t *mtext*, uint8_t *speed*)

5.55.2.8 int8_t mtext_update (mtext_t *mtext*)

5.56 mtext.h File Reference

Moving text.

```
#include "config.h"
#include "font.h"
#include "scroller.h"
#include "ticker.h"
```

Data Structures

- struct mtext_obj_t

Typedefs

- typedef mtext_obj_t * mtext_t

Enumerations

- enum { MTEXT_SPEED_SCALER = 16 }
- enum mtext_mode_t { MTEXT_MODE_REPLACE, MTEXT_MODE_SCROLL }

Functions

- mtext_t mtext_init (mtext_obj_t *dev, uint16_t poll_rate, font_t *font, uint8_t *image, uint8_t font_rows, uint8_t font_cols, uint8_t *screen, uint8_t rows, uint8_t cols)
- int8_t mtext_update (mtext_t mtext)
- void mtext_scroller_dir_set (mtext_t mtext, scroller_dir_t dir)
- scroller_dir_t mtext_scroller_dir_get (mtext_t mtext)
- void mtext_speed_set (mtext_t mtext, uint8_t speed)
- static void mtext_set (mtext_t mtext, const char *str)
- static const char * mtext_get (mtext_t mtext)
- static void mtext_mode_set (mtext_t mtext, mtext_mode_t mode)
- static mtext_mode_t mtext_mode_get (mtext_t mtext)

5.56.1 Detailed Description

Moving text.

Author:
M. P. Hayes, UCECE

Date:
8 April 2007

5.56.2 Typedef Documentation

5.56.2.1 typedef mtext_obj_t* mtext_t

5.56.3 Enumeration Type Documentation

5.56.3.1 anonymous enum

Enumerator:

MTEXT_SPEED_SCALER

5.56.3.2 enum mtext_mode_t

Enumerator:

MTEXT_MODE_REPLACE

MTEXT_MODE_SCROLL

5.56.4 Function Documentation

5.56.4.1 static const char* mtext_get (mtext_t *mtext*) [static]

5.56.4.2 mtext_t mtext_init (mtext_obj_t * *dev*, uint16_t *poll_rate*, font_t * *font*, uint8_t * *image*, uint8_t *font_rows*, uint8_t *font_cols*, uint8_t * *screen*, uint8_t *rows*, uint8_t *cols*)

5.56.4.3 static mtext_mode_t mtext_mode_get (mtext_t *mtext*) [static]

5.56.4.4 static void mtext_mode_set (mtext_t *mtext*, mtext_mode_t *mode*) [static]

5.56.4.5 scroller_dir_t mtext_scroller_dir_get (mtext_t *mtext*)

5.56.4.6 void mtext_scroller_dir_set (mtext_t *mtext*, scroller_dir_t *dir*)

5.56.4.7 static void mtext_set (mtext_t *mtext*, const char * *str*) [static]

5.56.4.8 void mtext_speed_set (mtext_t *mtext*, uint8_t *speed*)

5.56.4.9 int8_t mtext_update (mtext_t *mtext*)

5.57 mu1wire.c File Reference

```
#include <stdio.h>
#include "config.h"
#include "delay.h"
#include "u1wire.h"
#include "irq.h"
#include "port.h"
```

Defines

- #define U1WIRE_DEBUG 1
- #define U1WIRE_RELEASE() port_pin_config_pullup (U1WIRE_PORT, U1WIRE_BIT)
- #define U1WIRE_DRIVE()
- #define U1WIRE_TEST() port_pin_read (U1WIRE_PORT, U1WIRE_BIT)

Enumerations

- enum { U1WIRE_READ_ROM = 0x33, U1WIRE_SKIP_ROM = 0xcc, U1WIRE_MATCH_ROM = 0x55, U1WIRE_RECALL = 0xb8 }
- enum { U1WIRE_DELAY_OFFSET = 6, U1WIRE_ADDR_BYTES = 6 }

Functions

- int8_t u1wire_reset (void)
- void u1wire_bit_write (uint8_t value)
- void u1wire_byte_write (uint8_t value)
- uint8_t u1wire_bit_read (void)
- uint8_t u1wire_byte_read (void)
- int8_t u1wire_rom_code_read (u1wire_t dev)
- int8_t u1wire_command (u1wire_t dev, uint8_t command)
- int8_t u1wire_broadcast (uint8_t command)
- int8_t u1wire_read (u1wire_t dev, void *data, uint8_t bytes)
- int8_t u1wire_write (u1wire_t dev, void *data, uint8_t bytes)
- bool u1wire_ready_p (void)
- int8_t u1wire_init (u1wire_obj_t *devices, uint8_t devices_max)

5.57.1 Define Documentation

5.57.1.1 #define U1WIRE_DEBUG 1

5.57.1.2 #define U1WIRE_DRIVE()

Value:

```
do {port_pin_config_output (U1WIRE_PORT, U1WIRE_BIT); \
        port_pin_set_low (U1WIRE_PORT, U1WIRE_BIT);} while (0)
```

5.57.1.3 #define U1WIRE_RELEASE() port_pin_config_pullup (U1WIRE_PORT, U1WIRE_BIT)

5.57.1.4 #define U1WIRE_TEST() port_pin_read (U1WIRE_PORT, U1WIRE_BIT)

5.57.2 Enumeration Type Documentation

5.57.2.1 anonymous enum

Enumerator:

U1WIRE_READ_ROM

U1WIRE_SKIP_ROM

U1WIRE_MATCH_ROM

U1WIRE_RECALL

5.57.2.2 anonymous enum

Enumerator:

U1WIRE_DELAY_OFFSET

U1WIRE_ADDR_BYTES

5.57.3 Function Documentation

5.57.3.1 uint8_t u1wire_bit_read (void)

5.57.3.2 void u1wire_bit_write (uint8_t *value*)

5.57.3.3 int8_t u1wire_broadcast (uint8_t *command*)

5.57.3.4 uint8_t u1wire_byte_read (void)

5.57.3.5 void u1wire_byte_write (uint8_t *value*)

5.57.3.6 int8_t u1wire_command (u1wire_t *dev*, uint8_t *command*)

5.57.3.7 int8_t u1wire_init (u1wire_obj_t * *devices*, uint8_t *devices_max*)

5.57.3.8 int8_t u1wire_read (u1wire_t *dev*, void * *data*, uint8_t *bytes*)

5.57.3.9 bool u1wire_ready_p (void)

5.57.3.10 int8_t u1wire_reset (void)

5.57.3.11 int8_t u1wire_rom_code_read (u1wire_t *dev*)

5.57.3.12 int8_t u1wire_write (u1wire_t *dev*, void * *data*, uint8_t *bytes*)

5.58 mu1wire.h File Reference

```
#include "config.h"
```

Data Structures

- union mu1wire_rom_code_t
- struct mu1wire_obj_t

Typedefs

- typedef mu1wire_obj_t * mu1wire_t

Enumerations

- enum {

  MU1WIRE_ERR_BUS_LOW = 1, MU1WIRE_ERR_BUS_STUCK = 2, MU1WIRE_ERR_-PRESENCE_SHORT = 3, MU1WIRE_ERR_PRESENCE_LONG = 4,

  MU1WIRE_ERR_MULTIPLE_DEVICES = 5, MU1WIRE_ERR_BUS_HIGH = 6 }

Functions

- bool mu1wire_ready_p (void)
- int8_t mu1wire_reset (void)
- void mu1wire_bit_write (uint8_t value)
- void mu1wire_byte_write (uint8_t value)
- uint8_t mu1wire_bit_read (void)
- uint8_t mu1wire_byte_read (void)
- int8_t mu1wire_command (mu1wire_t dev, uint8_t command)
- int8_t mu1wire_broadcast (uint8_t command)
- int8_t mu1wire_read (mu1wire_t dev, void *data, uint8_t size)
- int8_t mu1wire_write (mu1wire_t dev, void *data, uint8_t size)
- int8_t mu1wire_init (mu1wire_obj_t *devices, uint8_t devices_max)
- void mu1wire_debug (mu1wire_t dev)

5.58.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.58.2 Typedef Documentation

5.58.2.1 typedef mu1wire_obj_t* mu1wire_t

5.58.3 Enumeration Type Documentation

5.58.3.1 anonymous enum

Enumerator:

MU1WIRE_ERR_BUS_LOW

MU1WIRE_ERR_BUS_STUCK

MU1WIRE_ERR_PRESENCE_SHORT

MU1WIRE_ERR_PRESENCE_LONG

MU1WIRE_ERR_MULTIPLE_DEVICES

MU1WIRE_ERR_BUS_HIGH

5.58.4 Function Documentation

5.58.4.1 uint8_t mu1wire_bit_read (void)

5.58.4.2 void mu1wire_bit_write (uint8_t *value*)

5.58.4.3 int8_t mu1wire_broadcast (uint8_t *command*)

5.58.4.4 uint8_t mu1wire_byte_read (void)

5.58.4.5 void mu1wire_byte_write (uint8_t *value*)

5.58.4.6 int8_t mu1wire_command (mu1wire_t *dev*, uint8_t *command*)

5.58.4.7 void mu1wire_debug (mu1wire_t *dev*)

5.58.4.8 int8_t mu1wire_init (mu1wire_obj_t * *devices*, uint8_t *devices_max*)

5.58.4.9 int8_t mu1wire_read (mu1wire_t *dev*, void * *data*, uint8_t *size*)

5.58.4.10 bool mu1wire_ready_p (void)

5.58.4.11 int8_t mu1wire_reset (void)

5.58.4.12 int8_t mu1wire_write (mu1wire_t *dev*, void * *data*, uint8_t *size*)

5.59 muxleds.c File Reference

```
#include "muxleds.h"
```

Defines

- #define MUXLEDS_TRANSPARENT

Functions

- void muxleds_set (muxleds_t muxleds, uint8_t bit, uint8_t val)
- void muxleds_toggle (muxleds_t muxleds, uint8_t bit)
- void muxleds_update (muxleds_t muxleds)
- muxleds_t muxleds_init (muxleds_obj_t *muxleds, const muxleds_cfg_t *row_cfg, uint8_t rows_num, const muxleds_cfg_t *col_cfg, uint8_t cols_num, uint8_t row_on, uint8_t col_on)

5.59.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.59.2 Define Documentation

5.59.2.1 #define MUXLEDS_TRANSPARENT

5.59.3 Function Documentation

5.59.3.1 muxleds_t muxleds_init (muxleds_obj_t * *muxleds*, const muxleds_cfg_t * *row_cfg*, uint8_t *rows_num*, const muxleds_cfg_t * *col_cfg*, uint8_t *cols_num*, uint8_t *row_on*, uint8_t *col_on*)

5.59.3.2 void muxleds_set (muxleds_t *muxleds*, uint8_t *bit*, uint8_t *val*)

5.59.3.3 void muxleds_toggle (muxleds_t *muxleds*, uint8_t *bit*)

5.59.3.4 void muxleds_update (muxleds_t *muxleds*)

5.60 muxleds.h File Reference

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct muxleds_cfg_t
- struct muxleds_row_t
- struct muxleds_col_t
- struct muxleds_obj_t

Defines

- #define MUXLEDS_ROWS_NUM 8
- #define MUXLEDS_COLS_NUM 8
- #define MUXLED_ROW_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}
- #define MUXLED_COL_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

Typedefs

- typedef muxleds_struct * muxleds_t

Functions

- void muxleds_set (muxleds_t muxleds, uint8_t bit, uint8_t val)
- void muxleds_toggle (muxleds_t muxleds, uint8_t bit)
- void muxleds_update (muxleds_t muxleds)
- muxleds_t muxleds_init (muxleds_obj_t *dev, const muxleds_cfg_t *row_cfg, uint8_t row_size, const muxleds_cfg_t *col_cfg, uint8_t col_size)

5.60.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.60.2 Define Documentation

5.60.2.1 #define MUXLED_COL_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.60.2.2 #define MUXLED_ROW_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.60.2.3 #define MUXLEDS_COLS_NUM 8

5.60.2.4 #define MUXLEDS_ROWS_NUM 8

5.60.3 Typedef Documentation

5.60.3.1 typedef struct muxleds_struct* muxleds_t

5.60.4 Function Documentation

5.60.4.1 muxleds_t muxleds_init (muxleds_obj_t * *dev*, const muxleds_cfg_t * *row_cfg*, uint8_t *row_size*, const muxleds_cfg_t * *col_cfg*, uint8_t *col_size*)

5.60.4.2 void muxleds_set (muxleds_t *muxleds*, uint8_t *bit*, uint8_t *val*)

5.60.4.3 void muxleds_toggle (muxleds_t *muxleds*, uint8_t *bit*)

5.60.4.4 void muxleds_update (muxleds_t *muxleds*)

5.61 nmea.c File Reference

```
#include <stdio.h>
#include "config.h"
```

Functions

- uint8_t nmea_checksum (const char *string)
- void nmea_puts (const char *string)

5.61.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.61.2 Function Documentation

5.61.2.1 uint8_t nmea_checksum (const char * *string*)

5.61.2.2 void nmea_puts (const char * *string*)

5.62 nmea.h File Reference

```
#include "config.h"
```

Defines

- #define NMEA_BUFFER_SIZE 80

Functions

- uint8_t nmea_checksum (const char *string)
- void nmea_puts (const char *string)

5.62.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.62.2 Define Documentation

5.62.2.1 #define NMEA_BUFFER_SIZE 80

5.62.3 Function Documentation

5.62.3.1 uint8_t nmea_checksum (const char * *string*)

5.62.3.2 void nmea_puts (const char * *string*)

5.63 nrf2401.c File Reference

```
#include "nrf2401.h"
#include "delay.h"
```

Defines

- #define CHAR_BIT 8
- #define NRF_DATA_READY_P(DEV) (port_pins_read ((DEV) → pins.dr_port, (DEV) → pins.dr_-bitmask))
- #define NRF_CONFIGURATION_REGISTER_SIZE 15
- #define NRF_CE_HIGH_SET(DEV) (port_pins_set_high ((DEV) → pins.ce_port, (DEV) → pins.ce_bitmask))
- #define NRF_CE_LOW_SET(DEV) (port_pins_set_low ((DEV) → pins.ce_port, (DEV) → pins.ce_bitmask))
- #define NRF_CS_HIGH_SET(DEV) (port_pins_set_high ((DEV) → pins.cs_port, (DEV) → pins.cs_bitmask))
- #define NRF_CS_LOW_SET(DEV) (port_pins_set_low ((DEV) → pins.cs_port, (DEV) → pins.cs_bitmask))
- #define NRF_FULL_CONFIGURE(DEV) (nrf_configure ((DEV), NRF_CONFIGURATION_-REGISTER_SIZE))

Enumerations

- enum { NRF_CHANNEL_NUMBER_MAX = 83 }
- enum { NRF_CONFIG_DELAY_US = 20 }
- enum { NRF_LINE_TIME_ENABLE_US = 10 }
- enum { NRF_T_SB_ACTIVE = 202 }

Functions

- nrf_t nrf_init (nrf_obj_t ∗dev, const nrf_cfg_t ∗cfg)
- bool nrf_setup (nrf_t rf, uint8_t payload_size)
- void nrf_configure (nrf_t rf, uint8_t size)
- uint8_t nrf_receive (nrf_t rf, uint8_t ∗data, uint8_t ms_to_wait)
- uint8_t nrf_transmit (nrf_t rf, rf_address_t ∗dst_address, uint8_t ∗data, uint8_t size)
- void nrf_rf_dir_set (nrf_t rf, uint8_t mode)
- void nrf_channel_set (nrf_t rf, uint8_t channel)
- void nrf_rf_power_set (nrf_t rf, uint8_t rf_power)
- void nrf_xtal_freq_set (nrf_t rf, uint8_t freq)
- void nrf_data_rate_set (nrf_t rf, uint8_t data_rate)
- void nrf_comms_mode_set (nrf_t rf, uint8_t mode)
- void nrf_single_or_dual_channel_set (nrf_t rf, uint8_t mode)
- void nrf_crc_status_set (nrf_t rf, uint8_t status)
- void nrf_crc_length_set (nrf_t rf, uint8_t length)
- void nrf_address_length_set (nrf_t rf, uint8_t length)
- void nrf_address1_set (nrf_t rf, rf_address_t ∗address)
- void nrf_address2_set (nrf_t rf, rf_address_t ∗address)
- void nrf_payload_length1_set (nrf_t rf, uint8_t length)

- void nrf_payload_length2_set (nrf_t rf, uint8_t length)
- void nrf_rf_standby (nrf_t rf)
- void nrf_rf_enable (nrf_t rf)
- uint8_t nrf_payload_length1_get (nrf_t rf)
- bool nrf_data_ready_p (nrf_t rf)

5.63.1 Detailed Description

Author:
Tony Culliford

Date:
7 December 2004

Description: Interface routines for the Nordic nRF2401 tranceiver chip

5.63.2 Define Documentation

5.63.2.1 #define CHAR_BIT 8

5.63.2.2 #define NRF_CE_HIGH_SET(DEV) (port_pins_set_high ((DEV) → pins.ce_port, (DEV) → pins.ce_bitmask))

5.63.2.3 #define NRF_CE_LOW_SET(DEV) (port_pins_set_low ((DEV) → pins.ce_port, (DEV) → pins.ce_bitmask))

5.63.2.4 #define NRF_CONFIGURATION_REGISTER_SIZE 15

5.63.2.5 #define NRF_CS_HIGH_SET(DEV) (port_pins_set_high ((DEV) → pins.cs_port, (DEV) → pins.cs_bitmask))

5.63.2.6 #define NRF_CS_LOW_SET(DEV) (port_pins_set_low ((DEV) → pins.cs_port, (DEV) → pins.cs_bitmask))

5.63.2.7 #define NRF_DATA_READY_P(DEV) (port_pins_read ((DEV) → pins.dr_port, (DEV) → pins.dr_bitmask))

5.63.2.8 #define NRF_FULL_CONFIGURE(DEV) (nrf_configure ((DEV), NRF_CONFIGURATION_REGISTER_SIZE))

5.63.3 Enumeration Type Documentation

5.63.3.1 anonymous enum

Enumerator:
NRF_CHANNEL_NUMBER_MAX

5.63.3.2 anonymous enum

Enumerator:
NRF_CONFIG_DELAY_US

5.63.3.3 anonymous enum

Enumerator:
NRF_LINE_TIME_ENABLE_US

5.63.3.4 anonymous enum

Enumerator:
NRF_T_SB_ACTIVE

5.63.4 Function Documentation

5.63.4.1 void nrf_address1_set (nrf_t *rf*, rf_address_t * *address*)

5.63.4.2 void nrf_address2_set (nrf_t *rf*, rf_address_t * *address*)

5.63.4.3 void nrf_address_length_set (nrf_t *rf*, uint8_t *length*)

5.63.4.4 void nrf_channel_set (nrf_t *rf*, uint8_t *channel*)

5.63.4.5 void nrf_comms_mode_set (nrf_t *rf*, uint8_t *mode*)

5.63.4.6 void nrf_configure (nrf_t *rf*, uint8_t *size*)

5.63.4.7 void nrf_crc_length_set (nrf_t *rf*, uint8_t *length*)

5.63.4.8 void nrf_crc_status_set (nrf_t *rf*, uint8_t *status*)

5.63.4.9 void nrf_data_rate_set (nrf_t *rf*, uint8_t *data_rate*)

5.63.4.10 bool nrf_data_ready_p (nrf_t *rf*)

5.63.4.11 nrf_t nrf_init (nrf_obj_t * *dev*, const nrf_cfg_t * *cfg*)

5.63.4.12 uint8_t nrf_payload_length1_get (nrf_t *rf*)

5.63.4.13 void nrf_payload_length1_set (nrf_t *rf*, uint8_t *length*)

5.63.4.14 void nrf_payload_length2_set (nrf_t *rf*, uint8_t *length*)

5.63.4.15 uint8_t nrf_receive (nrf_t *rf*, uint8_t * *data*, uint8_t *ms_to_wait*)

5.63.4.16 void nrf_rf_dir_set (nrf_t *rf*, uint8_t *mode*)

5.63.4.17 void nrf_rf_enable (nrf_t *rf*)

5.63.4.18 void nrf_rf_power_set (nrf_t *rf*, uint8_t *rf_power*)

5.63.4.19 void nrf_rf_standby (nrf_t *rf*)

5.63.4.20 bool nrf_setup (nrf_t *rf*, uint8_t *payload_size*)

5.63.4.21 void nrf_single_or_dual_channel_set (nrf_t *rf*, uint8_t *mode*)

5.63.4.22 uint8_t nrf_transmit (nrf_t *rf*, rf_address_t * *dst_address*, uint8_t * *data*, uint8_t *size*)

5.63.4.23 void nrf_xtal_freq_set (nrf_t *rf*, uint8_t *freq*)

5.64 nrf2401.h File Reference

```
#include "time.h"
#include "spi.h"
#include "port.h"
#include "config.h"
```

Data Structures

- struct rf_address_t
- struct nrf_config_bits_t
- union nrf_config_t
- struct nrf_cfg_t
- struct nrf_pins_t
- struct nrf_obj_t

Defines

- #define NRF_PAYLOAD_SIZE 25
- #define NRF_DEFAULT_ADDRESS {0xFD, 0xCA, 0x7E, 0xA9, 0x52}

Typedefs

- typedef nrf_obj_t * nrf_t

Enumerations

- enum { NRF_TX_MODE = 0, NRF_RX_MODE = 1 }
- enum { NRF_DIRECT = 0, NRF_SHOCKBURST = 1 }
- enum { NRF_CRC_8 = 0, NRF_CRC_16 = 1 }
- enum { NRF_CRC_DISABLED = 0, NRF_CRC_ENABLED = 1 }
- enum { NRF_RF_POWER_20, NRF_RF_POWER_10, NRF_RF_POWER_5, NRF_RF_POWER_-0 }
- enum { NRF_SINGLE_CHANNEL = 0, NRF_DUAL_CHANNEL = 1 }
- enum { NRF_DATA_250K = 0, NRF_DATA_1M = 1 }
- enum {

  NRF_XTAL_FREQ_4M, RF_XTAL_FREQ_8M, NRF_XTAL_FREQ_12M, NRF_XTAL_FREQ_-16M,

  NRF_XTAL_FREQ_20M }
- enum { NRF_TIME_OUT_ACK_MS = 10 }
- enum { NRF_ACK_DELAY_US = 600 }

Functions

- nrf_t nrf_init (nrf_obj_t *dev, const nrf_cfg_t *cfg)
- bool nrf_setup (nrf_t rf, uint8_t payload_size)
- void nrf_configure (nrf_t rf, uint8_t size)
- uint8_t nrf_transmit (nrf_t rf, rf_address_t *address, uint8_t *data, uint8_t size)
- uint8_t nrf_receive (nrf_t rf, uint8_t *data, uint8_t ms_to_wait)
- void nrf_rf_dir_set (nrf_t rf, uint8_t mode)
- void nrf_channel_set (nrf_t rf, uint8_t channel)
- void nrf_rf_power_set (nrf_t rf, uint8_t rf_power)
- void nrf_xtal_freq_set (nrf_t rf, uint8_t freq)
- void nrf_data_rate_set (nrf_t rf, uint8_t data_rate)
- void nrf_comms_mode_set (nrf_t rf, uint8_t mode)
- void nrf_single_or_dual_channel_set (nrf_t rf, uint8_t mode)
- void nrf_crc_status_set (nrf_t rf, uint8_t status)
- void nrf_crc_length_set (nrf_t rf, uint8_t length)
- void nrf_address_length_set (nrf_t rf, uint8_t length)
- void nrf_address1_set (nrf_t rf, rf_address_t *address)
- void nrf_address2_set (nrf_t rf, rf_address_t *address)
- void nrf_payload_length1_set (nrf_t rf, uint8_t length)
- void nrf_payload_length2_set (nrf_t rf, uint8_t length)
- void nrf_rf_standby (nrf_t rf)
- void nrf_rf_enable (nrf_t rf)
- uint8_t nrf_payload_length1_get (nrf_t rf)
- bool nrf_data_ready_p (nrf_t rf)

5.64.1 Detailed Description

Author:
Tony Culliford

Date:
7 December 2004

Description: Interface routines for the Nordic nRF2401 tranceiver chip

5.64.2 Define Documentation

5.64.2.1 #define NRF_DEFAULT_ADDRESS {0xFD, 0xCA, 0x7E, 0xA9, 0x52}

5.64.2.2 #define NRF_PAYLOAD_SIZE 25

5.64.3 Typedef Documentation

5.64.3.1 typedef nrf_obj_t* nrf_t

5.64.4 Enumeration Type Documentation

5.64.4.1 anonymous enum

Enumerator:
NRF_TX_MODE

NRF_RX_MODE

5.64.4.2 anonymous enum

Enumerator:
NRF_DIRECT
NRF_SHOCKBURST

5.64.4.3 anonymous enum

Enumerator:
NRF_CRC_8
NRF_CRC_16

5.64.4.4 anonymous enum

Enumerator:
NRF_CRC_DISABLED
NRF_CRC_ENABLED

5.64.4.5 anonymous enum

Enumerator:
NRF_RF_POWER_20
NRF_RF_POWER_10
NRF_RF_POWER_5
NRF_RF_POWER_0

5.64.4.6 anonymous enum

Enumerator:
NRF_SINGLE_CHANNEL
NRF_DUAL_CHANNEL

5.64.4.7 anonymous enum

Enumerator:
NRF_DATA_250K
NRF_DATA_1M

5.64.4.8 anonymous enum

Enumerator:
NRF_XTAL_FREQ_4M
RF_XTAL_FREQ_8M
NRF_XTAL_FREQ_12M
NRF_XTAL_FREQ_16M
NRF_XTAL_FREQ_20M

5.64.4.9 anonymous enum

Enumerator:
NRF_TIME_OUT_ACK_MS

5.64.4.10 anonymous enum

Enumerator:
NRF_ACK_DELAY_US

5.64.5 Function Documentation

5.64.5.1 void nrf_address1_set (nrf_t *rf*, rf_address_t * *address*)

5.64.5.2 void nrf_address2_set (nrf_t *rf*, rf_address_t * *address*)

5.64.5.3 void nrf_address_length_set (nrf_t *rf*, uint8_t *length*)

5.64.5.4 void nrf_channel_set (nrf_t *rf*, uint8_t *channel*)

5.64.5.5 void nrf_comms_mode_set (nrf_t *rf*, uint8_t *mode*)

5.64.5.6 void nrf_configure (nrf_t *rf*, uint8_t *size*)

5.64.5.7 void nrf_crc_length_set (nrf_t *rf*, uint8_t *length*)

5.64.5.8 void nrf_crc_status_set (nrf_t *rf*, uint8_t *status*)

5.64.5.9 void nrf_data_rate_set (nrf_t *rf*, uint8_t *data_rate*)

5.64.5.10 bool nrf_data_ready_p (nrf_t *rf*)

5.64.5.11 nrf_t nrf_init (nrf_obj_t * *dev*, const nrf_cfg_t * *cfg*)

5.64.5.12 uint8_t nrf_payload_length1_get (nrf_t *rf*)

5.64.5.13 void nrf_payload_length1_set (nrf_t *rf*, uint8_t *length*)

5.64.5.14 void nrf_payload_length2_set (nrf_t *rf*, uint8_t *length*)

5.64.5.15 uint8_t nrf_receive (nrf_t *rf*, uint8_t * *data*, uint8_t *ms_to_wait*)

5.64.5.16 void nrf_rf_dir_set (nrf_t *rf*, uint8_t *mode*)

5.64.5.17 void nrf_rf_enable (nrf_t *rf*)

5.64.5.18 void nrf_rf_power_set (nrf_t *rf*, uint8_t *rf_power*)

5.64.5.19 void nrf_rf_standby (nrf_t *rf*)

5.64.5.20 bool nrf_setup (nrf_t *rf*, uint8_t *payload_size*)

5.64.5.21 void nrf_single_or_dual_channel_set (nrf_t *rf*, uint8_t *mode*)

5.64.5.22 uint8_t nrf_transmit (nrf_t *rf*, rf_address_t * *address*, uint8_t * *data*, uint8_t *size*)

5.64.5.23 void nrf_xtal_freq_set (nrf_t *rf*, uint8_t *freq*)

5.65 nrf_config.h File Reference

```
#include "nrf2401.h"
```

Defines

- #define RF_RX_MODE_SET(DEV) nrf_rf_dir_set (DEV, NRF_RX_MODE)
- #define RF_TX_MODE_SET(DEV) nrf_rf_dir_set (DEV, NRF_TX_MODE)
- #define RF_DEVICE_ENABLE(DEV) nrf_rf_enable (DEV)
- #define RF_DEVICE_DISABLE(DEV) nrf_rf_standby (DEV)
- #define RF_DEVICE_CHANNEL_SET(DEV, CHANNEL) nrf_channel_set (DEV, (CHANNEL))
- #define RF_DEVICE_ADDRESS_SET(DEV, ADDRESS) nrf_address1_set (DEV, (ADDRESS))
- #define RF_TRANSMIT(DEV, ADDRESS, DATA, SIZE) nrf_transmit (DEV, ADDRESS, DATA, SIZE)
- #define RF_RECEIVE(DEV, DATA, MS_TO_WAIT) nrf_receive (DEV, DATA, MS_TO_WAIT)
- #define RF_INIT(DEV, CFG) nrf_init (DEV, CFG)
- #define RF_SETUP(DEV, SIZE) nrf_setup (DEV, SIZE)
- #define RF_DATA_READY_P(DEV) nrf_data_ready_p (DEV)

Typedefs

- typedef nrf_t rf_t
- typedef nrf_obj_t rf_obj_t
- typedef nrf_cfg_t rf_cfg_t

5.65.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.65.2 Define Documentation

5.65.2.1 #define RF_DATA_READY_P(DEV) nrf_data_ready_p (DEV)

5.65.2.2 #define RF_DEVICE_ADDRESS_SET(DEV, ADDRESS) nrf_address1_set (DEV, (ADDRESS))

5.65.2.3 #define RF_DEVICE_CHANNEL_SET(DEV, CHANNEL) nrf_channel_set (DEV, (CHANNEL))

5.65.2.4 #define RF_DEVICE_DISABLE(DEV) nrf_rf_standby (DEV)

5.65.2.5 #define RF_DEVICE_ENABLE(DEV) nrf_rf_enable (DEV)

5.65.2.6 #define RF_INIT(DEV, CFG) nrf_init (DEV, CFG)

5.65.2.7 #define RF_RECEIVE(DEV, DATA, MS_TO_WAIT) nrf_receive (DEV, DATA, MS_TO_WAIT)

5.65.2.8 #define RF_RX_MODE_SET(DEV) nrf_rf_dir_set (DEV, NRF_RX_MODE)

5.65.2.9 #define RF_SETUP(DEV, SIZE) nrf_setup (DEV, SIZE)

5.65.2.10 #define RF_TRANSMIT(DEV, ADDRESS, DATA, SIZE) nrf_transmit (DEV, ADDRESS, DATA, SIZE)

5.65.2.11 #define RF_TX_MODE_SET(DEV) nrf_rf_dir_set (DEV, NRF_TX_MODE)

5.65.3 Typedef Documentation

5.65.3.1 typedef nrf_cfg_t rf_cfg_t

5.65.3.2 typedef nrf_obj_t rf_obj_t

5.65.3.3 typedef nrf_t rf_t

5.66 pga.c File Reference

```
#include "pga.h"
```

Defines

- #define PGA_TRANSPARENT
- #define PGA_INSN_REGISTER_WRITE(REG) ((PGA_INSN_WRITE << 5) | (REG))
- #define PGA_INSN_GAIN_REGISTER_WRITE PGA_INSN_REGISTER_WRITE (PGA_GAIN_REGISTER)
- #define PGA_INSN_CHANNEL_REGISTER_WRITE PGA_INSN_REGISTER_WRITE (PGA_CHANNEL_REGISTER)
- #define PGA_PUTC(val) spi_putc(val)
- #define PGA_GETC() spi_getc()
- #define PGA_SELECT(PGA) port_pins_set_low ((PGA) → cs_port, (PGA) → cs_bitmask)
- #define PGA_DESELECT(PGA) port_pins_set_high ((PGA) → cs_port, (PGA) → cs_bitmask)

Enumerations

- enum { PGA_GAIN_REGISTER = 0, PGA_CHANNEL_REGISTER = 1 }
- enum { PGA_INSN_NOP = 0, PGA_INSN_SHUTDOWN = 1, PGA_INSN_WRITE = 2 }

Functions

- static void pga_send_command (pga_t pga, uint8_t command_byte_1, uint8_t command_byte_2)
- void pga_gain_set (pga_t pga, pga_gain_t gain)
- void pga_channel_set (pga_t pga, pga_channel_t channel)
- void pga_shutdown (pga_t pga)
- void pga_startup (pga_t pga)
- void pga_chip_select (pga_t pga)
- void pga_chip_deselect (pga_t pga)
- pga_t pga_init (pga_obj_t *pga, const pga_cfg_t *cfg)

5.66.1 Detailed Description

Author:
Micahel Hayes / Tony Culliford

Date:
8 February 2005

Description: Interface routines for Microchip PGAs (MCP6S21)

5.66.2 Define Documentation

5.66.2.1 #define PGA_DESELECT(PGA) port_pins_set_high ((PGA) → cs_port, (PGA) → cs_bitmask)

5.66.2.2 #define PGA_GETC() spi_getc()

5.66.2.3 #define PGA_INSN_CHANNEL_REGISTER_WRITE PGA_INSN_REGISTER_WRITE (PGA_CHANNEL_REGISTER)

5.66.2.4 #define PGA_INSN_GAIN_REGISTER_WRITE PGA_INSN_REGISTER_WRITE (PGA_GAIN_REGISTER)

5.66.2.5 #define PGA_INSN_REGISTER_WRITE(REG) ((PGA_INSN_WRITE << 5) | (REG))

5.66.2.6 #define PGA_PUTC(val) spi_putc(val)

5.66.2.7 #define PGA_SELECT(PGA) port_pins_set_low ((PGA) → cs_port, (PGA) → cs_bitmask)

5.66.2.8 #define PGA_TRANSPARENT

5.66.3 Enumeration Type Documentation

5.66.3.1 anonymous enum

Enumerator:

PGA_GAIN_REGISTER

PGA_CHANNEL_REGISTER

5.66.3.2 anonymous enum

Enumerator:

PGA_INSN_NOP

PGA_INSN_SHUTDOWN

PGA_INSN_WRITE

5.66.4 Function Documentation

5.66.4.1 void pga_channel_set (pga_t *pga*, pga_channel_t *channel*)

5.66.4.2 void pga_chip_deselect (pga_t *pga*)

5.66.4.3 void pga_chip_select (pga_t *pga*)

5.66.4.4 void pga_gain_set (pga_t *pga*, pga_gain_t *gain*)

5.66.4.5 pga_t pga_init (pga_obj_t * *pga*, const pga_cfg_t * *cfg*)

5.66.4.6 static void pga_send_command (pga_t *pga*, uint8_t *command_byte_1*, uint8_t *command_byte_2*) [static]

5.66.4.7 void pga_shutdown (pga_t *pga*)

5.66.4.8 void pga_startup (pga_t *pga*)

5.67 pga.h File Reference

```
#include "spi.h"
#include "config.h"
#include "port.h"
```

Data Structures

- struct pga_cfg_t
- struct pga_private_t

Defines

- #define PGA_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

Typedefs

- typedef pga_private_t pga_obj_t
- typedef pga_obj_t * pga_t

Enumerations

- enum pga_channel_t {

  PGA_CHANNEL_0 = 0, PGA_CHANNEL_1 = 1, PGA_CHANNEL_2 = 2, PGA_CHANNEL_3 = 3,

  PGA_CHANNEL_4 = 4, PGA_CHANNEL_5 = 5, PGA_CHANNEL_6 = 6, PGA_CHANNEL_7 = 7 }
- enum pga_gain_t {

  PGA_GAIN_1 = 0, PGA_GAIN_2 = 1, PGA_GAIN_4 = 2, PGA_GAIN_5 = 3,

  PGA_GAIN_8 = 4, PGA_GAIN_10 = 5, PGA_GAIN_16 = 6, PGA_GAIN_32 = 7 }

Functions

- void pga_gain_set (pga_t pga, pga_gain_t gain)
- void pga_channel_set (pga_t pga, pga_channel_t channel)
- void pga_startup (pga_t pga)
- void pga_shutdown (pga_t pga)
- void pga_chip_select (pga_t pga)
- void pga_chip_deselect (pga_t pga)
- pga_t pga_init (pga_obj_t *dev, const pga_cfg_t *cfg)

5.67.1 Detailed Description

Author:
Tony Culliford

Date:
8 February 2005

Description: Interface routines for Microchip PGAs (MCP6S2X)

5.67.2 Define Documentation

5.67.2.1 #define PGA_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.67.3 Typedef Documentation

5.67.3.1 typedef pga_private_t pga_obj_t

5.67.3.2 typedef pga_obj_t* pga_t

5.67.4 Enumeration Type Documentation

5.67.4.1 enum pga_channel_t

Enumerator:
PGA_CHANNEL_0
PGA_CHANNEL_1
PGA_CHANNEL_2
PGA_CHANNEL_3
PGA_CHANNEL_4
PGA_CHANNEL_5
PGA_CHANNEL_6
PGA_CHANNEL_7

5.67.4.2 enum pga_gain_t

Enumerator:
PGA_GAIN_1
PGA_GAIN_2
PGA_GAIN_4
PGA_GAIN_5
PGA_GAIN_8
PGA_GAIN_10
PGA_GAIN_16
PGA_GAIN_32

5.67.5 Function Documentation

5.67.5.1 void pga_channel_set (pga_t *pga*, pga_channel_t *channel*)

5.67.5.2 void pga_chip_deselect (pga_t *pga*)

5.67.5.3 void pga_chip_select (pga_t *pga*)

5.67.5.4 void pga_gain_set (pga_t *pga*, pga_gain_t *gain*)

5.67.5.5 pga_t pga_init (pga_obj_t * *dev*, const pga_cfg_t * *cfg*)

5.67.5.6 void pga_shutdown (pga_t *pga*)

5.67.5.7 void pga_startup (pga_t *pga*)

5.68 piezo.c File Reference

```
#include "piezo.h"
```

Functions

- piezo_t piezo_init (const piezo_cfg_t *cfg)

5.68.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
12 March 2003

5.68.2 Function Documentation

5.68.2.1 piezo_t piezo_init (const piezo_cfg_t * *cfg*)

5.69 piezo.h File Reference

```
#include "config.h"
#include "delay.h"
#include "port.h"
```

Data Structures

- struct piezo_cfg_t

Defines

- #define PIEZO_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

Typedefs

- typedef const piezo_cfg_t piezo_obj_t
- typedef piezo_obj_t ∗ piezo_t

Functions

- piezo_t piezo_init (const piezo_cfg_t ∗cfg)
- static void piezo_set (piezo_t piezo, uint8_t val)

5.69.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
12 March 2003

5.69.2 Define Documentation

5.69.2.1 #define PIEZO_CFG(PORT, PORTBIT) {(PORT), BIT (PORTBIT)}

5.69.3 Typedef Documentation

5.69.3.1 typedef const piezo_cfg_t piezo_obj_t

5.69.3.2 typedef piezo_obj_t∗ piezo_t

5.69.4 Function Documentation

5.69.4.1 piezo_t piezo_init (const piezo_cfg_t ∗ *cfg*)

5.69.4.2 static void piezo_set (piezo_t *piezo*, uint8_t *val*) [static]

5.70 piezo_beep.c File Reference

Piezo beeping routines. Note these block.

```
#include "piezo.h"
```

Defines

- #define PIEZO_SHORT_BEEP_TIME 30
- #define PIEZO_LONG_BEEP_TIME 200
- #define PIEZO_BEEP_PERIOD 200

Functions

- void piezo_beep (piezo_t piezo, uint16_t duration)
- void piezo_beep_short (piezo_t piezo)
- void piezo_beep_long (piezo_t piezo)

5.70.1 Detailed Description

Piezo beeping routines. Note these block.

Author:
M. P. Hayes, UCECE

Date:
12 April 2007

5.70.2 Define Documentation

5.70.2.1 #define PIEZO_BEEP_PERIOD 200

5.70.2.2 #define PIEZO_LONG_BEEP_TIME 200

5.70.2.3 #define PIEZO_SHORT_BEEP_TIME 30

5.70.3 Function Documentation

5.70.3.1 void piezo_beep (piezo_t *piezo*, uint16_t *duration*)

5.70.3.2 void piezo_beep_long (piezo_t *piezo*)

5.70.3.3 void piezo_beep_short (piezo_t *piezo*)

5.71 piezo_beep.h File Reference

```
#include "piezo.h"
```

Functions

- void piezo_beep (piezo_t dev, uint16_t duration)
- void piezo_beep_short (piezo_t dev)
- void piezo_beep_long (piezo_t dev)

5.71.1 Function Documentation

5.71.1.1 void piezo_beep (piezo_t *dev*, uint16_t *duration*)

5.71.1.2 void piezo_beep_long (piezo_t *dev*)

5.71.1.3 void piezo_beep_short (piezo_t *dev*)

5.72 rf.c File Reference

```
#include <limits.h>
#include <string.h>
#include <stdio.h>
#include "rf.h"
#include "time.h"
#include "delay.h"
#include "port.h"
```

Defines

- #define RF_DEBUG 0
- #define HIGH_BYTE(x) ((x) >> 8)
- #define LOW_BYTE(x) ((x) & 0xFF)

Functions

- static void rf_address_make (rf_address_t *address, rf_id_t id)
- void rf_node_make (rf_node_t *node, rf_id_t id, uint8_t channel)
- void rf_write_setup (rf_t rf, rf_node_t *dst_node)
- void rf_standby (rf_t rf)
- rf_size_t rf_transmit (rf_t rf, rf_node_t *dst_node, uint8_t *data, rf_size_t size)
- void rf_read_enable (rf_t rf)
- void rf_read_setup (rf_t rf, rf_node_t *node)
- rf_size_t rf_receive (rf_t rf, uint8_t *data, uint8_t ms_to_wait)
- rf_size_t rf_write (rf_t rf, rf_node_t *dst_node, uint8_t *data, rf_size_t size)
- rf_size_t rf_read (rf_t rf, uint8_t *data, rf_size_t size, uint8_t ms_to_wait)
- rf_t rf_setup (rf_obj_t *dev, rf_cfg_t *cfg)
- uint8_t rf_acknowledge_wait (rf_t rf, rf_node_t *node, uint8_t command, time_t *timestamp)
- void rf_acknowledge (rf_t rf, rf_node_t *dst_node, uint8_t command)
- rf_t rf_init (rf_obj_t *dev, rf_cfg_t *cfg)
- uint16_t rf_write_data (rf_t rf, rf_node_t *dst_node, uint8_t *data, uint16_t size)
- bool rf_read_ready_p (rf_t rf)
- rf_size_t rf_command (rf_t rf, rf_node_t *dst_node, rf_cmd_t command, void *data, rf_size_t data_size)
- void rf_command_no_ack (rf_t rf, rf_node_t *dst_node, rf_cmd_t command, void *data, rf_size_t data_size)

5.72.1 Detailed Description

Author:
Tony Culliford

Date:
8 December 2004

Description: Higher-level routines for interfacing to a wireless link.

5.72.2 Define Documentation

5.72.2.1 #define HIGH_BYTE(x) ((x) >> 8)

5.72.2.2 #define LOW_BYTE(x) ((x) & 0xFF)

5.72.2.3 #define RF_DEBUG 0

5.72.3 Function Documentation

5.72.3.1 void rf_acknowledge (rf_t *rf*, rf_node_t * *dst_node*, uint8_t *command*)

5.72.3.2 uint8_t rf_acknowledge_wait (rf_t *rf*, rf_node_t * *node*, uint8_t *command*, time_t * *timestamp*)

5.72.3.3 static void rf_address_make (rf_address_t * *address*, rf_id_t *id*) [static]

5.72.3.4 rf_size_t rf_command (rf_t *rf*, rf_node_t * *dst_node*, rf_cmd_t *command*, void * *data*, rf_size_t *data_size*)

5.72.3.5 void rf_command_no_ack (rf_t *rf*, rf_node_t * *dst_node*, rf_cmd_t *command*, void * *data*, rf_size_t *data_size*)

5.72.3.6 rf_t rf_init (rf_obj_t * *dev*, rf_cfg_t * *cfg*)

5.72.3.7 void rf_node_make (rf_node_t * *node*, rf_id_t *id*, uint8_t *channel*)

5.72.3.8 rf_size_t rf_read (rf_t *rf*, uint8_t * *data*, rf_size_t *size*, uint8_t *ms_to_wait*)

5.72.3.9 void rf_read_enable (rf_t *rf*)

5.72.3.10 bool rf_read_ready_p (rf_t *rf*)

5.72.3.11 void rf_read_setup (rf_t *rf*, rf_node_t * *node*)

5.72.3.12 rf_size_t rf_receive (rf_t *rf*, uint8_t * *data*, uint8_t *ms_to_wait*)

5.72.3.13 rf_t rf_setup (rf_obj_t * *dev*, rf_cfg_t * *cfg*)

5.72.3.14 void rf_standby (rf_t *rf*)

5.72.3.15 rf_size_t rf_transmit (rf_t *rf*, rf_node_t * *dst_node*, uint8_t * *data*, rf_size_t *size*)

5.72.3.16 rf_size_t rf_write (rf_t *rf*, rf_node_t * *dst_node*, uint8_t * *data*, rf_size_t *size*)

5.72.3.17 uint16_t rf_write_data (rf_t *rf*, rf_node_t * *dst_node*, uint8_t * *data*, uint16_t *size*)

5.72.3.18 void rf_write_setup (rf_t *rf*, rf_node_t * *dst_node*)

5.73 rf.h File Reference

```
#include "nrf_config.h"
```

Data Structures

- struct rf_node_t
- struct rf_probe_t

Defines

- #define RF_PROBE_NOT_FOUND 0xFD
- #define RF_PAYLOAD_SIZE NRF_PAYLOAD_SIZE
- #define RF_RETRIES_MAX 30

Typedefs

- typedef uint8_t rf_channel_t
- typedef uint8_t rf_size_t

Enumerations

- enum rf_id_t { RF_BROADCAST_SLAVE_ID = 0xFE, RF_MASTER_ID = 0xFF }
- enum { RF_BROADCAST_CHANNEL = 0 }
- enum { RF_READ_WAIT_MS = 2 }
- enum { RF_WRITE_WAIT_MS = 4 }
- enum { RF_READY_WAIT_US = 100 }
- enum { RF_READY_WAIT2_US = 300 }
- enum { RF_DEVICE_ID_SIZE = 8 }
- enum rf_cmd_t {

  RF_CMD_ACK = 0x61, RF_CMD_DATA_START, RF_CMD_CHANNEL_SET, RF_CMD_TIMESTAMP_REQ,

  RF_CMD_BROADCAST_MODE, RF_CMD_BROADCAST_EXIT, RF_CMD_DEVICE_ID_GET, RF_CMD_DEVICE_ID_RESPONSE,

  RF_CMD_ENUMERATE_DEVICE, RF_CMD_RESEND_PACKET, RF_CMD_ALL_PACKETS_RECEIVED, RF_CMD_SIZE }

Functions

- rf_address_t * rf_address_calc (rf_id_t id)
- rf_t rf_init (rf_obj_t *rf, rf_cfg_t *cfg)
- uint8_t rf_acknowledge_wait (rf_t rf, rf_node_t *node, rf_cmd_t command, time_t *timestamp)
- void rf_acknowledge (rf_t rf, rf_node_t *node, rf_cmd_t command)
- uint8_t rf_probes_enumerate (rf_t rf, rf_probe_t *probes, uint8_t probes_max)
- rf_id_t rf_enumeration_response (rf_t rf, uint8_t *device_id, uint8_t *version)
- void rf_read_enable (rf_t rf)
- void rf_write_setup (rf_t rf, rf_node_t *node)

- rf_size_t rf_write (rf_t rf, rf_node_t *node, uint8_t *data, rf_size_t size)
- uint16_t rf_write_data (rf_t rf, rf_node_t *node, uint8_t *data, uint16_t size)
- void rf_read_setup (rf_t rf, rf_node_t *node)
- bool rf_read_ready_p (rf_t rf)
- uint16_t rf_read_data (rf_t rf, rf_node_t *node, uint8_t *data, uint16_t size)
- rf_size_t rf_read (rf_t rf, uint8_t *data, rf_size_t size, uint8_t ms_to_wait)
- void rf_node_make (rf_node_t *node, rf_id_t id, rf_channel_t channel)
- rf_size_t rf_transmit (rf_t rf, rf_node_t *dst_node, uint8_t *data, rf_size_t size)
- rf_size_t rf_command (rf_t rf, rf_node_t *dst_node, rf_cmd_t command, void *data, rf_size_t data_size)
- void rf_command_no_ack (rf_t rf, rf_node_t *dst_node, rf_cmd_t command, void *data, rf_size_t data_size)
- void rf_standby (rf_t rf)

5.73.1 Detailed Description

Author:
Tony Culliford / Michael Hayes

Date:
8 December 2004

Description: Higher-level routines for interfacing to a wireless link.

5.73.2 Define Documentation

5.73.2.1 #define RF_PAYLOAD_SIZE NRF_PAYLOAD_SIZE

5.73.2.2 #define RF_PROBE_NOT_FOUND 0xFD

5.73.2.3 #define RF_RETRIES_MAX 30

5.73.3 Typedef Documentation

5.73.3.1 typedef uint8_t rf_channel_t

5.73.3.2 typedef uint8_t rf_size_t

5.73.4 Enumeration Type Documentation

5.73.4.1 anonymous enum

Enumerator:
RF_BROADCAST_CHANNEL

5.73.4.2 anonymous enum

Enumerator:
RF_READ_WAIT_MS

5.73.4.3 anonymous enum

Enumerator:
RF_WRITE_WAIT_MS

5.73.4.4 anonymous enum

Enumerator:
RF_READY_WAIT_US

5.73.4.5 anonymous enum

Enumerator:
RF_READY_WAIT2_US

5.73.4.6 anonymous enum

Enumerator:
RF_DEVICE_ID_SIZE

5.73.4.7 enum rf_cmd_t

Enumerator:
RF_CMD_ACK

RF_CMD_DATA_START

RF_CMD_CHANNEL_SET

RF_CMD_TIMESTAMP_REQ

RF_CMD_BROADCAST_MODE

RF_CMD_BROADCAST_EXIT

RF_CMD_DEVICE_ID_GET

RF_CMD_DEVICE_ID_RESPONSE

RF_CMD_ENUMERATE_DEVICE

RF_CMD_RESEND_PACKET

RF_CMD_ALL_PACKETS_RECEIVED

RF_CMD_SIZE

5.73.4.8 enum rf_id_t

Enumerator:
RF_BROADCAST_SLAVE_ID

RF_MASTER_ID

5.73.5 Function Documentation

5.73.5.1 void rf_acknowledge (rf_t *rf*, rf_node_t * *node*, rf_cmd_t *command*)

5.73.5.2 uint8_t rf_acknowledge_wait (rf_t *rf*, rf_node_t * *node*, rf_cmd_t *command*, time_t * *timestamp*)

5.73.5.3 rf_address_t* rf_address_calc (rf_id_t *id*)

5.73.5.4 rf_size_t rf_command (rf_t *rf*, rf_node_t * *dst_node*, rf_cmd_t *command*, void * *data*, rf_size_t *data_size*)

5.73.5.5 void rf_command_no_ack (rf_t *rf*, rf_node_t * *dst_node*, rf_cmd_t *command*, void * *data*, rf_size_t *data_size*)

5.73.5.6 rf_id_t rf_enumeration_response (rf_t *rf*, uint8_t * *device_id*, uint8_t * *version*)

5.73.5.7 rf_t rf_init (rf_obj_t * *rf*, rf_cfg_t * *cfg*)

5.73.5.8 void rf_node_make (rf_node_t * *node*, rf_id_t *id*, rf_channel_t *channel*)

5.73.5.9 uint8_t rf_probes_enumerate (rf_t *rf*, rf_probe_t * *probes*, uint8_t *probes_max*)

5.73.5.10 rf_size_t rf_read (rf_t *rf*, uint8_t * *data*, rf_size_t *size*, uint8_t *ms_to_wait*)

5.73.5.11 uint16_t rf_read_data (rf_t *rf*, rf_node_t * *node*, uint8_t * *data*, uint16_t *size*)

5.73.5.12 void rf_read_enable (rf_t *rf*)

5.73.5.13 bool rf_read_ready_p (rf_t *rf*)

5.73.5.14 void rf_read_setup (rf_t *rf*, rf_node_t * *node*)

5.73.5.15 void rf_standby (rf_t *rf*)

5.73.5.16 rf_size_t rf_transmit (rf_t *rf*, rf_node_t * *dst_node*, uint8_t * *data*, rf_size_t *size*)

5.73.5.17 rf_size_t rf_write (rf_t *rf*, rf_node_t * *node*, uint8_t * *data*, rf_size_t *size*)

5.73.5.18 uint16_t rf_write_data (rf_t *rf*, rf_node_t * *node*, uint8_t * *data*, uint16_t *size*)

5.73.5.19 void rf_write_setup (rf_t *rf*, rf_node_t * *node*)

5.74 rf_master.c File Reference

```
#include <string.h>
#include "rf.h"
#include "delay.h"
#include "port.h"
#include "target.h"
```

Defines

- #define RF_PROBE_RESPONSE_WAIT_MS 100
- #define RF_PROBE_RESPONSE_WAIT_NEXT_MS 5
- #define RF_ENUMERATE_REPEAT_NUM 3
- #define RF_UNACK_CMD_REPEAT_NUM 3
- #define RF_UNACK_CMD_REPEAT_DELAY_MS 2

Functions

- static uint8_t rf_probes_search (uint8_t *device_id, rf_probe_t *probes, uint8_t probes_num)
- uint8_t rf_probes_enumerate (rf_t rf, rf_probe_t *probes, uint8_t probes_max)
- uint16_t rf_read_data (rf_t rf, rf_node_t *node, uint8_t *data, uint16_t size)

5.74.1 Detailed Description

Author:
MPH

Description: Stripped from original rf file wireless.c to handle rf functions only used by the master node.

5.74.2 Define Documentation

5.74.2.1 #define RF_ENUMERATE_REPEAT_NUM 3

5.74.2.2 #define RF_PROBE_RESPONSE_WAIT_MS 100

5.74.2.3 #define RF_PROBE_RESPONSE_WAIT_NEXT_MS 5

5.74.2.4 #define RF_UNACK_CMD_REPEAT_DELAY_MS 2

5.74.2.5 #define RF_UNACK_CMD_REPEAT_NUM 3

5.74.3 Function Documentation

5.74.3.1 uint8_t rf_probes_enumerate (rf_t *rf*, rf_probe_t * *probes*, uint8_t *probes_max*)

5.74.3.2 static uint8_t rf_probes_search (uint8_t * *device_id*, rf_probe_t * *probes*, uint8_t *probes_num*) [static]

5.74.3.3 uint16_t rf_read_data (rf_t *rf*, rf_node_t * *node*, uint8_t * *data*, uint16_t *size*)

5.75 rf_slave.c File Reference

```
#include <string.h>
#include "rf.h"
#include "delay.h"
```

Enumerations

- enum { RF_PROBE_ENUMERATE_WAIT_MS = 5 }

Functions

- rf_id_t rf_enumeration_response (rf_t rf, uint8_t *device_id, uint8_t *version)

5.75.1 Detailed Description

Author:
Michael Hayes

Date:
25 March 2005

Description: Slave only wireless routines.

5.75.2 Enumeration Type Documentation

5.75.2.1 anonymous enum

Enumerator:
RF_PROBE_ENUMERATE_WAIT_MS

5.75.3 Function Documentation

5.75.3.1 rf_id_t rf_enumeration_response (rf_t *rf*, uint8_t * *device_id*, uint8_t * *version*)

5.76 ring.c File Reference

Ring buffer implementation.

```
#include <string.h>
#include "ring.h"
```

Defines

- #define RING_SIZE(RING) ((RING) → end - (RING) → top)
- #define RING_READ_NUM(RING, TMP)
- #define RING_WRITE_NUM(RING, TMP) (RING_SIZE (RING) - RING_READ_NUM (RING, TMP) - 1)
- #define RING_FULL_P(RING, TMP) (RING_WRITE_NUM (RING, TMP) == 0)
- #define RING_EMPTY_P(RING) ((RING) → in == (RING) → out)

Functions

- ring_size_t ring_empty_p (ring_t *ring)
- ring_size_t ring_read_num (ring_t *ring)
- ring_size_t ring_write_num (ring_t *ring)
- ring_size_t ring_init (ring_t *ring, void *buffer, ring_size_t size)
- ring_size_t ring_read (ring_t *ring, void *buffer, ring_size_t num)
- ring_size_t ring_write (ring_t *ring, const void *buffer, ring_size_t num)

5.76.1 Detailed Description

Ring buffer implementation.

Author:
M. P. Hayes, UCECE

Date:
15 May 2000

5.76.2 Define Documentation

5.76.2.1 #define RING_EMPTY_P(RING) ((RING) → in == (RING) → out)

5.76.2.2 #define RING_FULL_P(RING, TMP) (RING_WRITE_NUM (RING, TMP) == 0)

5.76.2.3 #define RING_READ_NUM(RING, TMP)

Value:

```
(((TMP) = ((RING)->in - (RING)->out)) < 0 \
      ? (TMP) + RING_SIZE (RING) : (TMP))
```

5.76.2.4 #define RING_SIZE(RING) ((RING) → end - (RING) → top)

5.76.2.5 #define RING_WRITE_NUM(RING, TMP) (RING_SIZE (RING) - RING_READ_NUM (RING, TMP) - 1)

5.76.3 Function Documentation

5.76.3.1 ring_size_t ring_empty_p (ring_t * *ring*)

5.76.3.2 ring_size_t ring_init (ring_t * *ring*, void * *buffer*, ring_size_t *size*)

5.76.3.3 ring_size_t ring_read (ring_t * *ring*, void * *buffer*, ring_size_t *num*)

5.76.3.4 ring_size_t ring_read_num (ring_t * *ring*)

5.76.3.5 ring_size_t ring_write (ring_t * *ring*, const void * *buffer*, ring_size_t *num*)

5.76.3.6 ring_size_t ring_write_num (ring_t * *ring*)

5.77 ring.h File Reference

Ring buffer interface.

`#include "config.h"`

Data Structures

- struct ring_struct

Typedefs

- typedef uint8_t ring_size_t
- typedef ring_struct ring_t

Functions

- bool ring_empty_p (ring_t *ring)
- ring_size_t ring_read (ring_t *ring, void *buffer, ring_size_t num)
- ring_size_t ring_write (ring_t *ring, const void *buffer, ring_size_t num)
- ring_size_t ring_init (ring_t *ring, void *buffer, ring_size_t size)
- ring_size_t ring_read_num (ring_t *ring)
- ring_size_t ring_write_num (ring_t *ring)

5.77.1 Detailed Description

Ring buffer interface.

Author:
M. P. Hayes, UCECE

Date:
15 May 2000

5.77.2 Typedef Documentation

5.77.2.1 typedef uint8_t ring_size_t

5.77.2.2 typedef struct ring_struct ring_t

5.77.3 Function Documentation

5.77.3.1 bool ring_empty_p (ring_t * *ring*)

5.77.3.2 ring_size_t ring_init (ring_t * *ring*, void * *buffer*, ring_size_t *size*)

5.77.3.3 ring_size_t ring_read (ring_t * *ring*, void * *buffer*, ring_size_t *num*)

5.77.3.4 ring_size_t ring_read_num (ring_t * *ring*)

5.77.3.5 ring_size_t ring_write (ring_t * *ring*, const void * *buffer*, ring_size_t *num*)

5.77.3.6 ring_size_t ring_write_num (ring_t * *ring*)

5.78 s_eeprom.c File Reference

```
#include "s_eeprom.h"
#include "spi.h"
#include "delay.h"
#include <stdio.h>
```

Defines

- #define SPI_EEPROM_TRANSPARENT
- #define SPI_EEPROM_PUTC(val) spi_putc(val)
- #define SPI_EEPROM_GETC() spi_getc()
- #define SPI_EEPROM_OP(op) SPI_EEPROM_PUTC (op)
- #define SPI_EEPROM_ENABLE(DEV) port_pins_set_low ((DEV) → cs_port, (DEV) → cs_-bitmask)
- #define SPI_EEPROM_DISABLE(DEV) port_pins_set_high ((DEV) → cs_port, (DEV) → cs_-bitmask)

Enumerations

- enum {
  SPI_EEPROM_OP_WRSR = 1, SPI_EEPROM_OP_WRITE = 2, SPI_EEPROM_OP_READ = 3, SPI_EEPROM_OP_WRDI = 4,
  SPI_EEPROM_OP_RDSR = 5, SPI_EEPROM_OP_WREN = 6 }
- enum {
  SPI_EEPROM_WPEN = BIT (7), SPI_EEPROM_BP1 = BIT (3), SPI_EEPROM_BP0 = BIT (2), SPI_EEPROM_WEL = BIT (1),
  SPI_EEPROM_WIP = BIT (0) }

Functions

- spi_eeprom_size_t spi_eeprom_read (spi_eeprom_t eeprom, spi_eeprom_addr_t addr, void ∗buffer, spi_eeprom_size_t size)
- spi_eeprom_size_t spi_eeprom_write (spi_eeprom_t eeprom, spi_eeprom_addr_t addr, const void ∗buffer, spi_eeprom_size_t size)
- uint8_t spi_eeprom_write_setup (spi_eeprom_t eeprom, spi_eeprom_addr_t addr)
- uint8_t spi_eeprom_disable (spi_eeprom_t eeprom)
- static void spi_eeprom_status_write (spi_eeprom_t eeprom, uint8_t data)
- spi_eeprom_t spi_eeprom_init (spi_eeprom_obj_t ∗eeprom, const spi_eeprom_cfg_t ∗cfg)

5.78.1 Detailed Description

Author:
Michael Hayes

Date:
06/08/03

Description: Routines to read/write SPI EEPROM.

5.78.2 Define Documentation

5.78.2.1 #define SPI_EEPROM_DISABLE(DEV) port_pins_set_high ((DEV) → cs_port, (DEV) → cs_bitmask)

5.78.2.2 #define SPI_EEPROM_ENABLE(DEV) port_pins_set_low ((DEV) → cs_port, (DEV) → cs_bitmask)

5.78.2.3 #define SPI_EEPROM_GETC() spi_getc()

5.78.2.4 #define SPI_EEPROM_OP(op) SPI_EEPROM_PUTC (op)

5.78.2.5 #define SPI_EEPROM_PUTC(val) spi_putc(val)

5.78.2.6 #define SPI_EEPROM_TRANSPARENT

5.78.3 Enumeration Type Documentation

5.78.3.1 anonymous enum

Enumerator:

SPI_EEPROM_OP_WRSR

SPI_EEPROM_OP_WRITE

SPI_EEPROM_OP_READ

SPI_EEPROM_OP_WRDI

SPI_EEPROM_OP_RDSR

SPI_EEPROM_OP_WREN

5.78.3.2 anonymous enum

Enumerator:

SPI_EEPROM_WPEN

SPI_EEPROM_BP1

SPI_EEPROM_BP0

SPI_EEPROM_WEL

SPI_EEPROM_WIP

5.78.4 Function Documentation

5.78.4.1 uint8_t spi_eeprom_disable (spi_eeprom_t *eeprom*)

5.78.4.2 spi_eeprom_t spi_eeprom_init (spi_eeprom_obj_t * *eeprom*, const spi_eeprom_cfg_t * *cfg*)

5.78.4.3 spi_eeprom_size_t spi_eeprom_read (spi_eeprom_t *eeprom*, spi_eeprom_addr_t *addr*, void * *buffer*, spi_eeprom_size_t *size*)

5.78.4.4 static void spi_eeprom_status_write (spi_eeprom_t *eeprom*, uint8_t *data*) [static]

5.78.4.5 spi_eeprom_size_t spi_eeprom_write (spi_eeprom_t *eeprom*, spi_eeprom_addr_t *addr*, const void * *buffer*, spi_eeprom_size_t *size*)

5.78.4.6 uint8_t spi_eeprom_write_setup (spi_eeprom_t *eeprom*, spi_eeprom_addr_t *addr*)

5.79 s_eeprom.h File Reference

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct spi_eeprom_cfg_t
- struct spi_eeprom_private_t

Typedefs

- typedef spi_eeprom_private_t spi_eeprom_obj_t
- typedef spi_eeprom_obj_t * spi_eeprom_t
- typedef uint16_t spi_eeprom_addr_t
- typedef uint16_t spi_eeprom_size_t

Functions

- spi_eeprom_t spi_eeprom_init (spi_eeprom_obj_t *dev, const spi_eeprom_cfg_t *cfg)
- spi_eeprom_size_t spi_eeprom_read (spi_eeprom_t dev, spi_eeprom_addr_t addr, void *buffer, spi_eeprom_size_t size)
- spi_eeprom_size_t spi_eeprom_write (spi_eeprom_t dev, spi_eeprom_addr_t addr, const void *buffer, spi_eeprom_size_t size)
- uint8_t spi_eeprom_write_setup (spi_eeprom_t dev, spi_eeprom_addr_t addr)
- uint8_t spi_eeprom_disable (spi_eeprom_t dev)

5.79.1 Typedef Documentation

5.79.1.1 typedef uint16_t spi_eeprom_addr_t

5.79.1.2 typedef spi_eeprom_private_t spi_eeprom_obj_t

5.79.1.3 typedef uint16_t spi_eeprom_size_t

5.79.1.4 typedef spi_eeprom_obj_t* spi_eeprom_t

5.79.2 Function Documentation

5.79.2.1 uint8_t spi_eeprom_disable (spi_eeprom_t *dev*)

5.79.2.2 spi_eeprom_t spi_eeprom_init (spi_eeprom_obj_t * *dev*, const spi_eeprom_cfg_t * *cfg*)

5.79.2.3 spi_eeprom_size_t spi_eeprom_read (spi_eeprom_t *dev*, spi_eeprom_addr_t *addr*, void * *buffer*, spi_eeprom_size_t *size*)

5.79.2.4 spi_eeprom_size_t spi_eeprom_write (spi_eeprom_t *dev*, spi_eeprom_addr_t *addr*, const void * *buffer*, spi_eeprom_size_t *size*)

5.79.2.5 uint8_t spi_eeprom_write_setup (spi_eeprom_t *dev*, spi_eeprom_addr_t *addr*)

5.80 scroller.c File Reference

```
#include "scroller.h"
#include <string.h>
```

Functions

- scroller_t scroller_init (scroller_t scroller, int rows, int cols, scroller_dir_t dir)
- int8_t scroller_update (scroller_t scroller, uint8_t *image, uint8_t *screen)
- void scroller_start (scroller_t scroller, uint8_t *image, uint8_t *screen)
- uint8_t scroller_speed_scale_get (scroller_t scroller)

5.80.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.80.2 Function Documentation

5.80.2.1 scroller_t scroller_init (scroller_t *scroller*, int *rows*, int *cols*, scroller_dir_t *dir*)

5.80.2.2 uint8_t scroller_speed_scale_get (scroller_t *scroller*)

5.80.2.3 void scroller_start (scroller_t *scroller*, uint8_t * *image*, uint8_t * *screen*)

5.80.2.4 int8_t scroller_update (scroller_t *scroller*, uint8_t * *image*, uint8_t * *screen*)

5.81 scroller.h File Reference

Image scroller.

```
#include "config.h"
```

Data Structures

- struct scroller_obj_t

Typedefs

- typedef scroller_obj_t * scroller_t

Enumerations

- enum scroller_dir_t {
  SCROLLER_OFF, SCROLLER_LEFT, SCROLLER_RIGHT, SCROLLER_DOWN,
  SCROLLER_UP }

Functions

- scroller_t scroller_init (scroller_t scroller, int rows, int cols, scroller_dir_t dir)
- int8_t scroller_update (scroller_t scroller, uint8_t *image, uint8_t *screen)
- void scroller_start (scroller_t scroller, uint8_t *image, uint8_t *screen)
- uint8_t scroller_speed_scale_get (scroller_t scroller)
- static void scroller_stop (scroller_t scroller)
- static void scroller_dir_set (scroller_t scroller, scroller_dir_t dir)
- static scroller_dir_t scroller_dir_get (scroller_t scroller)

5.81.1 Detailed Description

Image scroller.

Author:
M. P. Hayes, UCECE

Date:
7 April 2007

5.81.2 Typedef Documentation

5.81.2.1 typedef scroller_obj_t* scroller_t

5.81.3 Enumeration Type Documentation

5.81.3.1 enum scroller_dir_t

Enumerator:
SCROLLER_OFF

SCROLLER_LEFT
SCROLLER_RIGHT
SCROLLER_DOWN
SCROLLER_UP

5.81.4 Function Documentation

5.81.4.1 static scroller_dir_t scroller_dir_get (scroller_t *scroller*) [static]

5.81.4.2 static void scroller_dir_set (scroller_t *scroller*, scroller_dir_t *dir*) [static]

5.81.4.3 scroller_t scroller_init (scroller_t *scroller*, int *rows*, int *cols*, scroller_dir_t *dir*)

5.81.4.4 uint8_t scroller_speed_scale_get (scroller_t *scroller*)

5.81.4.5 void scroller_start (scroller_t *scroller*, uint8_t * *image*, uint8_t * *screen*)

5.81.4.6 static void scroller_stop (scroller_t *scroller*) [static]

5.81.4.7 int8_t scroller_update (scroller_t *scroller*, uint8_t * *image*, uint8_t * *screen*)

5.82 seq.c File Reference

```
#include <limits.h>
#include "seq.h"
```

Functions

- seq_t seq_init (seq_obj_t *seq, const char *(*callback)(void *data, const char *str), void *callback_data)
- int8_t seq_update (seq_t seq)

5.82.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.82.2 Function Documentation

5.82.2.1 seq_t seq_init (seq_obj_t * *seq*, const char *(*)(void *data, const char *str) *callback*, void * *callback_data*)

5.82.2.2 int8_t seq_update (seq_t *seq*)

5.83 seq.h File Reference

```
#include "config.h"
```

Data Structures

- struct seq_obj_t

Typedefs

- typedef seq_obj_t * seq_t

Functions

- seq_t seq_init (seq_obj_t *dev, const char *(*callback)(void *data, const char *str), void *callback_-data)
- static void seq_set (seq_t seq, const char *str)
- static const char * seq_get (seq_t seq)
- int8_t seq_update (seq_t seq)

5.83.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
1 April 2007

5.83.2 Typedef Documentation

5.83.2.1 typedef seq_obj_t* seq_t

5.83.3 Function Documentation

5.83.3.1 static const char* seq_get (seq_t *seq*) [static]

5.83.3.2 seq_t seq_init (seq_obj_t * *dev*, const char *(*)(void *data, const char *str) *callback*, void * *callback_data*)

5.83.3.3 static void seq_set (seq_t *seq*, const char * *str*) [static]

5.83.3.4 int8_t seq_update (seq_t *seq*)

5.84 sflash.c File Reference

```
#include "sflash.h"
```

Functions

- bool sflash_update (sflash_t sflash)

5.84.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
2 July 2007

5.84.2 Function Documentation

5.84.2.1 bool sflash_update (sflash_t *sflash*)

5.85 sflash.h File Reference

```
#include "config.h"
```

Data Structures

- struct sflash_obj_t

Typedefs

- typedef uint16_t sflash_pattern_t
- typedef sflash_obj_t * sflash_t

Functions

- static void sflash_pattern_set (sflash_t sflash, sflash_pattern_t pattern, sflash_pattern_t initial)
- static sflash_pattern_t sflash_pattern_get (sflash_t sflash)
- bool sflash_update (sflash_t)

5.85.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
2 July 2007

5.85.2 Typedef Documentation

5.85.2.1 typedef uint16_t sflash_pattern_t

5.85.2.2 typedef sflash_obj_t* sflash_t

5.85.3 Function Documentation

5.85.3.1 static sflash_pattern_t sflash_pattern_get (sflash_t *sflash*) [static]

5.85.3.2 static void sflash_pattern_set (sflash_t *sflash*, sflash_pattern_t *pattern*, sflash_pattern_t *initial*) [static]

5.85.3.3 bool sflash_update (sflash_t)

5.86 spi_adc.h File Reference

SPI ADC.

```
#include "config.h"
```

Enumerations

- enum spi_adc_mode_t { SPI_ADC_MODE_SINGLE_ENDED, SPI_ADC_MODE_DIFFERENTIAL, SPI_ADC_MODE_DIFFERENTIAL_INVERTED }

5.86.1 Detailed Description

SPI ADC.

Author:
M. P. Hayes, UCECE

Date:
09 August 2007

5.86.2 Enumeration Type Documentation

5.86.2.1 enum spi_adc_mode_t

Enumerator:

SPI_ADC_MODE_SINGLE_ENDED

SPI_ADC_MODE_DIFFERENTIAL

SPI_ADC_MODE_DIFFERENTIAL_INVERTED

5.87 spwm.c File Reference

```
#include "config.h"
#include "spwm.h"
```

Functions

- void spwm_period_set (spwm_t spwm, uint16_t period)
- void spwm_duty_set (spwm_t spwm, uint16_t duty)
- bool spwm_update (spwm_t spwm)
- spwm_t spwm_init (spwm_obj_t *dev)

5.87.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
13 March 2005

5.87.2 Function Documentation

5.87.2.1 void spwm_duty_set (spwm_t *spwm*, uint16_t *duty*)

5.87.2.2 spwm_t spwm_init (spwm_obj_t * *dev*)

5.87.2.3 void spwm_period_set (spwm_t *spwm*, uint16_t *period*)

5.87.2.4 bool spwm_update (spwm_t *spwm*)

5.88 spwm.h File Reference

```
#include "config.h"
#include "port.h"
```

Data Structures

- struct spwm_obj_t

Typedefs

- typedef spwm_struct * spwm_t

Functions

- void spwm_period_set (spwm_t spwm, uint16_t period)
- void spwm_duty_set (spwm_t spwm, uint16_t duty)
- bool spwm_update (spwm_t)
- spwm_t spwm_init (spwm_obj_t *info)

5.88.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
13 March 2005

5.88.2 Typedef Documentation

5.88.2.1 typedef struct spwm_struct* spwm_t

5.88.3 Function Documentation

5.88.3.1 void spwm_duty_set (spwm_t *spwm*, uint16_t *duty*)

5.88.3.2 spwm_t spwm_init (spwm_obj_t * *info*)

5.88.3.3 void spwm_period_set (spwm_t *spwm*, uint16_t *period*)

5.88.3.4 bool spwm_update (spwm_t)

5.89 squeaker.c File Reference

Play simple tunes with PWM.

```
#include "squeaker.h"
```

Defines

- #define SQUEAKER_TRANSPARENT 1
- #define SQUEAKER_HOLDOFF_TIME 50e-3

Enumerations

- enum { SQUEAKER_PRESCALER = 256 }
- enum { SQUEAKER_SCALE_SIZE = 12 }

Functions

- static void squeaker_ticker_set (squeaker_t squeaker)
- static void squeaker_note_fraction_set (squeaker_t squeaker, uint8_t note_fraction)
- void squeaker_note_set (squeaker_t squeaker, squeaker_period_t period, squeaker_period_t duty)
- static void squeaker_rest_play (squeaker_t squeaker)
- static void squeaker_note_play (squeaker_t squeaker, squeaker_note_t note)
- static squeaker_note_t squeaker_char_to_note (uint8_t ch)
- static const char * squeaker_scan (squeaker_t squeaker, const char *str)
- void squeaker_play (squeaker_t squeaker, const char *str)
- void squeaker_speed_set (squeaker_t squeaker, squeaker_speed_t speed)
- void squeaker_volume_set (squeaker_t squeaker, squeaker_volume_t volume)
- int8_t squeaker_update (squeaker_t squeaker)
- squeaker_t squeaker_init (squeaker_obj_t *squeaker, uint16_t poll_rate, squeaker_scale_t *scale_table)

5.89.1 Detailed Description

Play simple tunes with PWM.

Author:
M. P. Hayes, UCECE

Date:
14 April 2007

5.89.2 Define Documentation

5.89.2.1 #define SQUEAKER_HOLDOFF_TIME 50e-3

5.89.2.2 #define SQUEAKER_TRANSPARENT 1

5.89.3 Enumeration Type Documentation

5.89.3.1 anonymous enum

Enumerator:
SQUEAKER_PRESCALER

5.89.3.2 anonymous enum

Enumerator:
SQUEAKER_SCALE_SIZE

5.89.4 Function Documentation

5.89.4.1 static squeaker_note_t squeaker_char_to_note (uint8_t *ch*) [static]

5.89.4.2 squeaker_t squeaker_init (squeaker_obj_t * *squeaker*, uint16_t *poll_rate*, squeaker_scale_t * *scale_table*)

5.89.4.3 static void squeaker_note_fraction_set (squeaker_t *squeaker*, uint8_t *note_fraction*) [static]

5.89.4.4 static void squeaker_note_play (squeaker_t *squeaker*, squeaker_note_t *note*) [static]

5.89.4.5 void squeaker_note_set (squeaker_t *squeaker*, squeaker_period_t *period*, squeaker_period_t *duty*)

5.89.4.6 void squeaker_play (squeaker_t *squeaker*, const char * *str*)

5.89.4.7 static void squeaker_rest_play (squeaker_t *squeaker*) [static]

5.89.4.8 static const char* squeaker_scan (squeaker_t *squeaker*, const char * *str*) [static]

5.89.4.9 void squeaker_speed_set (squeaker_t *squeaker*, squeaker_speed_t *speed*)

5.89.4.10 static void squeaker_ticker_set (squeaker_t *squeaker*) [static]

5.89.4.11 int8_t squeaker_update (squeaker_t *squeaker*)

5.89.4.12 void squeaker_volume_set (squeaker_t *squeaker*, squeaker_volume_t *volume*)

5.90 squeaker.h File Reference

Play simple tunes with PWM.

```
#include "config.h"
#include "font.h"
#include "ticker.h"
```

Data Structures

- struct squeaker_private_t

Defines

- #define SQUEAKER_DIVISOR(POLL_RATE, FREQ) (POLL_RATE / FREQ + 0.5)
- #define SQUEAKER_SCALE_TABLE(POLL_RATE)

Typedefs

- typedef uint8_t squeaker_speed_t
- typedef uint8_t squeaker_scale_t
- typedef uint8_t squeaker_note_t
- typedef uint8_t squeaker_duration_t
- typedef uint8_t squeaker_period_t
- typedef uint8_t squeaker_volume_t
- typedef squeaker_private_t squeaker_obj_t
- typedef squeaker_obj_t * squeaker_t

Enumerations

- enum { SQUEAKER_OCTAVE_DEFAULT = 4 }
- enum { SQUEAKER_SPEED_DEFAULT = 200 }
- enum { SQUEAKER_NOTE_MIN = 40 }

Functions

- squeaker_t squeaker_init (squeaker_obj_t *dev, uint16_t poll_rate, squeaker_scale_t *scale_table)
- void squeaker_play (squeaker_t squeaker, const char *str)
- int8_t squeaker_update (squeaker_t squeaker)
- void squeaker_speed_set (squeaker_t squeaker, squeaker_speed_t speed)
- void squeaker_volume_set (squeaker_t squeaker, squeaker_volume_t volume)

5.90.1 Detailed Description

Play simple tunes with PWM.

Author:
M. P. Hayes, UCECE

Date:
14 April 2007

5.90.2 Define Documentation

5.90.2.1 #define SQUEAKER_DIVISOR(POLL_RATE, FREQ) (POLL_RATE / FREQ + 0.5)

5.90.2.2 #define SQUEAKER_SCALE_TABLE(POLL_RATE)

Value:

```
{SQUEAKER_DIVISOR (POLL_RATE, 82.41), \
    SQUEAKER_DIVISOR (POLL_RATE, 87.31), \
    SQUEAKER_DIVISOR (POLL_RATE, 92.50), \
    SQUEAKER_DIVISOR (POLL_RATE, 98.00), \
    SQUEAKER_DIVISOR (POLL_RATE, 103.83), \
    SQUEAKER_DIVISOR (POLL_RATE, 110.0), \
    SQUEAKER_DIVISOR (POLL_RATE, 116.54), \
    SQUEAKER_DIVISOR (POLL_RATE, 123.47), \
    SQUEAKER_DIVISOR (POLL_RATE, 130.81), \
    SQUEAKER_DIVISOR (POLL_RATE, 138.59), \
    SQUEAKER_DIVISOR (POLL_RATE, 146.83), \
    SQUEAKER_DIVISOR (POLL_RATE, 155.56)}
```

5.90.3 Typedef Documentation

5.90.3.1 typedef uint8_t squeaker_duration_t

5.90.3.2 typedef uint8_t squeaker_note_t

5.90.3.3 typedef squeaker_private_t squeaker_obj_t

5.90.3.4 typedef uint8_t squeaker_period_t

5.90.3.5 typedef uint8_t squeaker_scale_t

5.90.3.6 typedef uint8_t squeaker_speed_t

5.90.3.7 typedef squeaker_obj_t* squeaker_t

5.90.3.8 typedef uint8_t squeaker_volume_t

5.90.4 Enumeration Type Documentation

5.90.4.1 anonymous enum

Enumerator:
SQUEAKER_OCTAVE_DEFAULT

5.90.4.2 anonymous enum

Enumerator:
SQUEAKER_SPEED_DEFAULT

5.90.4.3 anonymous enum

Enumerator:
SQUEAKER_NOTE_MIN

5.90.5 Function Documentation

5.90.5.1 squeaker_t squeaker_init (squeaker_obj_t * *dev*, uint16_t *poll_rate*, squeaker_scale_t * *scale_table*)

5.90.5.2 void squeaker_play (squeaker_t *squeaker*, const char * *str*)

5.90.5.3 void squeaker_speed_set (squeaker_t *squeaker*, squeaker_speed_t *speed*)

5.90.5.4 int8_t squeaker_update (squeaker_t *squeaker*)

5.90.5.5 void squeaker_volume_set (squeaker_t *squeaker*, squeaker_volume_t *volume*)

5.91 squeaker2.c File Reference

```
#include "squeaker.h"
```

Defines

- #define SQUEAKER_TRANSPARENT 1
- #define SQUEAKER_HOLDOFF_TIME 50e-3

Enumerations

- enum { SQUEAKER_PRESCALER = 256 }
- enum { SQUEAKER_SCALE_SIZE = 12 }

Functions

- static void squeaker_ticker_set (squeaker_t *squeaker)
- static void squeaker_note_fraction_set (squeaker_t *squeaker, uint8_t note_fraction)
- void squeaker_note_set (squeaker_t *squeaker, squeaker_period_t period, squeaker_period_t duty)
- static void squeaker_rest_play (squeaker_t *squeaker)
- static void squeaker_note_play (squeaker_t *squeaker, squeaker_note_t note)
- static squeaker_note_t squeaker_char_to_note (uint8_t ch)
- static const char * squeaker_scan (squeaker_t *squeaker, const char *str)
- void squeaker_play (squeaker_t *squeaker, const char *str)
- void squeaker_speed_set (squeaker_t *squeaker, squeaker_speed_t speed)
- void squeaker_volume_set (squeaker_t *squeaker, squeaker_volume_t volume)
- int8_t squeaker_update (squeaker_t *squeaker)
- squeaker_t squeaker_init (squeaker_obj_t *squeaker, uint16_t poll_rate, squeaker_scale_t *scale_table)

5.91.1 Define Documentation

5.91.1.1 #define SQUEAKER_HOLDOFF_TIME 50e-3

5.91.1.2 #define SQUEAKER_TRANSPARENT 1

5.91.2 Enumeration Type Documentation

5.91.2.1 anonymous enum

Enumerator:
SQUEAKER_PRESCALER

5.91.2.2 anonymous enum

Enumerator:
SQUEAKER_SCALE_SIZE

5.91.3 Function Documentation

5.91.3.1 **static squeaker_note_t squeaker_char_to_note (uint8_t *ch*)** `[static]`

5.91.3.2 **squeaker_t squeaker_init (squeaker_obj_t * *squeaker*, uint16_t *poll_rate*, squeaker_scale_t * *scale_table*)**

5.91.3.3 **static void squeaker_note_fraction_set (squeaker_t * *squeaker*, uint8_t *note_fraction*)** `[static]`

5.91.3.4 **static void squeaker_note_play (squeaker_t * *squeaker*, squeaker_note_t *note*)** `[static]`

5.91.3.5 **void squeaker_note_set (squeaker_t * *squeaker*, squeaker_period_t *period*, squeaker_period_t *duty*)**

5.91.3.6 **void squeaker_play (squeaker_t * *squeaker*, const char * *str*)**

5.91.3.7 **static void squeaker_rest_play (squeaker_t * *squeaker*)** `[static]`

5.91.3.8 **static const char* squeaker_scan (squeaker_t * *squeaker*, const char * *str*)** `[static]`

5.91.3.9 **void squeaker_speed_set (squeaker_t * *squeaker*, squeaker_speed_t *speed*)**

5.91.3.10 **static void squeaker_ticker_set (squeaker_t * *squeaker*)** `[static]`

5.91.3.11 **int8_t squeaker_update (squeaker_t * *squeaker*)**

5.91.3.12 **void squeaker_volume_set (squeaker_t * *squeaker*, squeaker_volume_t *volume*)**

5.92 stext.c File Reference

Sequenced text.

```
#include "stext.h"
#include <limits.h>
```

Functions

- static const char * stext_display (void *data, const char *str)
- stext_t stext_init (stext_obj_t *stext, font_t *font, void(*callback)(void *data, uint8_t pixel, bool val), void *callback_data)

5.92.1 Detailed Description

Sequenced text.

Author:
M. P. Hayes, UCECE

Date:
2 April 2007

5.92.2 Function Documentation

5.92.2.1 static const char* stext_display (void * *data*, const char * *str*) [static]

5.92.2.2 stext_t stext_init (stext_obj_t * *stext*, font_t * *font*, void(*)(void *data, uint8_t pixel, bool val) *callback*, void * *callback_data*)

5.93 stext.h File Reference

Sequenced text.

```
#include "config.h"
#include "font.h"
#include "seq.h"
```

Data Structures

- struct stext_obj_t

Typedefs

- typedef stext_obj_t * stext_t

Enumerations

- enum stext_mode_t { STEXT_MODE_NORMAL, STEXT_MODE_CYCLE, STEXT_MODE_NUM }

Functions

- stext_t stext_init (stext_obj_t *dev, font_t *font, void(*callback)(void *data, uint8_t pixel, bool val), void *callback_data)
- static void stext_set (stext_t stext, const char *str)
- static const char * stext_get (stext_t stext)
- static int8_t stext_update (stext_t stext)

5.93.1 Detailed Description

Sequenced text.

Author:
M. P. Hayes, UCECE

Date:
2 April 2007

5.93.2 Typedef Documentation

5.93.2.1 typedef stext_obj_t* stext_t

5.93.3 Enumeration Type Documentation

5.93.3.1 enum stext_mode_t

Enumerator:
STEXT_MODE_NORMAL

STEXT_MODE_CYCLE
STEXT_MODE_NUM

5.93.4 Function Documentation

5.93.4.1 static const char* stext_get (stext_t *stext*) [static]

5.93.4.2 stext_t stext_init (stext_obj_t * *dev*, font_t * *font*, void(*)(void *data, uint8_t pixel, bool val) *callback*, void * *callback_data*)

5.93.4.3 static void stext_set (stext_t *stext*, const char * *str*) [static]

5.93.4.4 static int8_t stext_update (stext_t *stext*) [static]

5.94 ticker.c File Reference

5.94.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
2 April 2007

5.95 ticker.h File Reference

```
#include "config.h"
```

Data Structures

- struct ticker_t
- struct ticker16_t
- struct ticker8_t

Defines

- #define TICKER_INIT(DEV, PERIOD)
- #define TICKER_UPDATE(DEV) (–(DEV) → clock ? 0 : ((DEV) → clock = (DEV) → period))
- #define TICKER_START(DEV) (DEV) → clock = (DEV) → period;

5.95.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
2 April 2007

5.95.2 Define Documentation

5.95.2.1 #define TICKER_INIT(DEV, PERIOD)

Value:

```
(DEV)->period = (PERIOD);               \
    (DEV)->clock = (DEV)->period;
```

5.95.2.2 #define TICKER_START(DEV) (DEV) → clock = (DEV) → period;

5.95.2.3 #define TICKER_UPDATE(DEV) (–(DEV) → clock ? 0 : ((DEV) → clock = (DEV) → period))

5.96 time.c File Reference

Time routines.

```
#include "time.h"
#include "delay.h"
```

Functions

- void time_init (void)
- void time_delay_us (uint16_t us)
- time_t time_rf_timestamp_get (void)
- time_t time_current_time_get (void)
- uint32_t time_time2int (time_t time)
- void time_irq_enable (void)
- void time_irq_disable (void)

Variables

- static volatile uint16_t ms_ticks = 0

5.96.1 Detailed Description

Time routines.

Author:
Ian Downes, UCECE

Date:
12 January 2005

5.96.2 Function Documentation

5.96.2.1 time_t time_current_time_get (void)

5.96.2.2 void time_delay_us (uint16_t *us*)

5.96.2.3 void time_init (void)

5.96.2.4 void time_irq_disable (void)

5.96.2.5 void time_irq_enable (void)

5.96.2.6 time_t time_rf_timestamp_get (void)

5.96.2.7 uint32_t time_time2int (time_t *time*)

5.96.3 Variable Documentation

5.96.3.1 volatile uint16_t ms_ticks = 0 [static]

5.97 time.h File Reference

Time routines.

```
#include "config.h"
#include "irq.h"
```

Data Structures

- struct time

Defines

- #define TOPCNT (64000 - 1)
- #define DELAY_MIN 10
- #define DELAY_MAX 3999

Typedefs

- typedef time time_t

Enumerations

- enum delay_ret_t {
  ERRD, USTICKS, TIMEOUT, COMMS_INT,
  OTHER_INT }

Functions

- void time_init (void)
- time_t time_rf_timestamp_get (void)
- time_t time_current_time_get (void)
- void time_delay_us (uint16_t us)
- uint32_t time_time2int (time_t time)
- void time_irq_enable (void)
- void time_irq_disable (void)

5.97.1 Detailed Description

Time routines.

Author:
Ian Downes, UCECE

Date:
12 January 2005

5.97.2 Define Documentation

5.97.2.1 #define DELAY_MAX 3999

5.97.2.2 #define DELAY_MIN 10

5.97.2.3 #define TOPCNT (64000 - 1)

5.97.3 Typedef Documentation

5.97.3.1 typedef struct time time_t

5.97.4 Enumeration Type Documentation

5.97.4.1 enum delay_ret_t

Enumerator:

ERRD

USTICKS

TIMEOUT

COMMS_INT

OTHER_INT

5.97.5 Function Documentation

5.97.5.1 time_t time_current_time_get (void)

5.97.5.2 void time_delay_us (uint16_t *us*)

5.97.5.3 void time_init (void)

5.97.5.4 void time_irq_disable (void)

5.97.5.5 void time_irq_enable (void)

5.97.5.6 time_t time_rf_timestamp_get (void)

5.97.5.7 uint32_t time_time2int (time_t *time*)

5.98 tweeter.c File Reference

Generate PWM for a piezo tweeter.

```
#include "tweeter.h"
```

Defines

- #define TWEETER_TRANSPARENT 1
- #define TWEETER_HOLDOFF_TIME 50e-3

Enumerations

- enum { TWEETER_SCALE_SIZE = 12 }

Functions

- void tweeter_note_set (tweeter_t tweeter, tweeter_period_t period, tweeter_period_t duty)
- void tweeter_note_play (tweeter_t tweeter, tweeter_note_t note, uint8_t velocity)
- int8_t tweeter_update (tweeter_t tweeter)
- tweeter_t tweeter_init (tweeter_obj_t *tweeter, uint16_t poll_rate, tweeter_scale_t *scale_table)

5.98.1 Detailed Description

Generate PWM for a piezo tweeter.

Author:
M. P. Hayes, UCECE

Date:
20 April 2007

5.98.2 Define Documentation

5.98.2.1 #define TWEETER_HOLDOFF_TIME 50e-3

5.98.2.2 #define TWEETER_TRANSPARENT 1

5.98.3 Enumeration Type Documentation

5.98.3.1 anonymous enum

Enumerator:
TWEETER_SCALE_SIZE

5.98.4 Function Documentation

5.98.4.1 tweeter_t tweeter_init (tweeter_obj_t * *tweeter*, uint16_t *poll_rate*, tweeter_scale_t * *scale_table*)

5.98.4.2 void tweeter_note_play (tweeter_t *tweeter*, tweeter_note_t *note*, uint8_t *velocity*)

5.98.4.3 void tweeter_note_set (tweeter_t *tweeter*, tweeter_period_t *period*, tweeter_period_t *duty*)

5.98.4.4 int8_t tweeter_update (tweeter_t *tweeter*)

5.99 tweeter.h File Reference

Generate PWM for a piezo tweeter.

```
#include "config.h"
#include "font.h"
#include "ticker.h"
```

Data Structures

- struct tweeter_private_t

Defines

- #define TWEETER_DIVISOR(POLL_RATE, FREQ) (POLL_RATE / FREQ + 0.5)
- #define TWEETER_SCALE_TABLE(POLL_RATE)

Typedefs

- typedef uint8_t tweeter_note_t
- typedef uint8_t tweeter_duration_t
- typedef uint8_t tweeter_period_t
- typedef uint8_t tweeter_velocity_t
- typedef uint8_t tweeter_scale_t
- typedef tweeter_private_t tweeter_obj_t
- typedef tweeter_obj_t * tweeter_t

Enumerations

- enum { TWEETER_NOTE_MIN = 40 }

Functions

- int8_t tweeter_update (tweeter_t tweeter)
- void tweeter_note_play (tweeter_t tweeter, tweeter_note_t note, uint8_t velocity)
- tweeter_t tweeter_init (tweeter_obj_t *dev, uint16_t poll_rate, tweeter_scale_t *scale_table)

5.99.1 Detailed Description

Generate PWM for a piezo tweeter.

Author:
M. P. Hayes, UCECE

Date:
20 April 2007

5.99.2 Define Documentation

5.99.2.1 #define TWEETER_DIVISOR(POLL_RATE, FREQ) (POLL_RATE / FREQ + 0.5)

5.99.2.2 #define TWEETER_SCALE_TABLE(POLL_RATE)

Value:

```
{TWEETER_DIVISOR (POLL_RATE, 82.41), \
    TWEETER_DIVISOR (POLL_RATE, 87.31), \
    TWEETER_DIVISOR (POLL_RATE, 92.50), \
    TWEETER_DIVISOR (POLL_RATE, 98.00), \
    TWEETER_DIVISOR (POLL_RATE, 103.83), \
    TWEETER_DIVISOR (POLL_RATE, 110.0), \
    TWEETER_DIVISOR (POLL_RATE, 116.54), \
    TWEETER_DIVISOR (POLL_RATE, 123.47), \
    TWEETER_DIVISOR (POLL_RATE, 130.81), \
    TWEETER_DIVISOR (POLL_RATE, 138.59), \
    TWEETER_DIVISOR (POLL_RATE, 146.83), \
    TWEETER_DIVISOR (POLL_RATE, 155.56)}
```

5.99.3 Typedef Documentation

5.99.3.1 typedef uint8_t tweeter_duration_t

5.99.3.2 typedef uint8_t tweeter_note_t

5.99.3.3 typedef tweeter_private_t tweeter_obj_t

5.99.3.4 typedef uint8_t tweeter_period_t

5.99.3.5 typedef uint8_t tweeter_scale_t

5.99.3.6 typedef tweeter_obj_t* tweeter_t

5.99.3.7 typedef uint8_t tweeter_velocity_t

5.99.4 Enumeration Type Documentation

5.99.4.1 anonymous enum

Enumerator:

TWEETER_NOTE_MIN

5.99.5 Function Documentation

5.99.5.1 tweeter_t tweeter_init (tweeter_obj_t * *dev*, uint16_t *poll_rate*, tweeter_scale_t * *scale_table*)

5.99.5.2 void tweeter_note_play (tweeter_t *tweeter*, tweeter_note_t *note*, uint8_t *velocity*)

5.99.5.3 int8_t tweeter_update (tweeter_t *tweeter*)

5.100 u1wire.c File Reference

Low level routines to drive Dallas universal 1 wire bus. This only supports a single instance of a 1 wire bus.

```
#include <stdio.h>
#include "config.h"
#include "delay.h"
#include "u1wire.h"
#include "irq.h"
#include "port.h"
```

Defines

- #define U1WIRE_DEBUG 1
- #define U1WIRE_RELEASE() port_pin_config_pullup (U1WIRE_PORT, U1WIRE_BIT)
- #define U1WIRE_DRIVE()
- #define U1WIRE_TEST() port_pin_read (U1WIRE_PORT, U1WIRE_BIT)

Enumerations

- enum { U1WIRE_READ_ROM = 0x33, U1WIRE_SKIP_ROM = 0xcc, U1WIRE_MATCH_ROM = 0x55, U1WIRE_RECALL = 0xb8 }
- enum { U1WIRE_DELAY_OFFSET = 6, U1WIRE_ADDR_BYTES = 6 }

Functions

- int8_t u1wire_reset (void)
- void u1wire_bit_write (uint8_t value)
- void u1wire_byte_write (uint8_t value)
- uint8_t u1wire_bit_read (void)
- uint8_t u1wire_byte_read (void)
- int8_t u1wire_rom_code_read (u1wire_t dev)
- int8_t u1wire_command (u1wire_t dev, uint8_t command)
- int8_t u1wire_broadcast (uint8_t command)
- int8_t u1wire_read (void *data, uint8_t bytes)
- int8_t u1wire_write (void *data, uint8_t bytes)
- bool u1wire_ready_p (void)
- int8_t u1wire_init (u1wire_obj_t *devices, uint8_t devices_max)

5.100.1 Detailed Description

Low level routines to drive Dallas universal 1 wire bus. This only supports a single instance of a 1 wire bus.

Author:
M. P. Hayes

Date:
16 May 20002

5.100.2 Define Documentation

5.100.2.1 #define U1WIRE_DEBUG 1

5.100.2.2 #define U1WIRE_DRIVE()

Value:

```
do {port_pin_config_output (U1WIRE_PORT, U1WIRE_BIT); \
        port_pin_set_low (U1WIRE_PORT, U1WIRE_BIT);} while (0)
```

5.100.2.3 #define U1WIRE_RELEASE() port_pin_config_pullup (U1WIRE_PORT, U1WIRE_BIT)

5.100.2.4 #define U1WIRE_TEST() port_pin_read (U1WIRE_PORT, U1WIRE_BIT)

5.100.3 Enumeration Type Documentation

5.100.3.1 anonymous enum

Enumerator:
U1WIRE_READ_ROM

U1WIRE_SKIP_ROM

U1WIRE_MATCH_ROM

U1WIRE_RECALL

5.100.3.2 anonymous enum

Enumerator:
U1WIRE_DELAY_OFFSET

U1WIRE_ADDR_BYTES

5.100.4 Function Documentation

5.100.4.1 uint8_t u1wire_bit_read (void)

5.100.4.2 void u1wire_bit_write (uint8_t *value*)

5.100.4.3 int8_t u1wire_broadcast (uint8_t *command*)

5.100.4.4 uint8_t u1wire_byte_read (void)

5.100.4.5 void u1wire_byte_write (uint8_t *value*)

5.100.4.6 int8_t u1wire_command (u1wire_t *dev*, uint8_t *command*)

5.100.4.7 int8_t u1wire_init (u1wire_obj_t * *devices*, uint8_t *devices_max*)

5.100.4.8 int8_t u1wire_read (void * *data*, uint8_t *bytes*)

5.100.4.9 bool u1wire_ready_p (void)

5.100.4.10 int8_t u1wire_reset (void)

5.100.4.11 int8_t u1wire_rom_code_read (u1wire_t *dev*)

5.100.4.12 int8_t u1wire_write (void * *data*, uint8_t *bytes*)

5.101 u1wire.h File Reference

```
#include "config.h"
```

Data Structures

- union u1wire_rom_code_t
- struct u1wire_obj_t

Typedefs

- typedef u1wire_obj_t * u1wire_t

Enumerations

- enum {

  U1WIRE_ERR_BUS_LOW = 1, U1WIRE_ERR_BUS_STUCK = 2, U1WIRE_ERR_PRESENCE_SHORT = 3, U1WIRE_ERR_PRESENCE_LONG = 4,

  U1WIRE_ERR_MULTIPLE_DEVICES = 5, U1WIRE_ERR_BUS_HIGH = 6 }

Functions

- bool u1wire_ready_p (void)
- int8_t u1wire_reset (void)
- void u1wire_bit_write (uint8_t value)
- void u1wire_byte_write (uint8_t value)
- uint8_t u1wire_bit_read (void)
- uint8_t u1wire_byte_read (void)
- int8_t u1wire_command (u1wire_t dev, uint8_t command)
- int8_t u1wire_broadcast (uint8_t command)
- int8_t u1wire_read (void *data, uint8_t size)
- int8_t u1wire_write (void *data, uint8_t size)
- int8_t u1wire_init (u1wire_obj_t *devices, uint8_t devices_max)
- void u1wire_debug (u1wire_t dev)

5.101.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
08 June 2002

5.101.2 Typedef Documentation

5.101.2.1 typedef u1wire_obj_t* u1wire_t

5.101.3 Enumeration Type Documentation

5.101.3.1 anonymous enum

Enumerator:

U1WIRE_ERR_BUS_LOW

U1WIRE_ERR_BUS_STUCK

U1WIRE_ERR_PRESENCE_SHORT

U1WIRE_ERR_PRESENCE_LONG

U1WIRE_ERR_MULTIPLE_DEVICES

U1WIRE_ERR_BUS_HIGH

5.101.4 Function Documentation

5.101.4.1 uint8_t u1wire_bit_read (void)

5.101.4.2 void u1wire_bit_write (uint8_t *value*)

5.101.4.3 int8_t u1wire_broadcast (uint8_t *command*)

5.101.4.4 uint8_t u1wire_byte_read (void)

5.101.4.5 void u1wire_byte_write (uint8_t *value*)

5.101.4.6 int8_t u1wire_command (u1wire_t *dev*, uint8_t *command*)

5.101.4.7 void u1wire_debug (u1wire_t *dev*)

5.101.4.8 int8_t u1wire_init (u1wire_obj_t * *devices*, uint8_t *devices_max*)

5.101.4.9 int8_t u1wire_read (void * *data*, uint8_t *size*)

5.101.4.10 bool u1wire_ready_p (void)

5.101.4.11 int8_t u1wire_reset (void)

5.101.4.12 int8_t u1wire_write (void * *data*, uint8_t *size*)

5.102 u1wire_debug.c File Reference

```
#include "u1wire.h"
#include <stdio.h>
```

Defines

- #define U1WIRE_DEBUG 1

Functions

- void u1wire_debug (u1wire_t dev)

5.102.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.102.2 Define Documentation

5.102.2.1 #define U1WIRE_DEBUG 1

5.102.3 Function Documentation

5.102.3.1 void u1wire_debug (u1wire_t *dev*)

5.103 u1wire_discover.c File Reference

```
#include "dscrc8.h"
#include "u1wire.h"
```

Data Structures

- struct u1wire_state_t

Enumerations

- enum { U1WIRE_SEARCH = 0xf0 }

Functions

- static bool u1wire_search (u1wire_state_t *state, u1wire_rom_code_t *rom_code)
- int8_t u1wire_discover (u1wire_obj_t *devices, uint8_t devices_max)

5.103.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.103.2 Enumeration Type Documentation

5.103.2.1 anonymous enum

Enumerator:
U1WIRE_SEARCH

5.103.3 Function Documentation

5.103.3.1 int8_t u1wire_discover (u1wire_obj_t * *devices*, uint8_t *devices_max*)

5.103.3.2 static bool u1wire_search (u1wire_state_t * *state*, u1wire_rom_code_t * *rom_code*) [static]

5.104 u1wire_discover.h File Reference

This discovers devices on a Dallas universal one wire bus. Note this has been superseded by u1wire_-enumerate.

```
#include "u1wire.h"
```

Functions

- int8_t u1wire_discover (u1wire_obj_t *devices, uint8_t devices_max)

5.104.1 Detailed Description

This discovers devices on a Dallas universal one wire bus. Note this has been superseded by u1wire_-enumerate.

Author:
M. P. Hayes, UCECE

Date:
24 February 2005

5.104.2 Function Documentation

5.104.2.1 int8_t u1wire_discover (u1wire_obj_t * *devices*, uint8_t *devices_max*)

5.105 u1wire_enumerate.c File Reference

```
#include "dscrc8.h"
#include "u1wire.h"
#include "u1wire_enumerate.h"
```

Enumerations

- enum { U1WIRE_SEARCH = 0xf0 }

Functions

- static bool u1wire_search (u1wire_state_t *state, u1wire_rom_code_t *rom_code)
- u1wire_obj_t * u1wire_enumerate_next (u1wire_enumerate_t *info)
- u1wire_obj_t * u1wire_enumerate (u1wire_enumerate_t *info)

5.105.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.105.2 Enumeration Type Documentation

5.105.2.1 anonymous enum

Enumerator:
U1WIRE_SEARCH

5.105.3 Function Documentation

5.105.3.1 u1wire_obj_t* u1wire_enumerate (u1wire_enumerate_t * *info*)

5.105.3.2 u1wire_obj_t* u1wire_enumerate_next (u1wire_enumerate_t * *info*)

5.105.3.3 static bool u1wire_search (u1wire_state_t * *state*, u1wire_rom_code_t * *rom_code*)

[static]

5.106 u1wire_enumerate.h File Reference

This discovers devices on a Dallas universal one wire bus.

```
#include "u1wire.h"
```

Data Structures

- struct u1wire_state_t
- struct u1wire_enumerate_t

Functions

- u1wire_obj_t * u1wire_enumerate (u1wire_enumerate_t *info)
- u1wire_obj_t * u1wire_enumerate_next (u1wire_enumerate_t *info)

5.106.1 Detailed Description

This discovers devices on a Dallas universal one wire bus.

Author:
M. P. Hayes, UCECE

Date:
24 February 2005

5.106.2 Function Documentation

5.106.2.1 u1wire_obj_t* u1wire_enumerate (u1wire_enumerate_t * *info*)

5.106.2.2 u1wire_obj_t* u1wire_enumerate_next (u1wire_enumerate_t * *info*)

5.107 uint16toa.c File Reference

16 bit unsigned int to ASCII conversion.

```
#include "config.h"
```

Functions

- void uint16toa (uint16_t num, char *str, bool leading_zeroes)

5.107.1 Detailed Description

16 bit unsigned int to ASCII conversion.

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.107.2 Function Documentation

5.107.2.1 void uint16toa (uint16_t *num*, char * *str*, bool *leading_zeroes*)

Convert 16 bit unsigned integer to ASCII.

5.108 uint16toa.h File Reference

16 bit unsigned int to ASCII conversion.

```
#include "config.h"
```

Functions

- void uint16toa (uint16_t num, char *str, bool leading_zeroes)

5.108.1 Detailed Description

16 bit unsigned int to ASCII conversion.

Author:
M. P. Hayes, UCECE

Date:
21 Nov 2006

5.108.2 Function Documentation

5.108.2.1 void uint16toa (uint16_t *num*, char * *str*, bool *leading_zeroes*)

Convert 16 bit unsigned integer to ASCII.

5.109 uint8toa.c File Reference

```
#include "config.h"
```

Functions

- void uint8toa (uint8_t num, char *str, bool leading_zeroes)

5.109.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
15 May 2007

5.109.2 Function Documentation

5.109.2.1 void uint8toa (uint8_t *num*, char * *str*, bool *leading_zeroes*)

5.110 uint8toa.h File Reference

```
#include "config.h"
```

Functions

- void uint8toa (uint8_t num, char ∗str, bool leading_zeroes)

5.110.1 Detailed Description

Author:
M. P. Hayes, UCECE

Date:
21 Nov 2006

5.110.2 Function Documentation

5.110.2.1 void uint8toa (uint8_t *num*, char ∗ *str*, bool *leading_zeroes*)

5.111 usart.c File Reference

Unbuffered USART implementation.

```
#include "usart.h"
#include "peripherals.h"
```

Data Structures

- struct usart_dev_struct

Defines

- #define USART0_ENABLE (USART_NUM >= 1)
- #define USART1_ENABLE (USART_NUM >= 2)

Functions

- usart_t usart_init (uint8_t channel, uint16_t baud_divisor)
- bool usart_read_ready_p (usart_t usart)
- bool usart_write_ready_p (usart_t usart)
- bool usart_write_finished_p (usart_t usart)
- int8_t usart_getc (usart_t usart)
- int8_t usart_putc (usart_t usart, char ch)
- int8_t usart_puts (usart_t usart, const char *str)

5.111.1 Detailed Description

Unbuffered USART implementation.

Author:
M. P. Hayes, UCECE

Date:
21 June 2007

5.111.2 Define Documentation

5.111.2.1 #define USART0_ENABLE (USART_NUM >= 1)

5.111.2.2 #define USART1_ENABLE (USART_NUM >= 2)

5.111.3 Function Documentation

5.111.3.1 int8_t usart_getc (usart_t *usart*)

5.111.3.2 usart_t usart_init (uint8_t *channel*, uint16_t *baud_divisor*)

5.111.3.3 int8_t usart_putc (usart_t *usart*, char *ch*)

5.111.3.4 int8_t usart_puts (usart_t *usart*, const char * *str*)

5.111.3.5 bool usart_read_ready_p (usart_t *usart*)

5.111.3.6 bool usart_write_finished_p (usart_t *usart*)

5.111.3.7 bool usart_write_ready_p (usart_t *usart*)

5.112 usart.h File Reference

Unbuffered USART interface.

```
#include "config.h"
#include "usart0.h"
```

Defines

- #define USART_BAUD_DIVISOR(BAUD_RATE) USART0_BAUD_DIVISOR(BAUD_RATE)

Typedefs

- typedef usart_dev_struct usart_dev_t
- typedef usart_dev_t * usart_t

Functions

- usart_t usart_init (uint8_t channel, uint16_t baud_divisor)
- bool usart_read_ready_p (usart_t usart)
- bool usart_write_ready_p (usart_t usart)
- int8_t usart_getc (usart_t usart)
- int8_t usart_putc (usart_t usart, char ch)
- int8_t usart_puts (usart_t usart, const char *str)

5.112.1 Detailed Description

Unbuffered USART interface.

Author:
M. P. Hayes, UCECE

Date:
21 June 2007

5.112.2 Define Documentation

5.112.2.1 #define USART_BAUD_DIVISOR(BAUD_RATE) USART0_BAUD_DIVISOR(BAUD_RATE)

5.112.3 Typedef Documentation

5.112.3.1 typedef struct usart_dev_struct usart_dev_t

5.112.3.2 typedef usart_dev_t* usart_t

5.112.4 Function Documentation

5.112.4.1 int8_t usart_getc (usart_t *usart*)

5.112.4.2 usart_t usart_init (uint8_t *channel*, uint16_t *baud_divisor*)

5.112.4.3 int8_t usart_putc (usart_t *usart*, char *ch*)

5.112.4.4 int8_t usart_puts (usart_t *usart*, const char * *str*)

5.112.4.5 bool usart_read_ready_p (usart_t *usart*)

5.112.4.6 bool usart_write_ready_p (usart_t *usart*)

Index