# Pic Pack

Version 3.02
12/3/2010 1:44:00 PM

# Table of Contents

# Data Structure Index

## Data Structures

Here are the data structures with brief descriptions:



# File Index

## File List

Here is a list of all files with brief descriptions:

# Data Structure Documentation

## buffer_descriptor Struct Reference

`#include <pic_usb.h>`

### Data Fields

- uns16 addr
- uns8 count
- uns8 stat

---

### Detailed Description

Describes the USB endpoint buffer descriptor

Definition at line 159 of file pic_usb.h.

---

## Field Documentation

### uns16 buffer_descriptor::addr

Definition at line 162 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), and usb_send_data_chunk().

### uns8 buffer_descriptor::count

Definition at line 160 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), and usb_send_data_chunk().

### uns8 buffer_descriptor::stat

Definition at line 160 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_stall(), usb_handle_standard_request(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), usb_send_empty_data_pkt(), usb_send_one_byte(), usb_stall_ep0(), and usb_stall_on_in().

---

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

- pic_usb.h

---

# CDC_ACM_functional_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns8 capabilities
- uns8 descriptor_subtype
- uns8 descriptor_type
- uns8 length

---

## Detailed Description

Definition at line 367 of file pic_usb.h.

---

## Field Documentation

**uns8 CDC_ACM_functional_descriptor::capabilities**

Definition at line 368 of file pic_usb.h.

**uns8 CDC_ACM_functional_descriptor::descriptor_subtype**

Definition at line 368 of file pic_usb.h.

**uns8 CDC_ACM_functional_descriptor::descriptor_type**

Definition at line 368 of file pic_usb.h.

**uns8 CDC_ACM_functional_descriptor::length**

Definition at line 368 of file pic_usb.h.

---

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

- pic_usb.h

---

# CDC_call_mgt_functional_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns8 capabilities
- uns8 data_interface
- uns8 descriptor_subtype
- uns8 descriptor_type
- uns8 length

---

## Detailed Description

Definition at line 382 of file pic_usb.h.

---

## Field Documentation

**uns8 CDC_call_mgt_functional_descriptor::capabilities**

Definition at line 383 of file pic_usb.h.

**uns8 CDC_call_mgt_functional_descriptor::data_interface**

Definition at line 383 of file pic_usb.h.

**uns8 CDC_call_mgt_functional_descriptor::descriptor_subtype**

Definition at line 383 of file pic_usb.h.

**uns8 CDC_call_mgt_functional_descriptor::descriptor_type**

Definition at line 383 of file pic_usb.h.

**uns8 CDC_call_mgt_functional_descriptor::length**

Definition at line 383 of file pic_usb.h.

---

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

- pic_usb.h

---

# CDC_header_functional_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns16 CDC_version
- uns8 descriptor_subtype
- uns8 descriptor_type
- uns8 length

---

## Detailed Description

Definition at line 360 of file pic_usb.h.

---

## Field Documentation

**uns16 CDC_header_functional_descriptor::CDC_version**

Definition at line 364 of file pic_usb.h.

**uns8 CDC_header_functional_descriptor::descriptor_subtype**

Definition at line 361 of file pic_usb.h.

**uns8 CDC_header_functional_descriptor::descriptor_type**

Definition at line 361 of file pic_usb.h.

**uns8 CDC_header_functional_descriptor::length**

Definition at line 361 of file pic_usb.h.

---

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

- pic_usb.h

---

# CDC_union_functional_descriptor Struct Reference

`#include <pic_usb.h>`

## Data Fields

- uns8 descriptor_subtype
- uns8 descriptor_type
- uns8 length
- uns8 master_interface
- uns8 slave_interface

---

## Detailed Description

Definition at line 374 of file pic_usb.h.

---

## Field Documentation

**uns8 CDC_union_functional_descriptor::descriptor_subtype**

Definition at line 375 of file pic_usb.h.

**uns8 CDC_union_functional_descriptor::descriptor_type**

Definition at line 375 of file pic_usb.h.

**uns8 CDC_union_functional_descriptor::length**

Definition at line 375 of file pic_usb.h.

**uns8 CDC_union_functional_descriptor::master_interface**

Definition at line 375 of file pic_usb.h.

**uns8 CDC_union_functional_descriptor::slave_interface**

Definition at line 375 of file pic_usb.h.

---

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

- pic_usb.h

---

# configuration_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns8 attributes
- uns8 configuration_string_id
- uns8 configuration_value
- uns8 descriptor_type
- uns8 length
- uns8 max_power
- uns8 num_interfaces
- uns16 total_length

---

## Detailed Description

Configuration descriptor

Definition at line 312 of file pic_usb.h.

---

## Field Documentation

**uns8 configuration_descriptor::attributes**

Definition at line 316 of file pic_usb.h.

**uns8 configuration_descriptor::configuration_string_id**

Definition at line 316 of file pic_usb.h.

**uns8 configuration_descriptor::configuration_value**

Definition at line 316 of file pic_usb.h.

**uns8 configuration_descriptor::descriptor_type**

Definition at line 313 of file pic_usb.h.

**uns8 configuration_descriptor::length**

Definition at line 313 of file pic_usb.h.

**uns8 configuration_descriptor::max_power**

Definition at line 316 of file pic_usb.h.

**uns8 configuration_descriptor::num_interfaces**

Definition at line 316 of file pic_usb.h.

**uns16 configuration_descriptor::total_length**

Definition at line 315 of file pic_usb.h.

---

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

- pic_usb.h

---

# device_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns8 descriptor_type
- uns8 device_class
- uns8 device_protocol
- uns16 device_release
- uns8 device_subclass

- uns8 length
- uns8 manufacturer_string_id
- uns8 max_packet_size_ep0
- uns8 num_configurations
- uns16 product_id
- uns8 product_string_id
- uns8 serial_string_id
- uns16 usb_version
- uns16 vendor_id

---

## Detailed Description

Device descriptor

Definition at line 294 of file pic_usb.h.

---

## Field Documentation

### uns8 device_descriptor::descriptor_type

Definition at line 295 of file pic_usb.h.

### uns8 device_descriptor::device_class

Definition at line 298 of file pic_usb.h.

### uns8 device_descriptor::device_protocol

Definition at line 298 of file pic_usb.h.

### uns16 device_descriptor::device_release

Definition at line 302 of file pic_usb.h.

### uns8 device_descriptor::device_subclass

Definition at line 298 of file pic_usb.h.

### uns8 device_descriptor::length

Definition at line 295 of file pic_usb.h.

### uns8 device_descriptor::manufacturer_string_id

Definition at line 305 of file pic_usb.h.

**uns8 device_descriptor::max_packet_size_ep0**

Definition at line 301 of file pic_usb.h.

**uns8 device_descriptor::num_configurations**

Definition at line 305 of file pic_usb.h.

**uns16 device_descriptor::product_id**

Definition at line 302 of file pic_usb.h.

**uns8 device_descriptor::product_string_id**

Definition at line 305 of file pic_usb.h.

**uns8 device_descriptor::serial_string_id**

Definition at line 305 of file pic_usb.h.

**uns16 device_descriptor::usb_version**

Definition at line 297 of file pic_usb.h.

**uns16 device_descriptor::vendor_id**

Definition at line 302 of file pic_usb.h.

---

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

- pic_usb.h

---

# endpoint_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns8 attributes
- uns8 descriptor_type
- uns8 endpoint_address
- uns8 interval
- uns8 length

- uns16 max_packet_size

---

## Detailed Description

Endpoint descriptor

Definition at line 339 of file pic_usb.h.

---

## Field Documentation

**uns8 endpoint_descriptor::attributes**

Definition at line 340 of file pic_usb.h.

**uns8 endpoint_descriptor::descriptor_type**

Definition at line 340 of file pic_usb.h.

**uns8 endpoint_descriptor::endpoint_address**

Definition at line 340 of file pic_usb.h.

**uns8 endpoint_descriptor::interval**

Definition at line 345 of file pic_usb.h.

**uns8 endpoint_descriptor::length**

Definition at line 340 of file pic_usb.h.

**uns16 endpoint_descriptor::max_packet_size**

Definition at line 344 of file pic_usb.h.

---

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

- pic_usb.h

---

# hid_descriptor Struct Reference

```
#include <pic_usb.h>
```

## Data Fields

- uns16 class_descriptor_length
- uns8 class_descriptor_type
- uns8 country_code
- uns8 descriptor_type
- uns16 hid_spec
- uns8 length
- uns8 num_class_descriptors

---

## Detailed Description

Human Interface Device descriptor

Definition at line 349 of file pic_usb.h.

---

## Field Documentation

### uns16 hid_descriptor::class_descriptor_length

Definition at line 356 of file pic_usb.h.

### uns8 hid_descriptor::class_descriptor_type

Definition at line 353 of file pic_usb.h.

### uns8 hid_descriptor::country_code

Definition at line 353 of file pic_usb.h.

### uns8 hid_descriptor::descriptor_type

Definition at line 350 of file pic_usb.h.

### uns16 hid_descriptor::hid_spec

Definition at line 352 of file pic_usb.h.

### uns8 hid_descriptor::length

Definition at line 350 of file pic_usb.h.

### uns8 hid_descriptor::num_class_descriptors

Definition at line 353 of file pic_usb.h.

---

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

- pic_usb.h

---

# interface_descriptor Struct Reference

`#include <pic_usb.h>`

## Data Fields

- uns8 alternate_setting
- uns8 descriptor_type
- uns8 interface_class
- uns8 interface_number
- uns8 interface_protocol
- uns8 interface_string_id
- uns8 interface_subclass
- uns8 length
- uns8 num_endpoints

---

## Detailed Description

Interface descriptor

Definition at line 326 of file pic_usb.h.

---

## Field Documentation

### uns8 interface_descriptor::alternate_setting

Definition at line 327 of file pic_usb.h.

### uns8 interface_descriptor::descriptor_type

Definition at line 327 of file pic_usb.h.

### uns8 interface_descriptor::interface_class

Definition at line 327 of file pic_usb.h.

### uns8 interface_descriptor::interface_number

Definition at line 327 of file pic_usb.h.

**uns8 interface_descriptor::interface_protocol**

Definition at line 327 of file pic_usb.h.

**uns8 interface_descriptor::interface_string_id**

Definition at line 327 of file pic_usb.h.

**uns8 interface_descriptor::interface_subclass**

Definition at line 327 of file pic_usb.h.

**uns8 interface_descriptor::length**

Definition at line 327 of file pic_usb.h.

**uns8 interface_descriptor::num_endpoints**

Definition at line 327 of file pic_usb.h.

---

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

- pic_usb.h

---

# its2_packet Struct Reference

```
#include <its_mode2.h>
```

## Data Fields

- uns8 hop_count
- uns16 its_dest_id
- uns16 its_network_id
- uns16 its_source_id
- uns8 max_hop_count
- uns8 num_routes
- uns8 packet_type
- uns16 routers [ITS2_MAX_HOP_COUNT]
- uns8 sequence

---

## Detailed Description

ITS2 Packet definition

Definition at line 120 of file its_mode2.h.

## Field Documentation

### uns8 its2_packet::hop_count

Number of routes (hops) specified for this packet to take

Definition at line 128 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

### uns16 its2_packet::its_dest_id

ITS device ID of sender

Definition at line 125 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_print_queue(), and its2_router_queue_packet().

### uns16 its2_packet::its_network_id

Sequence of packet (incremented for each packet)

Definition at line 123 of file its_mode2.h.

Referenced by its2_print_packet(), and its2_router_queue_packet().

### uns16 its2_packet::its_source_id

Network ID of packet

Definition at line 124 of file its_mode2.h.

Referenced by its2_print_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

### uns8 its2_packet::max_hop_count

ITS device ID of destination

Definition at line 126 of file its_mode2.h.

Referenced by its2_print_packet(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

### uns8 its2_packet::num_routes

Maximum number of permitted routes

Definition at line 127 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_rebroadcast_net_discover_req(), and its2_transmit().

### uns8 its2_packet::packet_type

Definition at line 121 of file its_mode2.h.

Referenced by its2_print_packet(), and its2_router_queue_packet().

**uns16 its2_packet::routers[ITS2_MAX_HOP_COUNT]**

Number of routes (hops) already made by this packet

Definition at line 129 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_rebroadcast_net_discover_req(), and its2_transmit().

**uns8 its2_packet::sequence**

Type of packet (see its_common.h)

Definition at line 122 of file its_mode2.h.

Referenced by its2_print_packet(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

---

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

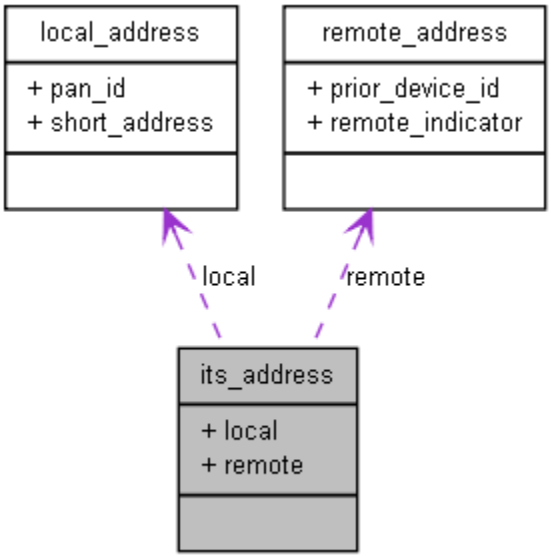
- its_mode2.h

---

# its_address Union Reference

`#include <its_common.h>`

Collaboration diagram for its_address:




## Data Fields

- local_address local
- remote_address remote

---

### Detailed Description

Union of local and remote addresses

Definition at line 93 of file its_common.h.

---

### Field Documentation

**local_address** **its_address::local**

Definition at line 94 of file its_common.h.

Referenced by its2_transmit(), its_add_local_device(), and its_transmit_to_handle().

**remote_address** **its_address::remote**

Definition at line 95 of file its_common.h.

Referenced by its2_forward_routed_packet(), and its_add_net_device().

---

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

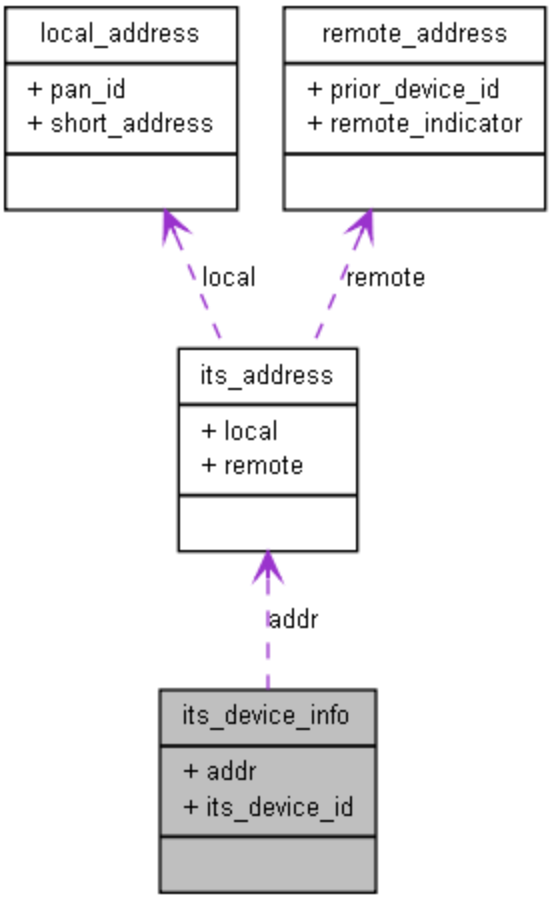
- its_common.h

---

## its_device_info Struct Reference

`#include <its_common.h>`

Collaboration diagram for its_device_info:




## Data Fields

- its_address addr
- uns16 its_device_id

---

## Detailed Description

Device info - its_device_id and address (local or remote)

Definition at line 99 of file its_common.h.

---

## Field Documentation

### its_address its_device_info::addr

Definition at line 101 of file its_common.h.

Referenced by its2_forward_routed_packet(), its2_transmit(), its_add_local_device(), its_add_net_device(), and its_transmit_to_handle().

**uns16 its_device_info::its_device_id**

Definition at line 100 of file its_common.h.

Referenced by its_add_local_device(), its_add_net_device(), its_init(), and its_transmit_to_handle().

---

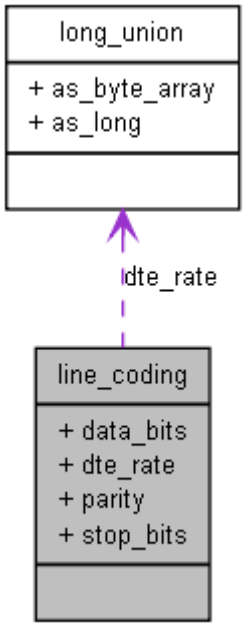
**The documentation for this struct was generated from the following file:**

- its_common.h

---

# line_coding Struct Reference

Collaboration diagram for line_coding:




## Data Fields

- uns8 data_bits
- long_union dte_rate
- uns8 parity
- uns8 stop_bits

---

## Detailed Description

Line coding struction that defines what (should) happen if we were actually USB to RS232 converter

Definition at line 84 of file usb_cdc_class.c.

---

## Field Documentation

### uns8 line_coding::data_bits

Definition at line 88 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

### long_union line_coding::dte_rate

Definition at line 85 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

### uns8 line_coding::parity

Definition at line 87 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

### uns8 line_coding::stop_bits

Definition at line 86 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

---

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

- usb_cdc_class.c

---

# local_address Struct Reference

`#include <its_common.h>`

## Data Fields

- uns16 pan_id
- uns16 short_address

---

## Detailed Description

Local address definition

Definition at line 81 of file its_common.h.

---

## Field Documentation

### uns16 local_address::pan_id

Definition at line 82 of file its_common.h.

Referenced by its2_transmit(), and its_add_local_device().

### uns16 local_address::short_address

Definition at line 83 of file its_common.h.

Referenced by its2_transmit(), its_add_local_device(), and its_transmit_to_handle().

---

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

- its_common.h

---

# long_union Union Reference

## Data Fields

- uns8 as_byte_array [4]
- long as_long

---

## Detailed Description

Definition at line 76 of file usb_cdc_class.c.

---

## Field Documentation

### uns8 long_union::as_byte_array[4]

Definition at line 78 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

### long long_union::as_long

Definition at line 77 of file usb_cdc_class.c.

Referenced by usb_cdc_setup(), and usb_handle_class_ctrl_write_callback().

---

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

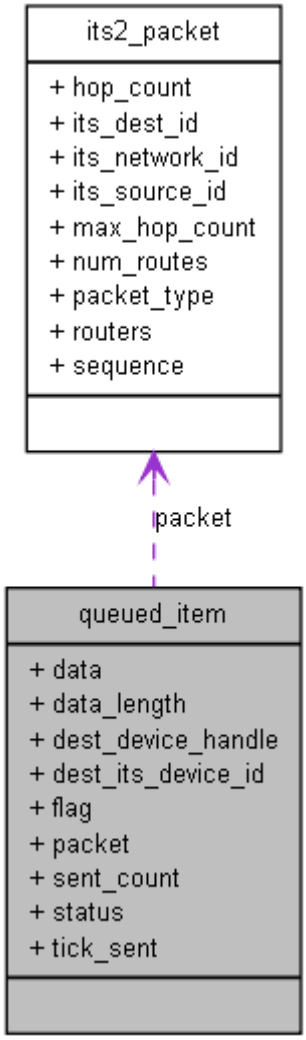
- usb_cdc_class.c

---

## queued_item Struct Reference

`#include <its_mode2.h>`

Collaboration diagram for queued_item:




### Data Fields

- uns8 * data
- uns8 data_length
- uns8 dest_device_handle
- uns16 dest_its_device_id
- uns8 flag

- its2_packet packet
- uns8 sent_count
- uns8 status
- uns16 tick_sent

---

## Detailed Description

Definition at line 132 of file its_mode2.h.

---

## Field Documentation

### uns8* queued_item::data

Definition at line 134 of file its_mode2.h.

Referenced by its2_delete_item_from_queue(), its2_forward_routed_packet(), its2_rebroadcast_net_discover_req(), its2_router_queue_packet(), and its2_transmit().

### uns8 queued_item::data_length

Definition at line 135 of file its_mode2.h.

Referenced by its2_delete_item_from_queue(), its2_forward_routed_packet(), its2_rebroadcast_net_discover_req(), its2_router_queue_packet(), and its2_transmit().

### uns8 queued_item::dest_device_handle

Definition at line 137 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_rebroadcast_net_discover_req(), and its2_transmit().

### uns16 queued_item::dest_its_device_id

Definition at line 136 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_router_queue_packet(), and its2_transmit().

### uns8 queued_item::flag

Definition at line 141 of file its_mode2.h.

Referenced by its2_delete_item_from_queue(), its2_forward_routed_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_router_init(), its2_router_queue_packet(), and its2_transmit().

### its2_packet queued_item::packet

Definition at line 133 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_router_queue_packet(), and its2_transmit().

**uns8 queued_item::sent_count**

Definition at line 139 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_router_queue_packet(), and its2_transmit().

**uns8 queued_item::status**

Definition at line 140 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), and its2_transmit().

**uns16 queued_item::tick_sent**

Definition at line 138 of file its_mode2.h.

Referenced by its2_print_queue(), its2_process_tx_queue(), and its2_transmit().

---

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

- its_mode2.h

---

# remote_address Struct Reference

`#include <its_common.h>`

## Data Fields

- uns16 prior_device_id
- uns16 remote_indicator

---

## Detailed Description

Remote address definition

Definition at line 87 of file its_common.h.

---

## Field Documentation

**uns16 remote_address::prior_device_id**

Definition at line 89 of file its_common.h.

Referenced by its_add_net_device().

**uns16 remote_address::remote_indicator**

Definition at line 88 of file its_common.h.

Referenced by its2_forward_routed_packet(), and its_add_net_device().

---

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

- its_common.h

---

# rf_config Struct Reference

`#include <pic_rf_2401a.h>`

## Data Fields

- uns8 address_ch1 [5]
- uns8 address_ch2 [5]
- uns8 address_width
- uns8 channel
- uns8 crystal
- uns8 options
- uns8 output_power
- uns8 payload_width_ch1
- uns8 payload_width_ch2

---

## Detailed Description

RF configuration structure

Definition at line 80 of file pic_rf_2401a.h.

---

## Field Documentation

**uns8 rf_config::address_ch1**

Address of channel 1

Definition at line 84 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::address_ch2**

Address of channel 2

Definition at line 83 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::address_width**

Address width in bits

Definition at line 85 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::channel**

Channel to operate on, 7 bits valid

Definition at line 88 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::crystal**

Crystal frequency look up, 3 bits valid

Definition at line 86 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::options**

Options (see option bits)

Definition at line 89 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::output_power**

Output power, 2 bits valid

Definition at line 87 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::payload_width_ch1**

Payload width of channel 1 in bits

Definition at line 82 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

**uns8 rf_config::payload_width_ch2**

Payload width of channel 2 in bits

Definition at line 81 of file pic_rf_2401a.h.

Referenced by pic_rf_init().

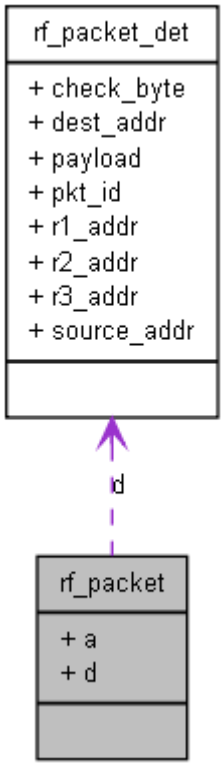
---

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

- pic_rf_2401a.h
- pic_rf_24l01.h

## rf_packet Union Reference

Collaboration diagram for rf_packet:




### Data Fields

- char a [PKT_PACKET_SIZE]
- rf_packet_det d

---

### Detailed Description

Definition at line 50 of file pic_packet.c.

---

### Field Documentation

#### char rf_packet::a[PKT_PACKET_SIZE]

Definition at line 52 of file pic_packet.c.

Referenced by pkt_calc_check_byte(), pkt_check_check_byte(), and pkt_send_packet().

rf_packet_det rf_packet::d

Definition at line 51 of file pic_packet.c.

Referenced by pkt_calc_check_byte(), pkt_check_check_byte(), pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_send_payload().

---

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

- pic_packet.c

---

# rf_packet_det Struct Reference

```
#include <pic_packet.h>
```

## Data Fields

- uns8 check_byte
- uns16 dest_addr
- uns8 payload [PKT_PAYLOAD_SIZE]
- uns16 pkt_id
- uns16 r1_addr
- uns16 r2_addr
- uns16 r3_addr
- uns16 source_addr

---

## Detailed Description

Internal packet structure. You shouldn't need to worry about this generally

Definition at line 190 of file pic_packet.h.

---

## Field Documentation

### uns8 rf_packet_det::check_byte

Definition at line 198 of file pic_packet.h.

Referenced by pkt_calc_check_byte(), and pkt_check_check_byte().

### uns16 rf_packet_det::dest_addr

Definition at line 193 of file pic_packet.h.

Referenced by pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_send_payload().

### uns8 rf_packet_det::payload[PKT_PAYLOAD_SIZE]

Definition at line 197 of file pic_packet.h.

Referenced by pkt_print_packet(), pkt_process_tx_queue(), and pkt_send_payload().

**uns16 rf_packet_det::pkt_id**

Definition at line 192 of file pic_packet.h.

Referenced by pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_send_payload().

**uns16 rf_packet_det::r1_addr**

Definition at line 194 of file pic_packet.h.

Referenced by pkt_print_packet(), pkt_process_tx_queue(), and pkt_send_payload().

**uns16 rf_packet_det::r2_addr**

Definition at line 195 of file pic_packet.h.

Referenced by pkt_print_packet(), and pkt_send_payload().

**uns16 rf_packet_det::r3_addr**

Definition at line 196 of file pic_packet.h.

Referenced by pkt_print_packet(), and pkt_send_payload().

**uns16 rf_packet_det::source_addr**

Definition at line 191 of file pic_packet.h.

Referenced by pkt_print_packet(), and pkt_send_payload().

---

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

- pic_packet.h

---

# seen_packet Struct Reference

```
#include <its_mode2.h>
```

## Data Fields

- uns16 its_source_id
- uns16 pkt_id
- uns8 sequence
- uns16 source_addr

## Detailed Description

Definition at line 144 of file its_mode2.h.

## Field Documentation

**uns16 seen_packet::its_source_id**

Definition at line 146 of file its_mode2.h.

Referenced by its2_router_init().

**uns16 seen_packet::pkt_id**

Definition at line 66 of file pic_packet.c.

Referenced by pkt_process_rf_data().

**uns8 seen_packet::sequence**

Definition at line 145 of file its_mode2.h.

**uns16 seen_packet::source_addr**

Definition at line 67 of file pic_packet.c.

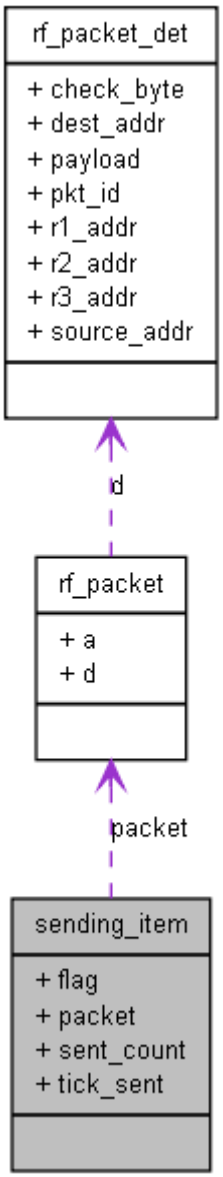
Referenced by pkt_init(), and pkt_process_rf_data().

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

- its_mode2.h
- pic_packet.c

# sending_item Struct Reference

Collaboration diagram for sending_item:




## Data Fields

- uns8 flag
- rf_packet packet
- uns8 sent_count
- uns16 tick_sent

---

## Detailed Description

Definition at line 57 of file pic_packet.c.

---

## Field Documentation

### uns8 sending_item::flag

Definition at line 61 of file pic_packet.c.

Referenced by pkt_init(), pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_queue_packet().

### rf_packet sending_item::packet

Definition at line 58 of file pic_packet.c.

Referenced by pkt_process_rf_data(), and pkt_process_tx_queue().

### uns8 sending_item::sent_count

Definition at line 60 of file pic_packet.c.

Referenced by pkt_process_tx_queue(), and pkt_queue_packet().

### uns16 sending_item::tick_sent

Definition at line 59 of file pic_packet.c.

Referenced by pkt_process_tx_queue().

---

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

- pic_packet.c

---

# setup_data_packet Struct Reference

`#include <pic_usb.h>`

## Data Fields

- uns8 bmRequestType
- uns8 bRequest
- uns16 wIndex
- uns16 wLength
- uns16 wValue

---

## Detailed Description

Describes a set up data packet, part of the control transfer

Definition at line 166 of file pic_usb.h.

## Field Documentation

### uns8 setup_data_packet::bmRequestType

Definition at line 167 of file pic_usb.h.

Referenced by usb_handle_transaction().

### uns8 setup_data_packet::bRequest

Definition at line 167 of file pic_usb.h.

Referenced by usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_standard_request(), and usb_handle_transaction().

### uns16 setup_data_packet::wIndex

Definition at line 169 of file pic_usb.h.

Referenced by usb_handle_transaction().

### uns16 setup_data_packet::wLength

Definition at line 169 of file pic_usb.h.

Referenced by usb_handle_class_request_callback(), usb_handle_standard_request(), and usb_handle_transaction().

### uns16 setup_data_packet::wValue

Definition at line 169 of file pic_usb.h.

Referenced by usb_handle_class_request_callback(), usb_handle_standard_request(), and usb_handle_transaction().

---

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

- pic_usb.h

---

# wpan_address Struct Reference

```
#include <wpan.h>
```

## Data Fields

- uns8 dest_address_type

- uns8 dest_ea [8]
- uns16 dest_pan_id
- uns16 dest_sa
- uns8 source_address_type
- uns8 source_ea [8]
- uns16 source_pan_id
- uns16 source_sa

---

## Detailed Description

Definition at line 84 of file wpan.h.

---

## Field Documentation

### uns8 wpan_address::dest_address_type

Definition at line 90 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

### uns8 wpan_address::dest_ea[8]

Definition at line 93 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

### uns16 wpan_address::dest_pan_id

Definition at line 91 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

### uns16 wpan_address::dest_sa

Definition at line 92 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

### uns8 wpan_address::source_address_type

Definition at line 85 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

### uns8 wpan_address::source_ea[8]

Definition at line 88 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

**uns16 wpan_address::source_pan_id**

Definition at line 86 of file wpan.h.

Referenced by wpan_data_received_callback(), wpan_handle_receive(), and wpan_print_address().

**uns16 wpan_address::source_sa**

Definition at line 87 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

---

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

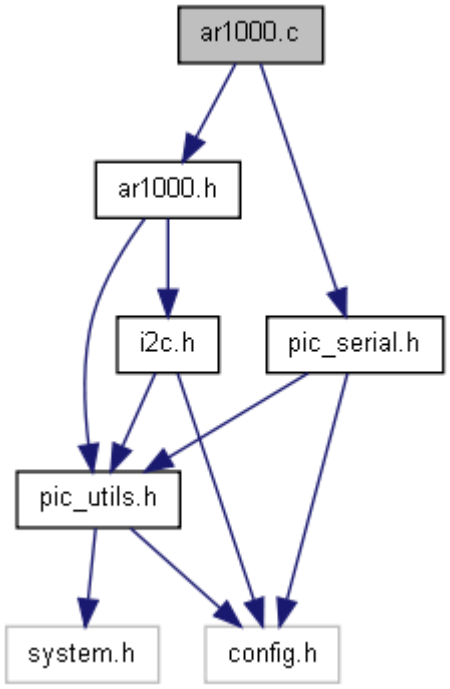
- wpan.h

# File Documentation

## ar1000.c File Reference

```
#include "ar1000.h"
#include "pic_serial.h"
```

Include dependency graph for ar1000.c:




### Functions

- uns8 ar1000_get_register (uns8 reg)
- void ar1000_init ()

- uns16 ar1000_read_register (uns8 reg)
- void ar1000_read_registers ()
- void ar1000_seek (uns16 frequency, bit seek_up)
- void ar1000_seek2 ()
- void ar1000_seek_more ()
- void ar1000_set_register (uns8 reg, uns8 data)
- void ar1000_set_seek_threshold (uns8 new_seek_threshold)
- void ar1000_set_volume (uns8 volume)
- void ar1000_setup_io ()
- *Setup AR1000 ports and pins.* void ar1000_test ()
- void ar1000_tune (uns16 frequency)
- void ar1000_write_register (uns8 reg, uns16 data)
- void ar1000_write_registers ()

## Variables

- uns8 ar1000_seek_threshold
- uns16 regs [18]
- rom uns8 vol_lookup []

---

## Function Documentation

### uns8 ar1000_get_register (uns8 *reg*)

Definition at line 55 of file ar1000.c.

References regs.

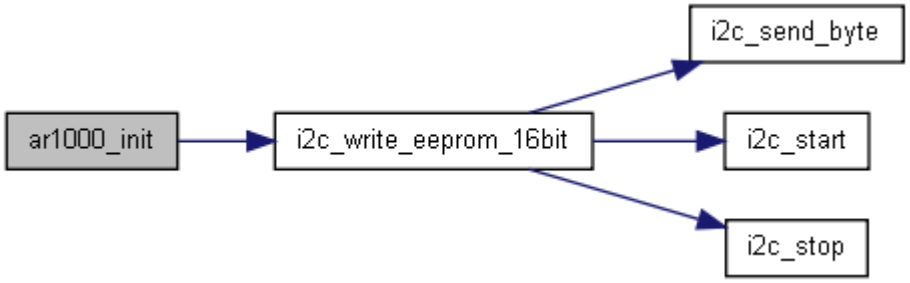
### void ar1000_init ()

Definition at line 224 of file ar1000.c.

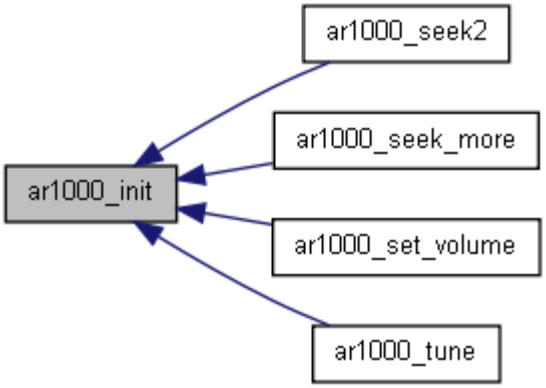
References AR1000_DEV_ADDR, ar1000_seek_threshold, i2c_write_eeprom_16bit(), regs, and uns8.

Referenced by ar1000_seek2(), ar1000_seek_more(), ar1000_set_volume(), and ar1000_tune().

Here is the call graph for this function:




Here is the caller graph for this function:

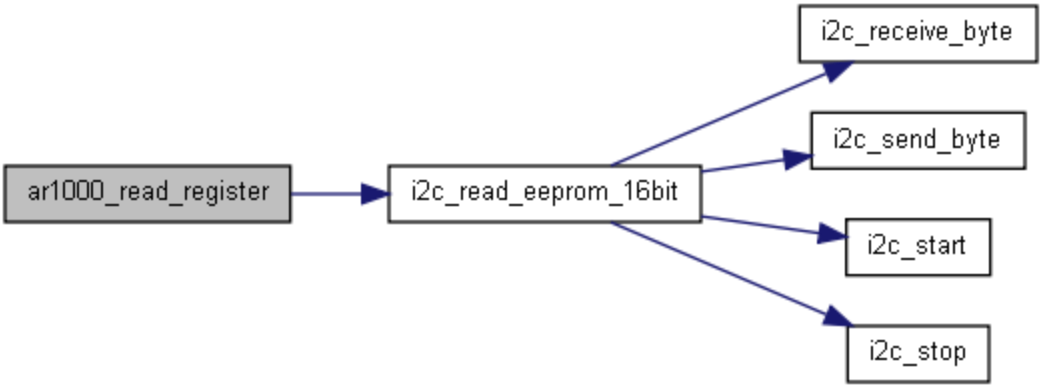



**uns16 ar1000_read_register (uns8 *reg*)**

Definition at line 70 of file ar1000.c.

References AR1000_DEV_ADDR, and i2c_read_eeprom_16bit().

Here is the call graph for this function:




**void ar1000_read_registers ()**

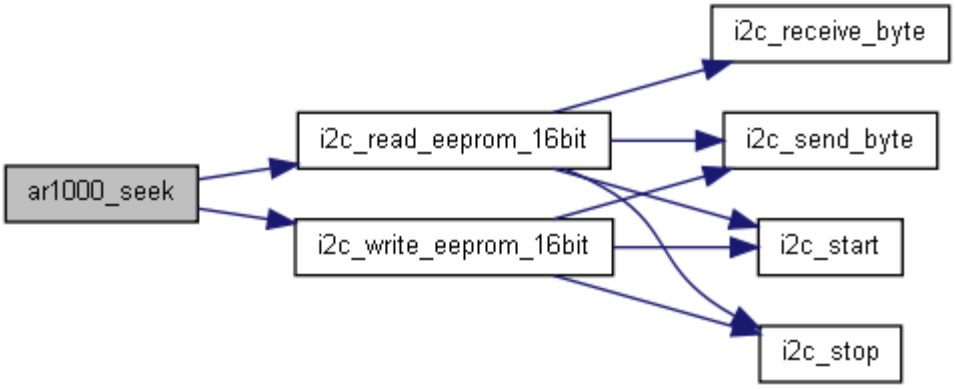
Definition at line 66 of file ar1000.c.

**void ar1000_seek (uns16 *frequency*, bit *seek_up*)**

Definition at line 83 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_seek_threshold, i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, R2_TUNE_ENABLE, R3_BAND_1, R3_SEEK_CHANNEL_SPACING, R3_SEEK_ENABLE, R3_SEEK_UP, and uns16.

Here is the call graph for this function:

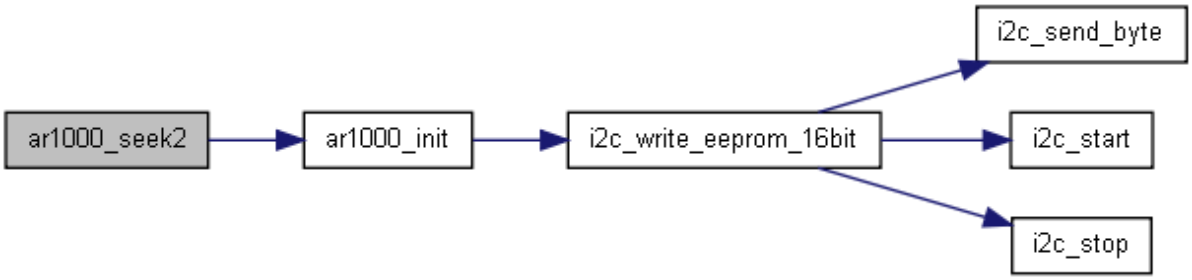



**void ar1000_seek2 ()**

Definition at line 234 of file ar1000.c.

References ar1000_init(), and regs.

Here is the call graph for this function:

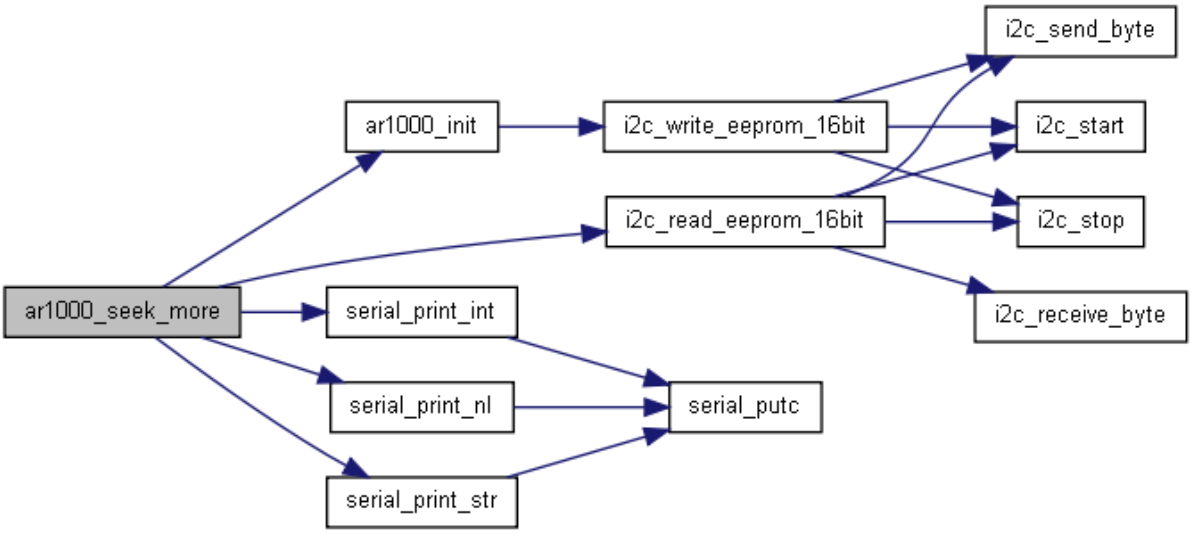



**void ar1000_seek_more ()**

Definition at line 273 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_init(), AR1000_STATUS, channel, i2c_read_eeprom_16bit(), R2_TUNE_ENABLE, regs, serial_print_int(), serial_print_nl(), serial_print_str(), and uns16.

Here is the call graph for this function:




**void ar1000_set_register (uns8 *reg*, uns8 *data*)**

Definition at line 50 of file ar1000.c.

References regs.

**void ar1000_set_seek_threshold (uns8 *new_seek_threshold*)**

Definition at line 219 of file ar1000.c.

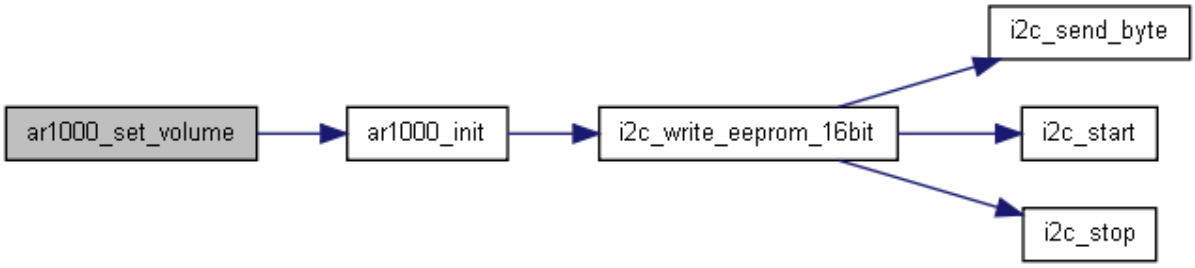
References ar1000_seek_threshold.

**void ar1000_set_volume (uns8 *volume*)**

Definition at line 321 of file ar1000.c.

References ar1000_init(), regs, uns16, uns8, and vol_lookup.

Here is the call graph for this function:




**void ar1000_setup_io ()**

Definition at line 46 of file ar1000.c.

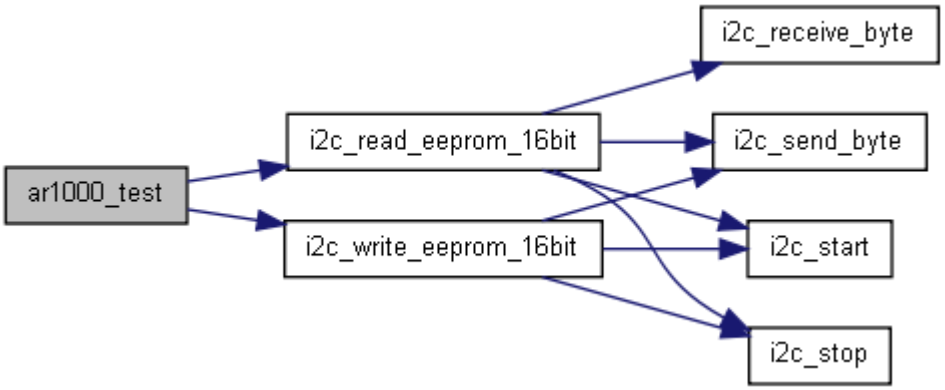
References i2c_setup.

**void ar1000_test ()**

Definition at line 255 of file ar1000.c.

References AR1000_DEV_ADDR, i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, and uns16.

Here is the call graph for this function:

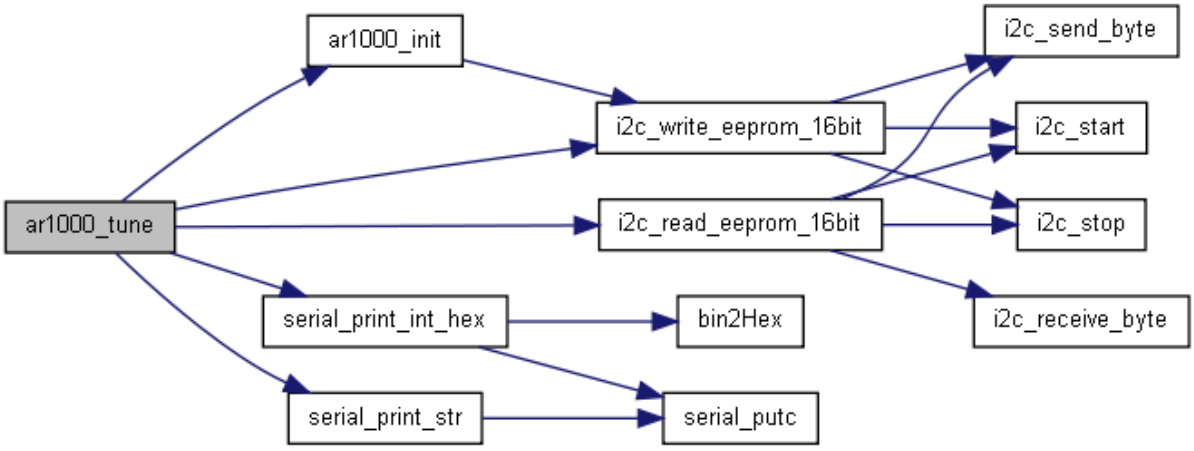



**void ar1000_tune (uns16 *frequency*)**

Definition at line 145 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_init(), i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, R2_TUNE_ENABLE, R3_BAND_1, R3_SEEK_CHANNEL_SPACING, R3_SEEK_ENABLE, regs, serial_print_int_hex(), serial_print_str(), and uns16.

Here is the call graph for this function:

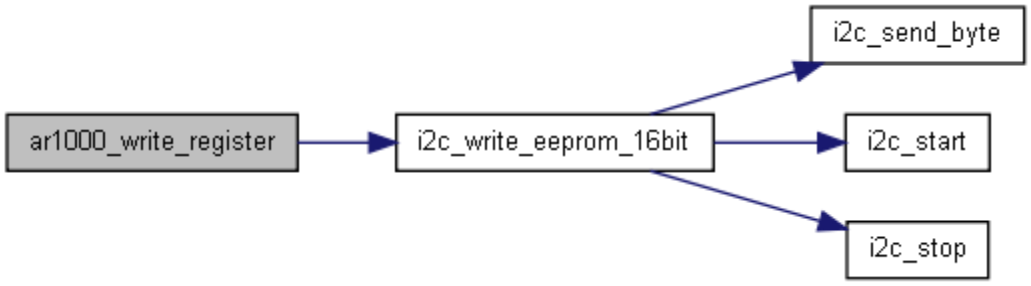



**void ar1000_write_register (uns8 *reg*, uns16 *data*)**

Definition at line 77 of file ar1000.c.

References AR1000_DEV_ADDR, and i2c_write_eeprom_16bit().

Here is the call graph for this function:




### void ar1000_write_registers ()

Definition at line 60 of file ar1000.c.

References uns8.

---

## Variable Documentation

### uns8 ar1000_seek_threshold

Definition at line 38 of file ar1000.c.

Referenced by ar1000_init(), ar1000_seek(), and ar1000_set_seek_threshold().

### uns16 regs[18]

```
Initial value: {
    0xffff,0x5b15,0xF4B9,0x8012,0x0400,0x28aa,0x4400,0x1ee7,
    0x7141,0x007d,0x82ce,0x4f55,0x970c,0xb845,0xfc2d,0x8097,
    0x04a1,0xdf6a}
```

Definition at line 41 of file ar1000.c.

Referenced by ar1000_get_register(), ar1000_init(), ar1000_seek2(), ar1000_seek_more(), ar1000_set_register(), ar1000_set_volume(), and ar1000_tune().

### rom uns8 vol_lookup[]

```
Initial value: {
0x0F,
0xCF,
0xDF,
0xEF,
0xFF,
0xEE,
0xFE,
0xED,
0xFD,
0xFB,
0xFA,
0xF9,
0xF7,
0xE6,
0xF6,
0xE5,
0xF5,
0xE3,
0xF3,
0xF2,
```

```
0xF1,
0xF0}
```

Definition at line 295 of file ar1000.c.

Referenced by ar1000_set_volume().

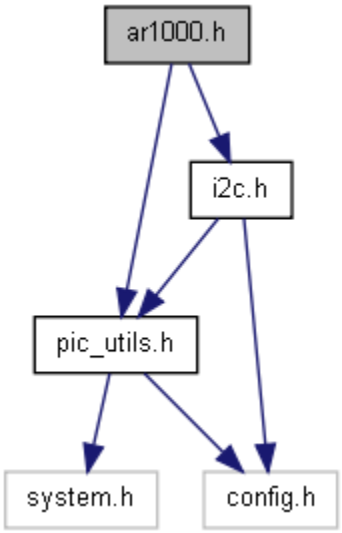
---

# ar1000.h File Reference

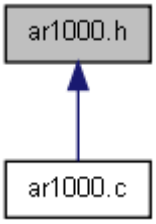
Routines to access the AR1000 FM radio chip.

```
#include "pic_utils.h"
#include "i2c.h"
```

Include dependency graph for ar1000.h:




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




## Defines

- #define AR1000_CHIP_ID 28
- #define AR1000_DEV_ADDR 0b00100000
- #define AR1000_DEV_ID 27
- #define AR1000_R0 0
- #define AR1000_R1 1
- #define AR1000_R10 10
- #define AR1000_R11 11

- #define AR1000_R13 13
- #define AR1000_R14 14
- #define AR1000_R15 15
- #define AR1000_R2 2
- #define AR1000_R3 3
- #define AR1000_RBS 20
- #define AR1000_RDS_1 21
- #define AR1000_RDS_2 22
- #define AR1000_RDS_3 23
- #define AR1000_RDS_4 24
- #define AR1000_RDS_5 25
- #define AR1000_RDS_6 26
- #define AR1000_RSSI 18
- #define ar1000_setup() ar1000_setup_io()
- *Setup AR1000 ports and pins.* #define AR1000_STATUS 19
- #define DEV_ID_MFID_0 0
- #define DEV_ID_MFID_1 1
- #define DEV_ID_MFID_10 10
- #define DEV_ID_MFID_11 11
- #define DEV_ID_MFID_2 2
- #define DEV_ID_MFID_3 3
- #define DEV_ID_MFID_4 4
- #define DEV_ID_MFID_5 5
- #define DEV_ID_MFID_6 6
- #define DEV_ID_MFID_7 7
- #define DEV_ID_MFID_8 8
- #define DEV_ID_MFID_9 9
- #define DEV_ID_VERSION_0 12
- #define DEV_ID_VERSION_1 13
- #define DEV_ID_VERSION_2 14
- #define DEV_ID_VERSION_3 15
- #define R0_ENABLE 0
- #define R0_INT_OSC_EN 15
- #define R10_SEEK_WRAP_ENABLE 3
- #define R11_AFC_HIGH_SIDE_b1 2
- #define R11_AFC_HIGH_SIDE_b2 0
- #define R11_AFC_INJECTION_CONTROL 15
- #define R11_HILO_SIDE 15
- #define R13_GPIO1_0 0
- #define R13_GPIO1_1 1
- #define R13_GPIO2_0 2
- #define R13_GPIO2_1 3
- #define R13_GPIO3_0 4
- #define R13_GPIO3_1 5
- #define R14_VOL2_0 12
- #define R14_VOL2_1 13
- #define R14_VOL2_2 14
- #define R14_VOL2_3 15
- #define R15_RDS_CTRL 0
- #define R15_RDS_MECC_0 3
- #define R15_RDS_MECC_1 4
- #define R15_RDS_STA_EN 5
- #define R1_DEEMP_SETTING 4

- #define R1_FORCE_MONO 3
- #define R1_HARD_MUTE_ENABLE 1
- #define R1_RDS_ENABLE 13
- #define R1_RDS_INT_ENABLE 6
- #define R1_SOFT_MUTE_ENABLE 2
- #define R1_STC_INT_ENABLE 5
- #define R2_CHAN_0 0
- #define R2_CHAN_1 1
- #define R2_CHAN_2 2
- #define R2_CHAN_3 3
- #define R2_CHAN_4 4
- #define R2_CHAN_5 5
- #define R2_CHAN_6 6
- #define R2_CHAN_7 7
- #define R2_CHAN_8 8
- #define R2_TUNE_ENABLE 9
- #define R3_BAND_0 11
- #define R3_BAND_1 12
- #define R3_SEEK_CHANNEL_SPACING 13
- #define R3_SEEK_ENABLE 14
- #define R3_SEEK_UP 15
- #define R3_SEEKTH_0 0
- #define R3_SEEKTH_1 1
- #define R3_SEEKTH_2 2
- #define R3_SEEKTH_3 3
- #define R3_SEEKTH_4 4
- #define R3_SEEKTH_5 5
- #define R3_SEEKTH_6 6
- #define R3_VOL_0 7
- #define R3_VOL_1 8
- #define R3_VOL_2 9
- #define R3_VOL_3 10
- #define STATUS_BIT_2 2
- #define STATUS_CHAN_0 7
- #define STATUS_CHAN_1 8
- #define STATUS_CHAN_2 9
- #define STATUS_CHAN_3 10
- #define STATUS_CHAN_4 11
- #define STATUS_CHAN_5 12
- #define STATUS_CHAN_6 13
- #define STATUS_CHAN_7 14
- #define STATUS_CHAN_8 15
- #define STATUS_RDS_DATA_READY 6
- #define STATUS_SEEK_FAIL 4
- #define STATUS_SEEK_TUNE_COMPLETE 5
- #define STATUS_STEREO 3

## Functions

- uns8 ar1000_get_register (uns8 reg)
- void ar1000_init ()
- uns16 ar1000_read_register (uns8 reg)
- void ar1000_read_registers ()

- void ar1000_seek (uns16 frequency, bit seek_up)
- void ar1000_seek2 ()
- void ar1000_seek_more ()
- void ar1000_set_register (uns8 reg, uns8 data)
- void ar1000_set_seek_threshold (uns8 new_seek_threshold)
- void ar1000_set_volume (uns8 volume)
- void ar1000_setup_io ()
- *Setup AR1000 ports and pins.* void ar1000_test ()
- void ar1000_tune (uns16 frequency)
- void ar1000_write_register (uns8 reg, uns16 data)
- void ar1000_write_registers ()

---

## Detailed Description

ITS networking routines - common to mode 1 and 2.

Definition in file ar1000.h.

---

## Define Documentation

### #define AR1000_CHIP_ID 28

Chip id - 0x1000 for RDS version, 0x1010 for non-rds version

Definition at line 325 of file ar1000.h.

### #define AR1000_DEV_ADDR 0b00100000

Definition at line 46 of file ar1000.h.

Referenced by ar1000_init(), ar1000_read_register(), ar1000_seek(), ar1000_seek_more(), ar1000_test(), ar1000_tune(), and ar1000_write_register().

### #define AR1000_DEV_ID 27

Definition at line 286 of file ar1000.h.

### #define AR1000_R0 0

Definition at line 52 of file ar1000.h.

### #define AR1000_R1 1

Definition at line 64 of file ar1000.h.

### #define AR1000_R10 10

Definition at line 138 of file ar1000.h.

**#define AR1000_R11 11**

Definition at line 145 of file ar1000.h.

**#define AR1000_R13 13**

GPIO controls in this register

Definition at line 160 of file ar1000.h.

**#define AR1000_R14 14**

Definition at line 177 of file ar1000.h.

**#define AR1000_R15 15**

Definition at line 190 of file ar1000.h.

**#define AR1000_R2 2**

Definition at line 85 of file ar1000.h.

**#define AR1000_R3 3**

Definition at line 106 of file ar1000.h.

**#define AR1000_RBS 20**

Definition at line 258 of file ar1000.h.

**#define AR1000_RDS_1 21**

Definition at line 262 of file ar1000.h.

**#define AR1000_RDS_2 22**

Definition at line 266 of file ar1000.h.

**#define AR1000_RDS_3 23**

Definition at line 270 of file ar1000.h.

**#define AR1000_RDS_4 24**

Definition at line 274 of file ar1000.h.

**#define AR1000_RDS_5 25**

Definition at line 278 of file ar1000.h.

**#define AR1000_RDS_6 26**

Definition at line 282 of file ar1000.h.

**#define AR1000_RSSI 18**

Definition at line 211 of file ar1000.h.

**#define ar1000_setup() ar1000_setup_io()**

define so as not to break existing code with new naming standard

Definition at line 334 of file ar1000.h.

**#define AR1000_STATUS 19**

Definition at line 220 of file ar1000.h.

Referenced by ar1000_seek_more().

**#define DEV_ID_MFID_0 0**

MFID (12 bits 5B1) bit 0

Definition at line 320 of file ar1000.h.

**#define DEV_ID_MFID_1 1**

MFID (12 bits 5B1) bit 1

Definition at line 318 of file ar1000.h.

**#define DEV_ID_MFID_10 10**

MFID (12 bits 5B1) bit 10

Definition at line 300 of file ar1000.h.

**#define DEV_ID_MFID_11 11**

MFID (12 bits 5B1) bit 11

Definition at line 298 of file ar1000.h.

**#define DEV_ID_MFID_2 2**

MFID (12 bits 5B1) bit 2

Definition at line 316 of file ar1000.h.

**#define DEV_ID_MFID_3 3**

MFID (12 bits 5B1) bit 3

Definition at line 314 of file ar1000.h.

**#define DEV_ID_MFID_4 4**

MFID (12 bits 5B1) bit 4

Definition at line 312 of file ar1000.h.

**#define DEV_ID_MFID_5 5**

MFID (12 bits 5B1) bit 5

Definition at line 310 of file ar1000.h.

**#define DEV_ID_MFID_6 6**

MFID (12 bits 5B1) bit 6

Definition at line 308 of file ar1000.h.

**#define DEV_ID_MFID_7 7**

MFID (12 bits 5B1) bit 7

Definition at line 306 of file ar1000.h.

**#define DEV_ID_MFID_8 8**

MFID (12 bits 5B1) bit 8

Definition at line 304 of file ar1000.h.

**#define DEV_ID_MFID_9 9**

MFID (12 bits 5B1) bit 9

Definition at line 302 of file ar1000.h.

**#define DEV_ID_VERSION_0 12**

Version of FM radio bit 0

Definition at line 295 of file ar1000.h.

**#define DEV_ID_VERSION_1 13**

Version of FM radio bit 1

Definition at line 293 of file ar1000.h.

**#define DEV_ID_VERSION_2 14**

Version of FM radio bit 2

Definition at line 291 of file ar1000.h.

**#define DEV_ID_VERSION_3 15**

Version of FM radio bit 3

Definition at line 289 of file ar1000.h.

**#define R0_ENABLE 0**

Definition at line 59 of file ar1000.h.

**#define R0_INT_OSC_EN 15**

Definition at line 55 of file ar1000.h.

**#define R10_SEEK_WRAP_ENABLE 3**

Seek wrap enable

Definition at line 141 of file ar1000.h.

**#define R11_AFC_HIGH_SIDE_b1 2**

High side control bits

Definition at line 151 of file ar1000.h.

**#define R11_AFC_HIGH_SIDE_b2 0**

Definition at line 152 of file ar1000.h.

**#define R11_AFC_INJECTION_CONTROL 15**

AFC injection control

Definition at line 148 of file ar1000.h.

**#define R11_HILO_SIDE 15**

Definition at line 149 of file ar1000.h.

**#define R13_GPIO1_0 0**

Definition at line 172 of file ar1000.h.

**#define R13_GPIO1_1 1**

GPIO1- 00-Disable 01-Reserved 10-Output logic 0 11-Output logic 1

Definition at line 171 of file ar1000.h.

**#define R13_GPIO2_0 2**

Definition at line 168 of file ar1000.h.

**#define R13_GPIO2_1 3**

GPIO2- 00-Disable 01-STC or RDS interrupt 10-Output logic 0 11-Output logic 1

Definition at line 167 of file ar1000.h.

**#define R13_GPIO3_0 4**

Definition at line 164 of file ar1000.h.

**#define R13_GPIO3_1 5**

GPIO3- 00-Disable 01-Stereo indication 10-Output logic 0 11-Output logic 1

Definition at line 163 of file ar1000.h.

**#define R14_VOL2_0 12**

Second volume settings (bit 0)

Definition at line 186 of file ar1000.h.

**#define R14_VOL2_1 13**

Second volume settings (bit 1)

Definition at line 184 of file ar1000.h.

**#define R14_VOL2_2 14**

Second volume settings (bit 2)

Definition at line 182 of file ar1000.h.

**#define R14_VOL2_3 15**

Second volume settings (bit 3)

Definition at line 180 of file ar1000.h.

**#define R15_RDS_CTRL 0**

RDS Control mode 0=block mode (RDSR asserted after 4 blocks received, data in blocks may not be correct. RBS1-4 will present the status of each block) 1=group mode (RDSR is asserted after 4 blocks received without any error)

Definition at line 202 of file ar1000.h.

**#define R15_RDS_MECC_0 3**

Definition at line 197 of file ar1000.h.

**#define R15_RDS_MECC_1 4**

RDS Error correction 0x=disable erorr correction 10=2 bit error correction 11=5 bit error correction

Definition at line 196 of file ar1000.h.

**#define R15_RDS_STA_EN 5**

RDS statistic data enable signal

Definition at line 193 of file ar1000.h.

**#define R1_DEEMP_SETTING 4**

De-emphasis 1=75us 0=50us

Definition at line 73 of file ar1000.h.

**#define R1_FORCE_MONO 3**

Force mono

Definition at line 75 of file ar1000.h.

**#define R1_HARD_MUTE_ENABLE 1**

Hard mute enable

Definition at line 79 of file ar1000.h.

Referenced by ar1000_seek(), ar1000_test(), and ar1000_tune().

**#define R1_RDS_ENABLE 13**

Enable RDS signal

Definition at line 67 of file ar1000.h.

**#define R1_RDS_INT_ENABLE 6**

Enable RDS interrupt

Definition at line 69 of file ar1000.h.

**#define R1_SOFT_MUTE_ENABLE 2**

Soft mute enable

Definition at line 77 of file ar1000.h.

**#define R1_STC_INT_ENABLE 5**

Seek Tune Complete (STC) interrupt enable

Definition at line 71 of file ar1000.h.

**#define R2_CHAN_0 0**

Definition at line 100 of file ar1000.h.

**#define R2_CHAN_1 1**

Definition at line 99 of file ar1000.h.

**#define R2_CHAN_2 2**

Definition at line 98 of file ar1000.h.

**#define R2_CHAN_3 3**

Definition at line 97 of file ar1000.h.

**#define R2_CHAN_4 4**

Definition at line 96 of file ar1000.h.

**#define R2_CHAN_5 5**

Definition at line 95 of file ar1000.h.

**#define R2_CHAN_6 6**

Definition at line 94 of file ar1000.h.

**#define R2_CHAN_7 7**

Definition at line 93 of file ar1000.h.

**#define R2_CHAN_8 8**

Definition at line 92 of file ar1000.h.

**#define R2_TUNE_ENABLE 9**

Tune channel enable

Definition at line 88 of file ar1000.h.

Referenced by ar1000_seek(), ar1000_seek_more(), and ar1000_tune().

**#define R3_BAND_0 11**

Band control 0=Japan narrow band 76Mhz - 90Mhz 1=Japan wide band 76Mhz - 108Mhz

Definition at line 117 of file ar1000.h.

**#define R3_BAND_1 12**

Band control 0=US/EUROPE 1=Japan

Definition at line 115 of file ar1000.h.

Referenced by ar1000_seek(), and ar1000_tune().

**#define R3_SEEK_CHANNEL_SPACING 13**

Seek channel spacing 1=100k 0=200k

Definition at line 113 of file ar1000.h.

Referenced by ar1000_seek(), and ar1000_tune().

**#define R3_SEEK_ENABLE 14**

Seek enable

Definition at line 111 of file ar1000.h.

Referenced by ar1000_seek(), and ar1000_tune().

**#define R3_SEEK_UP 15**

Seek direction 1=up 0=down

Definition at line 109 of file ar1000.h.

Referenced by ar1000_seek().

**#define R3_SEEKTH_0 0**

Definition at line 132 of file ar1000.h.

**#define R3_SEEKTH_1 1**

Definition at line 131 of file ar1000.h.

**#define R3_SEEKTH_2 2**

Definition at line 130 of file ar1000.h.

**#define R3_SEEKTH_3 3**

Definition at line 129 of file ar1000.h.

**#define R3_SEEKTH_4 4**

Definition at line 128 of file ar1000.h.

**#define R3_SEEKTH_5 5**

Definition at line 127 of file ar1000.h.

**#define R3_SEEKTH_6 6**

Definition at line 126 of file ar1000.h.

**#define R3_VOL_0 7**

Definition at line 123 of file ar1000.h.

**#define R3_VOL_1 8**

Definition at line 122 of file ar1000.h.

**#define R3_VOL_2 9**

Definition at line 121 of file ar1000.h.

**#define R3_VOL_3 10**

Definition at line 120 of file ar1000.h.

**#define STATUS_BIT_2 2**

Used but unknown function

Definition at line 254 of file ar1000.h.

**#define STATUS_CHAN_0 7**

Current tuned channel bit 0

Definition at line 239 of file ar1000.h.

**#define STATUS_CHAN_1 8**

Current tuned channel bit 1

Definition at line 237 of file ar1000.h.

**#define STATUS_CHAN_2 9**

Current tuned channel bit 2

Definition at line 235 of file ar1000.h.

**#define STATUS_CHAN_3 10**

Current tuned channel bit 3

Definition at line 233 of file ar1000.h.

**#define STATUS_CHAN_4 11**

Current tuned channel bit 4

Definition at line 231 of file ar1000.h.

**#define STATUS_CHAN_5 12**

Current tuned channel bit 5

Definition at line 229 of file ar1000.h.

**#define STATUS_CHAN_6 13**

Current tuned channel bit 6

Definition at line 227 of file ar1000.h.

**#define STATUS_CHAN_7 14**

Current tuned channel bit 7

Definition at line 225 of file ar1000.h.

**#define STATUS_CHAN_8 15**

Current tuned channel bit 8

Definition at line 223 of file ar1000.h.

**#define STATUS_RDS_DATA_READY 6**

RDS data recevied

Definition at line 242 of file ar1000.h.

**#define STATUS_SEEK_FAIL 4**

Seek has failed

Definition at line 248 of file ar1000.h.

**#define STATUS_SEEK_TUNE_COMPLETE 5**

Seek or Tune has completed

Definition at line 245 of file ar1000.h.

**#define STATUS_STEREO 3**

Stereo flag 1=stereo 0=mono

Definition at line 251 of file ar1000.h.

---

## Function Documentation

**uns8 ar1000_get_register (uns8 *reg*)**

Definition at line 55 of file ar1000.c.

References regs.

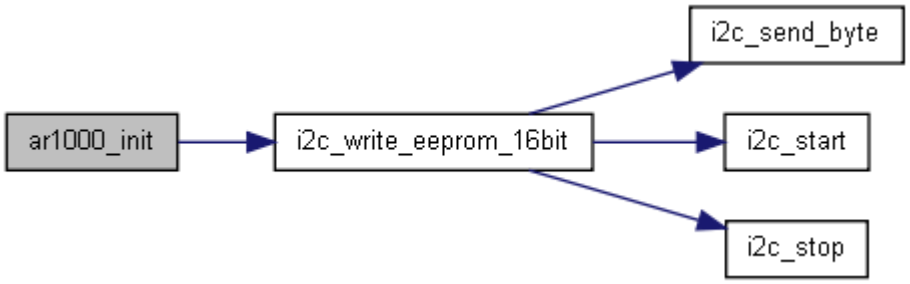
**void ar1000_init ()**

Definition at line 224 of file ar1000.c.

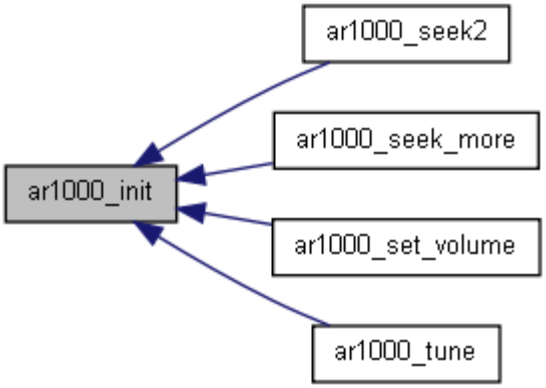
References AR1000_DEV_ADDR, ar1000_seek_threshold, i2c_write_eeprom_16bit(), regs, and uns8.

Referenced by ar1000_seek2(), ar1000_seek_more(), ar1000_set_volume(), and ar1000_tune().

Here is the call graph for this function:




Here is the caller graph for this function:

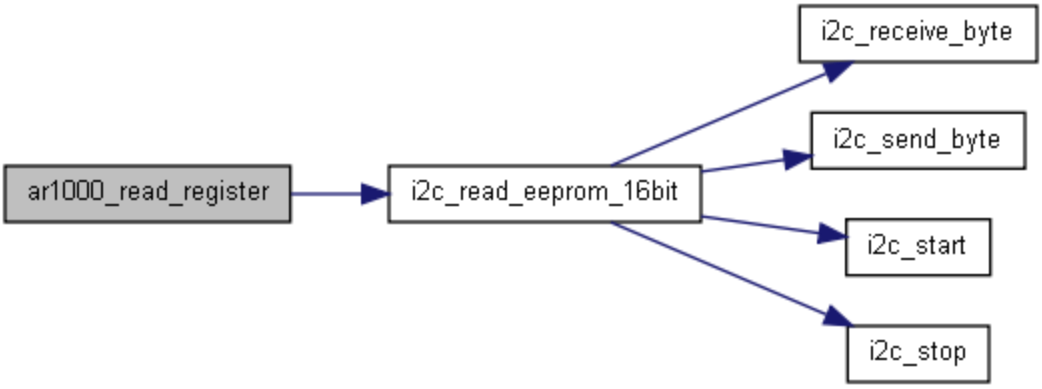



**uns16 ar1000_read_register (uns8 *reg*)**

Definition at line 70 of file ar1000.c.

References AR1000_DEV_ADDR, and i2c_read_eeprom_16bit().

Here is the call graph for this function:




**void ar1000_read_registers ()**

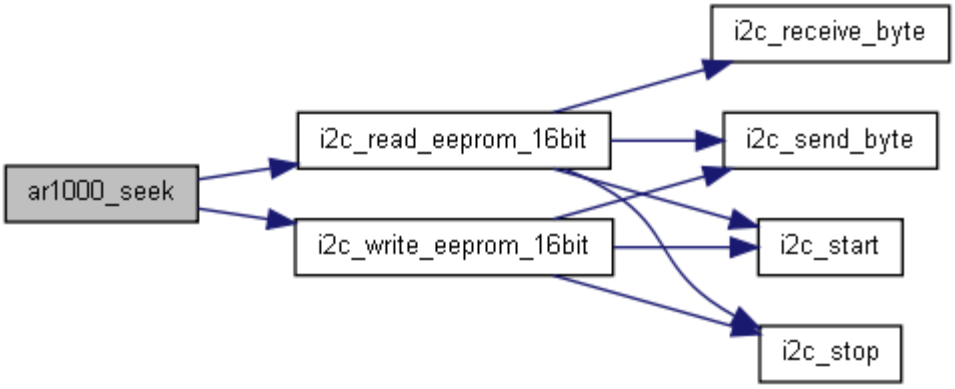
Definition at line 66 of file ar1000.c.

**void ar1000_seek (uns16 *frequency*, bit *seek_up*)**

Definition at line 83 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_seek_threshold, i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, R2_TUNE_ENABLE, R3_BAND_1, R3_SEEK_CHANNEL_SPACING, R3_SEEK_ENABLE, R3_SEEK_UP, and uns16.

Here is the call graph for this function:

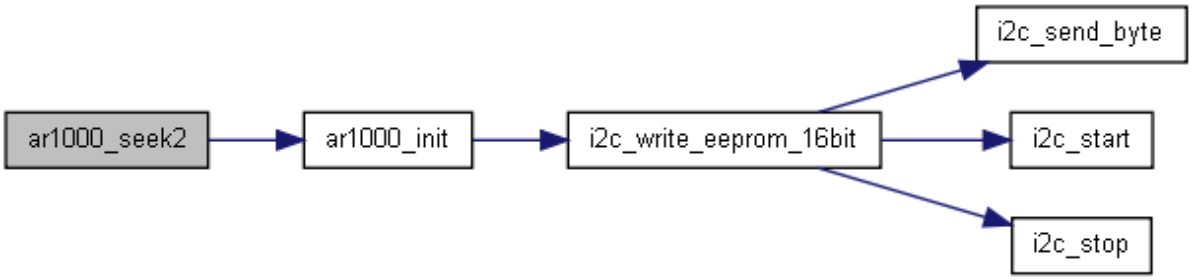



**void ar1000_seek2 ()**

Definition at line 234 of file ar1000.c.

References ar1000_init(), and regs.

Here is the call graph for this function:

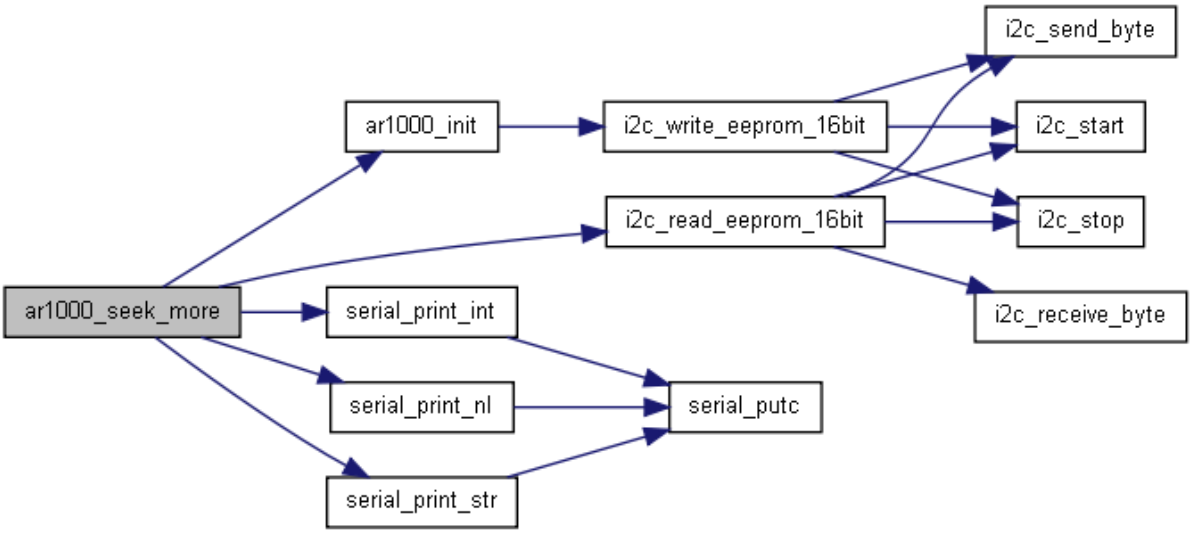



**void ar1000_seek_more ()**

Definition at line 273 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_init(), AR1000_STATUS, channel, i2c_read_eeprom_16bit(), R2_TUNE_ENABLE, regs, serial_print_int(), serial_print_nl(), serial_print_str(), and uns16.

Here is the call graph for this function:




**void ar1000_set_register (uns8 *reg*, uns8 *data*)**

Definition at line 50 of file ar1000.c.

References regs.

**void ar1000_set_seek_threshold (uns8 *new_seek_threshold*)**

Definition at line 219 of file ar1000.c.

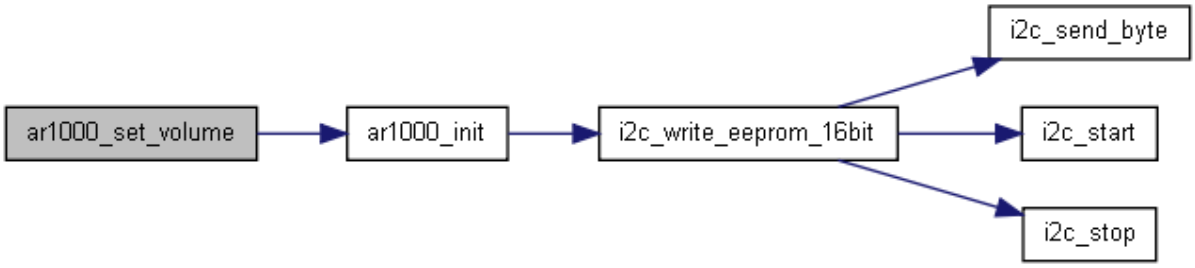
References ar1000_seek_threshold.

**void ar1000_set_volume (uns8 *volume*)**

Definition at line 321 of file ar1000.c.

References ar1000_init(), regs, uns16, uns8, and vol_lookup.

Here is the call graph for this function:




**void ar1000_setup_io ()**

Definition at line 46 of file ar1000.c.

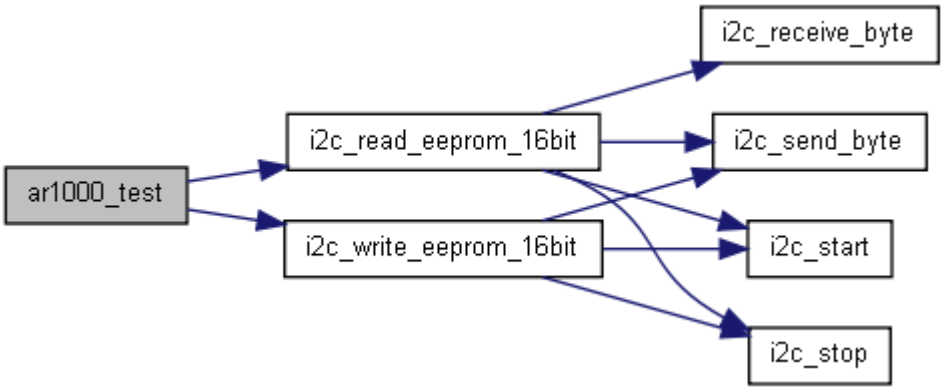
References i2c_setup.

**void ar1000_test ()**

Definition at line 255 of file ar1000.c.

References AR1000_DEV_ADDR, i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, and uns16.

Here is the call graph for this function:

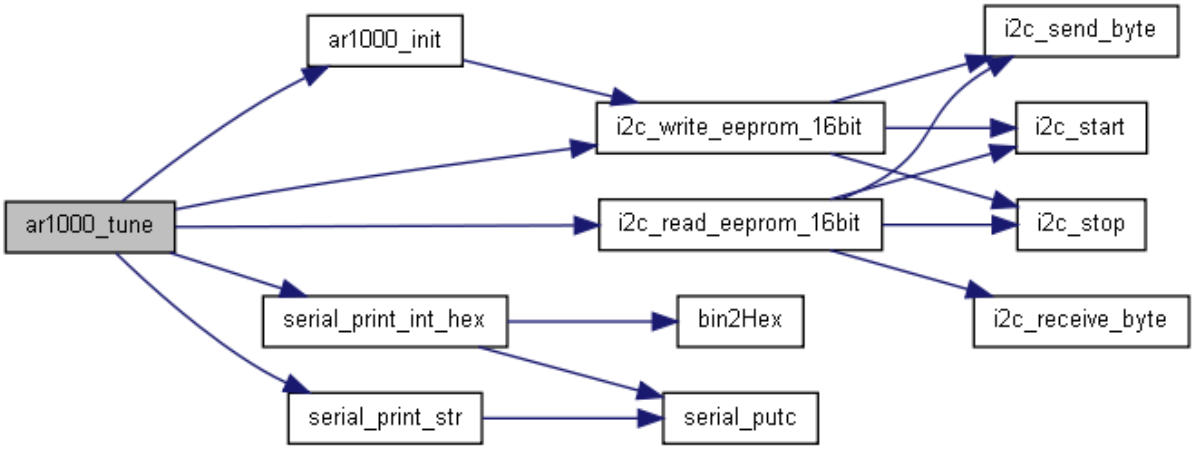



**void ar1000_tune (uns16 *frequency*)**

Definition at line 145 of file ar1000.c.

References AR1000_DEV_ADDR, ar1000_init(), i2c_read_eeprom_16bit(), i2c_write_eeprom_16bit(), R1_HARD_MUTE_ENABLE, R2_TUNE_ENABLE, R3_BAND_1, R3_SEEK_CHANNEL_SPACING, R3_SEEK_ENABLE, regs, serial_print_int_hex(), serial_print_str(), and uns16.

Here is the call graph for this function:

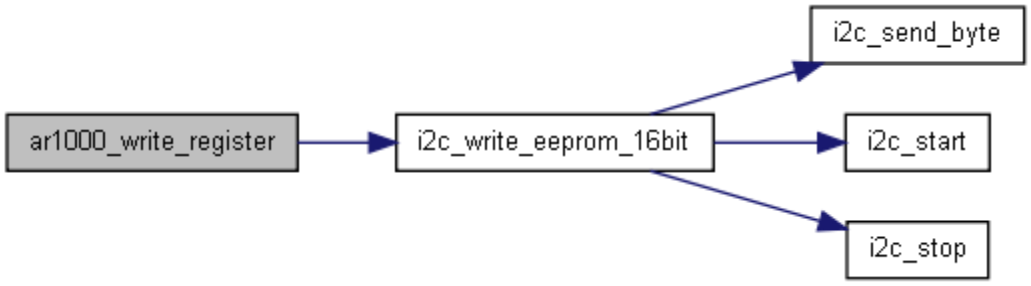



**void ar1000_write_register (uns8 *reg*, uns16 *data*)**

Definition at line 77 of file ar1000.c.

References AR1000_DEV_ADDR, and i2c_write_eeprom_16bit().

Here is the call graph for this function:




**void ar1000_write_registers ()**

Definition at line 60 of file ar1000.c.

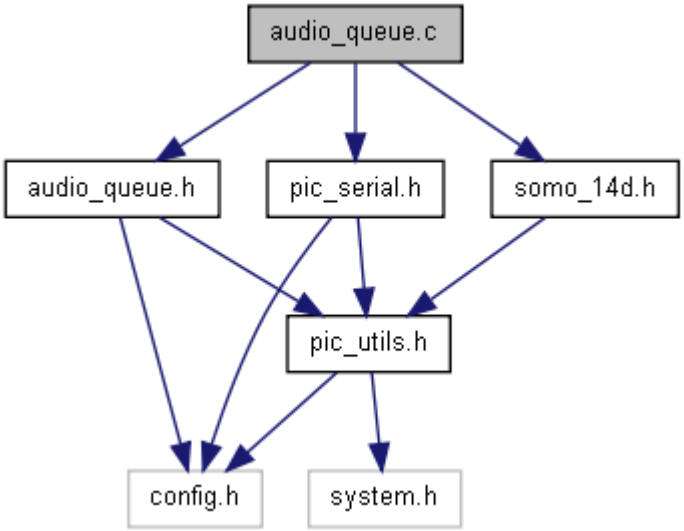
References uns8.

---

# audio_queue.c File Reference

```
#include "audio_queue.h"
#include "somo_14d.h"
#include "pic_serial.h"
```

Include dependency graph for audio_queue.c:




## Functions

- void audio_queue_add (uns8 phrase)
- void audio_queue_clear ()
- uns8 audio_queue_empty ()
- void audio_queue_process ()

## Variables

- uns8 aq_end = 0
- uns8 aq_start = 0
- bit audio_playing = 0
- uns8 audio_queue_fifo [AUDIO_QUEUE_FIFO_SIZE]

---

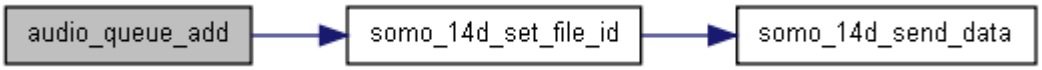
## Function Documentation

### void audio_queue_add (uns8 *phrase*)

Definition at line 51 of file audio_queue.c.

References aq_end, aq_start, audio_playing, audio_queue_fifo, somo_14d_set_file_id(), and uns8.

Here is the call graph for this function:

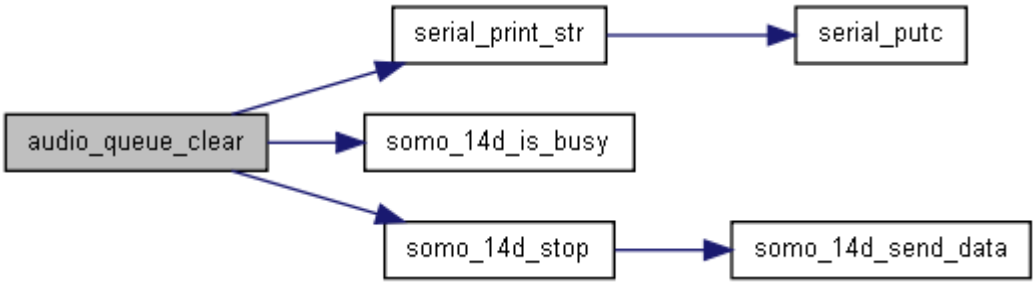



### void audio_queue_clear ()

Definition at line 105 of file audio_queue.c.

References aq_end, aq_start, audio_playing, end_crit_sec, serial_print_str(), somo_14d_is_busy(), somo_14d_stop(), and start_crit_sec.

Here is the call graph for this function:




### uns8 audio_queue_empty ()

Definition at line 121 of file audio_queue.c.

References audio_playing.

### void audio_queue_process ()

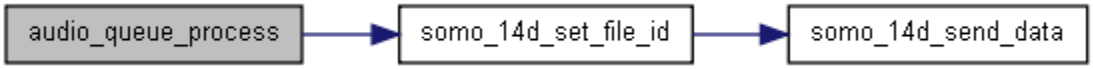
Call when audio file completes.

This routine will pluck the next file off the queue and start playing it. Assumes it is in an interrupt otherwise will need wrapping in critsec

Definition at line 81 of file audio_queue.c.

References aq_end, aq_start, audio_playing, audio_queue_fifo, somo_14d_set_file_id(), and uns8.

Here is the call graph for this function:




---

### Variable Documentation

**uns8 aq_end = 0**

Audio queue fifo end point

Definition at line 47 of file audio_queue.c.

Referenced by audio_queue_add(), audio_queue_clear(), and audio_queue_process().

**uns8 aq_start = 0**

Audio queue fifo start point

Definition at line 45 of file audio_queue.c.

Referenced by audio_queue_add(), audio_queue_clear(), and audio_queue_process().

**bit audio_playing = 0**

Audio playing at present

Definition at line 49 of file audio_queue.c.

Referenced by audio_queue_add(), audio_queue_clear(), audio_queue_empty(), and audio_queue_process().

**uns8 audio_queue_fifo[AUDIO_QUEUE_FIFO_SIZE]**

Audio queue fifo

Definition at line 43 of file audio_queue.c.

Referenced by audio_queue_add(), and audio_queue_process().

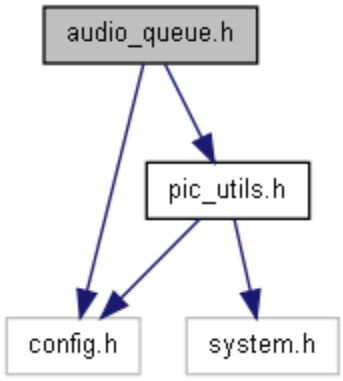
---

## audio_queue.h File Reference

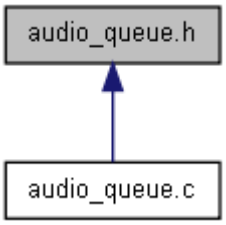
Queue audio files for the somo-14d.

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

Include dependency graph for audio_queue.h:




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




## Functions

- void audio_queue_add (uns8 phrase)
- void audio_queue_clear ()
- uns8 audio_queue_empty ()
- void audio_queue_process ()

---

## Detailed Description

Put the following into your config.h

- - - - - - - - - - - - - - - - - - - audio_queue defines
- - - - - - - - - - - - - - - - - - -

define AUDIO_QUEUE_FIFO_SIZE 5

Definition in file audio_queue.h.

---

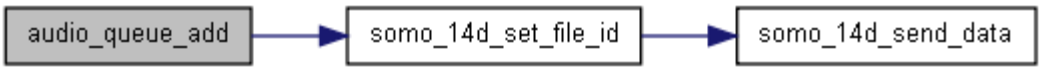
## Function Documentation

### void audio_queue_add (uns8 *phrase*)

Definition at line 51 of file audio_queue.c.

References aq_end, aq_start, audio_playing, audio_queue_fifo, somo_14d_set_file_id(), and uns8.

Here is the call graph for this function:

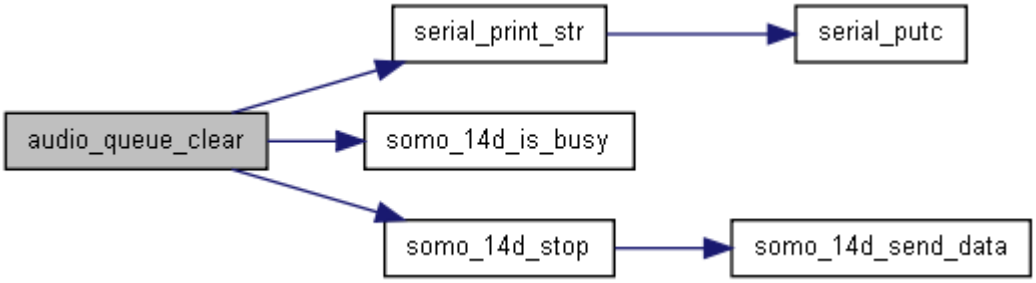



**void audio_queue_clear ()**

Definition at line 105 of file audio_queue.c.

References aq_end, aq_start, audio_playing, end_crit_sec, serial_print_str(), somo_14d_is_busy(), somo_14d_stop(), and start_crit_sec.

Here is the call graph for this function:




**uns8 audio_queue_empty ()**

Definition at line 121 of file audio_queue.c.

References audio_playing.

**void audio_queue_process ()**

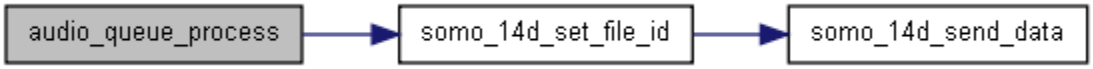
Call when audio file completes.

This routine will pluck the next file off the queue and start playing it. Assumes it is in an interrupt otherwise will need wrapping in critsec

Definition at line 81 of file audio_queue.c.

References aq_end, aq_start, audio_playing, audio_queue_fifo, somo_14d_set_file_id(), and uns8.

Here is the call graph for this function:

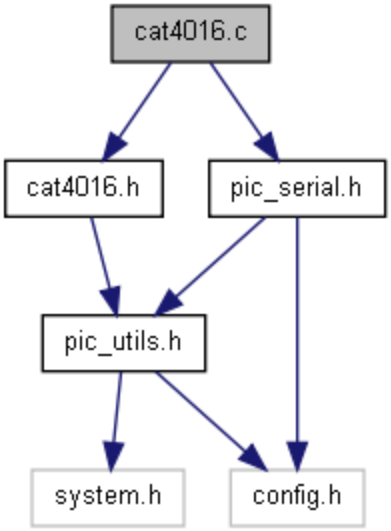



---

## cat4016.c File Reference

```
#include "cat4016.h"
#include "pic_serial.h"
```

Include dependency graph for cat4016.c:




## Functions

- void cat4016_enable_display (uns8 on)
- void cat4016_latch_data ()
- void cat4016_setup_io ()
- void cat4016_write_data (uns8 d1, uns8 d2)

---

## Function Documentation

### void cat4016_enable_display (uns8 *on*)

Definition at line 58 of file cat4016.c.

References clear_pin, and set_pin.

### void cat4016_latch_data ()

Definition at line 68 of file cat4016.c.

References clear_pin, and set_pin.

### void cat4016_setup_io ()

Definition at line 5 of file cat4016.c.

References clear_pin, make_output, and set_pin.

### void cat4016_write_data (uns8 *d1*, uns8 *d2*)

Definition at line 19 of file cat4016.c.

References change_pin_var, clear_pin, serial_putc(), set_pin, and uns8.

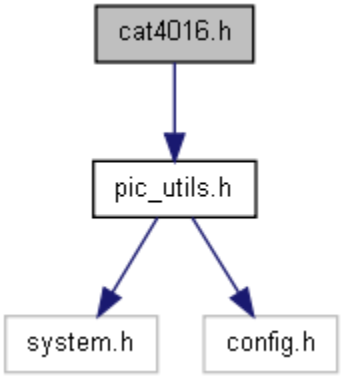
Here is the call graph for this function:


---

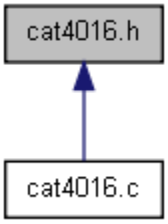
## cat4016.h File Reference

```
#include "pic_utils.h"
```

Include dependency graph for cat4016.h:




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




### Defines

- #define __cat4016_H

### Functions

- void cat4016_enable_display (uns8 on)
- void cat4016_latch_data ()
- void cat4016_setup_io ()
- void cat4016_write_data (uns8 d1, uns8 d2)

---

### Define Documentation

#### #define __cat4016_H

Definition at line 2 of file cat4016.h.

---

## Function Documentation

### void cat4016_enable_display (uns8 *on*)

Definition at line 58 of file cat4016.c.

References clear_pin, and set_pin.

### void cat4016_latch_data ()

Definition at line 68 of file cat4016.c.

References clear_pin, and set_pin.

### void cat4016_setup_io ()

Definition at line 5 of file cat4016.c.

References clear_pin, make_output, and set_pin.

### void cat4016_write_data (uns8 *d1*, uns8 *d2*)

Definition at line 19 of file cat4016.c.

References change_pin_var, clear_pin, serial_putc(), set_pin, and uns8.

Here is the call graph for this function:

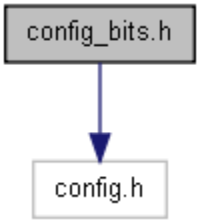

---

# config_bits.h File Reference

Trying the impossible task of creating some commonality around the config settings.

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

Include dependency graph for config_bits.h:



### Detailed Description

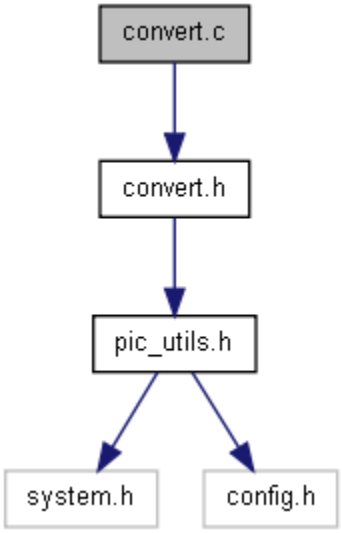
Definition in file config_bits.h.

---

## convert.c File Reference

```
#include "convert.h"
```

Include dependency graph for convert.c:




### Functions

- uns8 convert_to_dec1 (uns8 data)
- uns8 convert_to_dec2 (uns8 data)
- uns8 convert_to_dec2b (uns8 data)
- char * temp_to_str (uns8 int_part, uns8 fract_part, char *buffer)

---

### Function Documentation

#### uns8 convert_to_dec1 (uns8 *data*)

Definition at line 80 of file convert.c.

#### uns8 convert_to_dec2 (uns8 *data*)

Definition at line 101 of file convert.c.

**uns8 convert_to_dec2b (uns8 *data*)**

Definition at line 58 of file convert.c.

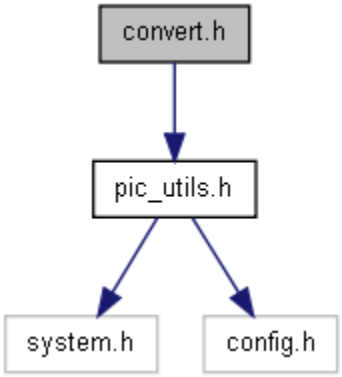
**char* temp_to_str (uns8 *int_part*, uns8 *fract_part*, char * *buffer*)**

Definition at line 124 of file convert.c.

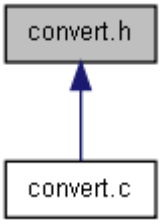
---

# convert.h File Reference

Convert pseudo-decimal to strings (typically temperature conversion).
`#include "pic_utils.h"`
Include dependency graph for convert.h:




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




## Functions

- uns8 convert_to_dec1 (uns8 data)
- uns8 convert_to_dec2 (uns8 data)
- uns8 convert_to_dec2b (uns8 data)
- char * temp_to_str (uns8 int_part, uns8 fract_part, char *buffer)

---

## Detailed Description

Definition in file convert.h.

---

## Function Documentation

**uns8 convert_to_dec1 (uns8 *data*)**

Definition at line 80 of file convert.c.

**uns8 convert_to_dec2 (uns8 *data*)**

Definition at line 101 of file convert.c.

**uns8 convert_to_dec2b (uns8 *data*)**

Definition at line 58 of file convert.c.

**char* temp_to_str (uns8 *int_part*, uns8 *fract_part*, char * *buffer*)**

Definition at line 124 of file convert.c.

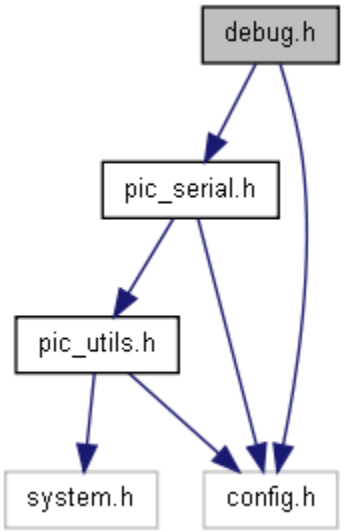
---

# debug.h File Reference

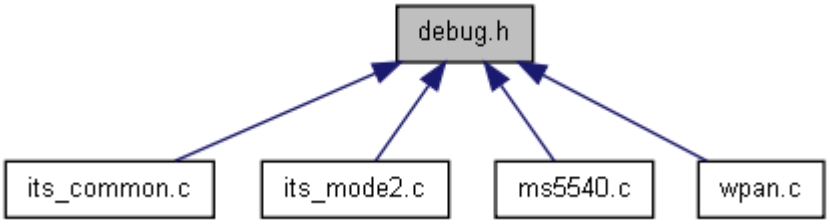
A nice way of printing debug, allowing it to be compiled out for production.

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

Include dependency graph for debug.h:



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




## Defines

- #define debug_int(i)
- #define debug_int_hex(i)
- #define debug_int_hex_16bit(i)
- #define debug_nl()
- #define debug_putc(c)
- #define debug_spc()
- #define debug_str(str)
- #define debug_var(str, i)

---

## Detailed Description

Definition in file debug.h.

---

## Define Documentation

### #define debug_int(i)

Definition at line 64 of file debug.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_process_tx_queue(), its2_transmit(), mrf24j40_receive_callback(), and mrf24j40_transmit_callback().

### #define debug_int_hex(i)

Definition at line 65 of file debug.h.

Referenced by wpan_data_received_callback().

### #define debug_int_hex_16bit(i)

Definition at line 66 of file debug.h.

Referenced by its2_forward_routed_packet(), its2_print_packet(), its2_process_tx_queue(), its2_transmit(), its_add_local_device(), its_add_net_device(), and its_print_devices().

#### #define debug_nl()

Definition at line 67 of file debug.h.

Referenced by its2_device_process(), its2_print_queue(), its2_router_init(), its_print_devices(), and wpan_data_received_callback().

#### #define debug_putc(c)

Definition at line 61 of file debug.h.

Referenced by its2_print_packet(), its2_transmit(), and wpan_data_received_callback().

#### #define debug_spc()

Definition at line 68 of file debug.h.

Referenced by its2_forward_routed_packet().

#### #define debug_str(str)

Definition at line 62 of file debug.h.

Referenced by its2_device_init(), its2_device_process(), its2_forward_routed_packet(), its2_print_packet(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_respond_local_addr(), its2_router_handle_association(), its2_router_init(), its2_router_queue_packet(), its2_transmit(), its_add_local_device(), its_add_net_device(), its_print_devices(), mrf24j40_receive_callback(), mrf24j40_transmit_callback(), wpan_data_received_callback(), and wpan_print_address().

#### #define debug_var(str, i)

Definition at line 63 of file debug.h.

Referenced by its2_device_process(), its2_forward_routed_packet(), its2_print_queue(), its2_rebroadcast_net_discover_req(), its2_router_init(), its2_router_queue_packet(), and its2_transmit().

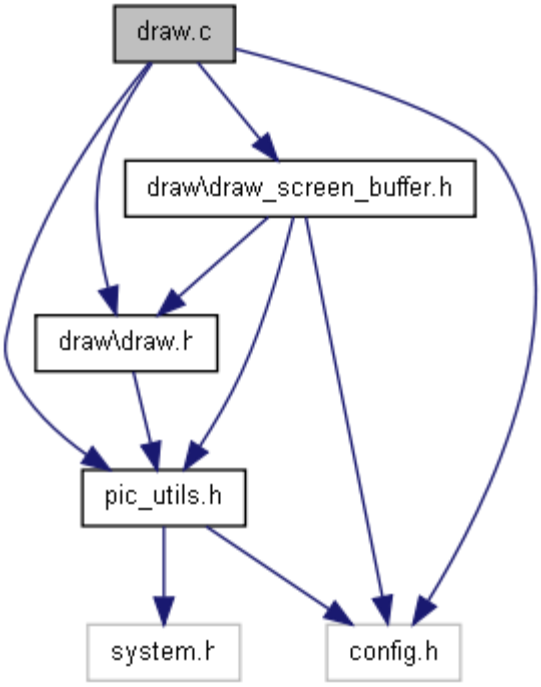
---

## draw.c File Reference

Buffered graphics routines.

```
#include "pic_utils.h"
#include "draw\draw.h"
#include "config.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for draw.c:




## Defines

- #define FONT_FIRST_CHAR 32
- #define FONT_HEIGHT 7
- #define FONT_LAST_CHAR 127

## Functions

- void draw_bitmap (uns8 x, uns8 y, uns8 colour, char *bitmap)
- void draw_circle (int x_centre, int y_centre, int r, uns8 colour)
- void draw_circle2 (int x_centre, int y_centre, int r, uns8 colour)
- void draw_circle_lines (int ctr_x, int ctr_y, int pt_x, int pt_y, uns8 colour)
- void draw_circle_points (int ctr_x, int ctr_y, int pt_x, int pt_y, uns8 colour)
- void draw_circle_points2 (int ctr_x, int ctr_y, int pt_x, int pt_y, uns8 colour)
- void draw_clear_screen ()
- void draw_filled_circle (int x_centre, int y_centre, int r, uns8 colour)
- uns8 draw_get_pixel (uns8 x, uns8 y)
- void draw_init ()
- uns16 draw_length_str (char *str)
- void draw_line (uns8 x0, uns8 y0, uns8 x1, uns8 y1, uns8 colour)
- void draw_print_buffer ()
- void draw_print_str (uns8 x, uns8 y, uns8 width, uns8 start_pixel, uns8 colour, char *str)
- void draw_rect (uns8 x, uns8 y, uns16 width, uns8 height, uns8 colour)
- void draw_set_pixel (uns8 x, uns8 y, uns8 colour)
- void draw_setup_io ()

## Variables

- rom char PicPack5x7_bitmap_0 [1]
- rom char PicPack5x7_bitmap_1 [1]
- uns16 PicPack5x7_index [1]

## Detailed Description

Definition in file draw.c.

## Define Documentation

### #define FONT_FIRST_CHAR 32

Definition at line 507 of file draw.c.

### #define FONT_HEIGHT 7

Definition at line 509 of file draw.c.

### #define FONT_LAST_CHAR 127

Definition at line 508 of file draw.c.

## Function Documentation

### void draw_bitmap (uns8 *x*, uns8 *y*, uns8 *colour*, char * *bitmap*)

Definition at line 591 of file draw.c.

References draw_set_pixel(), and uns8.

Referenced by draw_tests_run().

Here is the call graph for this function:

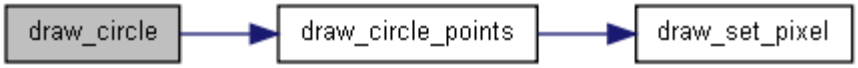

Here is the caller graph for this function:


### void draw_circle (int *x_centre*, int *y_centre*, int *r*, uns8 *colour*)

Definition at line 390 of file draw.c.

References draw_circle_points().

Here is the call graph for this function:



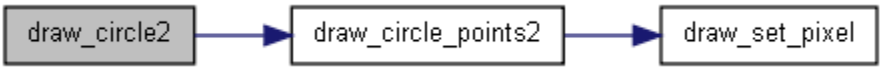

**void draw_circle2 (int *x_centre*, int *y_centre*, int *r*, uns8 *colour*)**

Definition at line 426 of file draw.c.

References draw_circle_points2().

Referenced by draw_tests_run().

Here is the call graph for this function:




Here is the caller graph for this function:

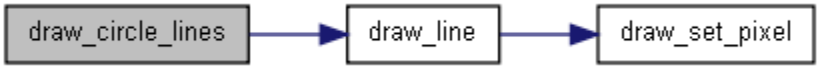

**void draw_circle_lines (int *ctr_x*, int *ctr_y*, int *pt_x*, int *pt_y*, uns8 *colour*)**

Definition at line 345 of file draw.c.

References draw_line().

Referenced by draw_filled_circle().

Here is the call graph for this function:




Here is the caller graph for this function:


**void draw_circle_points (int *ctr_x*, int *ctr_y*, int *pt_x*, int *pt_y*, uns8 *colour*)**

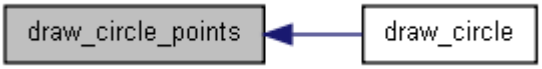
Definition at line 375 of file draw.c.

References draw_set_pixel().

Referenced by draw_circle().

Here is the call graph for this function:


Here is the caller graph for this function:



**void draw_circle_points2 (int *ctr_x*, int *ctr_y*, int *pt_x*, int *pt_y*, uns8 *colour*)**

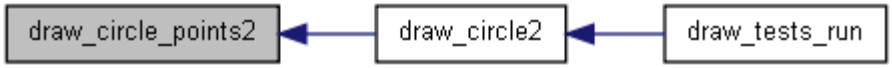
Definition at line 411 of file draw.c.

References draw_set_pixel().

Referenced by draw_circle2().

Here is the call graph for this function:


Here is the caller graph for this function:



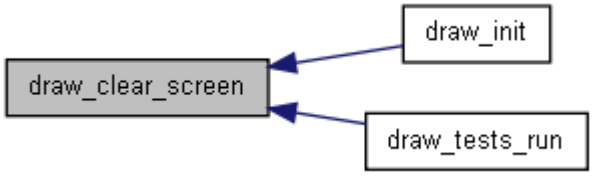

**void draw_clear_screen ()**

Definition at line 49 of file draw.c.

References draw_buffer0, DRAW_TOTAL_BUFFER_SIZE, and uns8.

Referenced by draw_init(), and draw_tests_run().

Here is the caller graph for this function:




**void draw_filled_circle (int *x_centre*, int *y_centre*, int *r*, uns8 *colour*)**

Definition at line 352 of file draw.c.

References draw_circle_lines().

Here is the call graph for this function:


**uns8 draw_get_pixel (uns8 *x*, uns8 *y*)**

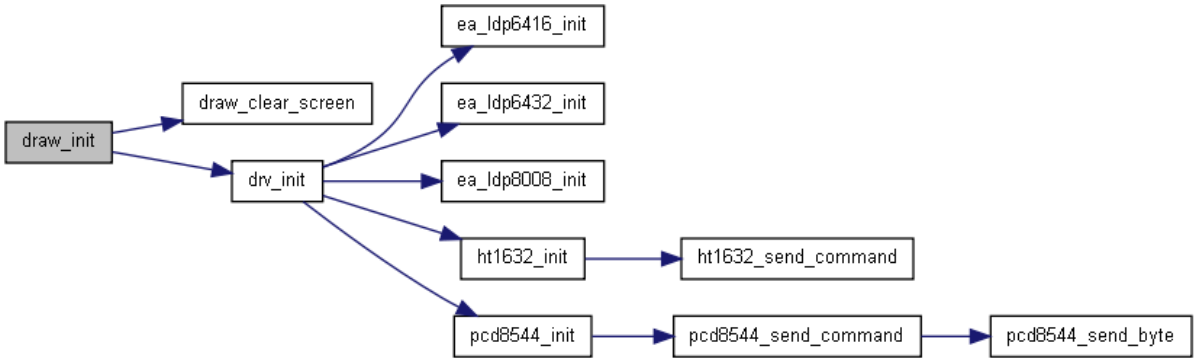
Definition at line 246 of file draw.c.

**void draw_init ()**

Definition at line 128 of file draw.c.

References draw_clear_screen(), and drv_init().

Here is the call graph for this function:




### uns16 draw_length_str (char * *str*)

Definition at line 514 of file draw.c.

References PicPack5x7_index, uns16, and uns8.

### void draw_line (uns8 *x0*, uns8 *y0*, uns8 *x1*, uns8 *y1*, uns8 *colour*)

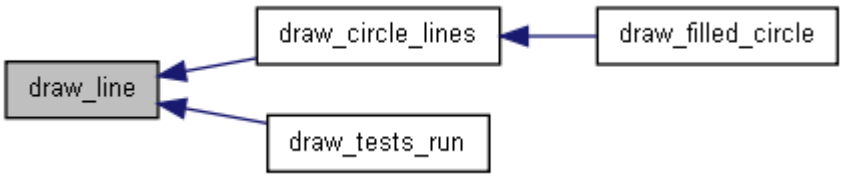
Definition at line 307 of file draw.c.

References draw_set_pixel().

Referenced by draw_circle_lines(), and draw_tests_run().

Here is the call graph for this function:


Here is the caller graph for this function:

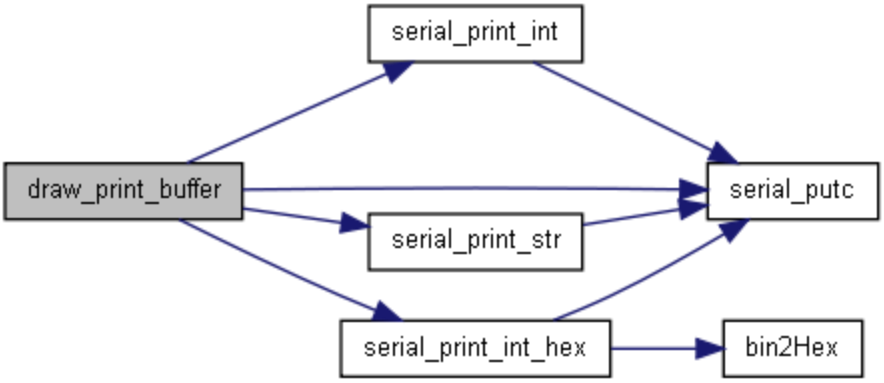



### void draw_print_buffer ()

Definition at line 266 of file draw.c.

References draw_buffer0, DRAW_PIXELS_PER_BYTE, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), uns16, and uns8.

Here is the call graph for this function:




**void draw_print_str (uns8 *x*, uns8 *y*, uns8 *width*, uns8 *start_pixel*, uns8 *colour*, char * *str*)**

Definition at line 529 of file draw.c.

References draw_set_pixel(), PicPack5x7_bitmap_0, PicPack5x7_bitmap_1, PicPack5x7_index, uns16, and uns8.

Referenced by draw_tests_run().

Here is the call graph for this function:


Here is the caller graph for this function:


**void draw_rect (uns8 *x*, uns8 *y*, uns16 *width*, uns8 *height*, uns8 *colour*)**

Definition at line 251 of file draw.c.

References draw_set_pixel(), and uns16.

Referenced by draw_tests_run().

Here is the call graph for this function:


Here is the caller graph for this function:

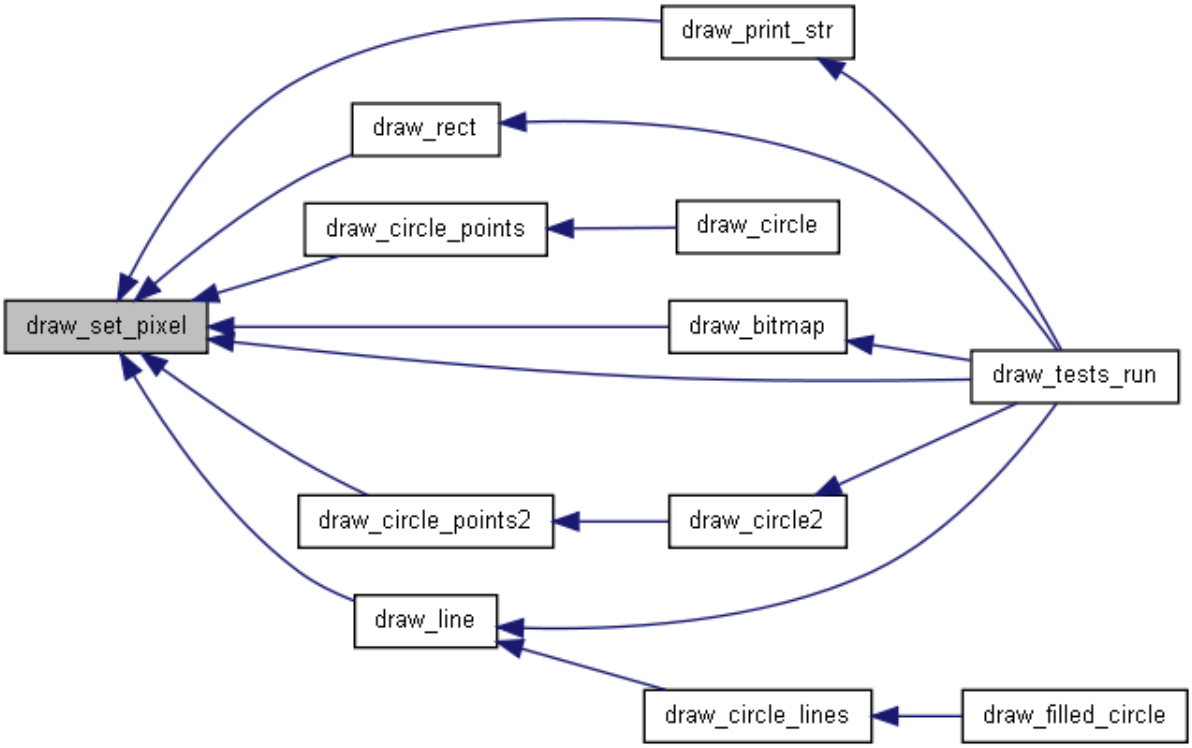

**void draw_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 135 of file draw.c.

References draw_buffer0, uns16, and uns8.

Referenced by draw_bitmap(), draw_circle_points(), draw_circle_points2(), draw_line(), draw_print_str(), draw_rect(), and draw_tests_run().

Here is the caller graph for this function:

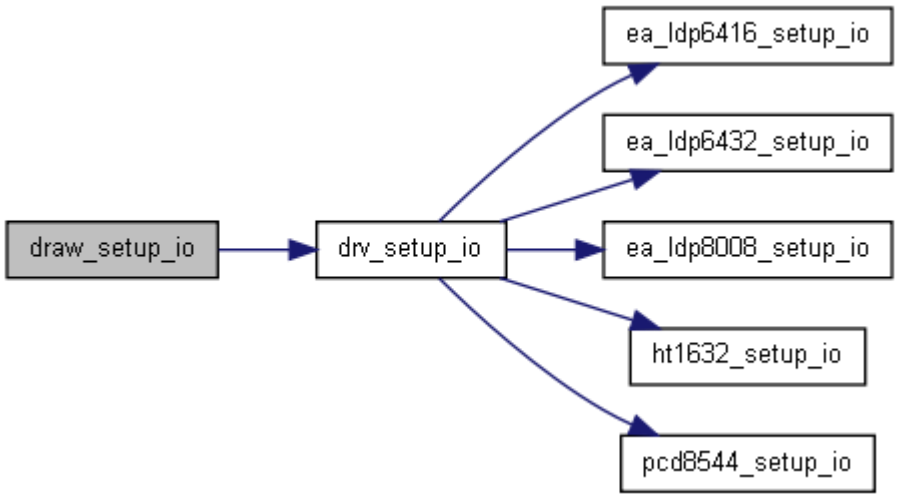



### void draw_setup_io ()

Definition at line 124 of file draw.c.

References drv_setup_io().

Here is the call graph for this function:



## Variable Documentation

### rom char PicPack5x7_bitmap_0[1]

Definition at line 54 of file draw_font_picpack_5x7.c.

Referenced by draw_print_str().

### rom char PicPack5x7_bitmap_1[1]

Definition at line 123 of file draw_font_picpack_5x7.c.

Referenced by draw_print_str().

### uns16 PicPack5x7_index[1]

Definition at line 157 of file draw_font_picpack_5x7.c.

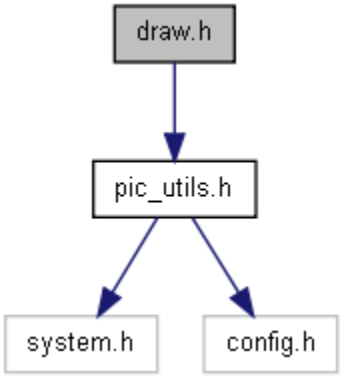
Referenced by draw_length_str(), and draw_print_str().

# draw.h File Reference

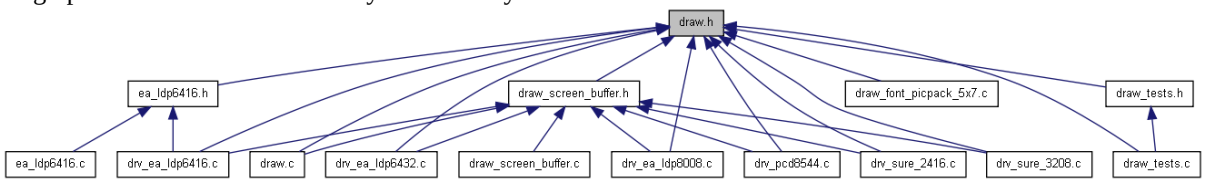
Buffered graphics routines.

```
#include "pic_utils.h"
```

Include dependency graph for draw.h:




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



## Defines

- #define BOTTOM_LEFT 1
- #define draw_paint() drv_paint()
- #define DRAW_PIXELS_PER_BYTE (8 / DRAW_BITS_PER_PIXEL)
- #define draw_set_display_brightness(brightness) drv_set_display_brightness(brightness)
- #define drv_setup() drv_setup_io()
- #define HORIZONTAL 0
- #define TOP_LEFT 0
- #define VERTICAL 1

## Functions

- void draw_bitmap (uns8 x, uns8 y, uns8 colour, char *bitmap)
- void draw_circle (int x_centre, int y_centre, int r, uns8 colour)
- void draw_circle2 (int x_centre, int y_centre, int r, uns8 colour)
- void draw_clear_screen ()
- uns8 draw_get_pixel (uns8 x, uns8)
- void draw_init ()
- uns16 draw_length_str (char *str)
- void draw_line (uns8 x0, uns8 y0, uns8 x1, uns8 y1, uns8 colour)
- void draw_print_buffer ()
- void draw_print_str (uns8 x, uns8 y, uns8 width, uns8 start_pixel, uns8 colour, char *str)
- void draw_rect (uns8 x, uns8 y, uns16 width, uns8 height, uns8 colour)
- void draw_set_pixel (uns8 x, uns8 y, uns8 colour)
- void draw_setup_io ()
- void drv_init ()
- void drv_paint ()
- void drv_print_buffer ()
- void drv_refresh ()
- void drv_set_display_brightness (uns8 brightness)
- void drv_setup_io ()

---

## Detailed Description

You will need to pick a hardware buffering mode for the draw routines.

Draw buffer addressing

4 | U V W X Y 3 | P Q R S T 2 | K L M N O 1 | F G H I J 0 | A B C D E --------- 0 1 2 3 4 DRAW_HW_Y_ORIGIN == BOTTOM_LEFT DRAW_HW_BUFFER_ORIENTATION == HORIZONTAL

0 | A B C D E 1 | F G H I J 2 | K L M N O 3 | P Q R S T 4 | U V W X Y --------- 0 1 2 3 4 DRAW_HW_Y_ORIGIN == TOP_LEFT DRAW_HW_BUFFER_ORIENTATION == HORIZONTAL

0 | E J O T Y 1 | D I N S X 2 | C H M R W 3 | B G L Q V 4 | A F K P U --------- 0 1 2 3 4 DRAW_HW_Y_ORIGIN == BOTTOM_LEFT DRAW_HW_BUFFER_ORIENTATION == VERTICAL

0 | A F K P U 1 | B G L Q V 2 | C H M R W 3 | D I N S X 4 | E J O T Y --------- 0 1 2 3 4 DRAW_HW_Y_ORIGIN == TOP_LEFT DRAW_HW_BUFFER_ORIENTATION == VERTICAL

Put the following in your config.h:

```
// - - - - - - - - - - - - - - - - - - - -
// Draw defines
// - - - - - - - - - - - - - - - - - - - -

#define DRAW_PIXELS_HIGH 24
#define DRAW_PIXELS_WIDE 16
#define DRAW_BITS_PER_PIXEL 1

#define DRAW_HW_Y_ORIGIN TOP_LEFT
// or BOTTOM LEFT

#define DRAW_HW_BUFFER_ORIENTATION VERTICAL
// or HORIZONTAL

//Enable debug to see what's happening under the hood
//#define DRAW DEBUG

// - - - - - - - - - - - - - - - - - - - -
```

Definition in file draw.h.

---

## Define Documentation

### #define BOTTOM_LEFT 1

Definition at line 120 of file draw.h.

### #define draw_paint() drv_paint()

Definition at line 136 of file draw.h.

Referenced by draw_tests_run().

### #define DRAW_PIXELS_PER_BYTE (8 / DRAW_BITS_PER_PIXEL)

Definition at line 122 of file draw.h.

Referenced by draw_print_buffer(), and drv_paint().

### #define draw_set_display_brightness(brightness) drv_set_display_brightness(brightness)

Definition at line 157 of file draw.h.

### #define drv_setup() drv_setup_io()

Definition at line 151 of file draw.h.

### #define HORIZONTAL 0

Definition at line 116 of file draw.h.

**#define TOP_LEFT 0**

Definition at line 119 of file draw.h.

**#define VERTICAL 1**

Definition at line 117 of file draw.h.

---

## Function Documentation

**void draw_bitmap (uns8 *x*, uns8 *y*, uns8 *colour*, char * *bitmap*)**

Definition at line 591 of file draw.c.

References draw_set_pixel(), and uns8.

Referenced by draw_tests_run().

Here is the call graph for this function:

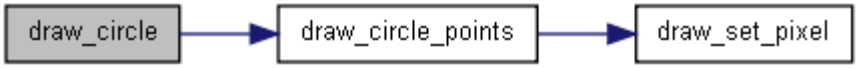

Here is the caller graph for this function:


**void draw_circle (int *x_centre*, int *y_centre*, int *r*, uns8 *colour*)**

Definition at line 390 of file draw.c.

References draw_circle_points().

Here is the call graph for this function:



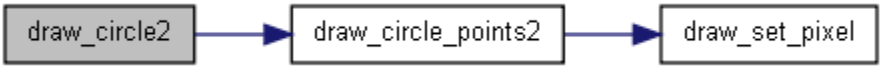

**void draw_circle2 (int *x_centre*, int *y_centre*, int *r*, uns8 *colour*)**

Definition at line 426 of file draw.c.

References draw_circle_points2().

Referenced by draw_tests_run().

Here is the call graph for this function:




Here is the caller graph for this function:

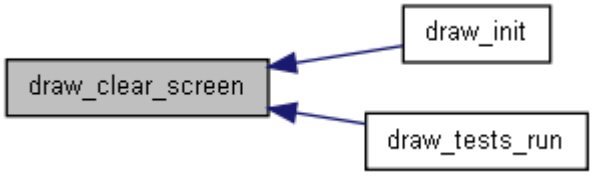

**void draw_clear_screen ()**

Definition at line 49 of file draw.c.

References draw_buffer0, DRAW_TOTAL_BUFFER_SIZE, and uns8.

Referenced by draw_init(), and draw_tests_run().

Here is the caller graph for this function:




**uns8 draw_get_pixel (uns8 *x*, uns8)**

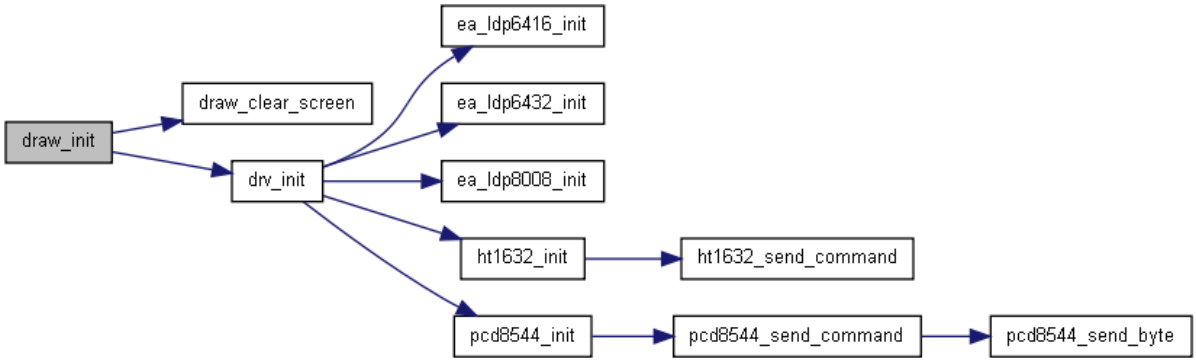
Definition at line 246 of file draw.c.

**void draw_init ()**

Definition at line 128 of file draw.c.

References draw_clear_screen(), and drv_init().

Here is the call graph for this function:




**uns16 draw_length_str (char * *str*)**

Definition at line 514 of file draw.c.

References PicPack5x7_index, uns16, and uns8.

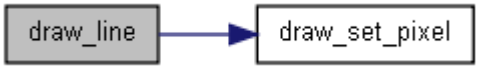
**void draw_line (uns8 *x0*, uns8 *y0*, uns8 *x1*, uns8 *y1*, uns8 *colour*)**

Definition at line 307 of file draw.c.

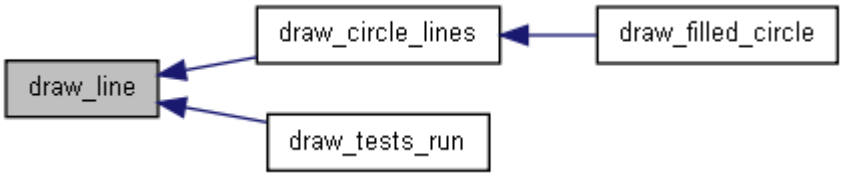
References draw_set_pixel().

Referenced by draw_circle_lines(), and draw_tests_run().

Here is the call graph for this function:




Here is the caller graph for this function:

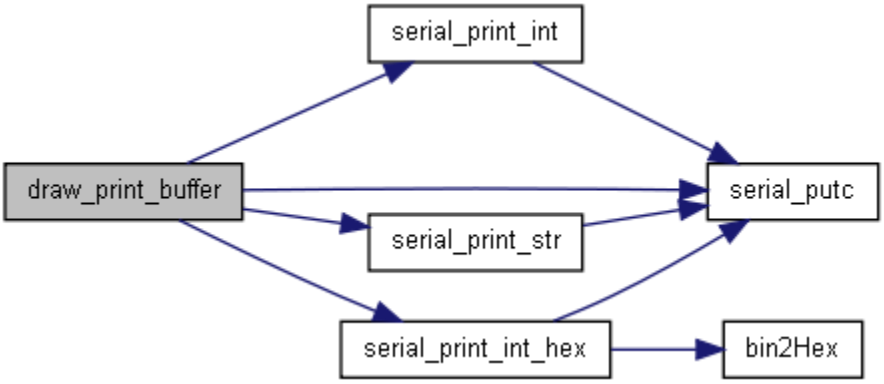



**void draw_print_buffer ()**

Definition at line 266 of file draw.c.

References draw_buffer0, DRAW_PIXELS_PER_BYTE, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), uns16, and uns8.

Here is the call graph for this function:




**void draw_print_str (uns8 *x*, uns8 *y*, uns8 *width*, uns8 *start_pixel*, uns8 *colour*, char * *str*)**

Definition at line 529 of file draw.c.

References draw_set_pixel(), PicPack5x7_bitmap_0, PicPack5x7_bitmap_1, PicPack5x7_index, uns16, and uns8.

Referenced by draw_tests_run().

Here is the call graph for this function:


Here is the caller graph for this function:


**void draw_rect (uns8 *x*, uns8 *y*, uns16 *width*, uns8 *height*, uns8 *colour*)**

Definition at line 251 of file draw.c.

References draw_set_pixel(), and uns16.

Referenced by draw_tests_run().

Here is the call graph for this function:


Here is the caller graph for this function:

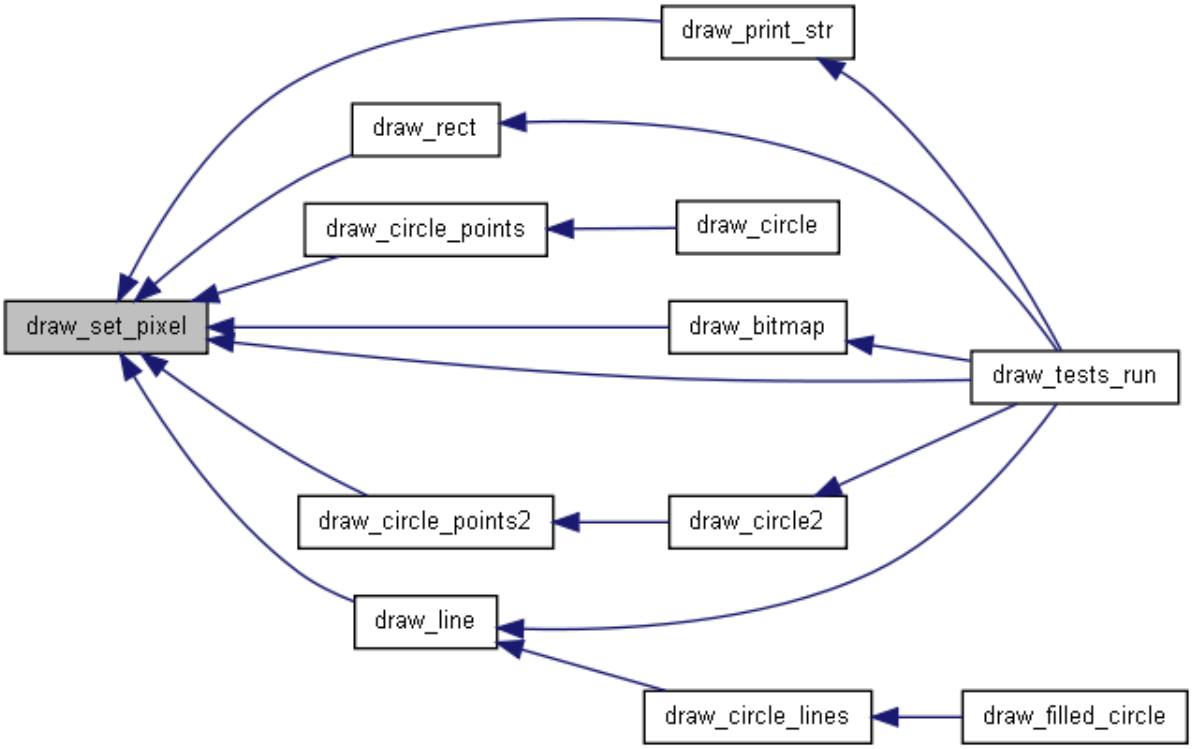

**void draw_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 135 of file draw.c.

References draw_buffer0, uns16, and uns8.

Referenced by draw_bitmap(), draw_circle_points(), draw_circle_points2(), draw_line(), draw_print_str(), draw_rect(), and draw_tests_run().

Here is the caller graph for this function:

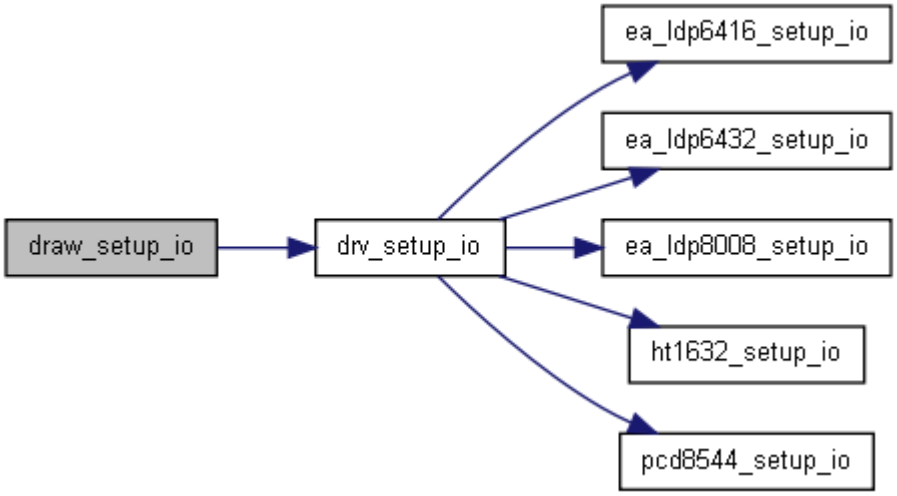



**void draw_setup_io ()**

Definition at line 124 of file draw.c.

References drv_setup_io().

Here is the call graph for this function:



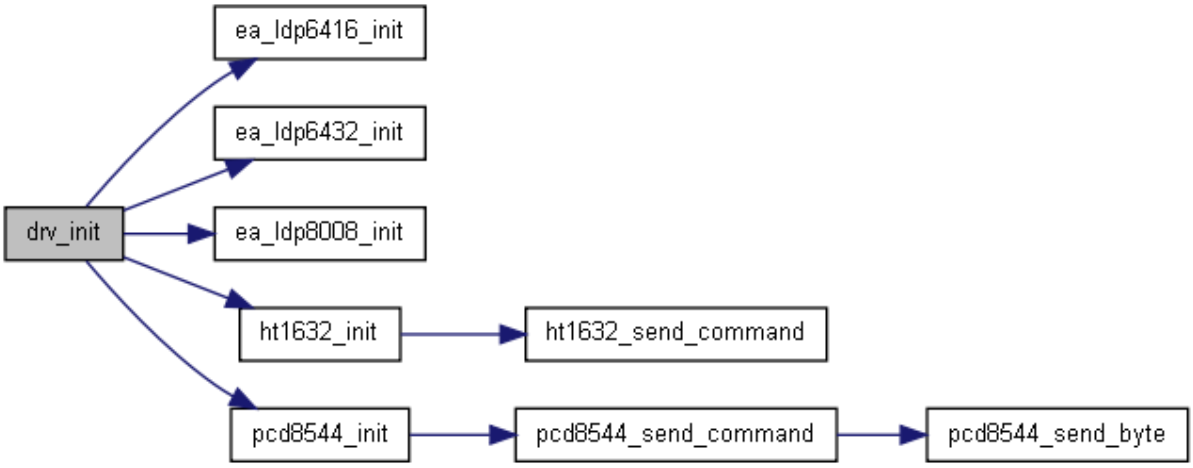

**void drv_init ()**

Definition at line 287 of file drv_ea_ldp6416.c.

References ea_ldp6416_init(), ea_ldp6432_init(), ea_ldp8008_init(), HT1632_CMD_PMOS_16_COMMON, ht1632_init(), and pcd8544_init().

Referenced by draw_init().

Here is the call graph for this function:



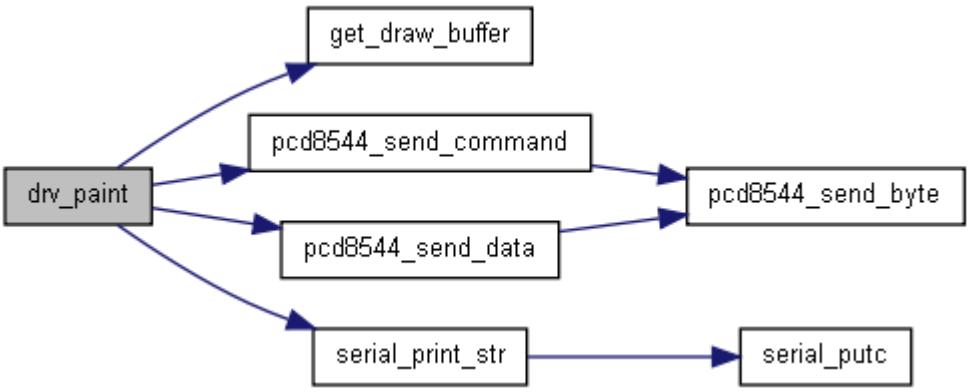

Here is the caller graph for this function:


### void drv_paint ()

Definition at line 73 of file drv_ea_ldp6416.c.

References buffer0, buffer1, clear_pin, draw_buffer0, DRAW_PIXELS_PER_BYTE, end_crit_sec, get_draw_buffer(), pcd8544_send_command(), pcd8544_send_data(), serial_print_str(), set_pin, start_crit_sec, uns16, and uns8.

Here is the call graph for this function:

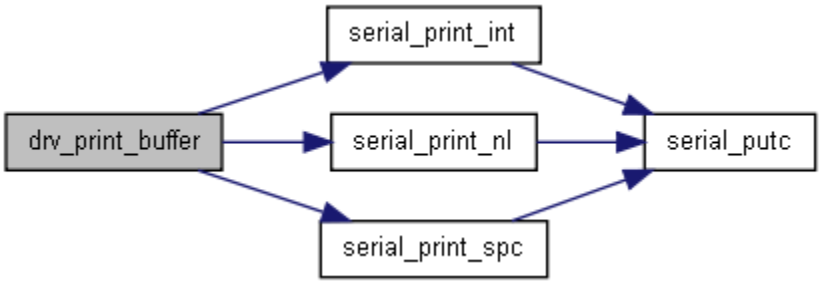



### void drv_print_buffer ()

Definition at line 94 of file drv_ea_ldp6416.c.

References buffer0, serial_print_int(), serial_print_nl(), serial_print_spc(), and uns8.

Here is the call graph for this function:




### void drv_refresh ()

Definition at line 116 of file drv_ea_ldp6416.c.

References bright_count, bright_level, buffer0, buffer1, buffer_position, buffer_position0, clear_pin, current_buffer, current_row, MAX_BRIGHTNESS, set_pin, set_pins_r1_g1, set_pins_r1_g1_r2_g2, set_pins_r_g, and uns8.

### void drv_set_display_brightness (uns8 *brightness*)

Definition at line 108 of file drv_ea_ldp6416.c.

References bright_level, and MAX_BRIGHTNESS.

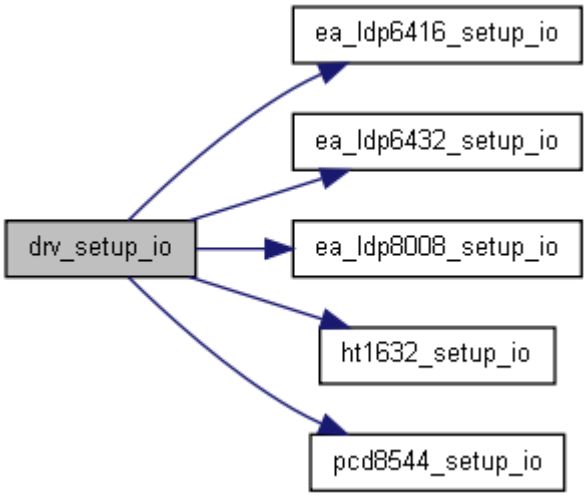
### void drv_setup_io ()

Definition at line 283 of file drv_ea_ldp6416.c.

References ea_ldp6416_setup_io(), ea_ldp6432_setup_io(), ea_ldp8008_setup_io(), ht1632_setup_io(), and pcd8544_setup_io().

Referenced by draw_setup_io().

Here is the call graph for this function:



Here is the caller graph for this function:

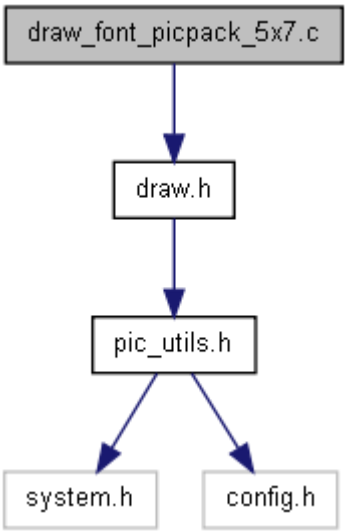

---

## draw_font_picpack_5x7.c File Reference

Simple 5x7 font for Draw library.

```
#include "draw.h"
```

Include dependency graph for draw_font_picpack_5x7.c:




### Defines

- #define FONT_FIRST_CHAR 32
- #define FONT_LAST_CHAR 128

### Variables

- rom char PicPack5x7_bitmap_0 []
- rom char PicPack5x7_bitmap_1 []
- uns16 PicPack5x7_index []

---

### Detailed Description

Definition in file draw_font_picpack_5x7.c.

---

## Define Documentation

### #define FONT_FIRST_CHAR 32

Definition at line 258 of file draw_font_picpack_5x7.c.

### #define FONT_LAST_CHAR 128

Definition at line 259 of file draw_font_picpack_5x7.c.

---

## Variable Documentation

### rom char PicPack5x7_bitmap_0[]

Definition at line 54 of file draw_font_picpack_5x7.c.
Referenced by draw_print_str().

### rom char PicPack5x7_bitmap_1[]

Definition at line 123 of file draw_font_picpack_5x7.c.
Referenced by draw_print_str().

### uns16 PicPack5x7_index[]

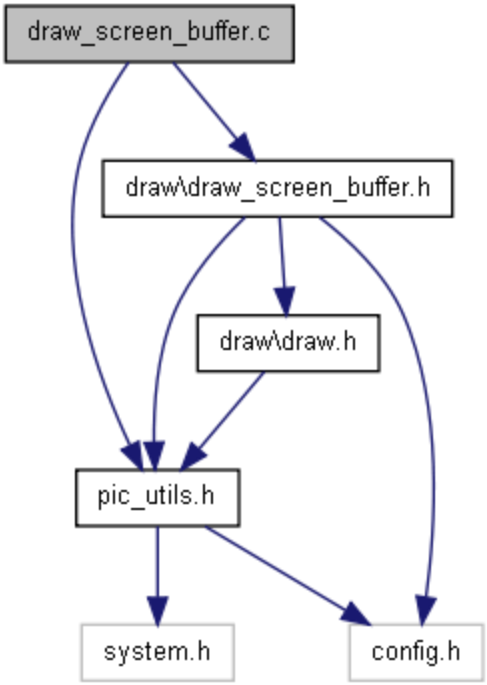
Definition at line 157 of file draw_font_picpack_5x7.c.
Referenced by draw_length_str(), and draw_print_str().

---

# draw_screen_buffer.c File Reference

```
#include "pic_utils.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for draw_screen_buffer.c:




## Functions

- uns8 get_draw_buffer (uns16 address)
- void set_draw_buffer (uns16 address, uns8 data)

## Variables

- uns8 draw_buffer0 [DRAW_TOTAL_BUFFER_SIZE]

---

## Function Documentation

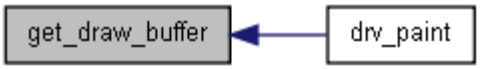
### uns8 get_draw_buffer (uns16 *address*)

Definition at line 33 of file draw_screen_buffer.c.

References draw_buffer0.

Referenced by drv_paint().

Here is the caller graph for this function:




### void set_draw_buffer (uns16 *address*, uns8 *data*)

Definition at line 55 of file draw_screen_buffer.c.

References draw_buffer0.

## Variable Documentation

### uns8 draw_buffer0[DRAW_TOTAL_BUFFER_SIZE]

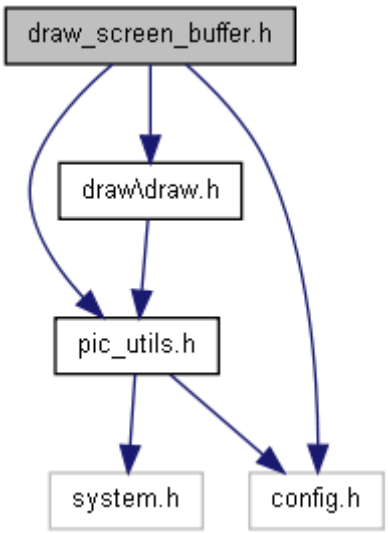
Definition at line 8 of file draw_screen_buffer.c.

Referenced by draw_clear_screen(), draw_print_buffer(), draw_set_pixel(), drv_paint(), get_draw_buffer(), and set_draw_buffer().

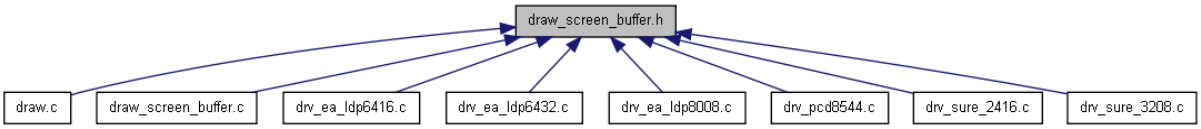
# draw_screen_buffer.h File Reference

```
#include "pic_utils.h"
#include "config.h"
#include "draw\draw.h"
```

Include dependency graph for draw_screen_buffer.h:




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




## Defines

- #define DRAW_BUFFERS 1
- #define DRAW_TOTAL_BUFFER_SIZE (DRAW_PIXELS_WIDE * DRAW_PIXELS_HIGH / DRAW_PIXELS_PER_BYTE)

## Functions

- uns8 get_draw_buffer (uns16 address)
- void set_draw_buffer (uns16 address, uns8 data)

## Variables

- uns8 draw_buffer0 [DRAW_TOTAL_BUFFER_SIZE]

---

## Define Documentation

### #define DRAW_BUFFERS 1

Definition at line 20 of file draw_screen_buffer.h.

### #define DRAW_TOTAL_BUFFER_SIZE (DRAW_PIXELS_WIDE * DRAW_PIXELS_HIGH / DRAW_PIXELS_PER_BYTE)

Definition at line 14 of file draw_screen_buffer.h.

Referenced by draw_clear_screen().

---

## Function Documentation

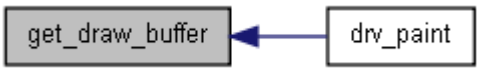
### uns8 get_draw_buffer (uns16 *address*)

Definition at line 33 of file draw_screen_buffer.c.

References draw_buffer0.

Referenced by drv_paint().

Here is the caller graph for this function:




### void set_draw_buffer (uns16 *address*, uns8 *data*)

Definition at line 55 of file draw_screen_buffer.c.

References draw_buffer0.

---

## Variable Documentation

### uns8 draw_buffer0[DRAW_TOTAL_BUFFER_SIZE]

Definition at line 8 of file draw_screen_buffer.c.

Referenced by draw_clear_screen(), draw_print_buffer(), draw_set_pixel(), drv_paint(), get_draw_buffer(), and set_draw_buffer().

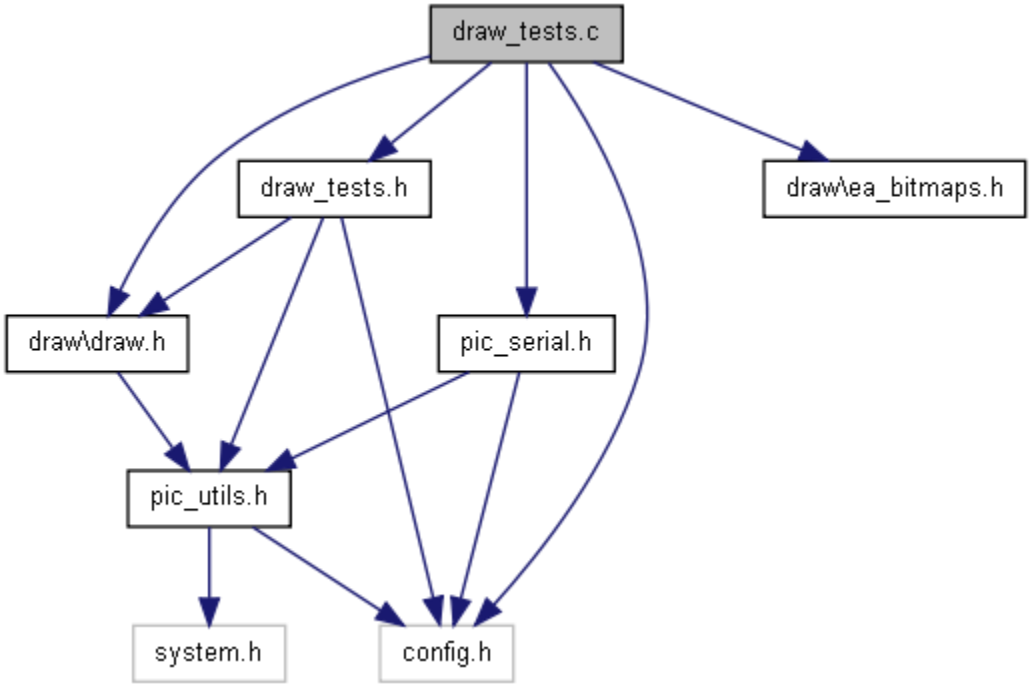
---

## draw_tests.c File Reference

Routines to test the Draw graphics functions.

```
#include "draw_tests.h"
#include "pic_serial.h"
#include "draw\draw.h"
#include "draw\ea_bitmaps.h"
#include "config.h"
```

Include dependency graph for draw_tests.c:




### Defines

- #define TEST_RADIUS DRAW_PIXELS_WIDE / 2 -1

### Functions

- uns8 draw_tests_run (uns8 test_num)

---

### Detailed Description

Definition in file draw_tests.c.

## Define Documentation

### #define TEST_RADIUS  DRAW_PIXELS_WIDE / 2 -1

Referenced by draw_tests_run().

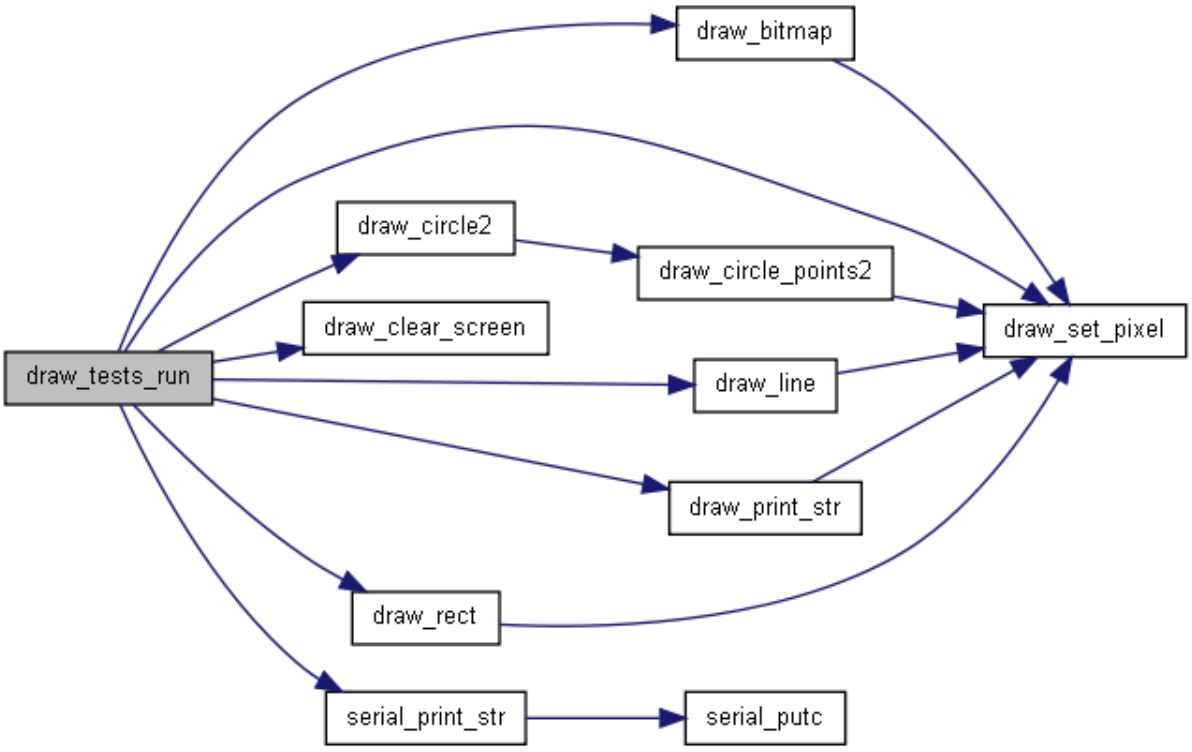
## Function Documentation

### uns8 draw_tests_run (uns8 *test_num*)

Definition at line 46 of file draw_tests.c.

References a_big_bitmap, a_bitmap, adventures_bitmap, draw_bitmap(), DRAW_BOTTOM_PIXEL_Y, draw_circle2(), draw_clear_screen(), draw_line(), draw_paint, draw_print_str(), draw_rect(), draw_set_pixel(), DRAW_TOP_PIXEL_Y, e_big_bitmap, e_bitmap, embedded_bitmap, serial_print_str(), TEST_RADIUS, TEST_RESULT_NO_MORE_TESTS, TEST_RESULT_NOT_APPLICABLE, TEST_RESULT_RAN, and uns8.

Here is the call graph for this function:

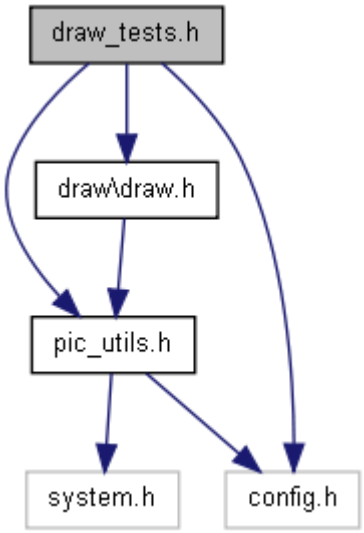


# draw_tests.h File Reference

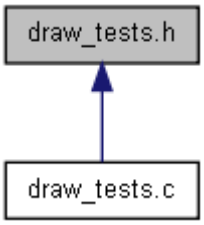
Routines to test the Draw graphics functions.

```
#include "pic_utils.h"
#include "config.h"
#include "draw\draw.h"
```

Include dependency graph for draw_tests.h:




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




## Defines

- #define DRAW_BOTTOM_PIXEL_Y DRAW_PIXELS_HIGH - 1
- #define DRAW_TOP_PIXEL_Y 0
- #define TEST_RESULT_NO_MORE_TESTS 2
- #define TEST_RESULT_NOT_APPLICABLE 1
- #define TEST_RESULT_RAN 0

## Functions

- uns8 draw_tests_run (uns8 test_num)

---

## Detailed Description

Definition in file draw_tests.h.

## Define Documentation

### #define DRAW_BOTTOM_PIXEL_Y DRAW_PIXELS_HIGH - 1

Definition at line 50 of file draw_tests.h.

Referenced by draw_tests_run().

### #define DRAW_TOP_PIXEL_Y 0

Definition at line 49 of file draw_tests.h.

Referenced by draw_tests_run().

### #define TEST_RESULT_NO_MORE_TESTS 2

Definition at line 58 of file draw_tests.h.

Referenced by draw_tests_run().

### #define TEST_RESULT_NOT_APPLICABLE 1

Definition at line 57 of file draw_tests.h.

Referenced by draw_tests_run().

### #define TEST_RESULT_RAN 0

Definition at line 56 of file draw_tests.h.

Referenced by draw_tests_run().

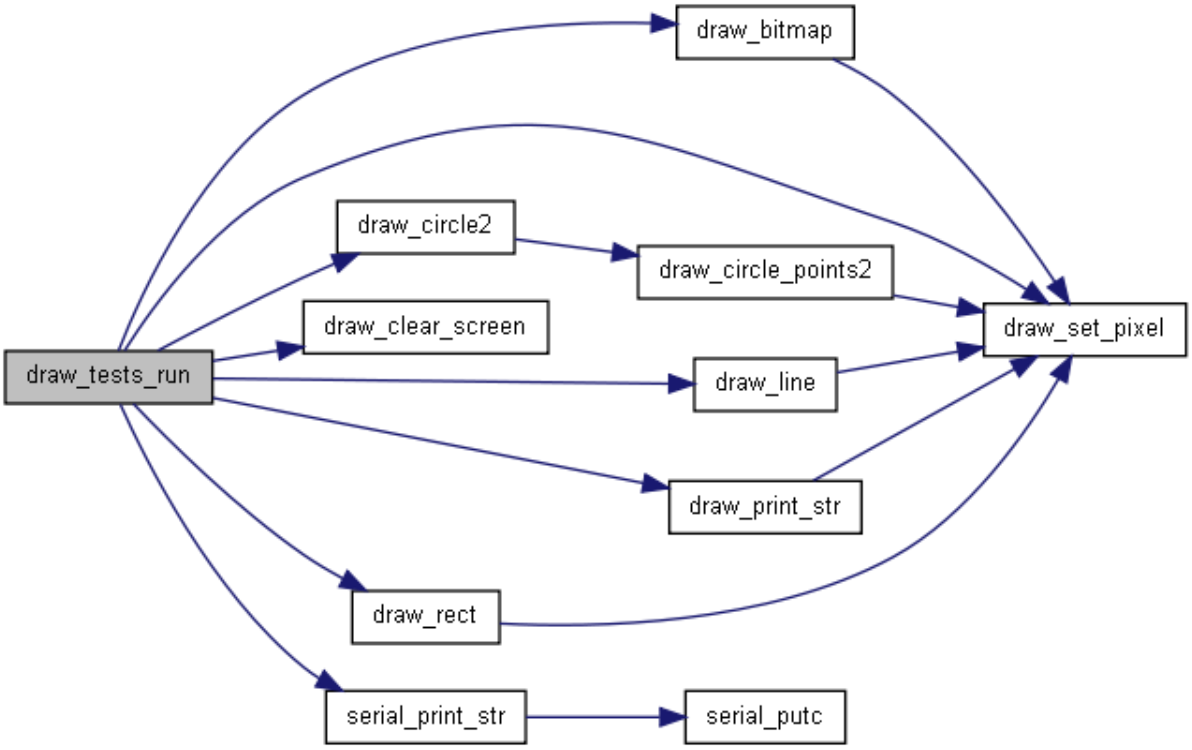
## Function Documentation

### uns8 draw_tests_run (uns8 *test_num*)

Definition at line 46 of file draw_tests.c.

References a_big_bitmap, a_bitmap, adventures_bitmap, draw_bitmap(), DRAW_BOTTOM_PIXEL_Y, draw_circle2(), draw_clear_screen(), draw_line(), draw_paint, draw_print_str(), draw_rect(), draw_set_pixel(), DRAW_TOP_PIXEL_Y, e_big_bitmap, e_bitmap, embedded_bitmap, serial_print_str(), TEST_RADIUS, TEST_RESULT_NO_MORE_TESTS, TEST_RESULT_NOT_APPLICABLE, TEST_RESULT_RAN, and uns8.

Here is the call graph for this function:



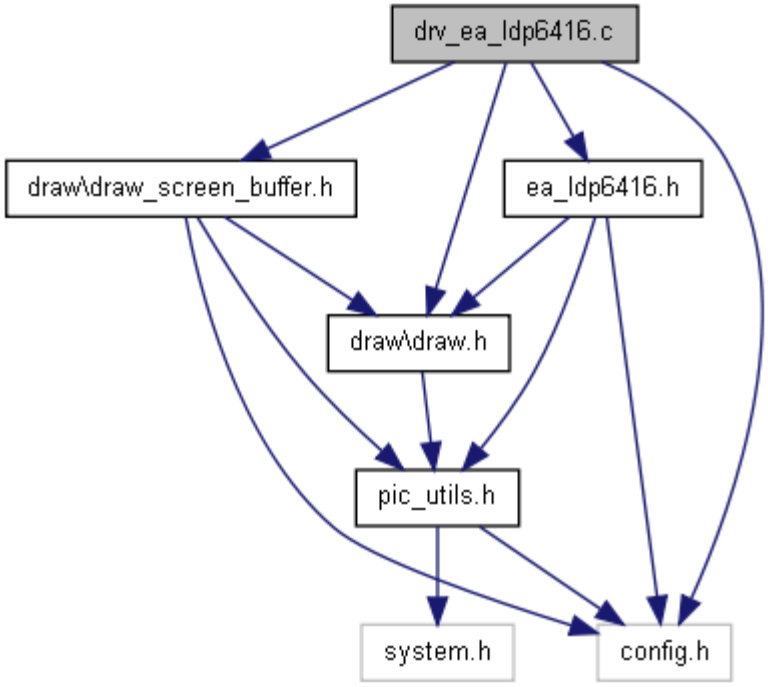

---

## drv_ea_ldp6416.c File Reference

Draw drivers for Embedded Adventures LDP-6416 LED panel and similar.

```
#include "ea_ldp6416.h"
#include "config.h"
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_ea_ldp6416.c:




## Defines

- #define set_pins_r1_g1()

## Functions

- uns8 drv_get_pixel (uns8 x, uns8 y)
- void drv_init ()
- void drv_paint ()
- void drv_print_buffer ()
- void drv_refresh ()
- void drv_set_display_brightness (uns8 brightness)
- void drv_setup_io ()

## Variables

- uns8 bright_count = 0
- uns8 bright_level = 0
- char buffer0 [256]
- uns8 buffer_position = 0
- uns8 current_buffer = 0
- uns8 current_row = 0

---

## Detailed Description

Definition in file drv_ea_ldp6416.c.

---

## Define Documentation

### #define set_pins_r1_g1()

```
Value:set_pin(ea_ldp6416_r1_port, ea_ldp6416_r1_pin); \
    set_pin(ea_ldp6416_g1_port, ea_ldp6416_g1_pin);
```

Definition at line 48 of file drv_ea_ldp6416.c.

Referenced by drv_refresh().

---

## Function Documentation

### uns8 drv_get_pixel (uns8 *x*, uns8 *y*)

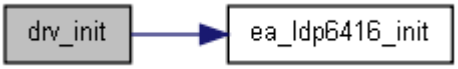
Definition at line 279 of file drv_ea_ldp6416.c.

### void drv_init ()

Definition at line 287 of file drv_ea_ldp6416.c.

References ea_ldp6416_init().

Here is the call graph for this function:




### void drv_paint ()

Definition at line 73 of file drv_ea_ldp6416.c.

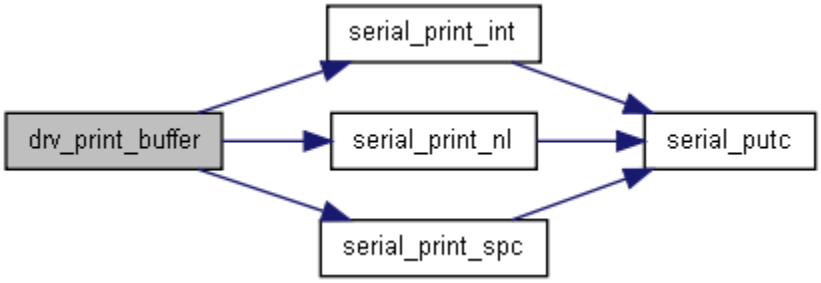
References buffer0, buffer1, draw_buffer0, and uns8.

### void drv_print_buffer ()

Definition at line 94 of file drv_ea_ldp6416.c.

References buffer0, serial_print_int(), serial_print_nl(), serial_print_spc(), and uns8.

Here is the call graph for this function:



### void drv_refresh ()

Definition at line 116 of file drv_ea_ldp6416.c.

References bright_count, bright_level, buffer0, buffer1, buffer_position, clear_pin, current_buffer, current_row, set_pin, set_pins_r1_g1, and uns8.

### void drv_set_display_brightness (uns8 *brightness*)

Definition at line 108 of file drv_ea_ldp6416.c.

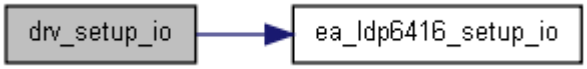
References bright_level.

### void drv_setup_io ()

Definition at line 283 of file drv_ea_ldp6416.c.

References ea_ldp6416_setup_io().

Here is the call graph for this function:




---

## Variable Documentation

### uns8 bright_count = 0

Definition at line 54 of file drv_ea_ldp6416.c.

Referenced by drv_refresh().

### uns8 bright_level = 0

Definition at line 55 of file drv_ea_ldp6416.c.

Referenced by drv_refresh(), and drv_set_display_brightness().

### char buffer0[256]

Definition at line 62 of file drv_ea_ldp6416.c.

Referenced by drv_paint(), drv_print_buffer(), and drv_refresh().

### uns8 buffer_position = 0

Definition at line 56 of file drv_ea_ldp6416.c.

Referenced by drv_refresh().

**uns8 current_buffer = 0**

Definition at line 60 of file drv_ea_ldp6416.c.

Referenced by drv_refresh().

**uns8 current_row = 0**

Definition at line 52 of file drv_ea_ldp6416.c.

Referenced by drv_refresh().

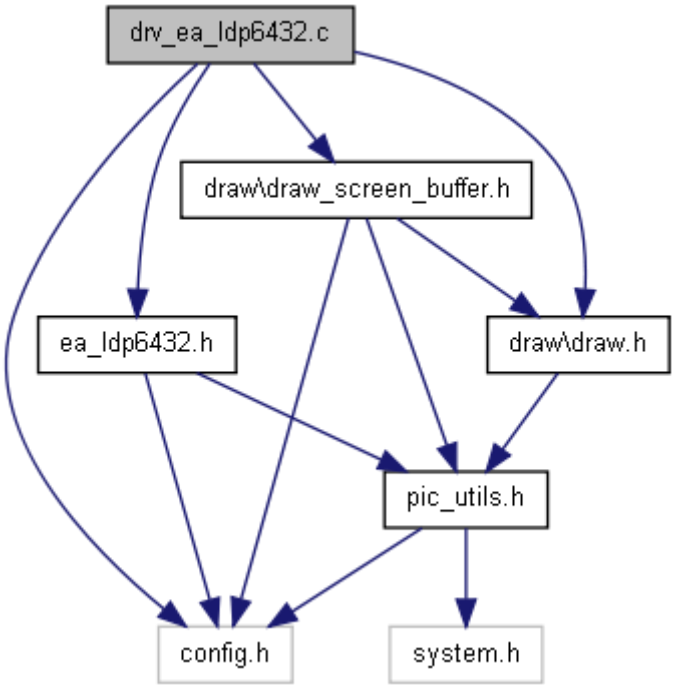
---

# drv_ea_ldp6432.c File Reference

Draw drivers for Embedded Adventures LDP-6432 LED panel and similar.

```
#include "ea_ldp6432.h"
#include "config.h"
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_ea_ldp6432.c:




## Defines

- #define MAX_BRIGHTNESS 3
- #define set_pins_r1_g1_r2_g2()

## Functions

- void drv_clear_screen ()
- uns8 drv_get_pixel (uns8 x, uns8 y)
- void drv_init ()
- void drv_paint ()
- void drv_print_buffer ()
- void drv_refresh ()
- void drv_set_display_brightness (uns8 brightness)
- void drv_setup_io ()

## Variables

- uns8 bright_count = 0
- uns8 bright_level = 3
- char buffer0 [256]
- char buffer1 [256]
- uns8 buffer_position0 = 0
- uns8 current_row = 0

---

## Detailed Description

Definition in file drv_ea_ldp6432.c.

---

## Define Documentation

### #define MAX_BRIGHTNESS 3

Definition at line 71 of file drv_ea_ldp6432.c.

Referenced by drv_refresh(), and drv_set_display_brightness().

### #define set_pins_r1_g1_r2_g2()

```
Value:set_pin(ea_ldp6432_r1_port, ea_ldp6432_r1_pin); \
    set_pin(ea_ldp6432_r2_port, ea_ldp6432_r2_pin); \
    set_pin(ea_ldp6432_g1_port, ea_ldp6432_g1_pin); \
    set_pin(ea_ldp6432_g2_port, ea_ldp6432_g2_pin)
```

Definition at line 49 of file drv_ea_ldp6432.c.

Referenced by drv_refresh().

---

## Function Documentation

### void drv_clear_screen ()

Definition at line 331 of file drv_ea_ldp6432.c.

### uns8 drv_get_pixel (uns8 *x*, uns8 *y*)

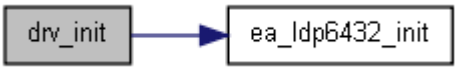
Definition at line 327 of file drv_ea_ldp6432.c.

### void drv_init ()

Definition at line 339 of file drv_ea_ldp6432.c.

References ea_ldp6432_init().

Here is the call graph for this function:




### void drv_paint ()

Definition at line 74 of file drv_ea_ldp6432.c.

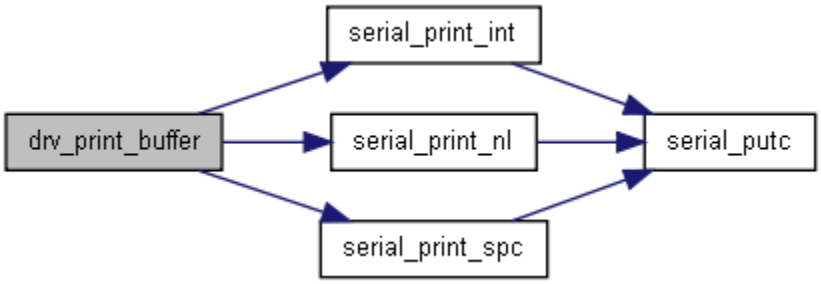
References buffer0, buffer1, draw_buffer0, and uns8.

### void drv_print_buffer ()

Definition at line 312 of file drv_ea_ldp6432.c.

References buffer0, serial_print_int(), serial_print_nl(), serial_print_spc(), and uns8.

Here is the call graph for this function:




### void drv_refresh ()

Definition at line 98 of file drv_ea_ldp6432.c.

References bright_count, bright_level, buffer0, buffer1, buffer_position0, clear_pin, current_row, MAX_BRIGHTNESS, set_pin, set_pins_r1_g1_r2_g2, and uns8.

### void drv_set_display_brightness (uns8 *brightness*)

Definition at line 90 of file drv_ea_ldp6432.c.

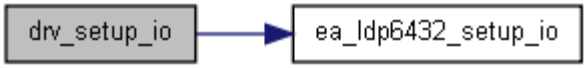
References bright_level, and MAX_BRIGHTNESS.

### void drv_setup_io ()

Definition at line 335 of file drv_ea_ldp6432.c.

References ea_ldp6432_setup_io().

Here is the call graph for this function:




---

### Variable Documentation

**uns8 bright_count = 0**

Definition at line 64 of file drv_ea_ldp6432.c.

**uns8 bright_level = 3**

Definition at line 65 of file drv_ea_ldp6432.c.

**char buffer0[256]**

Definition at line 55 of file drv_ea_ldp6432.c.

**char buffer1[256]**

Definition at line 56 of file drv_ea_ldp6432.c.

Referenced by drv_paint(), and drv_refresh().

**uns8 buffer_position0 = 0**

Definition at line 66 of file drv_ea_ldp6432.c.

Referenced by drv_refresh().

**uns8 current_row = 0**

Definition at line 62 of file drv_ea_ldp6432.c.

---

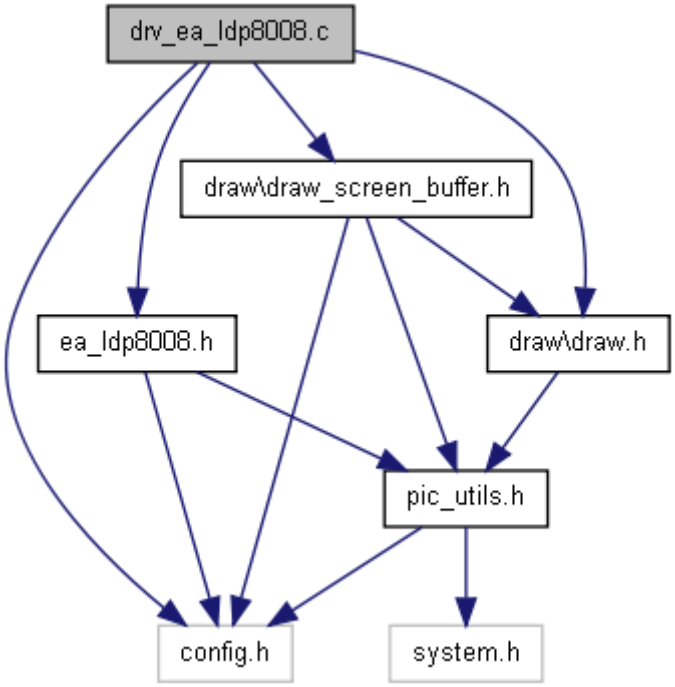
## drv_ea_ldp8008.c File Reference

Draw drivers for Embedded Adventures LDP-8008 LED panel and similar.

```
#include "ea_ldp8008.h"
#include "config.h"
```

```
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_ea_ldp8008.c:




## Defines

- #define MAX_BRIGHTNESS 3
- #define set_pins_r_g()

## Functions

- uns8 drv_get_pixel (uns8 x, uns8 y)
- void drv_init ()
- void drv_paint ()
- void drv_print_buffer ()
- void drv_refresh ()
- void drv_set_display_brightness (uns8 brightness)
- void drv_setup_io ()

## Variables

- uns8 bright_count = 0
- uns8 bright_level = 3
- char buffer0 [256]
- uns8 buffer_position0 = 0
- uns8 current_row = 0

---

## Detailed Description

Definition in file drv_ea_ldp8008.c.

## Define Documentation

### #define MAX_BRIGHTNESS 3

Definition at line 57 of file drv_ea_ldp8008.c.

### #define set_pins_r_g()

```
Value:set_pin(ea_ldp8008_r_port, ea_ldp8008_r_pin); \
    set_pin(ea_ldp8008_g_port, ea_ldp8008_g_pin);
```

Definition at line 48 of file drv_ea_ldp8008.c.

Referenced by drv_refresh().

## Function Documentation

### uns8 drv_get_pixel (uns8 *x*, uns8 *y*)

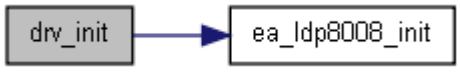
Definition at line 290 of file drv_ea_ldp8008.c.

### void drv_init ()

Definition at line 298 of file drv_ea_ldp8008.c.

References ea_ldp8008_init().

Here is the call graph for this function:




### void drv_paint ()

Definition at line 72 of file drv_ea_ldp8008.c.

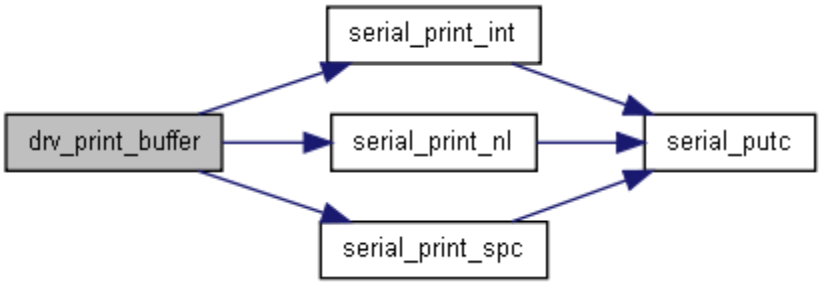
References buffer0, buffer1, draw_buffer0, end_crit_sec, start_crit_sec, and uns8.

### void drv_print_buffer ()

Definition at line 93 of file drv_ea_ldp8008.c.

References buffer0, serial_print_int(), serial_print_nl(), serial_print_spc(), and uns8.

Here is the call graph for this function:




**void drv_refresh ()**

Definition at line 115 of file drv_ea_ldp8008.c.

References bright_count, bright_level, buffer0, buffer_position0, clear_pin, current_row, MAX_BRIGHTNESS, set_pin, set_pins_r_g, and uns8.

**void drv_set_display_brightness (uns8 *brightness*)**

Definition at line 107 of file drv_ea_ldp8008.c.

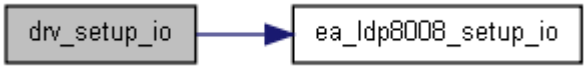
References bright_level, and MAX_BRIGHTNESS.

**void drv_setup_io ()**

Definition at line 294 of file drv_ea_ldp8008.c.

References ea_ldp8008_setup_io().

Here is the call graph for this function:




---

## Variable Documentation

**uns8 bright_count = 0**

Definition at line 54 of file drv_ea_ldp8008.c.

**uns8 bright_level = 3**

Definition at line 55 of file drv_ea_ldp8008.c.

**char buffer0[256]**

Definition at line 61 of file drv_ea_ldp8008.c.

**uns8 buffer_position0 = 0**

Definition at line 56 of file drv_ea_ldp8008.c.

**uns8 current_row = 0**

Definition at line 52 of file drv_ea_ldp8008.c.

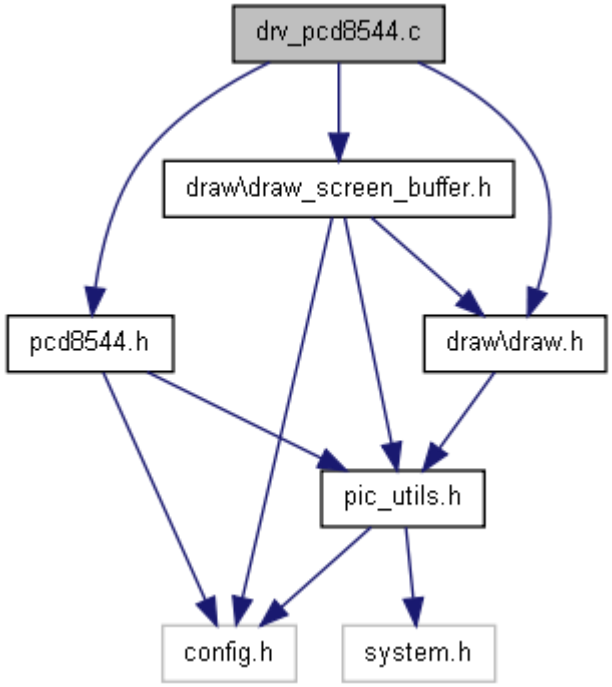
---

# drv_pcd8544.c File Reference

Draw drivers for PCD8544 based LCD display (Nokia 3310).

```
#include "pcd8544.h"
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_pcd8544.c:




## Functions

- void drv_init ()
- void drv_paint ()
- void drv_setup_io ()

---

## Detailed Description

Definition in file drv_pcd8544.c.

---

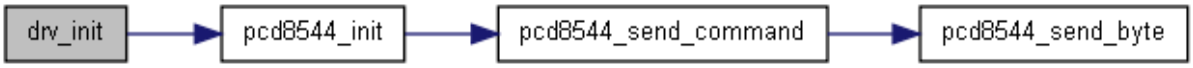
## Function Documentation

### void drv_init ()

Definition at line 89 of file drv_pcd8544.c.

References pcd8544_init().

Here is the call graph for this function:

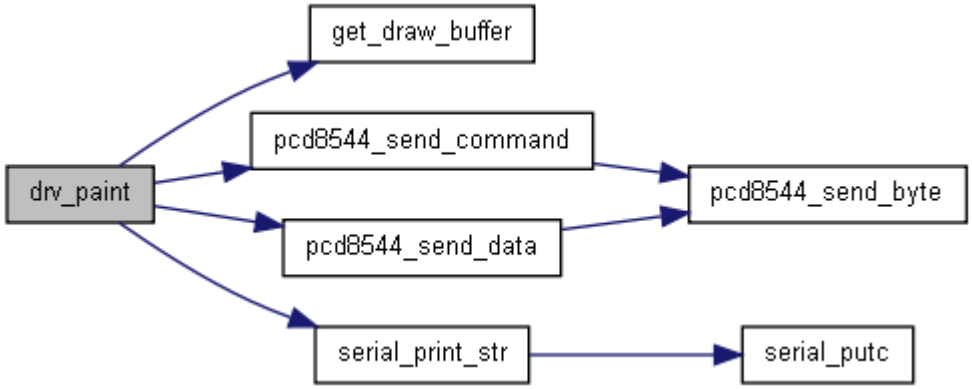



### void drv_paint ()

Definition at line 45 of file drv_pcd8544.c.

References DRAW_PIXELS_PER_BYTE, get_draw_buffer(), pcd8544_send_command(), pcd8544_send_data(), serial_print_str(), uns16, and uns8.

Here is the call graph for this function:

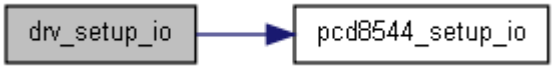



### void drv_setup_io ()

Definition at line 85 of file drv_pcd8544.c.

References pcd8544_setup_io().

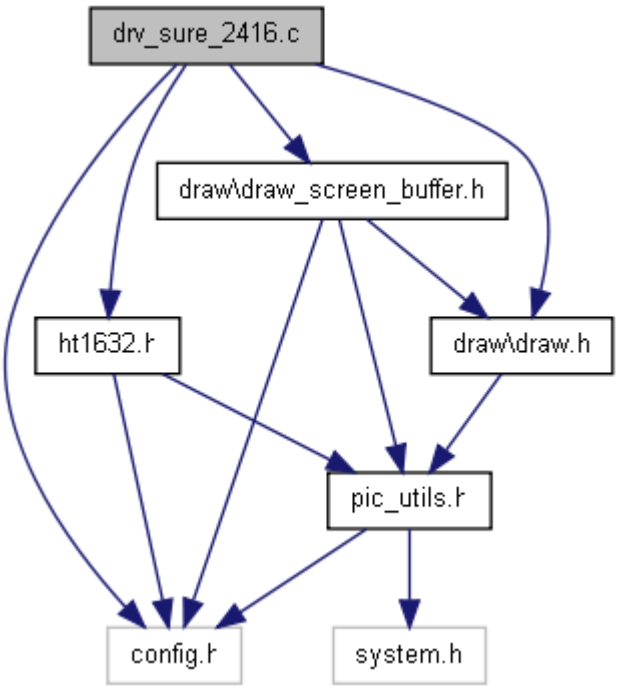
Here is the call graph for this function:



# drv_sure_2416.c File Reference

```
#include "config.h"
#include "ht1632.h"
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_sure_2416.c:




## Functions

- void drv_clear_screen ()
- uns8 drv_get_pixel (uns8 x, uns8 y)
- void drv_init ()
- void drv_paint ()
- void drv_setup_io ()

## Function Documentation

### void drv_clear_screen ()

Definition at line 203 of file drv_sure_2416.c.

### uns8 drv_get_pixel (uns8 *x*, uns8 *y*)

Definition at line 199 of file drv_sure_2416.c.

### void drv_init ()

Definition at line 211 of file drv_sure_2416.c.

References HT1632_CMD_PMOS_16_COMMON, and ht1632_init().

Here is the call graph for this function:


### void drv_paint ()

Definition at line 137 of file drv_sure_2416.c.

References clear_pin, draw_buffer0, set_pin, uns16, and uns8.

### void drv_setup_io ()

Definition at line 207 of file drv_sure_2416.c.

References ht1632_setup_io().

Here is the call graph for this function:

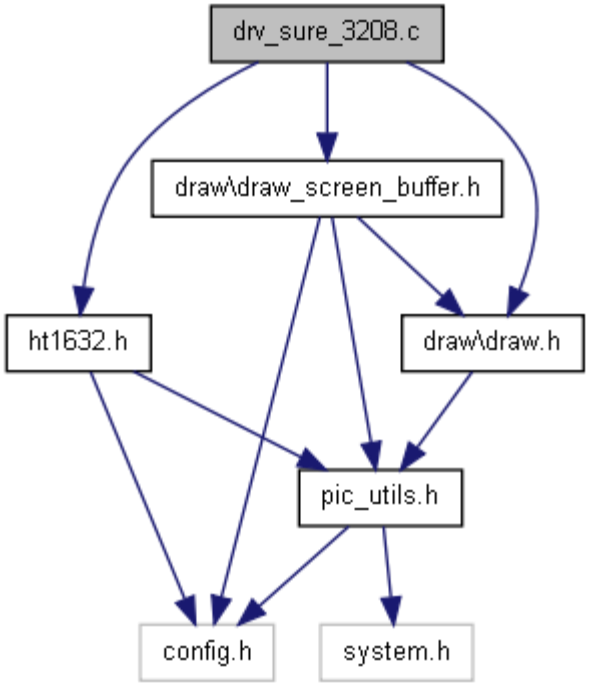

---

## drv_sure_3208.c File Reference

Draw drivers for HT1632 based displays such as Sure 3208 and similar.

```
#include "ht1632.h"
#include "draw\draw.h"
#include "draw\draw_screen_buffer.h"
```

Include dependency graph for drv_sure_3208.c:




## Functions

- void drv_clear_screen ()
- uns8 drv_get_pixel (uns8 x, uns8 y)
- void drv_init ()
- void drv_paint ()
- void drv_setup_io ()

---

## Detailed Description

Definition in file drv_sure_3208.c.

---

## Function Documentation

### void drv_clear_screen ()

Definition at line 161 of file drv_sure_3208.c.

### uns8 drv_get_pixel (uns8 *x*, uns8 *y*)

Definition at line 157 of file drv_sure_3208.c.

### void drv_init ()

Definition at line 169 of file drv_sure_3208.c.

References ht1632_init().

Referenced by draw_init().

Here is the call graph for this function:


Here is the caller graph for this function:


### void drv_paint ()

Definition at line 47 of file drv_sure_3208.c.

References clear_pin, draw_buffer0, set_pin, uns16, and uns8.

### void drv_setup_io ()

Definition at line 165 of file drv_sure_3208.c.

References ht1632_setup_io().

Referenced by draw_setup_io().

Here is the call graph for this function:

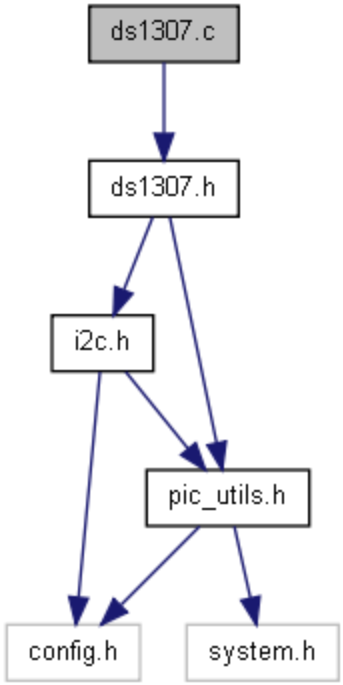

Here is the caller graph for this function:


---

## ds1307.c File Reference

```
#include "ds1307.h"
```

Include dependency graph for ds1307.c:




## Functions

- uns8 bcd_to_dec (uns8 bcd)
- uns8 dec_to_bcd (uns8 dec)
- uns8 rtc_get_config ()
- *Get the config register from the ds1307.* uns8 rtc_get_date ()
- *Get the date register from the ds1307.* uns8 rtc_get_day ()
- *Get the day register from the ds1307.* uns8 rtc_get_hours ()
- *Get the decoded hours register from the ds1307.* uns8 rtc_get_minutes ()
- *Get the decoded minutes register from the ds1307.* uns8 rtc_get_month ()
- *Get the month register from the ds1307.* uns8 rtc_get_seconds ()
- *Get the decoded seconds register from the ds1307.* uns8 rtc_get_year ()
- *Get the year register from the ds1307.* uns8 rtc_set_config (uns8 config)
- *Set the config register in the ds1307.* void rtc_set_date (uns8 date)
- *Set the date register from the ds1307.* void rtc_set_day (uns8 day)
- *Set the day of the week register from the ds1307.* void rtc_set_hours (uns8 hours)
- *Set the hours register in the ds1307.* void rtc_set_minutes (uns16 minutes)
- *Set the minutes register from the ds1307.* void rtc_set_month (uns8 month)
- *Set the month register in the ds1307.* void rtc_set_seconds (uns8 seconds)
- *Set the seconds register in the ds1307.* void rtc_set_year (uns16 year)
- *Set the year register from the ds1307.* void rtc_setup_io ()
- *Setup ports and pins for use in the ds1307.* void rtc_start_clock ()
- *Starts the clock in the ds1307.* void rtc_stop_clock ()

*Stop the clock in the ds1307.*

---

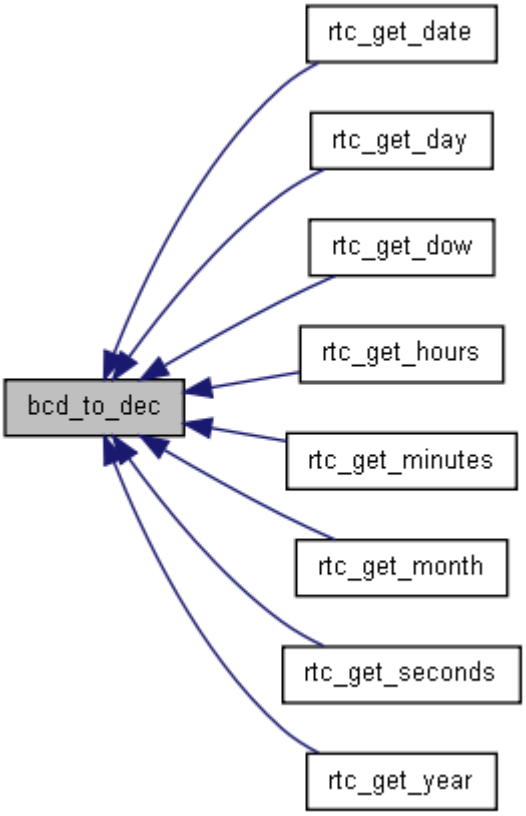
## Function Documentation

### uns8 bcd_to_dec (uns8 *bcd*)

Definition at line 39 of file ds1307.c.

Referenced by rtc_get_date(), rtc_get_day(), rtc_get_dow(), rtc_get_hours(), rtc_get_minutes(), rtc_get_month(), rtc_get_seconds(), and rtc_get_year().

Here is the caller graph for this function:

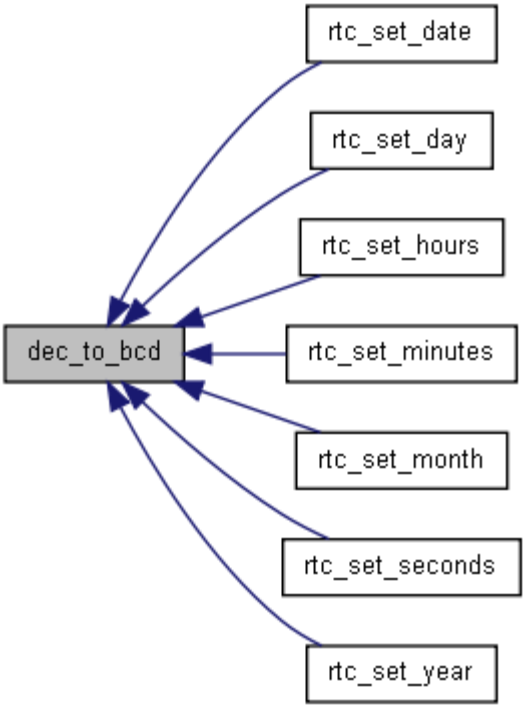



### uns8 dec_to_bcd (uns8 *dec*)

Definition at line 43 of file ds1307.c.

Referenced by rtc_set_date(), rtc_set_day(), rtc_set_hours(), rtc_set_minutes(), rtc_set_month(), rtc_set_seconds(), and rtc_set_year().

Here is the caller graph for this function:




**uns8 rtc_get_config ()**

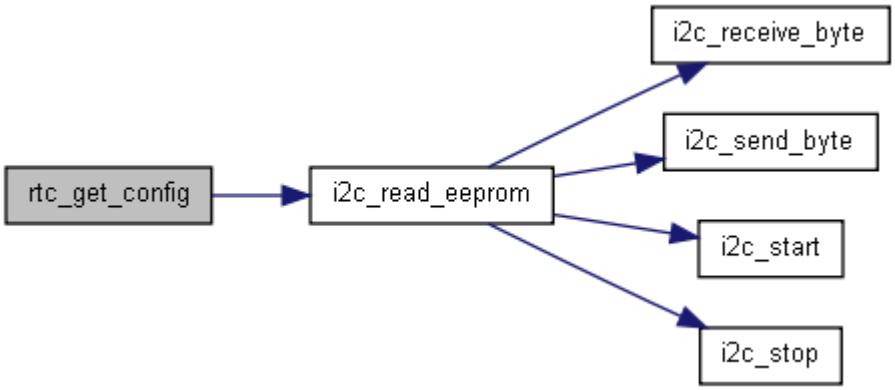
Returns the config register from the ds1307. Bit 7 - Out - Value on SQWE pin if not outputting square wave Bit 6 - 0 Bit 5 - 0 Bit 4 - SQWE - Enable square wave output Bit 3 - 0 Bit 2 - 0 Bit 1 - RS1 Bit 0 - RS0

RS1/0 determin the speed of the square wave output. Set to 0/0 for 1 Hz.

Definition at line 76 of file ds1307.c.

References ds1307_control_register, ds1307_device, and i2c_read_eeprom().

Here is the call graph for this function:




**uns8 rtc_get_date ()**

Get the date register from the m41t81s.

Returns the date in month from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 28/29/30/31 depending on month

Definition at line 65 of file ds1307.c.

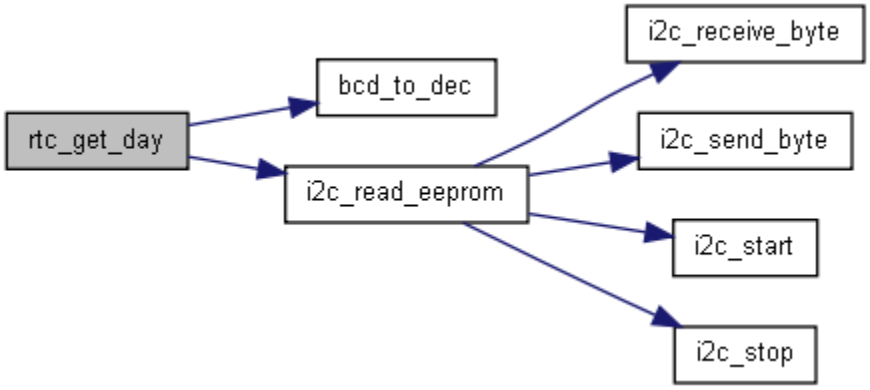
### uns8 rtc_get_day ()

Returns the day of the week from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 1 through 7

Definition at line 61 of file ds1307.c.

References bcd_to_dec(), ds1307_day_register, ds1307_device, and i2c_read_eeprom().

Here is the call graph for this function:




### uns8 rtc_get_hours ()

Get the decoded hours register from the m41t81s.

Returns hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. These routines assume the ds1307 is running in 24 hour mode. Range - 0 through 23

Definition at line 50 of file ds1307.c.

### uns8 rtc_get_minutes ()

Get the decoded minutes register from the m41t81s.

Returns the number of minutes past the hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 47 of file ds1307.c.

### uns8 rtc_get_month ()

Get the month register from the m41t81s.

Returns the month of the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 12

Definition at line 69 of file ds1307.c.

### uns8 rtc_get_seconds ()

Get the decoded seconds register from the m41t81s.

Returns seconds from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 57 of file ds1307.c.

### uns8 rtc_get_year ()

Get the year register from the m41t81s.

Returns the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 0 through 99

Definition at line 72 of file ds1307.c.

### uns8 rtc_set_config (uns8 *config*)

Set the config register in the m41t81s.

Sets the config register in the ds1307.

Bit 7 - Out - Value on SQWE pin if not outputting square wave Bit 6 - 0 Bit 5 - 0 Bit 4 - SQWE - Enable square wave output Bit 3 - 0 Bit 2 - 0 Bit 1 - RS1 Bit 0 - RS0

RS1/0 determin the speed of the square wave output. Set to 0/0 for 1 Hz.

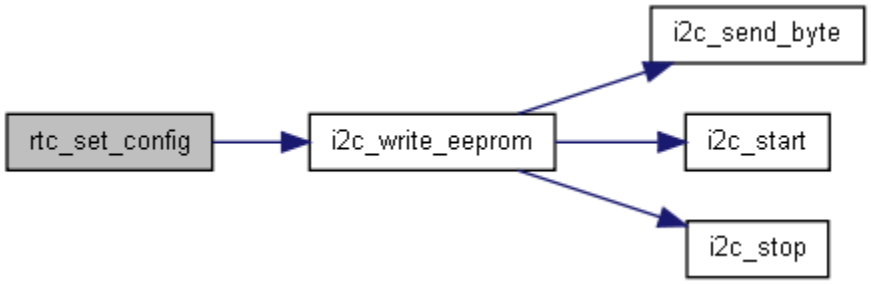
**Parameters:**

*config* Value to set the config register to

Definition at line 80 of file ds1307.c.

References ds1307_control_register, ds1307_device, and i2c_write_eeprom().

Here is the call graph for this function:




### void rtc_set_date (uns8 *date*)

Set the date register from the m41t81s.

Changes the date in the ds1307.

**Parameters:**

*seconds* Value to set date to

Definition at line 102 of file ds1307.c.

### void rtc_set_day (uns8 *day*)

Set the day of the week register from the m41t81s.

Changes the day of the week in the ds1307.

**Parameters:**

*seconds* Value to set day to

Definition at line 99 of file ds1307.c.

### void rtc_set_hours (uns8 *hours*)

Set the hours register in the m41t81s.

Changes the hours in the ds1307. Forces the ds1307 into 24 hour mode.

Definition at line 110 of file ds1307.c.

### void rtc_set_minutes (uns16 *minutes*)

Changes the minutes in the ds1307.

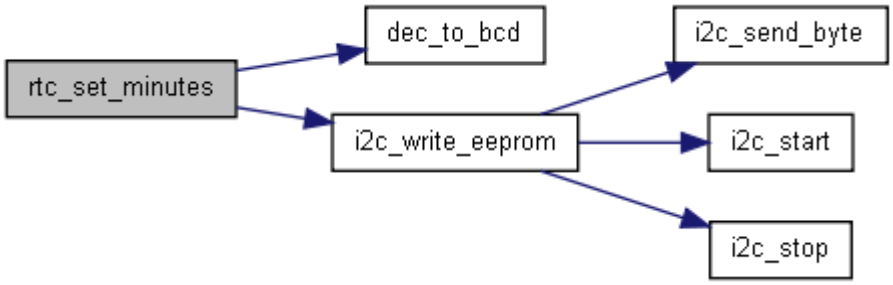
**Parameters:**

*seconds* Value to set minutes to

Definition at line 96 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_minutes_register, and i2c_write_eeprom().

Here is the call graph for this function:




### void rtc_set_month (uns8 *month*)

Set the month register in the m41t81s.

Changes the month in the ds1307.

Definition at line 115 of file ds1307.c.

### void rtc_set_seconds (uns8 *seconds*)

Set the seconds register in the m41t81s.

Changes the seconds in the ds1307.

**Parameters:**

*seconds* Value to set seconds to

Definition at line 106 of file ds1307.c.

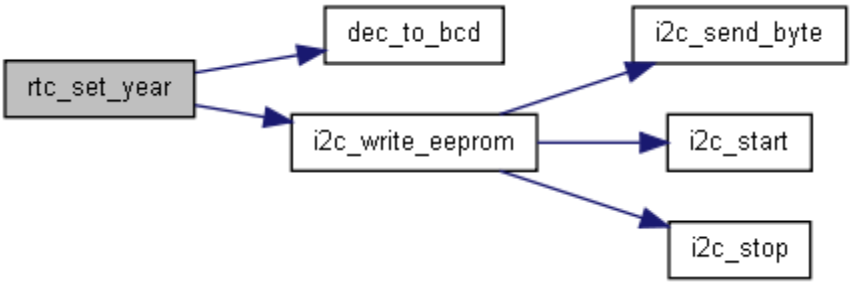
### void rtc_set_year (uns16 *year*)

Changes the year in the ds1307.

Definition at line 93 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_year_register, and i2c_write_eeprom().

Here is the call graph for this function:




**void rtc_setup_io ()**

Setup ports and pins for use in the m41t81s.

Calls i2c_setup() to configure ports and pins ready for use

Definition at line 119 of file ds1307.c.

**void rtc_start_clock ()**

Starts the clock in the m41t81s.

Resume time in the ds1307

Definition at line 89 of file ds1307.c.

**void rtc_stop_clock ()**

Stop the clock in the m41t81s.

Pauses time in the ds1307

Definition at line 85 of file ds1307.c.

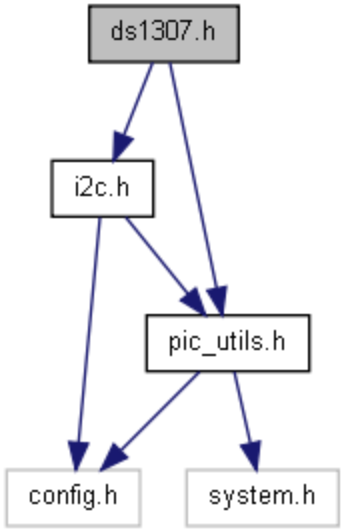
---

## ds1307.h File Reference

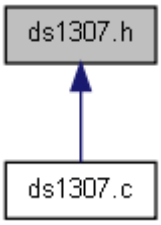
Routines for communicating with the ds1307 real time clock.

```
#include "i2c.h"
#include "pic_utils.h"
```

Include dependency graph for ds1307.h:




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




## Defines

- #define __ds1307_h defined
- #define ds1307_control_register 0x07
- #define ds1307_date_register 0x04
- #define ds1307_day_register 0x03
- #define ds1307_device 0xD0
- #define ds1307_hours_register 0x02
- #define ds1307_minutes_register 0x01
- #define ds1307_month_register 0x05
- #define ds1307_seconds_register 0x00
- #define ds1307_year_register 0x06
- #define rtc_setup() rtc_setup_io()

## Functions

- uns8 rtc_get_config ()
- *Get the config register from the ds1307.* uns8 rtc_get_date ()
- *Get the date register from the ds1307.* uns8 rtc_get_day ()
- *Get the day register from the ds1307.* uns8 rtc_get_hours ()
- *Get the decoded hours register from the ds1307.* uns8 rtc_get_minutes ()
- *Get the decoded minutes register from the ds1307.* uns8 rtc_get_month ()
- *Get the month register from the ds1307.* uns8 rtc_get_seconds ()
- *Get the decoded seconds register from the ds1307.* uns8 rtc_get_year ()
- *Get the year register from the ds1307.* uns8 rtc_set_config (uns8 config)

- *Set the config register in the ds1307.* void rtc_set_date (uns8 date)
- *Set the date register from the ds1307.* void rtc_set_day (uns8 day)
- *Set the day of the week register from the ds1307.* void rtc_set_hours (uns8 hours)
- *Set the hours register in the ds1307.* void rtc_set_minutes (uns16 minutes)
- *Set the minutes register from the ds1307.* void rtc_set_month (uns8 month)
- *Set the month register in the ds1307.* void rtc_set_seconds (uns8 seconds)
- *Set the seconds register in the ds1307.* void rtc_set_year (uns16 year)
- *Set the year register from the ds1307.* void rtc_setup_io ()
- *Setup ports and pins for use in the ds1307.* void rtc_start_clock ()
- *Starts the clock in the ds1307.* void rtc_stop_clock ()

*Stop the clock in the ds1307.*

---

## Detailed Description

Definition in file ds1307.h.

---

## Define Documentation

### #define __ds1307_h defined

Definition at line 44 of file ds1307.h.

### #define ds1307_control_register 0x07

ds1307 control register

Definition at line 67 of file ds1307.h.

Referenced by rtc_get_config(), and rtc_set_config().

### #define ds1307_date_register 0x04

ds1307 date in month register

Definition at line 61 of file ds1307.h.

Referenced by rtc_get_date(), and rtc_set_date().

### #define ds1307_day_register 0x03

ds1307 day of week register

Definition at line 59 of file ds1307.h.

Referenced by rtc_get_day(), and rtc_set_day().

### #define ds1307_device 0xD0

The ds1307 device address

Definition at line 50 of file ds1307.h.

Referenced by rtc_get_config(), rtc_get_date(), rtc_get_day(), rtc_get_hours(), rtc_get_minutes(), rtc_get_month(), rtc_get_seconds(), rtc_get_year(), rtc_set_config(), rtc_set_date(), rtc_set_day(), rtc_set_hours(), rtc_set_minutes(), rtc_set_month(), rtc_set_seconds(), rtc_set_year(), rtc_start_clock(), and rtc_stop_clock().

#### #define ds1307_hours_register 0x02

ds1307 hours register

Definition at line 57 of file ds1307.h.

Referenced by rtc_get_hours(), and rtc_set_hours().

#### #define ds1307_minutes_register 0x01

ds1307 minutes register

Definition at line 55 of file ds1307.h.

Referenced by rtc_get_minutes(), and rtc_set_minutes().

#### #define ds1307_month_register 0x05

ds1307 month register

Definition at line 63 of file ds1307.h.

Referenced by rtc_get_month(), and rtc_set_month().

#### #define ds1307_seconds_register 0x00

ds1307 seconds register

Definition at line 53 of file ds1307.h.

Referenced by rtc_get_seconds(), rtc_set_seconds(), rtc_start_clock(), and rtc_stop_clock().

#### #define ds1307_year_register 0x06

ds1307 year register

Definition at line 65 of file ds1307.h.

Referenced by rtc_get_year(), and rtc_set_year().

#### #define rtc_setup() rtc_setup_io()

Definition at line 264 of file ds1307.h.

---

## Function Documentation

#### uns8 rtc_get_config ()

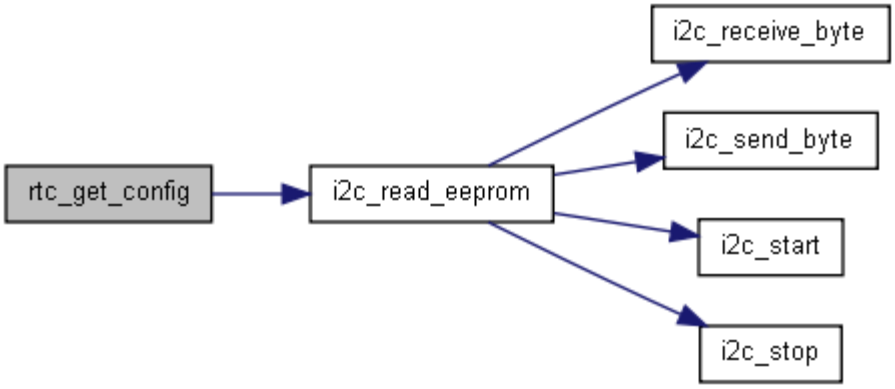
Returns the config register from the ds1307. Bit 7 - Out - Value on SQWE pin if not outputting square wave Bit 6 - 0 Bit 5 - 0 Bit 4 - SQWE - Enable square wave output Bit 3 - 0 Bit 2 - 0 Bit 1 - RS1 Bit 0 - RS0

RS1/0 determin the speed of the square wave output. Set to 0/0 for 1 Hz.

Definition at line 76 of file ds1307.c.

References ds1307_control_register, ds1307_device, and i2c_read_eeprom().

Here is the call graph for this function:




**uns8 rtc_get_date ()**

Returns the date in month from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 28/29/30/31 depending on month

Definition at line 65 of file ds1307.c.

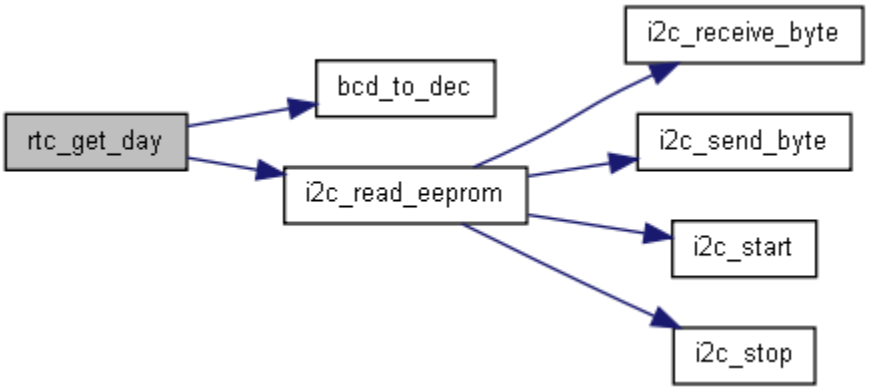
**uns8 rtc_get_day ()**

Returns the day of the week from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 1 through 7

Definition at line 61 of file ds1307.c.

References bcd_to_dec(), ds1307_day_register, ds1307_device, and i2c_read_eeprom().

Here is the call graph for this function:




**uns8 rtc_get_hours ()**

Returns hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. These routines assume the ds1307 is running in 24 hour mode. Range - 0 through 23

Definition at line 50 of file ds1307.c.

**uns8 rtc_get_minutes ()**

Returns the number of minutes past the hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 47 of file ds1307.c.

**uns8 rtc_get_month ()**

Returns the month of the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 12

Definition at line 69 of file ds1307.c.

**uns8 rtc_get_seconds ()**

Returns seconds from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 57 of file ds1307.c.

**uns8 rtc_get_year ()**

Returns the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 0 through 99

Definition at line 72 of file ds1307.c.

**uns8 rtc_set_config (uns8 *config*)**

Sets the config register in the ds1307.

Bit 7 - Out - Value on SQWE pin if not outputting square wave Bit 6 - 0 Bit 5 - 0 Bit 4 - SQWE - Enable square wave output Bit 3 - 0 Bit 2 - 0 Bit 1 - RS1 Bit 0 - RS0

RS1/0 determin the speed of the square wave output. Set to 0/0 for 1 Hz.

**Parameters:**

*config* Value to set the config register to

Definition at line 80 of file ds1307.c.

**void rtc_set_date (uns8 *date*)**

Changes the date in the ds1307.

**Parameters:**

*seconds* Value to set date to

Definition at line 102 of file ds1307.c.

**void rtc_set_day (uns8 *day*)**

Changes the day of the week in the ds1307.

**Parameters:**

*seconds* Value to set day to

Definition at line 99 of file ds1307.c.

**void rtc_set_hours (uns8 *hours*)**

Changes the hours in the ds1307. Forces the ds1307 into 24 hour mode.

Definition at line 110 of file ds1307.c.

**void rtc_set_minutes (uns16 *minutes*)**

Changes the minutes in the ds1307.

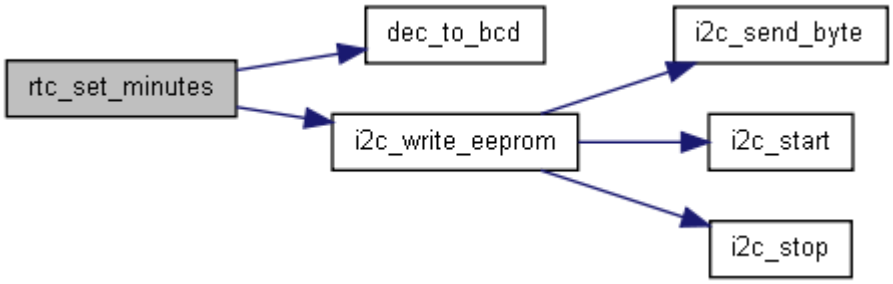
**Parameters:**

*seconds* Value to set minutes to

Definition at line 96 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_minutes_register, and i2c_write_eeprom().

Here is the call graph for this function:




### void rtc_set_month (uns8 *month*)

Changes the month in the ds1307.

Definition at line 115 of file ds1307.c.

### void rtc_set_seconds (uns8 *seconds*)

Changes the seconds in the ds1307.

**Parameters:**

*seconds* Value to set seconds to

Definition at line 106 of file ds1307.c.

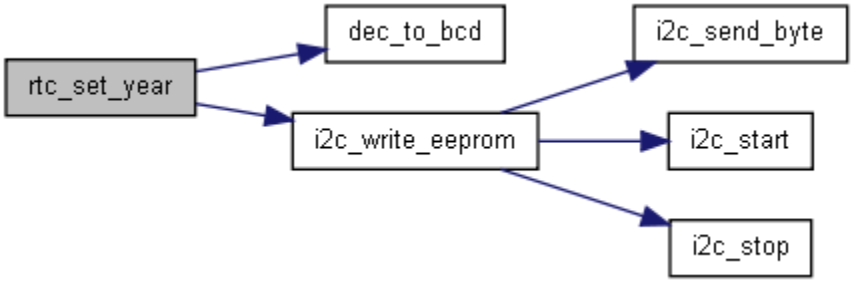
### void rtc_set_year (uns16 *year*)

Changes the year in the ds1307.

Definition at line 93 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_year_register, and i2c_write_eeprom().

Here is the call graph for this function:




### void rtc_setup_io ()

Calls i2c_setup() to configure ports and pins ready for use

Definition at line 119 of file ds1307.c.

**void rtc_start_clock ()**

Resume time in the ds1307

Definition at line 89 of file ds1307.c.

**void rtc_stop_clock ()**

Pauses time in the ds1307

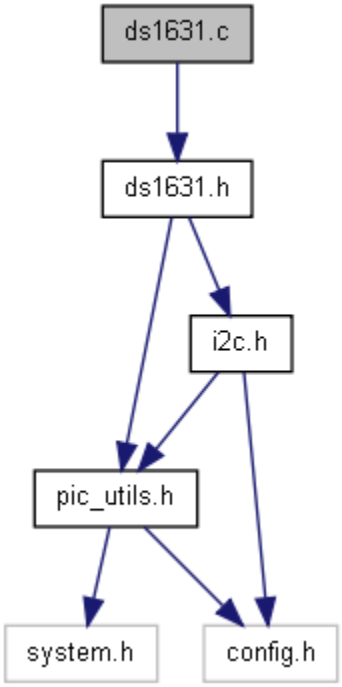
Definition at line 85 of file ds1307.c.

---

## ds1631.c File Reference

```
#include "ds1631.h"
```

Include dependency graph for ds1631.c:




### Functions

- void ds1631_convert_temp (uns8 addr)
- *Start temperature conversion on ds1631.* uns8 ds1631_get_config (uns8 addr)
- *Get ds1631 config register.* uns16 ds1631_get_temp (uns8 addr)
- *Read temperature from ds1631.* void ds1631_set_config (uns8 addr, uns8 config)
- *Set ds1631 config register.* void ds1631_setup ()

---

## Function Documentation

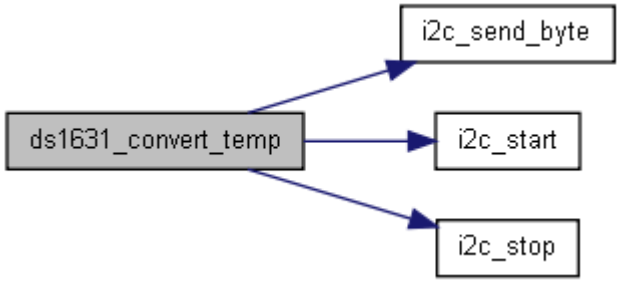
### void ds1631_convert_temp (uns8 *addr*)

This routine starts the temperature conversion in the ds1631. Issue this command before actually reading the temperature.

Definition at line 52 of file ds1631.c.

References ds1631_start_convert, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:



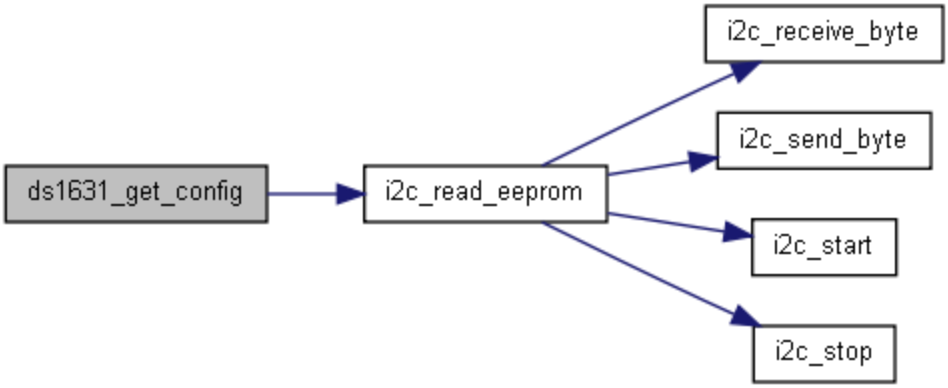

### uns8 ds1631_get_config (uns8 *addr*)

Gets the ds1631 config register (memory location 0x01)

Definition at line 47 of file ds1631.c.

References ds1631_access_config, and i2c_read_eeprom().

Here is the call graph for this function:



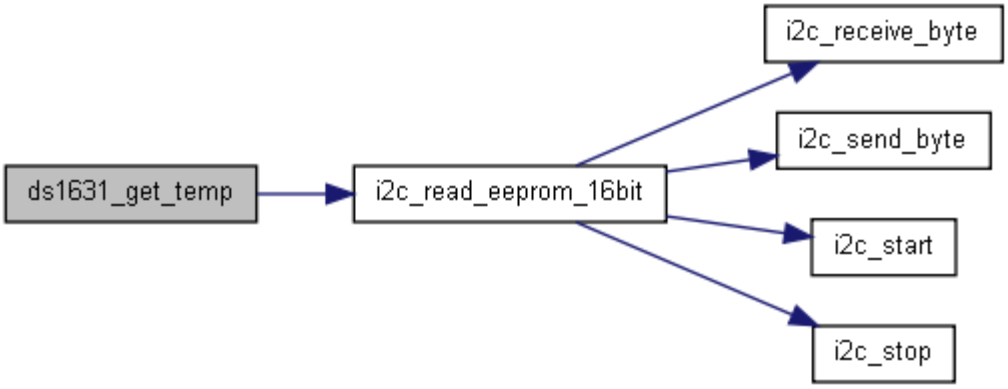

### uns16 ds1631_get_temp (uns8 *addr*)

Returns 16bit raw temperature register from ds1631. Note that if you are in one-shot mode (the default) you must have already issued a start_convert and waited until it is complete (to check for completion you can either wait long enough, or query the config register to check).

Definition at line 59 of file ds1631.c.

References ds1631_read_temp, and i2c_read_eeprom_16bit().

Here is the call graph for this function:



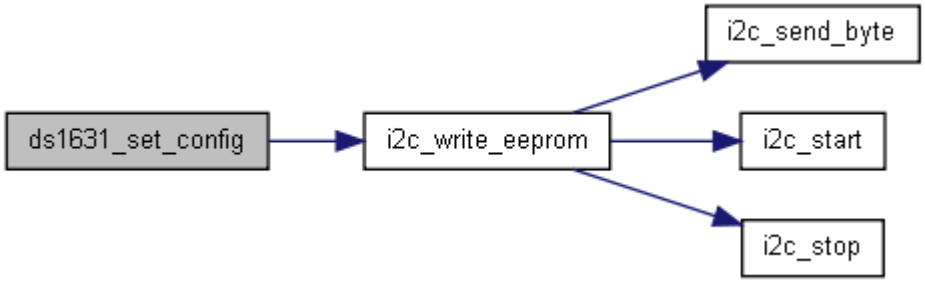

**void ds1631_set_config (uns8 *addr*, uns8 *config*)**

Sets the ds1631 config register

Definition at line 42 of file ds1631.c.

References ds1631_access_config, and i2c_write_eeprom().

Here is the call graph for this function:




**void ds1631_setup ()**

Definition at line 38 of file ds1631.c.

References i2c_setup.

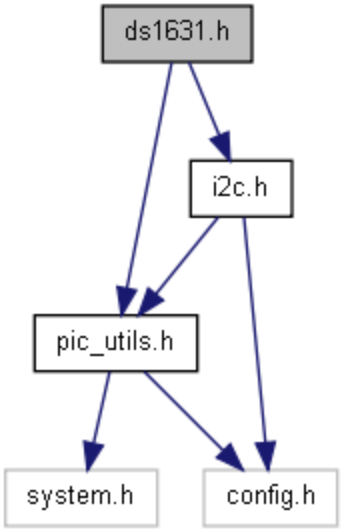
---

## ds1631.h File Reference

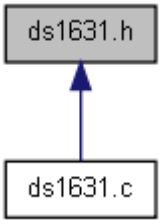
DS1631 temperature sensor routines.

```
#include "pic_utils.h"
#include "i2c.h"
```

Include dependency graph for ds1631.h:




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




## Defines

- #define ds1631_access_config 0xAC
- #define ds1631_access_th 0xA1
- #define ds1631_access_tl 0xA2
- #define ds1631_read_temp 0xAA
- #define ds1631_setup() ds1631_setup_io()
- *Setup ds1631 ports and pins.* #define ds1631_software_por 0x54
- #define ds1631_start_convert 0x51
- #define ds1631_stop_convert 0x22

## Functions

- void ds1631_convert_temp (uns8 addr)
- *Start temperature conversion on ds1631.* uns8 ds1631_get_config (uns8 addr)
- *Get ds1631 config register.* uns16 ds1631_get_temp (uns8 addr)
- *Read temperature from ds1631.* void ds1631_set_config (uns8 addr, uns8 config)
- *Set ds1631 config register.* void ds1631_setup_io (void)

---

## Detailed Description

Definition in file ds1631.h.

---

## Define Documentation

### #define ds1631_access_config 0xAC

Definition at line 50 of file ds1631.h.

Referenced by ds1631_get_config(), and ds1631_set_config().

### #define ds1631_access_th 0xA1

Definition at line 48 of file ds1631.h.

### #define ds1631_access_tl 0xA2

Definition at line 49 of file ds1631.h.

### #define ds1631_read_temp 0xAA

Definition at line 47 of file ds1631.h.

Referenced by ds1631_get_temp().

### #define ds1631_setup() ds1631_setup_io()

Definition at line 58 of file ds1631.h.

### #define ds1631_software_por 0x54

Definition at line 51 of file ds1631.h.

### #define ds1631_start_convert 0x51

Definition at line 45 of file ds1631.h.

Referenced by ds1631_convert_temp().

### #define ds1631_stop_convert 0x22

Definition at line 46 of file ds1631.h.

---

## Function Documentation

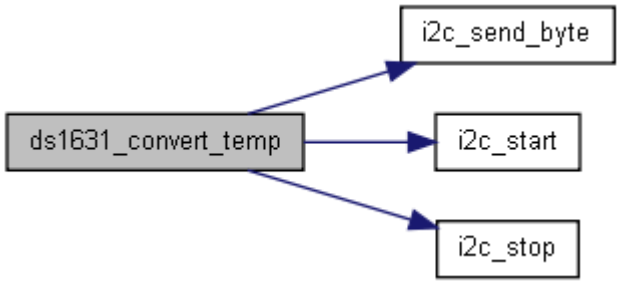
### void ds1631_convert_temp (uns8 *addr*)

This routine starts the temperature conversion in the ds1631. Issue this command before actually reading the temperature.

Definition at line 52 of file ds1631.c.

References ds1631_start_convert, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:



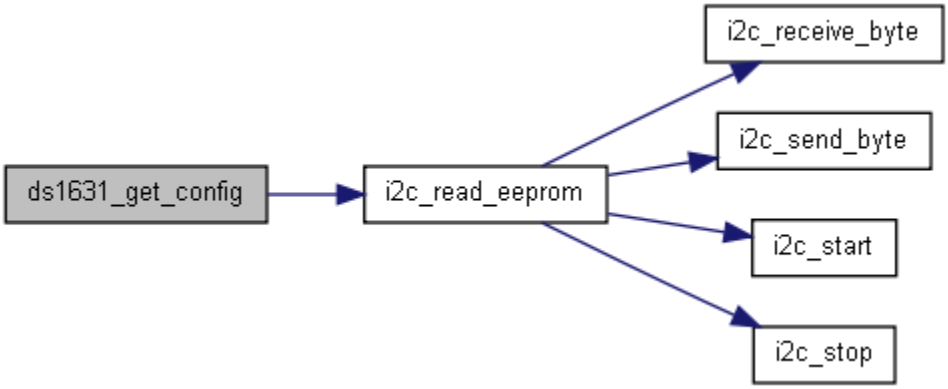

**uns8 ds1631_get_config (uns8 *addr*)**

Gets the ds1631 config register (memory location 0x01)

Definition at line 47 of file ds1631.c.

References ds1631_access_config, and i2c_read_eeprom().

Here is the call graph for this function:



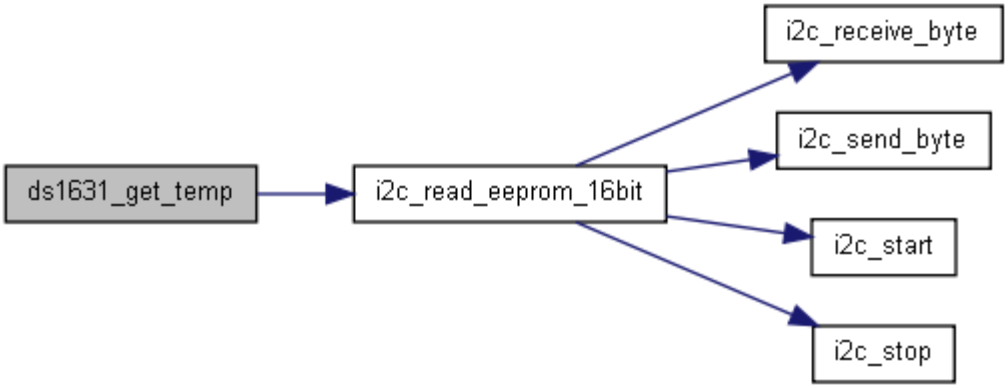

**uns16 ds1631_get_temp (uns8 *addr*)**

Returns 16bit raw temperature register from ds1631. Note that if you are in one-shot mode (the default) you must have already issued a start_convert and waited until it is complete (to check for completion you can either wait long enough, or query the config register to check).

Definition at line 59 of file ds1631.c.

References ds1631_read_temp, and i2c_read_eeprom_16bit().

Here is the call graph for this function:



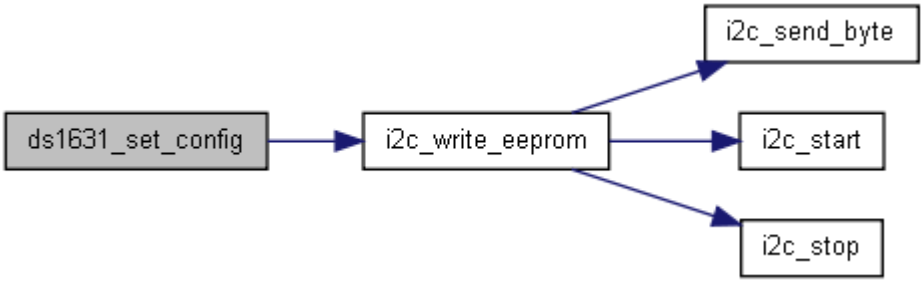

**void ds1631_set_config (uns8 *addr*, uns8 *config*)**

Sets the ds1631 config register

Definition at line 42 of file ds1631.c.

References ds1631_access_config, and i2c_write_eeprom().

Here is the call graph for this function:




**void ds1631_setup_io (void)**

---

# ea_bitmaps.c File Reference

Bitmaps for draw library testing.

## Variables

- char a_big_bitmap []
- char a_bitmap []
- char adventures_bitmap []
- char e_big_bitmap []
- char e_bitmap []
- char embedded_bitmap []

---

## Detailed Description

Definition in file ea_bitmaps.c.

---

## Variable Documentation

### char a_big_bitmap[]

Definition at line 126 of file ea_bitmaps.c.

Referenced by draw_tests_run().

### char a_bitmap[]

```
Initial value: {
0x12,
0x10,
0x01,
0x24,
0x80,0x3F,0xC0,0x7F,0xC0,0x7F,0xE0,0xFF,0xE0,0xF1,0xE0,0xF1,0xE0,0xF1,0xE0,0xF1,
0xE6,0xF1,0xEF,0xF1,0xEF,0xF1,0xEF,0xF1,0xEF,0xF1,0xEF,0xF1,0xFF,0xFF,0xFE,0x7F,
0xFE,0x7F,0xF8,0x3F,
}
```

Definition at line 74 of file ea_bitmaps.c.

Referenced by draw_tests_run().

### char adventures_bitmap[]

```
Initial value: {
0x40,
0x08,
0x01,
0x40,
0x70,0x90,0x90,0x90,0x94,0x94,0x7C,0x00,0x78,0x84,0x84,0x84,0x84,0x84,0x7F,0x00,
0x0C,0x18,0x60,0xC0,0x70,0x18,0x0C,0x00,0x78,0xA4,0xA4,0x24,0x24,0x38,0x00,0xF8,
0x04,0x04,0x04,0x04,0x04,0xF8,0x04,0xFF,0x04,0x04,0x7C,0x80,0x80,0x80,0x80,0xFC,
0x00,0xF8,0x04,0x04,0x00,0x78,0xA4,0xA4,0x24,0x24,0x38,0x00,0x98,0xA4,0xA4,0xC4,
}
```

Definition at line 52 of file ea_bitmaps.c.

Referenced by draw_tests_run().

### char e_big_bitmap[]

Definition at line 84 of file ea_bitmaps.c.

Referenced by draw_tests_run().

### char e_bitmap[]

```
Initial value: {
0x11,
0x10,
0x01,
0x22,

0xF8,0x1F,0xFE,0x7F,0xFE,0x7F,0xFF,0xFF,0xCF,0xF3,0xCF,0xF3,0xCF,0xF3,0xCF,0xF3,
0xCF,0xF3,0xCF,0xF3,0xCF,0x03,0xCF,0x03,0xCF,0x03,0xFF,0x03,0xFE,0x03,0xFE,0x03,
0xFC,0x03,
```

```
}
```

Definition at line 63 of file ea_bitmaps.c.

Referenced by draw_tests_run().

### char embedded_bitmap[]

```
Initial value: {
0x40,
0x08,
0x01,
0x40,
0x78,0xA4,0xA4,0x24,0x24,0x38,0x00,0x00,0xF8,0x04,0x04,0xFC,0x04,0x04,0x04,0xF8,
0x00,0x7F,0x84,0x84,0x84,0x84,0x84,0x78,0x00,0x00,0x78,0xA4,0xA4,0x24,0x24,0x38,
0x00,0x78,0x84,0x84,0x84,0x84,0x84,0x7F,0x00,0x78,0x84,0x84,0x84,0x84,0x84,0x7F,
0x00,0x00,0x78,0xA4,0xA4,0x24,0x24,0x38,0x00,0x78,0x84,0x84,0x84,0x84,0x84,0x7F,
}
```

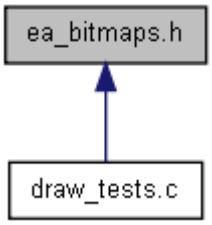
Definition at line 40 of file ea_bitmaps.c.

Referenced by draw_tests_run().

---

# ea_bitmaps.h File Reference

Bitmaps for draw library testing.
This graph shows which files directly or indirectly include this file:




## Variables

- char a_big_bitmap [1]
- char a_bitmap [1]
- char adventures_bitmap [1]
- char e_big_bitmap [1]
- char e_bitmap [1]
- char embedded_bitmap [1]

---

## Detailed Description

Definition in file ea_bitmaps.h.

---

## Variable Documentation

**char a_big_bitmap[1]**

Definition at line 126 of file ea_bitmaps.c.
Referenced by draw_tests_run().

**char a_bitmap[1]**

Definition at line 74 of file ea_bitmaps.c.
Referenced by draw_tests_run().

**char adventures_bitmap[1]**

Definition at line 52 of file ea_bitmaps.c.
Referenced by draw_tests_run().

**char e_big_bitmap[1]**

Definition at line 84 of file ea_bitmaps.c.
Referenced by draw_tests_run().

**char e_bitmap[1]**

Definition at line 63 of file ea_bitmaps.c.
Referenced by draw_tests_run().

**char embedded_bitmap[1]**

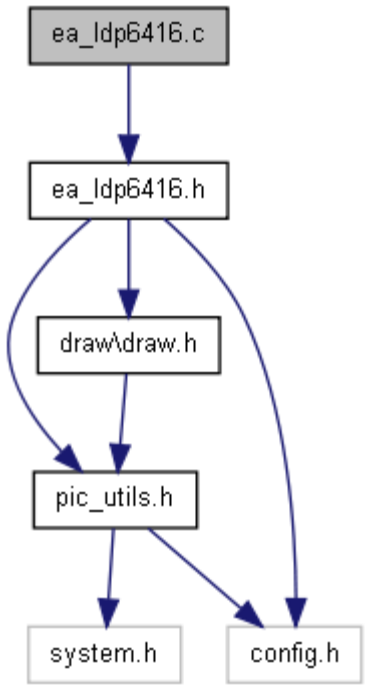
Definition at line 40 of file ea_bitmaps.c.
Referenced by draw_tests_run().

---

# ea_ldp6416.c File Reference

```
#include "ea_ldp6416.h"
```

Include dependency graph for ea_ldp6416.c:




## Functions

- void ea_ldp6416_init ()
- void ea_ldp6416_setup_io ()

---

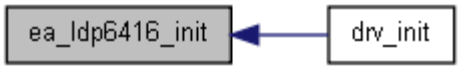
## Function Documentation

### void ea_ldp6416_init ()

Definition at line 63 of file ea_ldp6416.c.

Referenced by drv_init().

Here is the caller graph for this function:



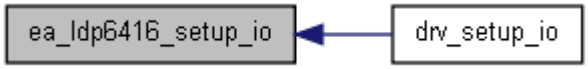

### void ea_ldp6416_setup_io ()

Definition at line 40 of file ea_ldp6416.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:

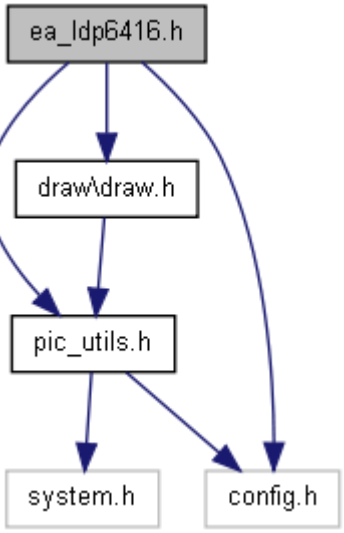


# ea_ldp6416.h File Reference

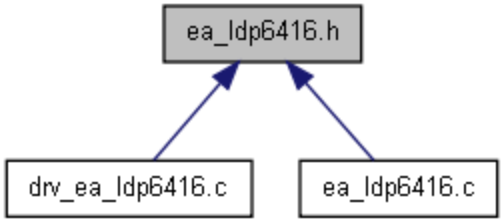
Support routines for LDP-6416 LED display panel.

```
#include "pic_utils.h"
#include "draw\draw.h"
#include "config.h"
```

Include dependency graph for ea_ldp6416.h:




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




## Defines

- #define ea_ldp6416_setup() ea_ldp6416_setup_io()

## Functions

- void ea_ldp6416_init ()
- void ea_ldp6416_setup_io ()

## Detailed Description

Definition in file ea_ldp6416.h.

---

## Define Documentation

### #define ea_ldp6416_setup()  ea_ldp6416_setup_io()

Definition at line 139 of file ea_ldp6416.h.

---

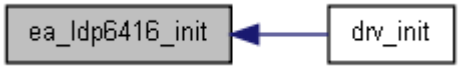
## Function Documentation

### void ea_ldp6416_init ()

Definition at line 63 of file ea_ldp6416.c.

Referenced by drv_init().

Here is the caller graph for this function:



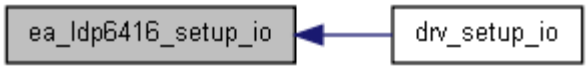

### void ea_ldp6416_setup_io ()

Definition at line 40 of file ea_ldp6416.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:

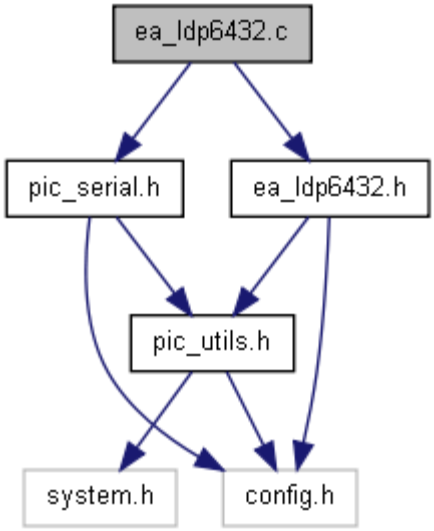



---

# ea_ldp6432.c File Reference

```
#include "ea_ldp6432.h"
#include "pic_serial.h"
```

Include dependency graph for ea_ldp6432.c:




## Functions

- void ea_ldp6432_init ()
- void ea_ldp6432_setup_io ()

---

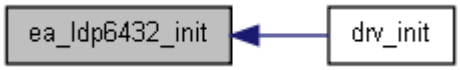
## Function Documentation

### void ea_ldp6432_init ()

Definition at line 67 of file ea_ldp6432.c.

Referenced by drv_init().

Here is the caller graph for this function:



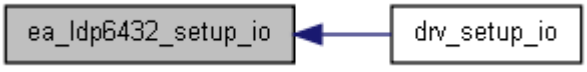

### void ea_ldp6432_setup_io ()

Definition at line 44 of file ea_ldp6432.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:



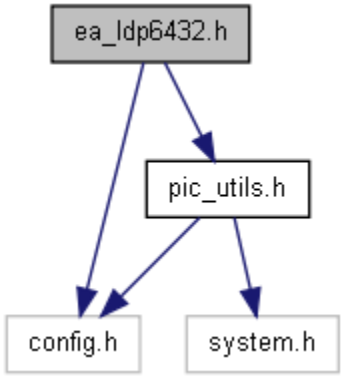

---

# ea_ldp6432.h File Reference

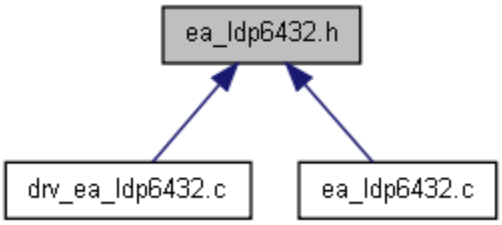
Support routines for the LDP-6432 LED display panel.

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

Include dependency graph for ea_ldp6432.h:




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




## Defines

- #define ea_ldp6432_setup() ea_ldp6432_setup_io()

## Functions

- void ea_ldp6432_init ()
- void ea_ldp6432_setup_io ()

---

## Detailed Description

Definition in file ea_ldp6432.h.

---

## Define Documentation

### #define ea_ldp6432_setup() ea_ldp6432_setup_io()

Definition at line 92 of file ea_ldp6432.h.

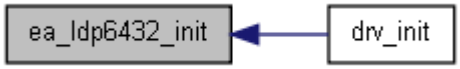
## Function Documentation

### void ea_ldp6432_init ()

Definition at line 67 of file ea_ldp6432.c.

Referenced by drv_init().

Here is the caller graph for this function:



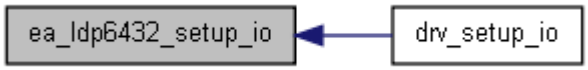

### void ea_ldp6432_setup_io ()

Definition at line 44 of file ea_ldp6432.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

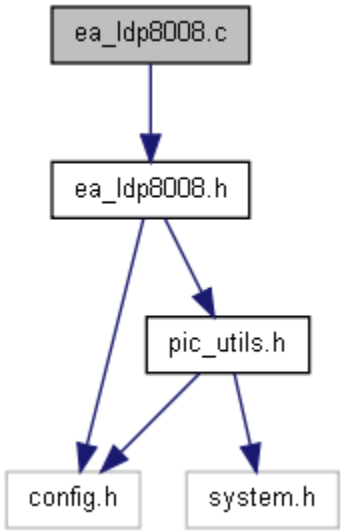
Here is the caller graph for this function:




# ea_ldp8008.c File Reference

```
#include "ea_ldp8008.h"
```

Include dependency graph for ea_ldp8008.c:



## Functions

- void ea_ldp8008_init ()
- void ea_ldp8008_setup_io ()

---

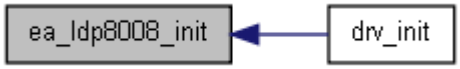
## Function Documentation

### void ea_ldp8008_init ()

Definition at line 60 of file ea_ldp8008.c.

Referenced by drv_init().

Here is the caller graph for this function:



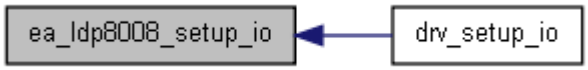

### void ea_ldp8008_setup_io ()

Definition at line 40 of file ea_ldp8008.c.

References clear_pin, and make_output.

Referenced by drv_setup_io().

Here is the caller graph for this function:



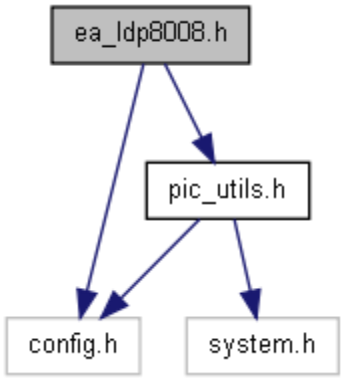

---

# ea_ldp8008.h File Reference

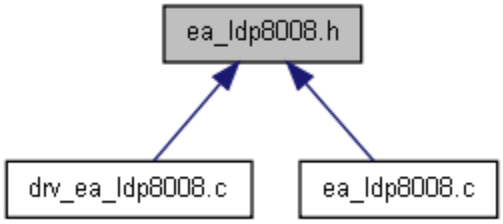
Support for LDP-8008 80x08 LED panel.

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

Include dependency graph for ea_ldp8008.h:



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




## Defines

- #define ea_ldp8008_setup() ea_ldp8008_setup_io()

## Functions

- void ea_ldp8008_init ()
- void ea_ldp8008_setup_io ()

---

## Detailed Description

Definition in file ea_ldp8008.h.

---

## Define Documentation

### #define ea_ldp8008_setup() ea_ldp8008_setup_io()

Definition at line 86 of file ea_ldp8008.h.

---

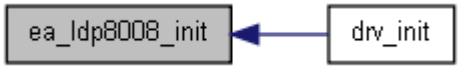
## Function Documentation

### void ea_ldp8008_init ()

Definition at line 60 of file ea_ldp8008.c.

Referenced by drv_init().

Here is the caller graph for this function:



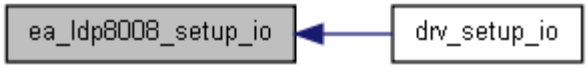

### void ea_ldp8008_setup_io ()

Definition at line 40 of file ea_ldp8008.c.

References clear_pin, and make_output.

Referenced by drv_setup_io().

Here is the caller graph for this function:

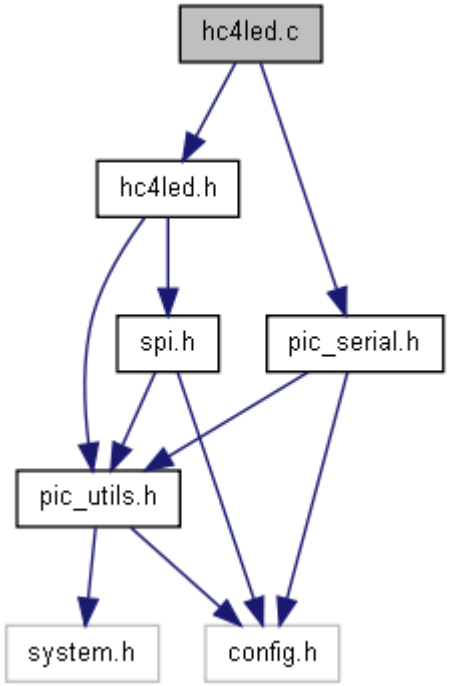



---

## hc4led.c File Reference

```
#include "hc4led.h"
#include "pic_serial.h"
```

Include dependency graph for hc4led.c:




### Functions

- uns8 hc4led_convert (uns8 digit)
- void hc4led_setup ()
- void hc4led_write_str (char *data)

---

### Function Documentation

#### uns8 hc4led_convert (uns8 *digit*)

Definition at line 43 of file hc4led.c.

Referenced by hc4led_write_str().

Here is the caller graph for this function:


### void hc4led_setup ()

Definition at line 39 of file hc4led.c.

References spi_setup().

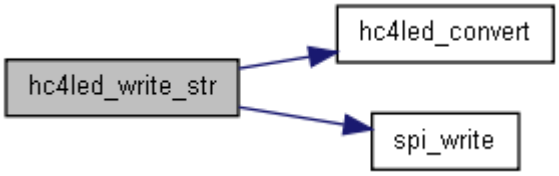
Here is the call graph for this function:


### void hc4led_write_str (char * *data*)

Definition at line 60 of file hc4led.c.

References hc4led_convert(), spi_write(), and uns8.

Here is the call graph for this function:



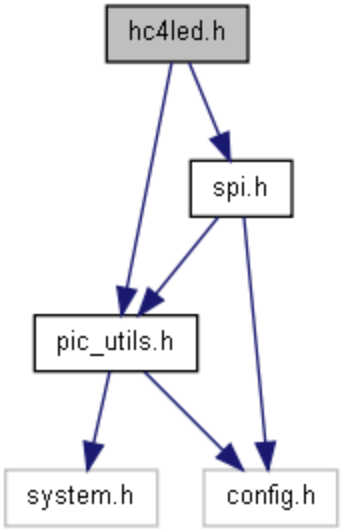

---

## hc4led.h File Reference

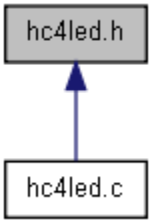
Routines to access four digit LED display.

```
#include "spi.h"
#include "pic_utils.h"
```

Include dependency graph for hc4led.h:




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




## Defines

- #define __hc4led_h include

## Functions

- void hc4led_setup ()
- void hc4led_write_str (char *data)

---

## Detailed Description

Definition in file hc4led.h.

---

## Define Documentation

### #define __hc4led_h include

Definition at line 41 of file hc4led.h.

---

## Function Documentation

### void hc4led_setup ()

Definition at line 39 of file hc4led.c.

References spi_setup().

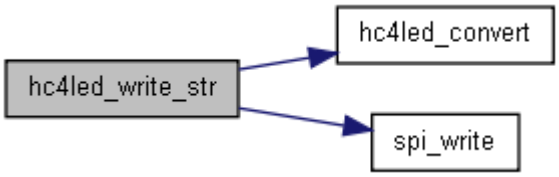
Here is the call graph for this function:


### void hc4led_write_str (char * *data*)

Definition at line 60 of file hc4led.c.

References hc4led_convert(), spi_write(), and uns8.

Here is the call graph for this function:

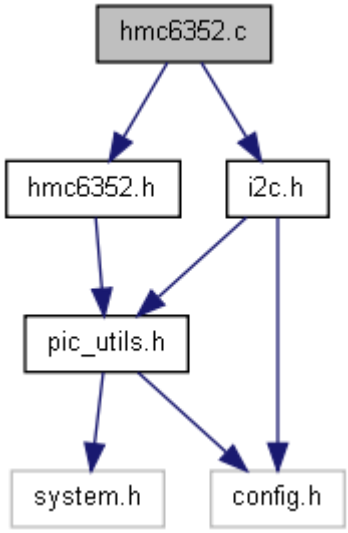



---

# hmc6352.c File Reference

```
#include "hmc6352.h"
#include "i2c.h"
```

Include dependency graph for hmc6352.c:



## Functions

- void hmc6352_enter_cal ()
- void hmc6352_exit_cal ()
- uns16 hmc6352_get_data ()
- uns8 hmc6352_read_eeprom (uns8 addr)
- uns8 hmc6352_read_ram (uns8 addr)
- void hmc6352_save_op_mode ()
- void hmc6352_set_mode (uns8 mode)
- void hmc6352_setup_io ()
- void hmc6352_sleep ()
- void hmc6352_update_bridge_offsets ()
- void hmc6352_wake ()
- void hmc6352_write_eeprom (uns8 addr, uns8 data)
- void hmc6352_write_ram (uns8 addr, uns8 data)

---

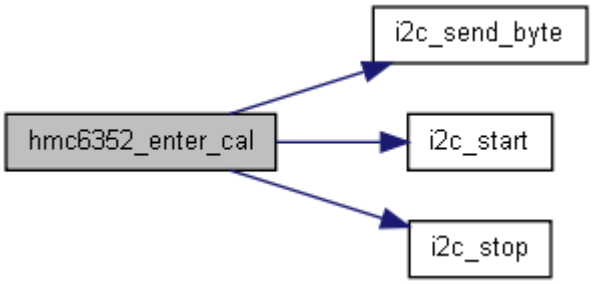
## Function Documentation

### void hmc6352_enter_cal ()

Definition at line 160 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_enter_cal_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

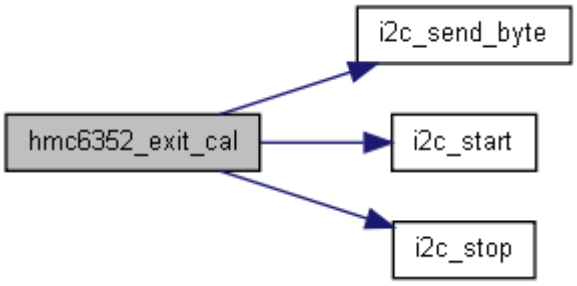



### void hmc6352_exit_cal ()

Definition at line 171 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_exit_cal_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

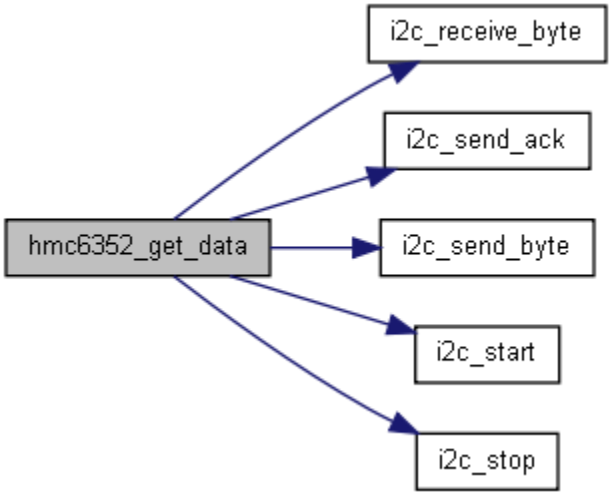



**uns16 hmc6352_get_data ()**

Definition at line 193 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_get_data_cmd, hmc6352_read, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns16.

Here is the call graph for this function:

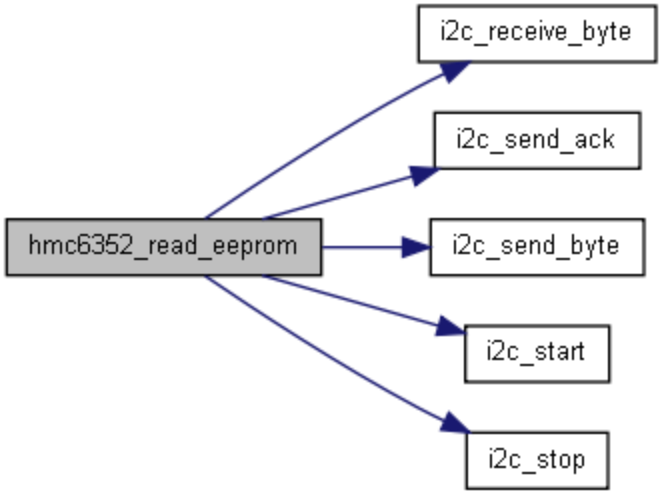



**uns8 hmc6352_read_eeprom (uns8 *addr*)**

Definition at line 55 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_read, hmc6352_read_from_eeprom, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns8.

Here is the call graph for this function:



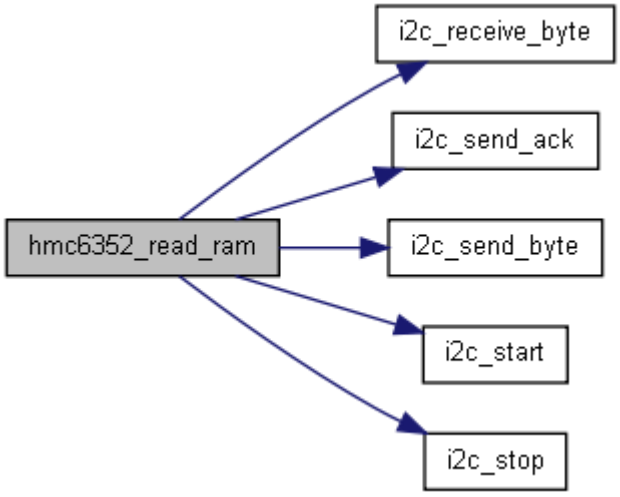

**uns8 hmc6352_read_ram (uns8 *addr*)**

Definition at line 99 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_read, hmc6352_read_from_ram, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns8.

Referenced by hmc6352_set_mode().

Here is the call graph for this function:



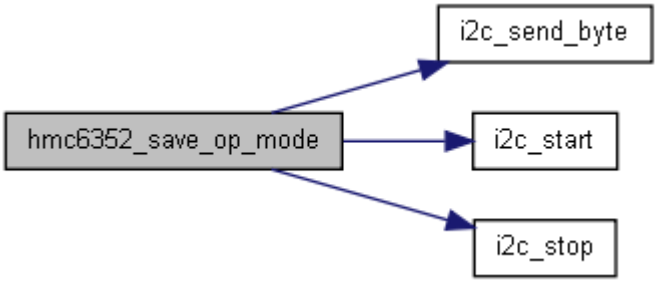

Here is the caller graph for this function:


**void hmc6352_save_op_mode ()**

Definition at line 182 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_save_op_mode_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

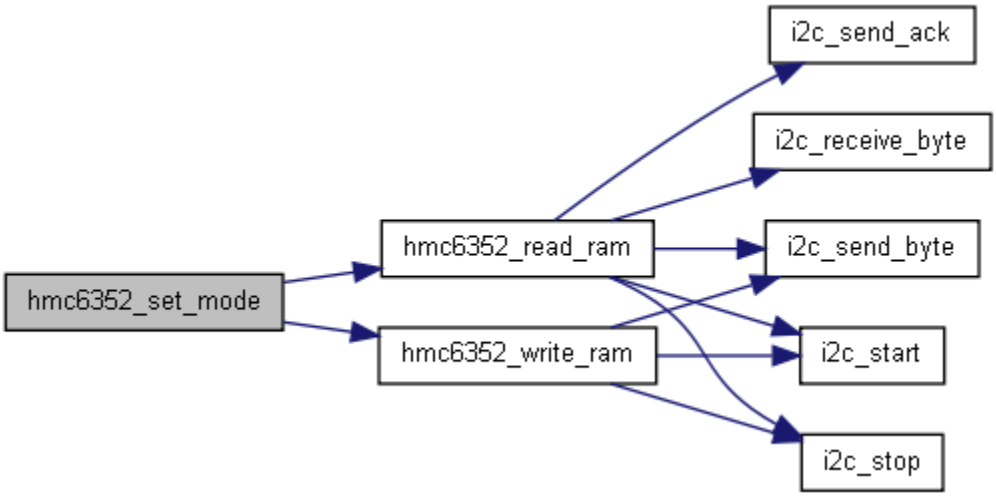



### void hmc6352_set_mode (uns8 *mode*)

Definition at line 225 of file hmc6352.c.

References hmc6352_mode_continuous, hmc6352_mode_query, hmc6352_mode_standby, hmc6352_read_ram(), hmc6352_write_ram(), and uns8.

Here is the call graph for this function:




### void hmc6352_setup_io ()

Definition at line 247 of file hmc6352.c.

References i2c_setup_io().

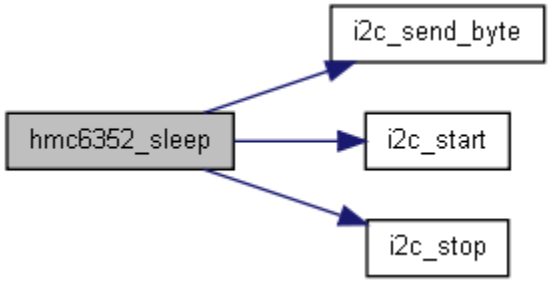
Here is the call graph for this function:


### void hmc6352_sleep ()

Definition at line 125 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_sleep_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

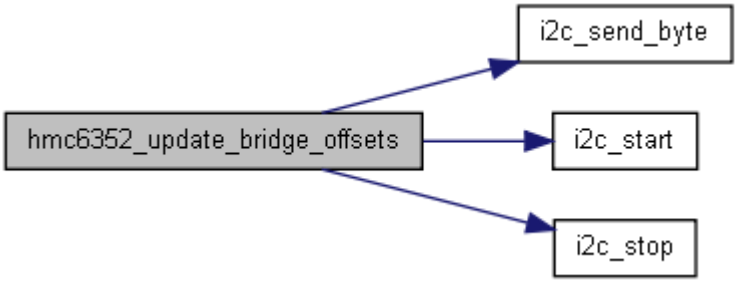



### void hmc6352_update_bridge_offsets ()

Definition at line 149 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_update_bridge_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

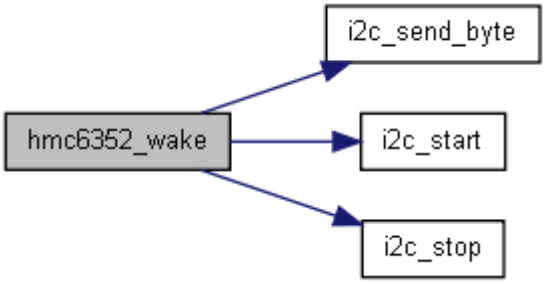



### void hmc6352_wake ()

Definition at line 137 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_wake_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

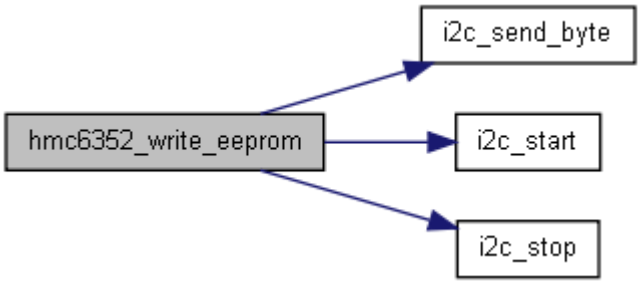


### void hmc6352_write_eeprom (uns8 *addr*, uns8 *data*)

Definition at line 40 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_write, hmc6352_write_to_eeprom, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:



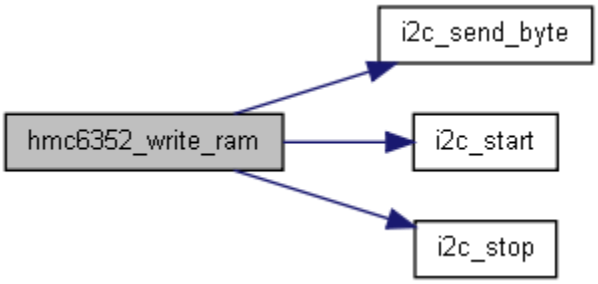

### void hmc6352_write_ram (uns8 *addr*, uns8 *data*)

Definition at line 82 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_write, hmc6352_write_to_ram, i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by hmc6352_set_mode().

Here is the call graph for this function:




Here is the caller graph for this function:

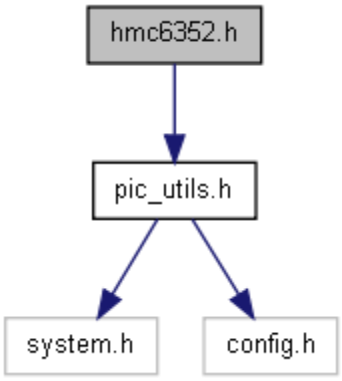

---

## hmc6352.h File Reference

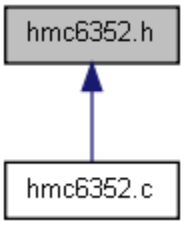
Routines for communicating with the hmc6352 digital compass.

```
#include "pic_utils.h"
```

Include dependency graph for hmc6352.h:




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




## Defines

- #define hmc6352_device_addr 0x42
- #define hmc6352_ee_slave_addr 0x00
- #define hmc6352_ee_time_delay 0x05
- #define hmc6352_ee_x_offset_lsb 0x02
- #define hmc6352_ee_x_offset_msb 0x01
- #define hmc6352_ee_y_offset_lsb 0x04
- #define hmc6352_ee_y_offset_msb 0x03
- #define hmc6352_enter_cal_cmd 0x43
- #define hmc6352_exit_cal_cmd 0x45
- #define hmc6352_get_data_cmd 0x41
- #define hmc6352_mode_continuous 0x02
- #define hmc6352_mode_query 0x01
- #define hmc6352_mode_standby 0x00
- #define hmc6352_num_summed 0x06
- #define hmc6352_op_mode 0x08
- #define hmc6352_ram_op_mode_control 0x74
- #define hmc6352_ram_output_data_control 0x4e
- #define hmc6352_read 0x01
- #define hmc6352_read_from_eeprom 0x72
- #define hmc6352_read_from_ram 0x67
- #define hmc6352_save_op_mode_cmd 0x4c
- #define hmc6352_sleep_cmd 0x53
- #define hmc6352_software_ver 0x07
- #define hmc6352_update_bridge_cmd 0x4F
- #define hmc6352_wake_cmd 0x57
- #define hmc6352_write 0x00
- #define hmc6352_write_to_eeprom 0x77

- #define hmc6352_write_to_ram 0x47

## Functions

- void hmc6352_enter_cal ()
- void hmc6352_exit_cal ()
- uns16 hmc6352_get_data ()
- uns8 hmc6352_read_eeprom (uns8 addr)
- uns8 hmc6352_read_ram (uns8 addr)
- void hmc6352_save_op_mode ()
- void hmc6352_set_mode (uns8 mode)
- void hmc6352_setup_io ()
- void hmc6352_sleep ()
- void hmc6352_update_bridge_offsets ()
- void hmc6352_wake ()
- void hmc6352_write_eeprom (uns8 addr, uns8 data)
- void hmc6352_write_ram (uns8 addr, uns8 data)

---

## Detailed Description

Definition in file hmc6352.h.

---

## Define Documentation

### #define hmc6352_device_addr 0x42

Definition at line 47 of file hmc6352.h.

Referenced by hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), and hmc6352_write_ram().

### #define hmc6352_ee_slave_addr 0x00

Definition at line 63 of file hmc6352.h.

### #define hmc6352_ee_time_delay 0x05

Definition at line 68 of file hmc6352.h.

### #define hmc6352_ee_x_offset_lsb 0x02

Definition at line 65 of file hmc6352.h.

### #define hmc6352_ee_x_offset_msb 0x01

Definition at line 64 of file hmc6352.h.

**#define hmc6352_ee_y_offset_lsb 0x04**

Definition at line 67 of file hmc6352.h.

**#define hmc6352_ee_y_offset_msb 0x03**

Definition at line 66 of file hmc6352.h.

**#define hmc6352_enter_cal_cmd 0x43**

Definition at line 58 of file hmc6352.h.
Referenced by hmc6352_enter_cal().

**#define hmc6352_exit_cal_cmd 0x45**

Definition at line 59 of file hmc6352.h.
Referenced by hmc6352_exit_cal().

**#define hmc6352_get_data_cmd 0x41**

Definition at line 61 of file hmc6352.h.
Referenced by hmc6352_get_data().

**#define hmc6352_mode_continuous 0x02**

Definition at line 75 of file hmc6352.h.
Referenced by hmc6352_set_mode().

**#define hmc6352_mode_query 0x01**

Definition at line 74 of file hmc6352.h.
Referenced by hmc6352_set_mode().

**#define hmc6352_mode_standby 0x00**

Definition at line 73 of file hmc6352.h.
Referenced by hmc6352_set_mode().

**#define hmc6352_num_summed 0x06**

Definition at line 69 of file hmc6352.h.

**#define hmc6352_op_mode 0x08**

Definition at line 71 of file hmc6352.h.

**#define hmc6352_ram_op_mode_control 0x74**

Definition at line 77 of file hmc6352.h.

**#define hmc6352_ram_output_data_control 0x4e**

Definition at line 78 of file hmc6352.h.

**#define hmc6352_read 0x01**

Definition at line 48 of file hmc6352.h.
Referenced by hmc6352_get_data(), hmc6352_read_eeprom(), and hmc6352_read_ram().

**#define hmc6352_read_from_eeprom 0x72**

Definition at line 52 of file hmc6352.h.
Referenced by hmc6352_read_eeprom().

**#define hmc6352_read_from_ram 0x67**

Definition at line 54 of file hmc6352.h.
Referenced by hmc6352_read_ram().

**#define hmc6352_save_op_mode_cmd 0x4c**

Definition at line 60 of file hmc6352.h.
Referenced by hmc6352_save_op_mode().

**#define hmc6352_sleep_cmd 0x53**

Definition at line 55 of file hmc6352.h.
Referenced by hmc6352_sleep().

**#define hmc6352_software_ver 0x07**

Definition at line 70 of file hmc6352.h.

**#define hmc6352_update_bridge_cmd 0x4F**

Definition at line 57 of file hmc6352.h.

Referenced by hmc6352_update_bridge_offsets().

#### #define hmc6352_wake_cmd 0x57

Definition at line 56 of file hmc6352.h.

Referenced by hmc6352_wake().

#### #define hmc6352_write 0x00

Definition at line 49 of file hmc6352.h.

Referenced by hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), and hmc6352_write_ram().

#### #define hmc6352_write_to_eeprom 0x77

Definition at line 51 of file hmc6352.h.

Referenced by hmc6352_write_eeprom().

#### #define hmc6352_write_to_ram 0x47

Definition at line 53 of file hmc6352.h.

Referenced by hmc6352_write_ram().

---

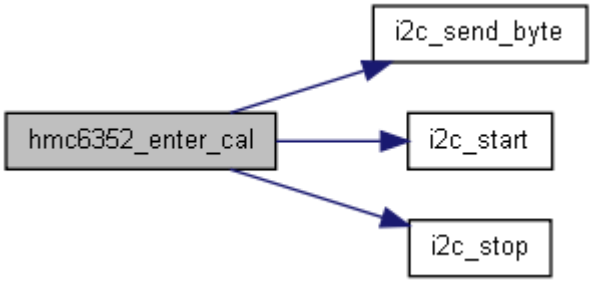
## Function Documentation

#### void hmc6352_enter_cal ()

Definition at line 160 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_enter_cal_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

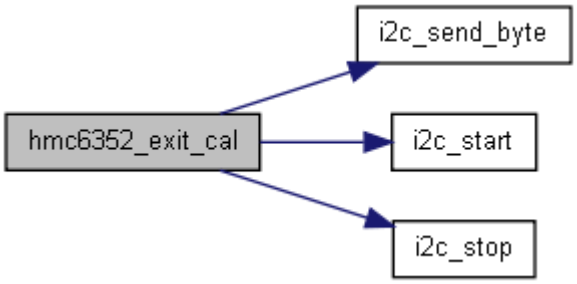


### void hmc6352_exit_cal ()

Definition at line 171 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_exit_cal_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

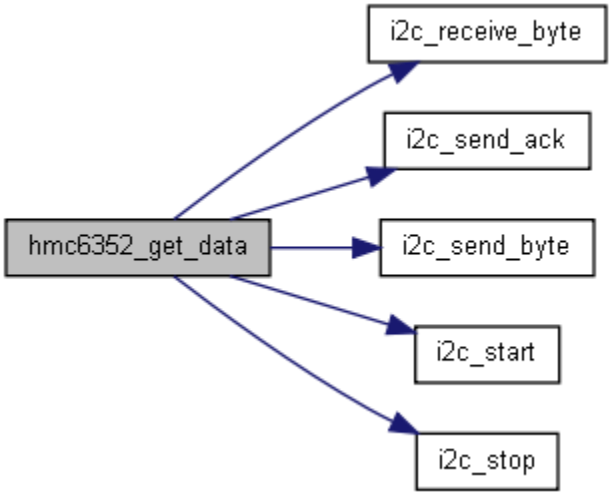



### uns16 hmc6352_get_data ()

Definition at line 193 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_get_data_cmd, hmc6352_read, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns16.

Here is the call graph for this function:

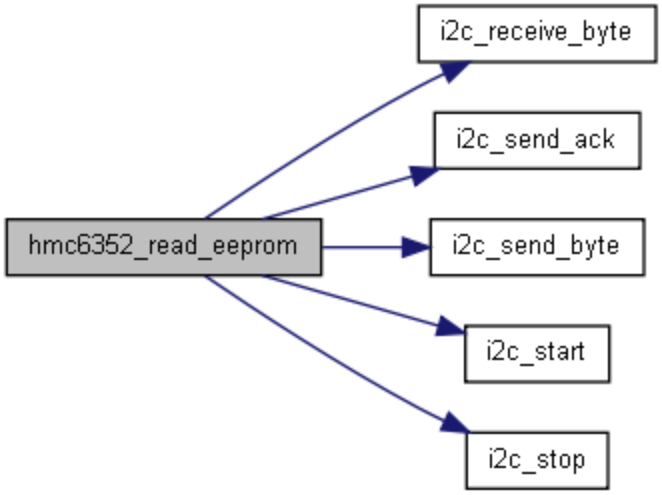



### uns8 hmc6352_read_eeprom (uns8 *addr*)

Definition at line 55 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_read, hmc6352_read_from_eeprom, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns8.

Here is the call graph for this function:



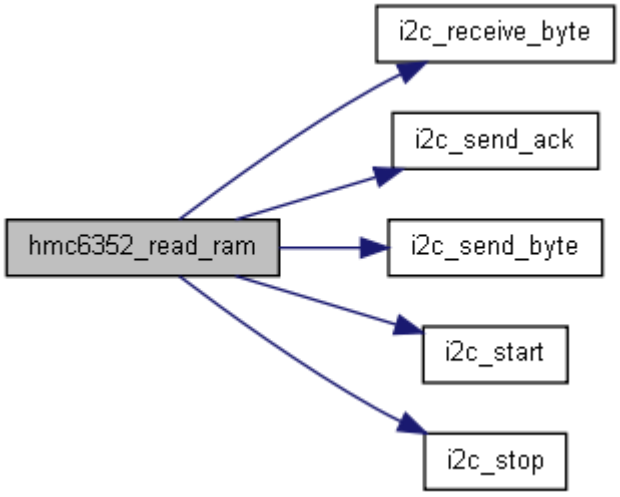

**uns8 hmc6352_read_ram (uns8 *addr*)**

Definition at line 99 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_read, hmc6352_read_from_ram, hmc6352_write, i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns8.

Referenced by hmc6352_set_mode().

Here is the call graph for this function:



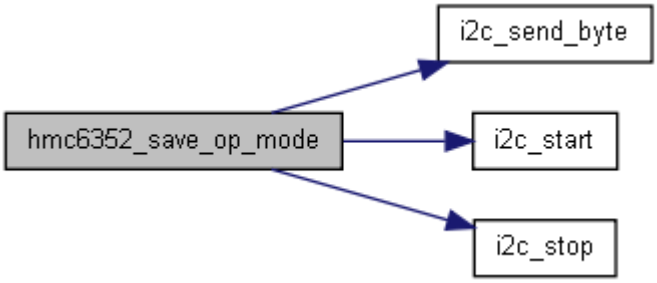

Here is the caller graph for this function:


**void hmc6352_save_op_mode ()**

Definition at line 182 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_save_op_mode_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

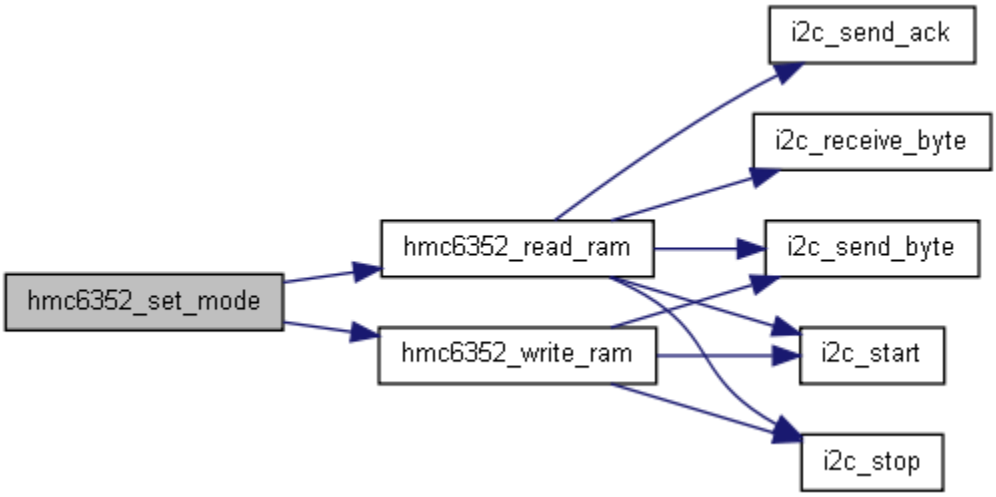



### void hmc6352_set_mode (uns8 *mode*)

Definition at line 225 of file hmc6352.c.

References hmc6352_mode_continuous, hmc6352_mode_query, hmc6352_mode_standby, hmc6352_read_ram(), hmc6352_write_ram(), and uns8.

Here is the call graph for this function:




### void hmc6352_setup_io ()

Definition at line 247 of file hmc6352.c.

References i2c_setup_io().

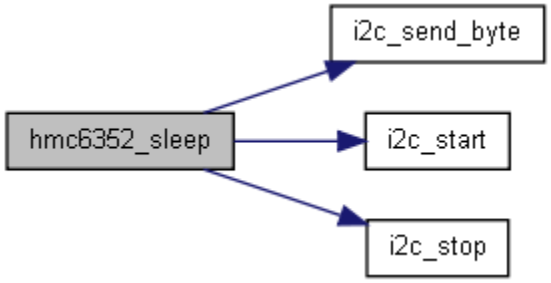
Here is the call graph for this function:


### void hmc6352_sleep ()

Definition at line 125 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_sleep_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

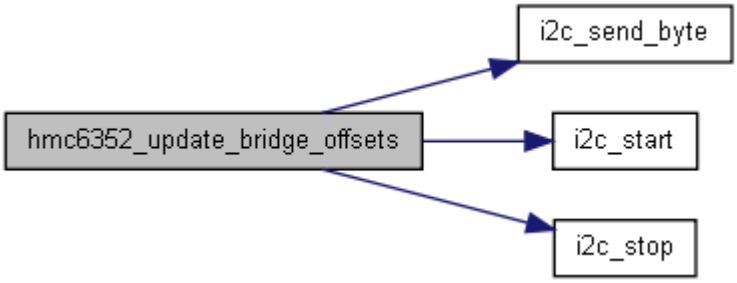



### void hmc6352_update_bridge_offsets ()

Definition at line 149 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_update_bridge_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

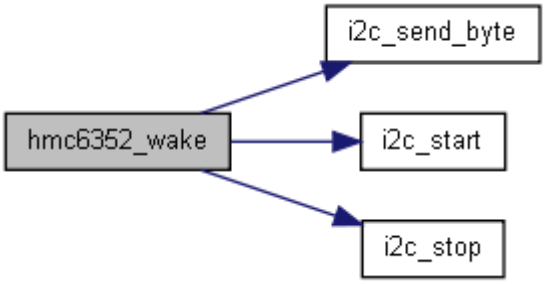



### void hmc6352_wake ()

Definition at line 137 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_wake_cmd, hmc6352_write, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:

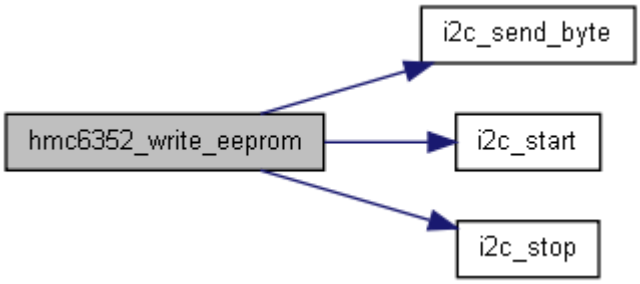


**void hmc6352_write_eeprom (uns8 *addr*, uns8 *data*)**

Definition at line 40 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_write, hmc6352_write_to_eeprom, i2c_send_byte(), i2c_start(), and i2c_stop().

Here is the call graph for this function:



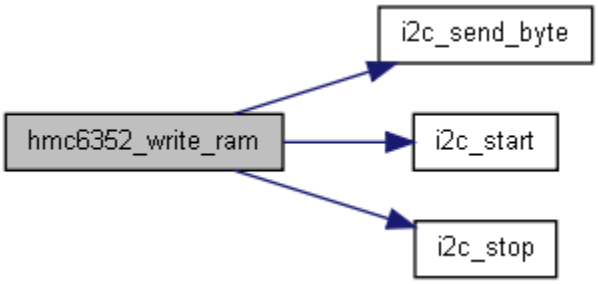

**void hmc6352_write_ram (uns8 *addr*, uns8 *data*)**

Definition at line 82 of file hmc6352.c.

References hmc6352_device_addr, hmc6352_write, hmc6352_write_to_ram, i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by hmc6352_set_mode().

Here is the call graph for this function:



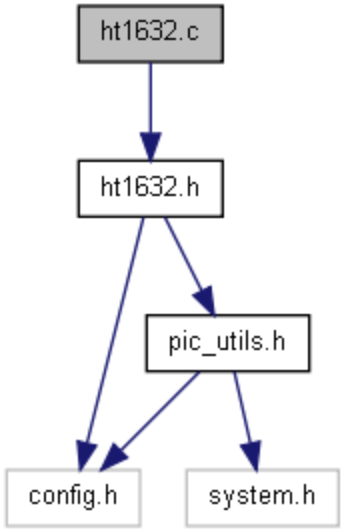

Here is the caller graph for this function:


---

## ht1632.c File Reference

```
#include "ht1632.h"
```

Include dependency graph for ht1632.c:




## Functions

- void ht1632_fill (uns8 colour)
- void ht1632_fill2 (uns8 colour)
- void ht1632_init (uns8 hw_config)
- void ht1632_send_command (uns8 command)
- void ht1632_set_brightness (uns8 brightness)
- void ht1632_set_pixel (uns8 x, uns8 y, uns8 colour)
- void ht1632_setup_io ()
- void ht1632_write (uns8 mem_addr, uns8 data)

---

## Function Documentation

### void ht1632_fill (uns8 *colour*)

Definition at line 326 of file ht1632.c.

References ht1632_write(), and uns8.

Here is the call graph for this function:


### void ht1632_fill2 (uns8 *colour*)

Definition at line 341 of file ht1632.c.

References clear_pin, HT1632_CMD_LEDS_OFF, HT1632_CMD_LEDS_ON, ht1632_send_command(), set_pin, and uns16.

Here is the call graph for this function:


### void ht1632_init (uns8 *hw_config*)

Definition at line 75 of file ht1632.c.

References HT1632_CMD_CLK_MASTER_MODE, HT1632_CMD_LEDS_ON, HT1632_CMD_SYS_DISABLE, HT1632_CMD_SYS_ENABLE, and ht1632_send_command().

Referenced by drv_init().

Here is the call graph for this function:


Here is the caller graph for this function:

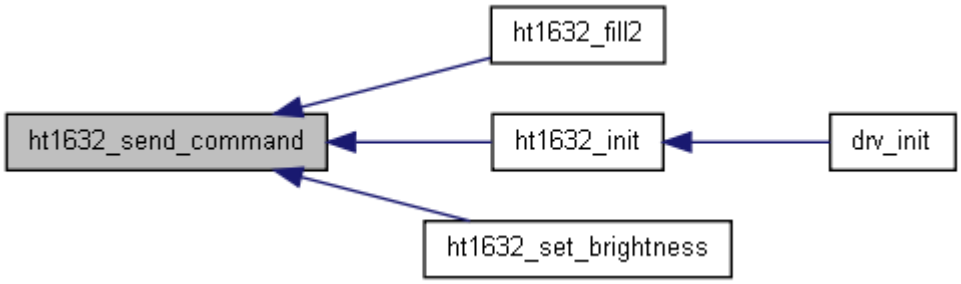

### void ht1632_send_command (uns8 *command*)

Definition at line 86 of file ht1632.c.

References clear_pin, set_pin, and uns8.

Referenced by ht1632_fill2(), ht1632_init(), and ht1632_set_brightness().

Here is the caller graph for this function:




### void ht1632_set_brightness (uns8 *brightness*)

Definition at line 206 of file ht1632.c.

References ht1632_send_command().

Here is the call graph for this function:

**void ht1632_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 211 of file ht1632.c.

References clear_pin, make_input, make_output, set_pin, test_pin, and uns8.

**void ht1632_setup_io ()**

Definition at line 41 of file ht1632.c.

References make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:


**void ht1632_write (uns8 *mem_addr*, uns8 *data*)**

Definition at line 142 of file ht1632.c.

References change_pin_var, clear_pin, set_pin, and uns8.

Referenced by ht1632_fill().

Here is the caller graph for this function:

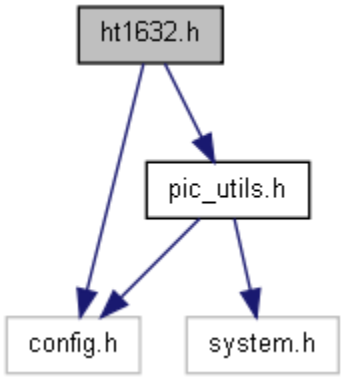

---

## ht1632.h File Reference

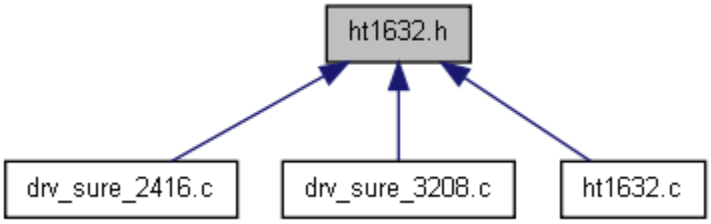
Holtek LED matrix display routines, used in Sure 2416 display and others.

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

Include dependency graph for ht1632.h:



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




## Defines

- #define HT1632_CMD_BLINK_OFF 0b00001000
- #define HT1632_CMD_BLINK_ON 0b00001001
- #define HT1632_CMD_CLK_MASTER_MODE 0b00010100
- #define HT1632_CMD_CLK_SLAVE_MODE 0b00010000
- #define HT1632_CMD_CLK_SOURCE_EXT 0b00011100
- #define HT1632_CMD_CLK_SOURCE_INT_RC 0b00011000
- #define HT1632_CMD_LEDS_OFF 0b00000010
- #define HT1632_CMD_LEDS_ON 0b00000011
- #define HT1632_CMD_NMOS_16_COMMON 0b00100100
- #define HT1632_CMD_NMOS_8_COMMON 0b00100000
- #define HT1632_CMD_PMOS_16_COMMON 0b00101100
- #define HT1632_CMD_PMOS_8_COMMON 0b00101000
- #define HT1632_CMD_SYS_DISABLE 0b00000000
- #define HT1632_CMD_SYS_ENABLE 0b00000001

## Functions

- void ht1632_clear ()
- void ht1632_fill (uns8 colour)
- void ht1632_fill2 (uns8 colour)
- uns8 ht1632_get_pixel (uns8 x, uns8 y)
- void ht1632_horizontal_line (uns8 x, uns8 y, uns8 length, uns8 colour)
- void ht1632_init (uns8 hw_config)
- void ht1632_send_command (uns8 command)
- void ht1632_set_brightness (uns8 brightness)
- void ht1632_set_pixel (uns8 x, uns8 y, uns8 colour)
- void ht1632_setup_io ()
- void ht1632_vertical_line (uns8 x, uns8 y, uns8 length, uns8 colour)
- void ht1632_write (uns8 mem_addr, uns8 data)

---

## Detailed Description

Routines to communicate with Holtek HT1632 led matrix display chip

Definition in file ht1632.h.

---

## Define Documentation

### #define HT1632_CMD_BLINK_OFF 0b00001000

Definition at line 104 of file ht1632.h.

### #define HT1632_CMD_BLINK_ON 0b00001001

Definition at line 105 of file ht1632.h.

### #define HT1632_CMD_CLK_MASTER_MODE 0b00010100

Definition at line 95 of file ht1632.h.
Referenced by ht1632_init().

### #define HT1632_CMD_CLK_SLAVE_MODE 0b00010000

Definition at line 96 of file ht1632.h.

### #define HT1632_CMD_CLK_SOURCE_EXT 0b00011100

Definition at line 99 of file ht1632.h.

### #define HT1632_CMD_CLK_SOURCE_INT_RC 0b00011000

Definition at line 98 of file ht1632.h.

### #define HT1632_CMD_LEDS_OFF 0b00000010

Definition at line 101 of file ht1632.h.
Referenced by ht1632_fill2().

### #define HT1632_CMD_LEDS_ON 0b00000011

Definition at line 102 of file ht1632.h.
Referenced by ht1632_fill2(), and ht1632_init().

### #define HT1632_CMD_NMOS_16_COMMON 0b00100100

Definition at line 91 of file ht1632.h.

### #define HT1632_CMD_NMOS_8_COMMON 0b00100000

Definition at line 90 of file ht1632.h.

**#define HT1632_CMD_PMOS_16_COMMON 0b00101100**

Definition at line 93 of file ht1632.h.

Referenced by drv_init().

**#define HT1632_CMD_PMOS_8_COMMON 0b00101000**

Definition at line 92 of file ht1632.h.

**#define HT1632_CMD_SYS_DISABLE 0b00000000**

Definition at line 87 of file ht1632.h.

Referenced by ht1632_init().

**#define HT1632_CMD_SYS_ENABLE 0b00000001**

Definition at line 88 of file ht1632.h.

Referenced by ht1632_init().

---

## Function Documentation

**void ht1632_clear ()**

**void ht1632_fill (uns8 *colour*)**

Definition at line 326 of file ht1632.c.

References ht1632_write(), and uns8.

Here is the call graph for this function:


**void ht1632_fill2 (uns8 *colour*)**

Definition at line 341 of file ht1632.c.

References clear_pin, HT1632_CMD_LEDS_OFF, HT1632_CMD_LEDS_ON, ht1632_send_command(), set_pin, and uns16.

Here is the call graph for this function:

**uns8 ht1632_get_pixel (uns8 *x*, uns8 *y*)**

**void ht1632_horizontal_line (uns8 *x*, uns8 *y*, uns8 *length*, uns8 *colour*)**

**void ht1632_init (uns8 *hw_config*)**

Definition at line 75 of file ht1632.c.

References HT1632_CMD_CLK_MASTER_MODE, HT1632_CMD_LEDS_ON, HT1632_CMD_SYS_DISABLE, HT1632_CMD_SYS_ENABLE, and ht1632_send_command().

Referenced by drv_init().

Here is the call graph for this function:


Here is the caller graph for this function:

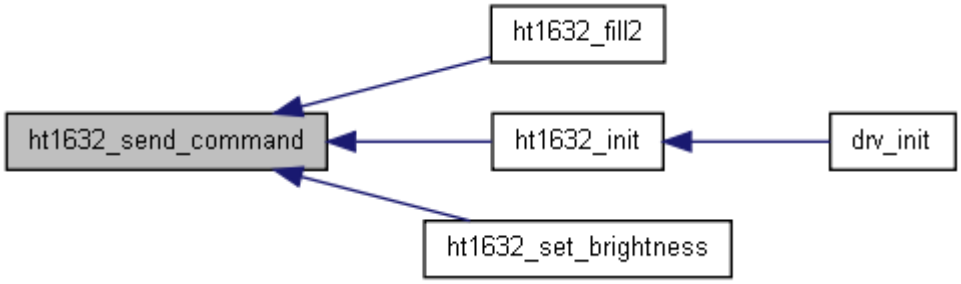

**void ht1632_send_command (uns8 *command*)**

Definition at line 86 of file ht1632.c.

References clear_pin, set_pin, and uns8.

Referenced by ht1632_fill2(), ht1632_init(), and ht1632_set_brightness().

Here is the caller graph for this function:




**void ht1632_set_brightness (uns8 *brightness*)**

Definition at line 206 of file ht1632.c.

References ht1632_send_command().

Here is the call graph for this function:

**void ht1632_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 211 of file ht1632.c.

References clear_pin, make_input, make_output, set_pin, test_pin, and uns8.

**void ht1632_setup_io ()**

Definition at line 41 of file ht1632.c.

References make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:


**void ht1632_vertical_line (uns8 *x*, uns8 *y*, uns8 *length*, uns8 *colour*)**

**void ht1632_write (uns8 *mem_addr*, uns8 *data*)**

Definition at line 142 of file ht1632.c.

References change_pin_var, clear_pin, set_pin, and uns8.

Referenced by ht1632_fill().

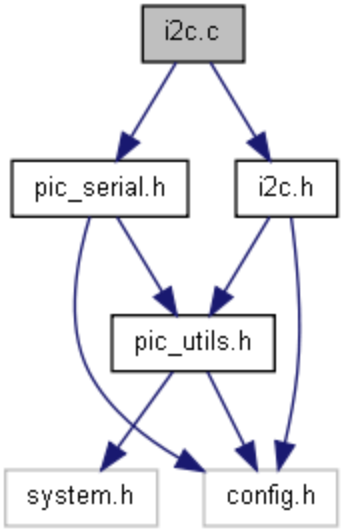
Here is the caller graph for this function:


---

## i2c.c File Reference

```
#include "i2c.h"
#include "pic_serial.h"
```

Include dependency graph for i2c.c:




## Defines

- #define DELAY_AMOUNT 10

## Functions

- void i2c_ack_polling (uns8 device_address)
- uns8 i2c_read_eeprom (uns8 device_address, uns8 mem_address)
- *Read an 8 bit byte over I2C buss.* uns16 i2c_read_eeprom_16bit (uns8 device_address, uns8 mem_address)
- *Read 16 bits of data over I2C buss.* uns8 i2c_receive_byte (void)
- *Receive byte from I2C buss.* void i2c_send_ack (void)
- *Send an ACK.* void i2c_send_byte (uns8 data)
- *Send a byte to I2C buss.* void i2c_setup_io ()
- *Setup ports and pins for I2C communication.* void i2c_start (void)
- *Send start signal to I2C buss.* void i2c_stop (void)
- *Send stop signal to I2C buss.* void i2c_write_eeprom (uns8 device_address, uns8 mem_address, uns8 data)
- *Write an 8 bit byte ove I2C buss.* void i2c_write_eeprom_16bit (uns8 device_address, uns8 mem_address, uns16 data)

*Write a 16 bit value over I2C buss.*

---

## Define Documentation

### #define DELAY_AMOUNT 10

Definition at line 39 of file i2c.c.

Referenced by i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), and i2c_stop().

---

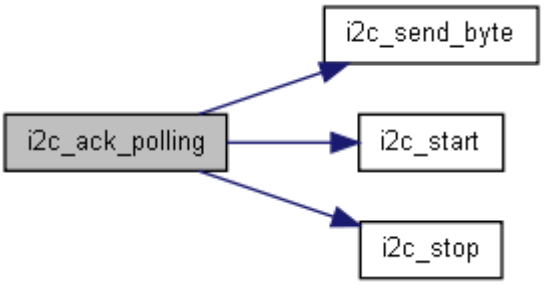
## Function Documentation

### void i2c_ack_polling (uns8 *device_address*)

Definition at line 41 of file i2c.c.

References i2c_send_byte(), i2c_start(), i2c_stop(), and test_pin.

Here is the call graph for this function:




### uns8 i2c_read_eeprom (uns8 *device_address*, uns8 *mem_address*)

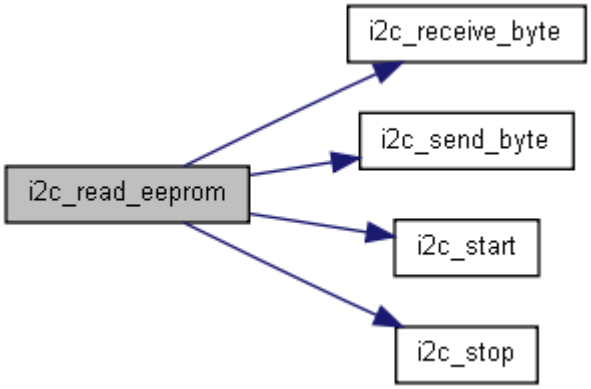
Read an 8 bit byte from the specified device at the memory address

Definition at line 76 of file i2c.c.

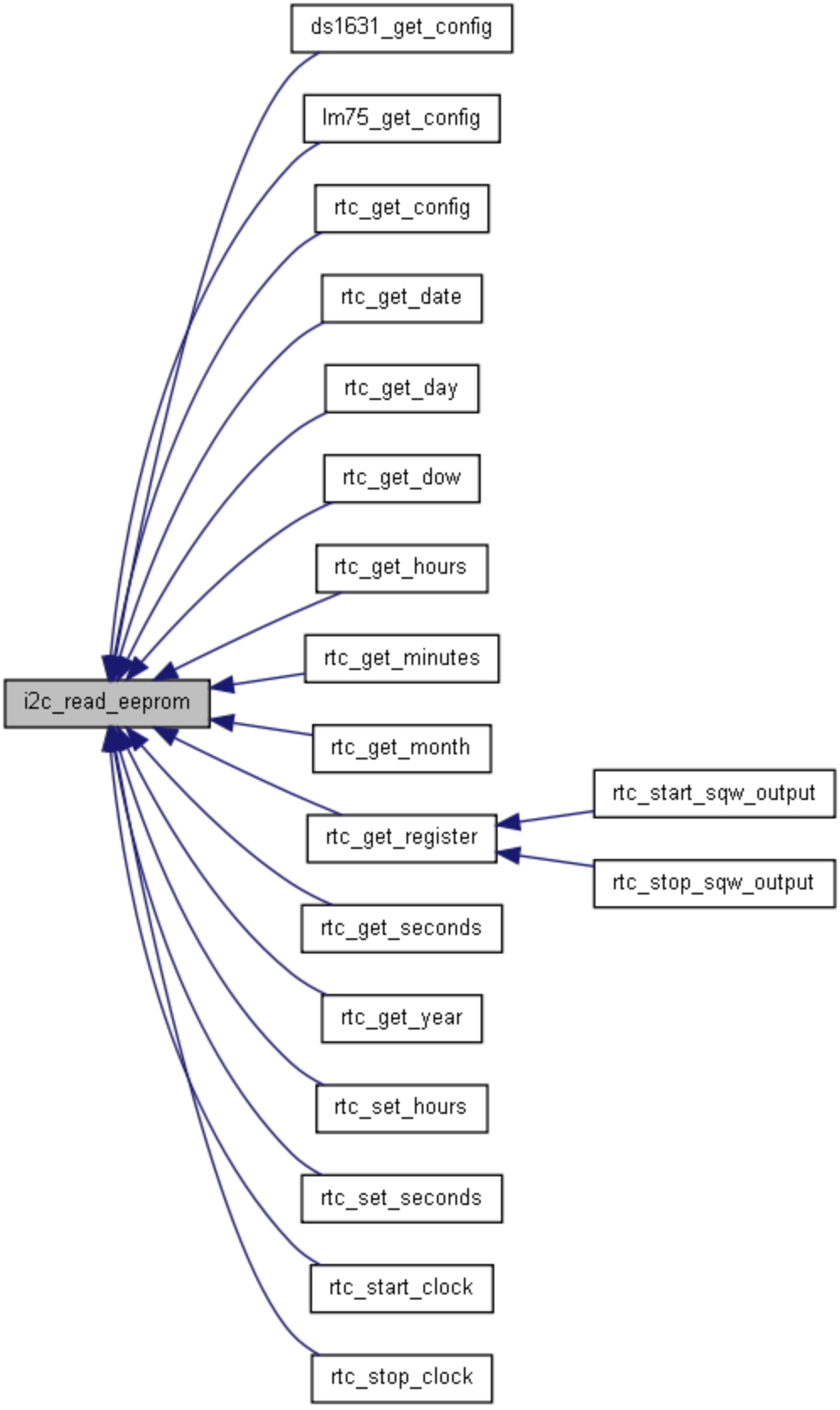
References clear_pin, DELAY_AMOUNT, i2c_receive_byte(), i2c_send_byte(), i2c_start(), i2c_stop(), i2c_write_sda, set_pin, and uns8.

Referenced by ds1631_get_config(), lm75_get_config(), rtc_get_config(), rtc_get_date(), rtc_get_day(), rtc_get_dow(), rtc_get_hours(), rtc_get_minutes(), rtc_get_month(), rtc_get_register(), rtc_get_seconds(), rtc_get_year(), rtc_set_hours(), rtc_set_seconds(), rtc_start_clock(), and rtc_stop_clock().

Here is the call graph for this function:




Here is the caller graph for this function:


ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
i2c_read_eeprom
rtc_get_month
rtc_start_sqw_output
rtc_get_register
rtc_stop_sqw_output
rtc_get_seconds
rtc_get_year
rtc_set_hours
rtc_set_seconds
rtc_start_clock
rtc_stop_clock

**uns16 i2c_read_eeprom_16bit (uns8 *device_address*, uns8 *mem_address*)**

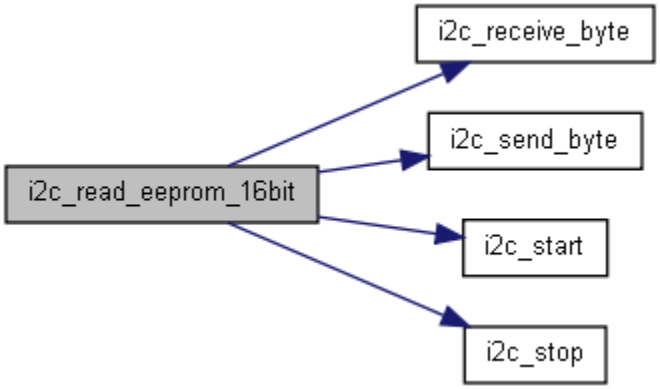
Read a 16 bit chunk of data from the given device and memory address

Definition at line 103 of file i2c.c.

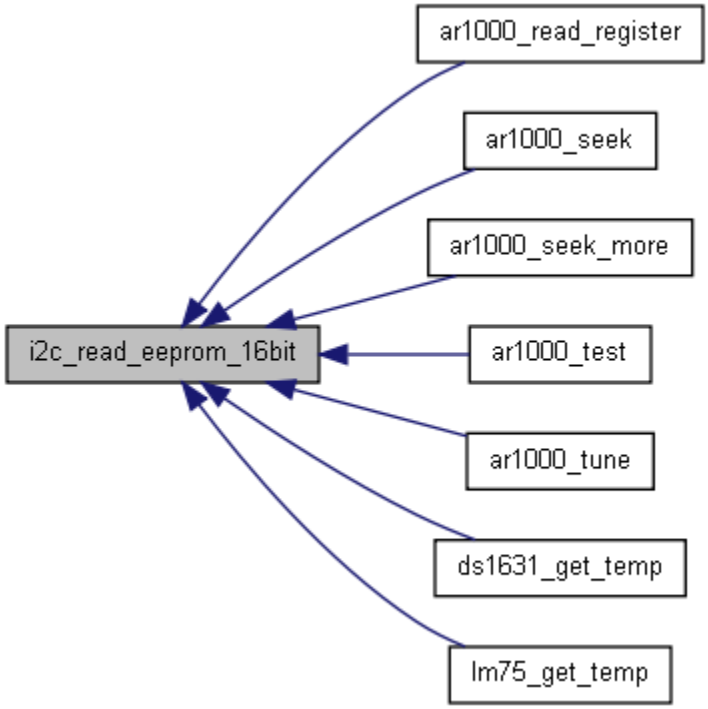
References clear_pin, DELAY_AMOUNT, i2c_receive_byte(), i2c_send_byte(), i2c_start(), i2c_stop(), i2c_write_sda, set_pin, and uns16.

Referenced by ar1000_read_register(), ar1000_seek(), ar1000_seek_more(), ar1000_test(), ar1000_tune(), ds1631_get_temp(), and lm75_get_temp().

Here is the call graph for this function:




Here is the caller graph for this function:




**uns8 i2c_receive_byte ()**

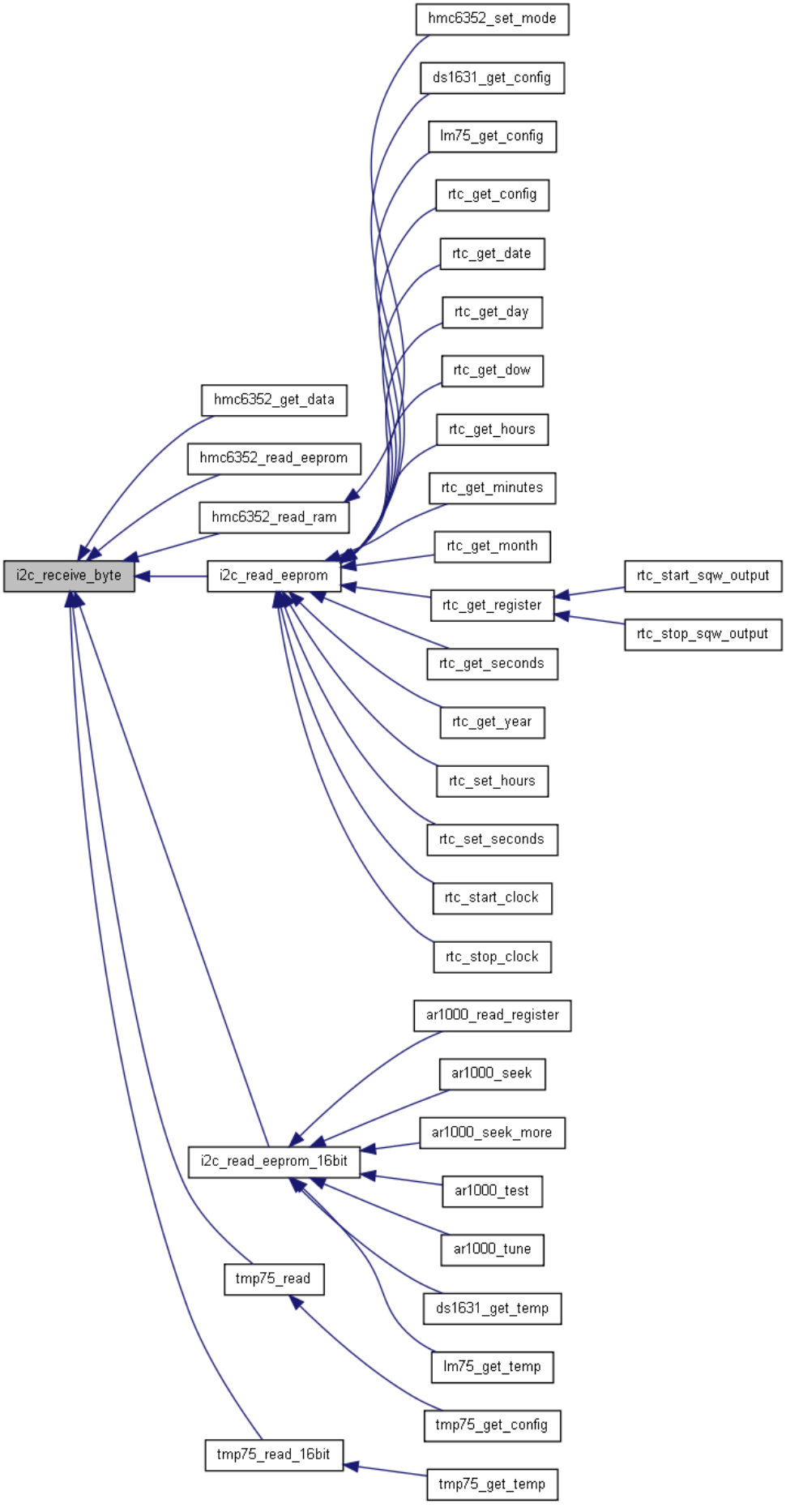
Clock in a byte over I2C lines. Note that an acknowledge is not sent/received

Definition at line 186 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_read_sda, set_pin, test_pin, and uns8.

Referenced by hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), tmp75_read(), and tmp75_read_16bit().

Here is the caller graph for this function:


hmc6352_set_mode
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_start_sqw_output
rtc_stop_sqw_output
rtc_get_seconds
rtc_get_year
rtc_set_hours
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
i2c_receive_byte
i2c_read_eeprom
ar1000_read_register
ar1000_seek
ar1000_seek_more
i2c_read_eeprom_16bit
ar1000_test
ar1000_tune
tmp75_read
ds1631_get_temp
lm75_get_temp
tmp75_get_config
tmp75_read_16bit
tmp75_get_temp

**void i2c_send_ack (void)**

Sends an I2C acknowledge (bit)

Definition at line 140 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), and tmp75_read_16bit().

Here is the caller graph for this function:


**void i2c_send_byte (uns8 *data*)**

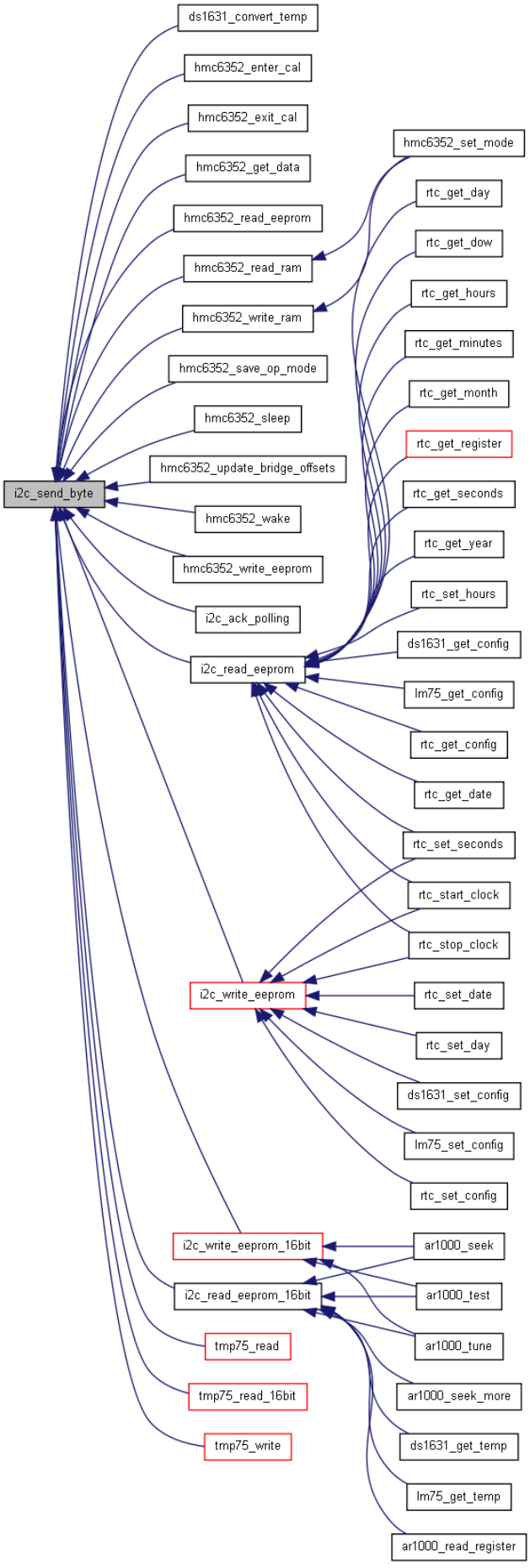
Clock out a byte from the I2C buss. An acknowledge is "received" (although ignored).

Definition at line 206 of file i2c.c.

References change_pin, clear_pin, DELAY_AMOUNT, i2c_read_sda, i2c_write_sda, set_pin, and uns8.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_send_byte
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
i2c_write_eeprom
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

### void i2c_setup_io ()

Setup ports and pins for i2c output.

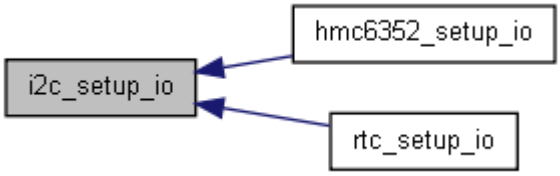
Set port and pins correctly for I2C communication

Definition at line 231 of file i2c.c.

References make_input, and make_output.

Referenced by hmc6352_setup_io(), and rtc_setup_io().

Here is the caller graph for this function:




### void i2c_start (void)

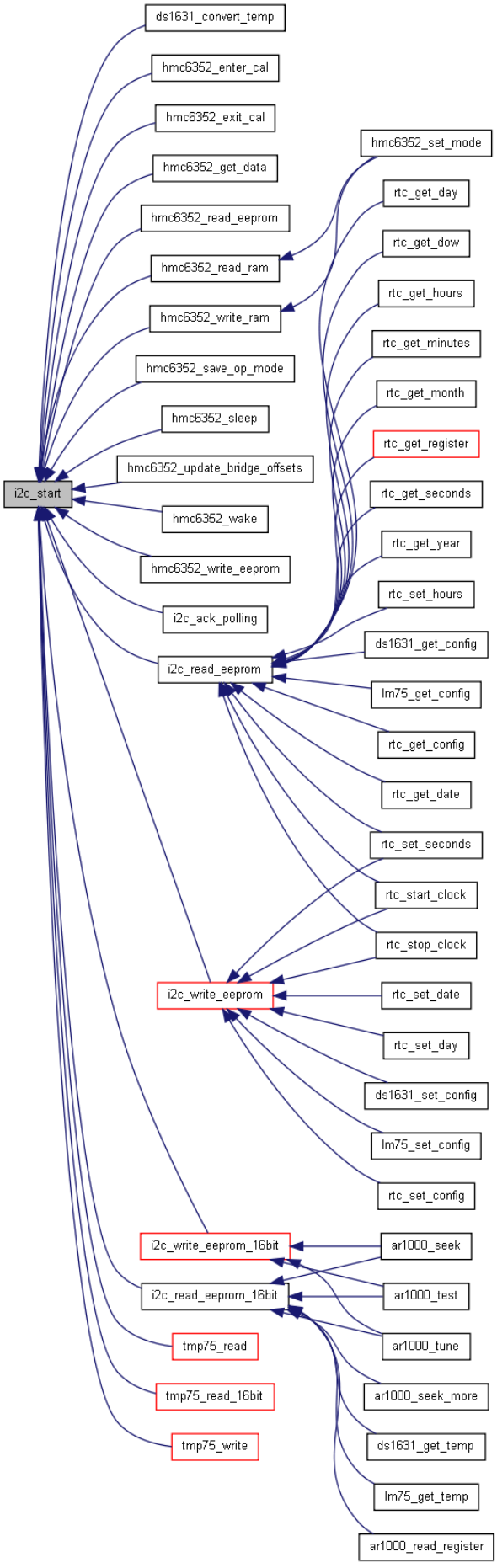
Signals a start condition on the I2C buss.

Definition at line 155 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_start
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
i2c_write_eeprom
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

**void i2c_stop (void)**

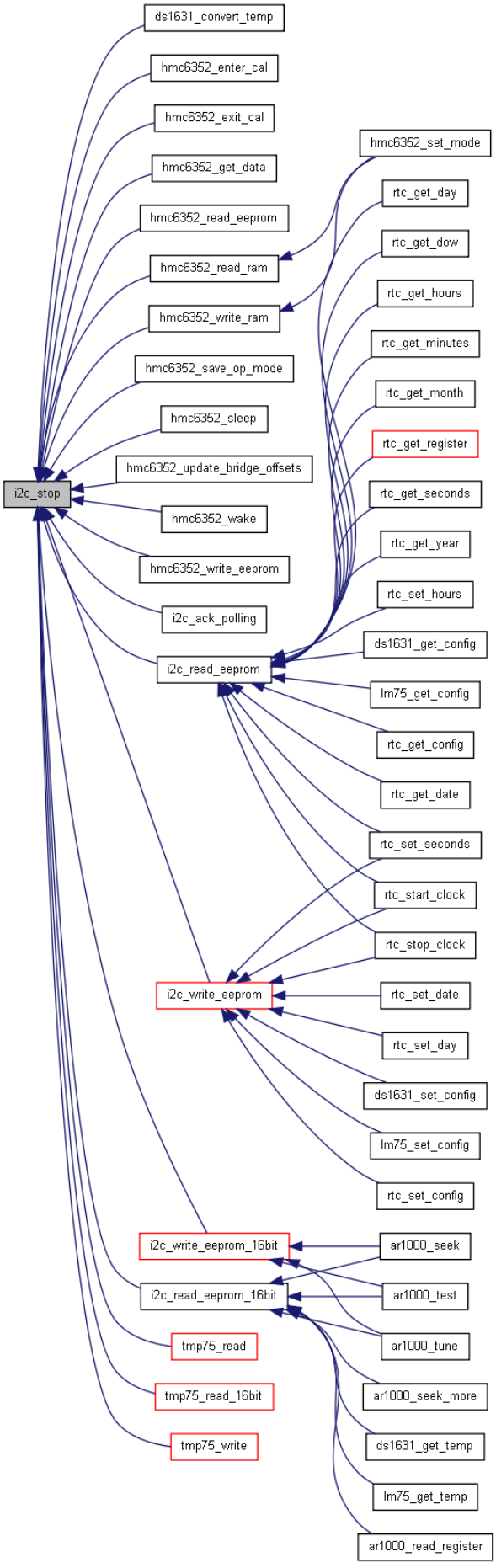
Signals a stop condition on the I2C buss.

Definition at line 171 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_stop
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
i2c_write_eeprom
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

**void i2c_write_eeprom (uns8 *device_address*, uns8 *mem_address*, uns8 *data*)**

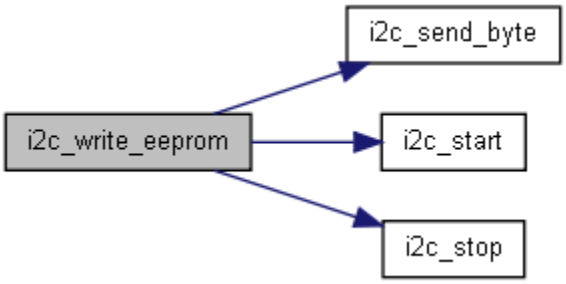
Write a byte to a given device address at the memory address

Definition at line 51 of file i2c.c.

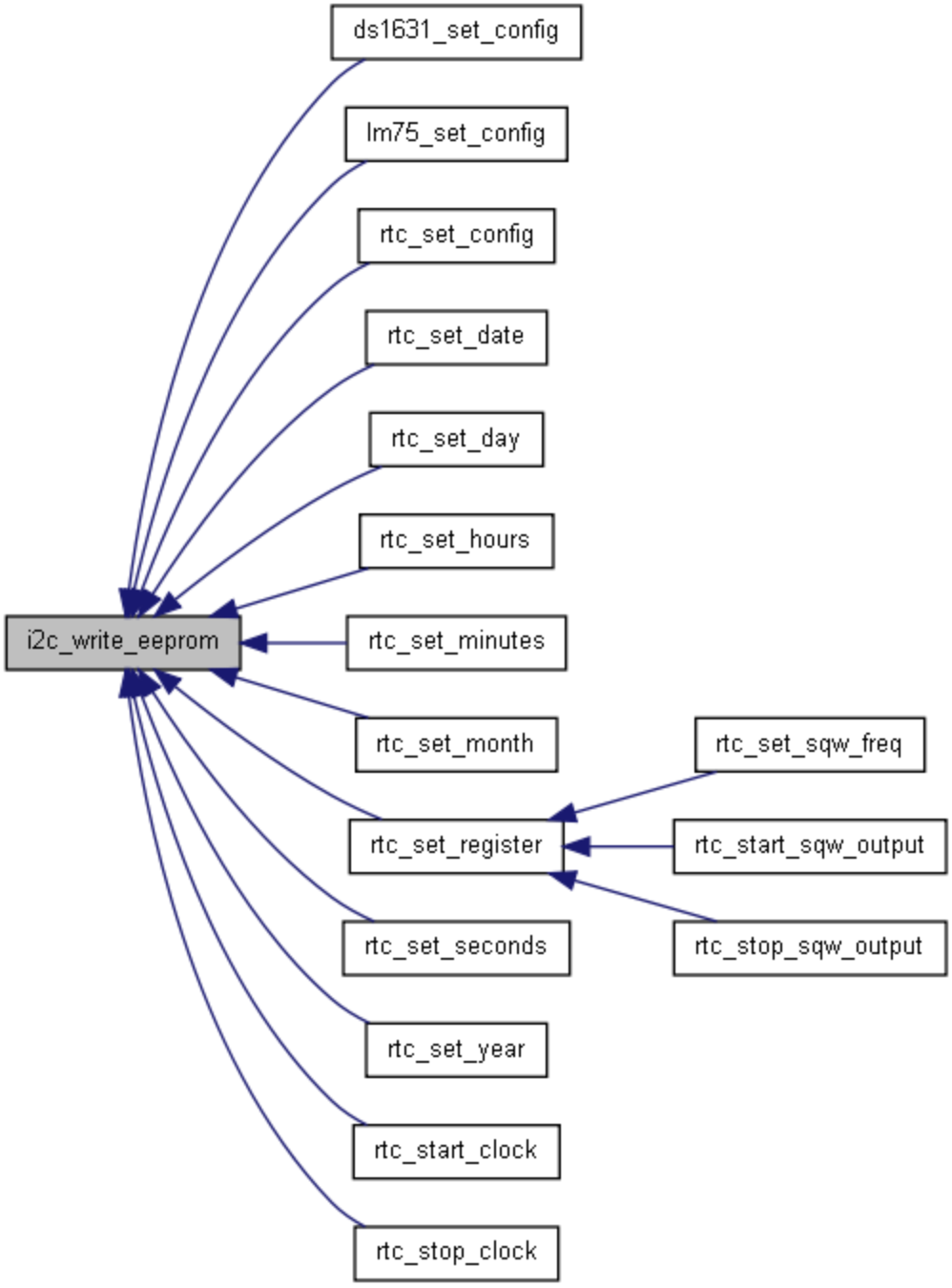
References i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by ds1631_set_config(), lm75_set_config(), rtc_set_config(), rtc_set_date(), rtc_set_day(), rtc_set_hours(), rtc_set_minutes(), rtc_set_month(), rtc_set_register(), rtc_set_seconds(), rtc_set_year(), rtc_start_clock(), and rtc_stop_clock().

Here is the call graph for this function:




Here is the caller graph for this function:




**void i2c_write_eeprom_16bit (uns8 *device_address*, uns8 *mem_address*, uns16 *data*)**

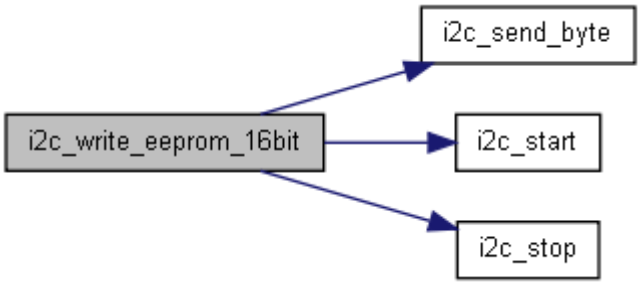
Write a byte to a given device address at the memory address

Definition at line 63 of file i2c.c.

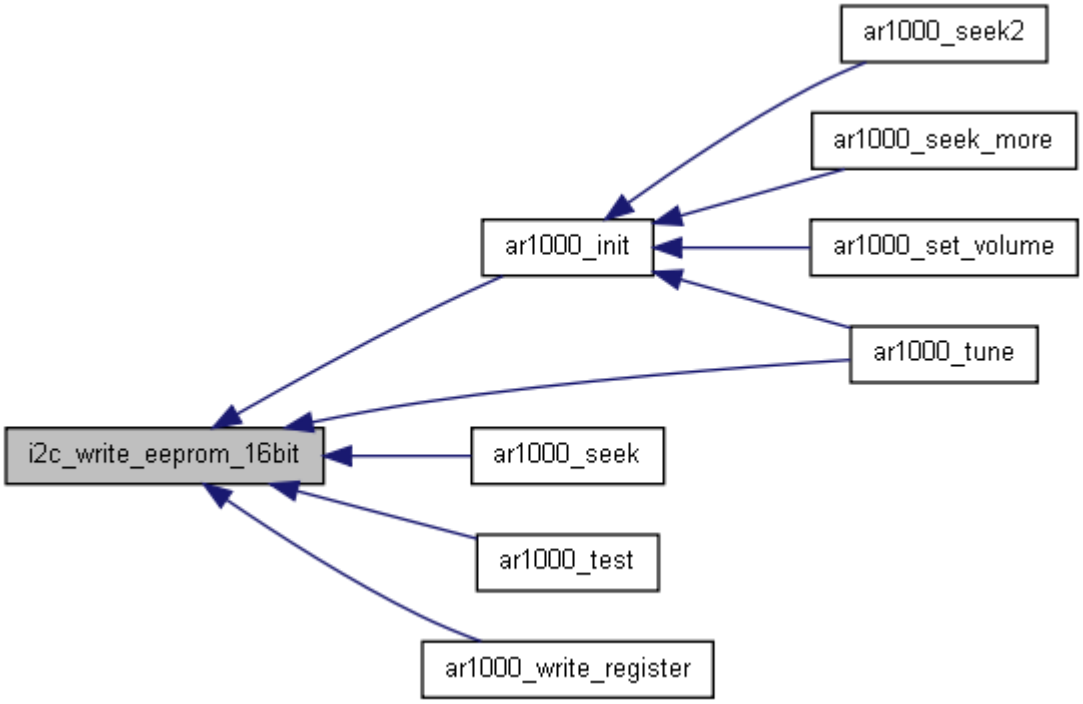
References i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by ar1000_init(), ar1000_seek(), ar1000_test(), ar1000_tune(), and ar1000_write_register().

Here is the call graph for this function:




Here is the caller graph for this function:



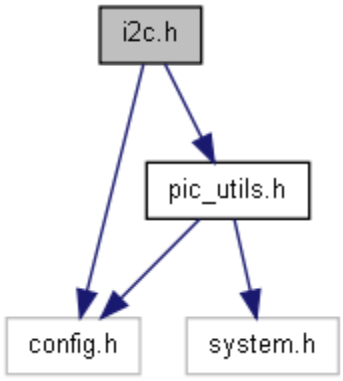

---

## i2c.h File Reference

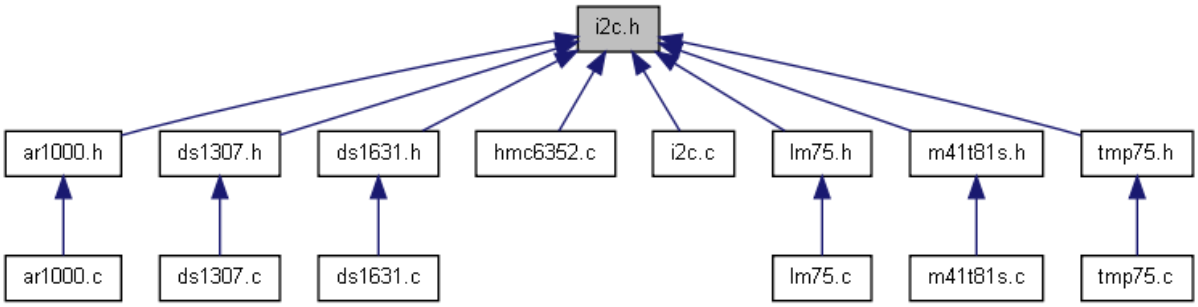
I2C software routines.

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

Include dependency graph for i2c.h:




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




## Defines

- #define __i2c_h defined
- #define i2c_read_sda() set_bit(tris_array[i2c_sda_port - PORTA], i2c_sda_pin);
- #define i2c_setup() i2c_setup_io()
- #define i2c_write_sda() clear_bit(tris_array[i2c_sda_port - PORTA], i2c_sda_pin);

## Functions

- uns8 i2c_read_eeprom (uns8 device_address, uns8 mem_address)
- *Read an 8 bit byte over I2C buss.* uns16 i2c_read_eeprom_16bit (uns8 device_address, uns8 mem_address)
- *Read 16 bits of data over I2C buss.* uns8 i2c_receive_byte ()
- *Receive byte from I2C buss.* void i2c_send_ack (void)
- *Send an ACK.* void i2c_send_byte (uns8 data)
- *Send a byte to I2C buss.* void i2c_setup_io ()
- *Setup ports and pins for I2C communication.* void i2c_start (void)
- *Send start signal to I2C buss.* void i2c_stop (void)
- *Send stop signal to I2C buss.* void i2c_write_eeprom (uns8 device_address, uns8 mem_address, uns8 data)
- *Write an 8 bit byte ove I2C buss.* void i2c_write_eeprom_16bit (uns8 device_address, uns8 mem_address, uns16 data)

*Write a 16 bit value over I2C buss.*

---

## Detailed Description

I2C communication routines. Although all standard functions are provided, you should only need to use i2c_setup, i2c_read_eeprom and i2c_write_eeprom

Definition in file i2c.h.

## Define Documentation

### #define __i2c_h  defined

Definition at line 45 of file i2c.h.

### #define i2c_read_sda()  set_bit(tris_array[i2c_sda_port - PORTA], i2c_sda_pin);

Change SDA line to read mode

Definition at line 70 of file i2c.h.

Referenced by i2c_receive_byte(), and i2c_send_byte().

### #define i2c_setup()  i2c_setup_io()

Definition at line 140 of file i2c.h.

Referenced by ar1000_setup_io(), ds1631_setup(), lm75_setup(), and tmp75_setup().

### #define i2c_write_sda()  clear_bit(tris_array[i2c_sda_port - PORTA], i2c_sda_pin);

Change SDA line to write mode

Definition at line 68 of file i2c.h.

Referenced by i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_send_ack(), i2c_send_byte(), i2c_start(), and i2c_stop().

## Function Documentation

### uns8 i2c_read_eeprom (uns8 *device_address*, uns8 *mem_address*)

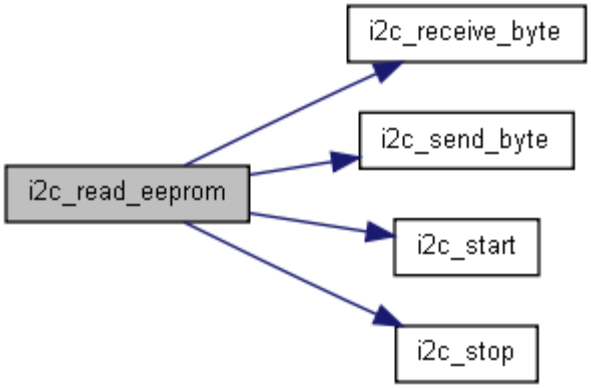
Read an 8 bit byte from the specified device at the memory address

Definition at line 76 of file i2c.c.

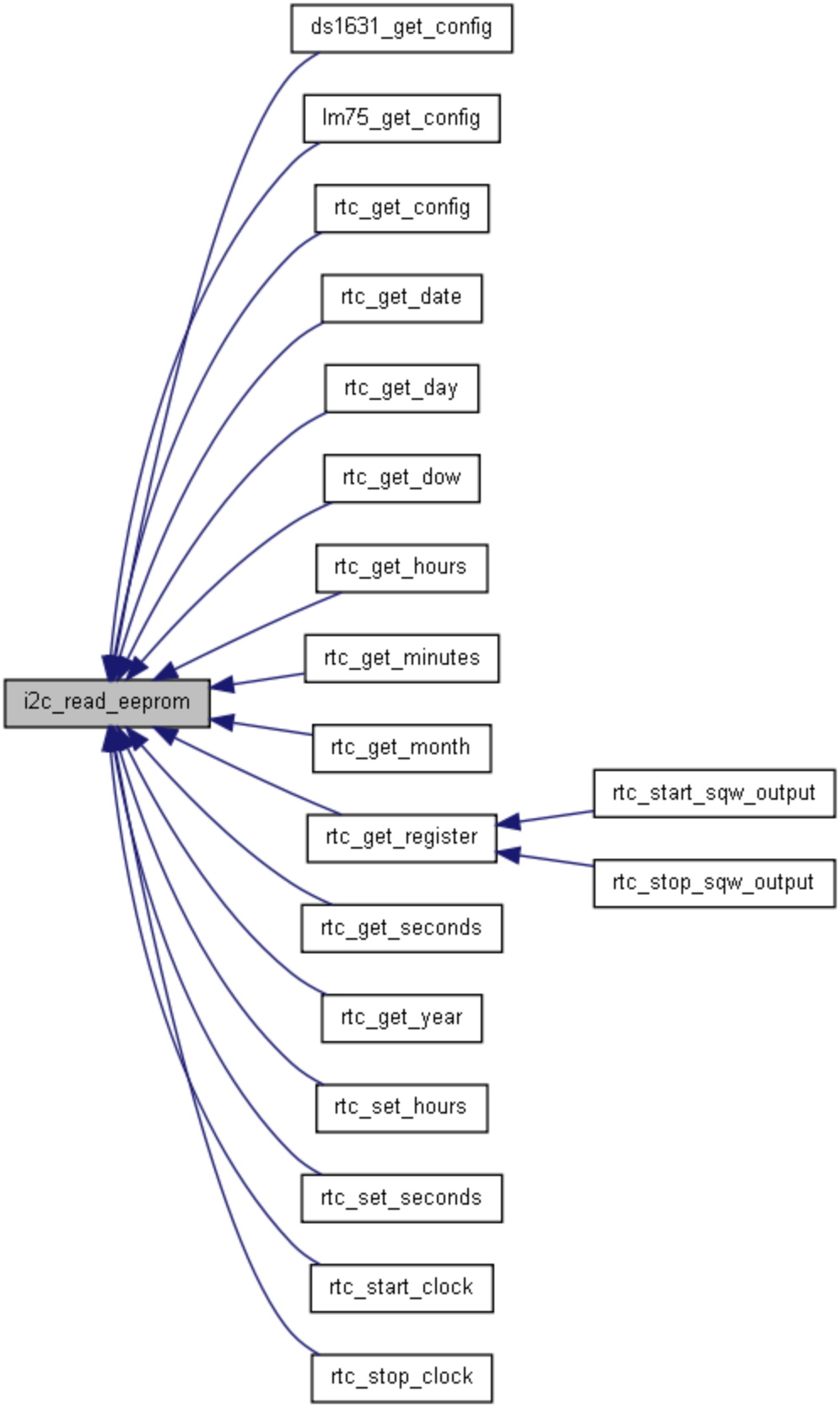
References clear_pin, DELAY_AMOUNT, i2c_receive_byte(), i2c_send_byte(), i2c_start(), i2c_stop(), i2c_write_sda, set_pin, and uns8.

Referenced by ds1631_get_config(), lm75_get_config(), rtc_get_config(), rtc_get_date(), rtc_get_day(), rtc_get_dow(), rtc_get_hours(), rtc_get_minutes(), rtc_get_month(), rtc_get_register(), rtc_get_seconds(), rtc_get_year(), rtc_set_hours(), rtc_set_seconds(), rtc_start_clock(), and rtc_stop_clock().

Here is the call graph for this function:




Here is the caller graph for this function:


ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
i2c_read_eeprom
rtc_get_month
rtc_start_sqw_output
rtc_get_register
rtc_stop_sqw_output
rtc_get_seconds
rtc_get_year
rtc_set_hours
rtc_set_seconds
rtc_start_clock
rtc_stop_clock

**uns16 i2c_read_eeprom_16bit (uns8 *device_address*, uns8 *mem_address*)**

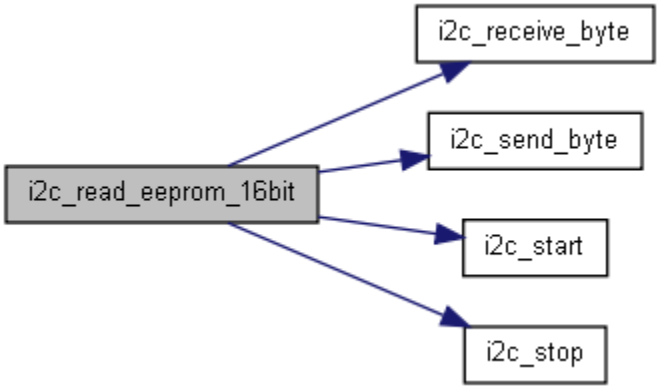
Read a 16 bit chunk of data from the given device and memory address

Definition at line 103 of file i2c.c.

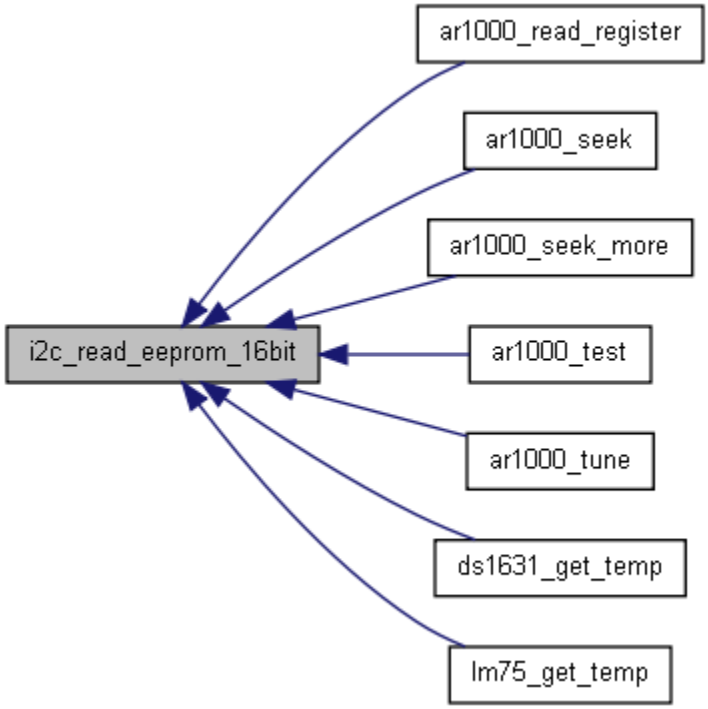
References clear_pin, DELAY_AMOUNT, i2c_receive_byte(), i2c_send_byte(), i2c_start(), i2c_stop(), i2c_write_sda, set_pin, and uns16.

Referenced by ar1000_read_register(), ar1000_seek(), ar1000_seek_more(), ar1000_test(), ar1000_tune(), ds1631_get_temp(), and lm75_get_temp().

Here is the call graph for this function:




Here is the caller graph for this function:




**uns8 i2c_receive_byte ()**

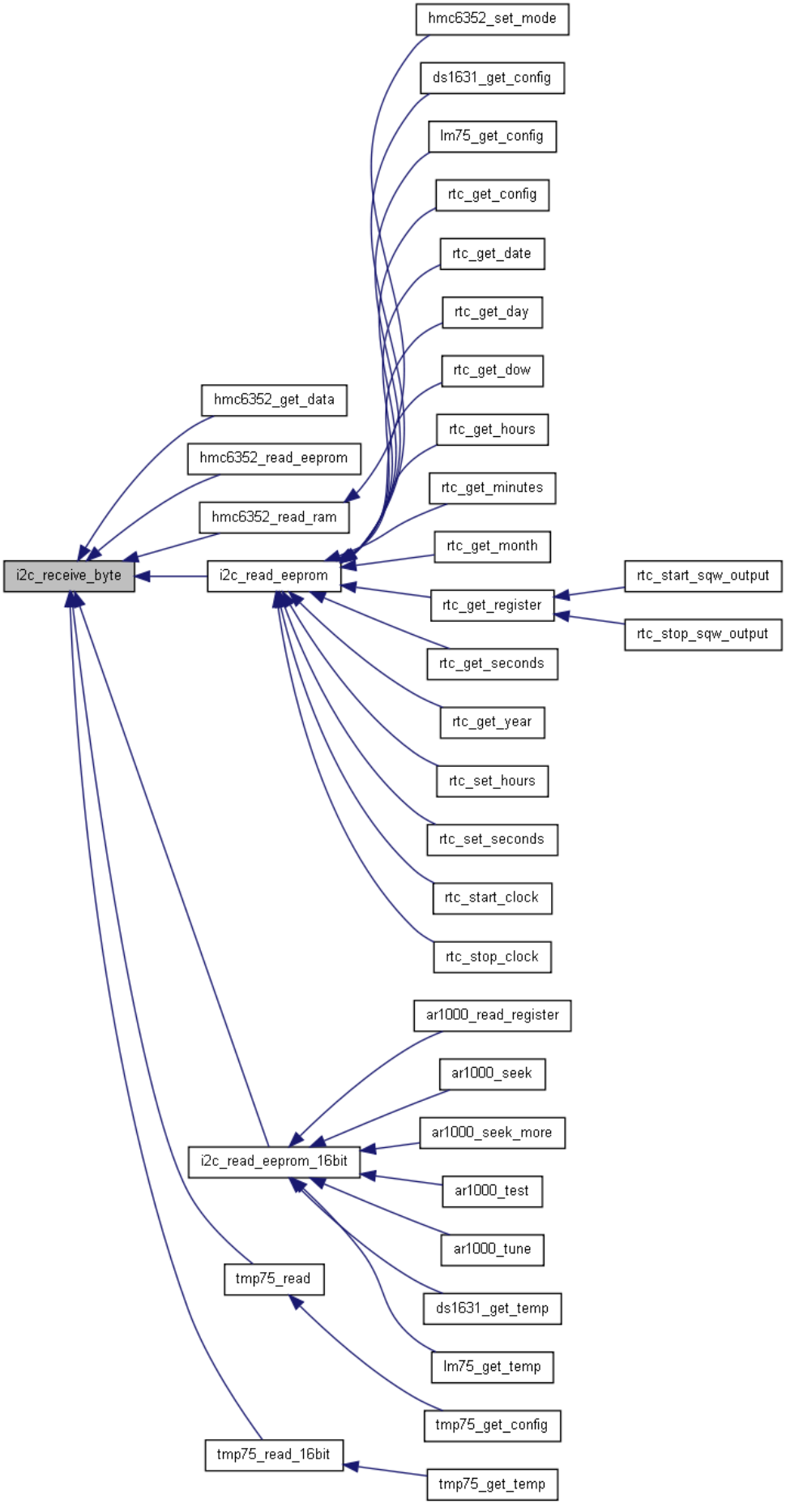
Clock in a byte over I2C lines. Note that an acknowledge is not sent/received

Definition at line 186 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_read_sda, set_pin, test_pin, and uns8.

Referenced by hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), tmp75_read(), and tmp75_read_16bit().

Here is the caller graph for this function:


hmc6352_set_mode
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_start_sqw_output
rtc_stop_sqw_output
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
i2c_receive_byte
i2c_read_eeprom
ar1000_read_register
ar1000_seek
ar1000_seek_more
i2c_read_eeprom_16bit
ar1000_test
ar1000_tune
tmp75_read
ds1631_get_temp
lm75_get_temp
tmp75_get_config
tmp75_read_16bit
tmp75_get_temp

**void i2c_send_ack (void)**

Sends an I2C acknowledge (bit)

Definition at line 140 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), and tmp75_read_16bit().

Here is the caller graph for this function:


**void i2c_send_byte (uns8 *data*)**

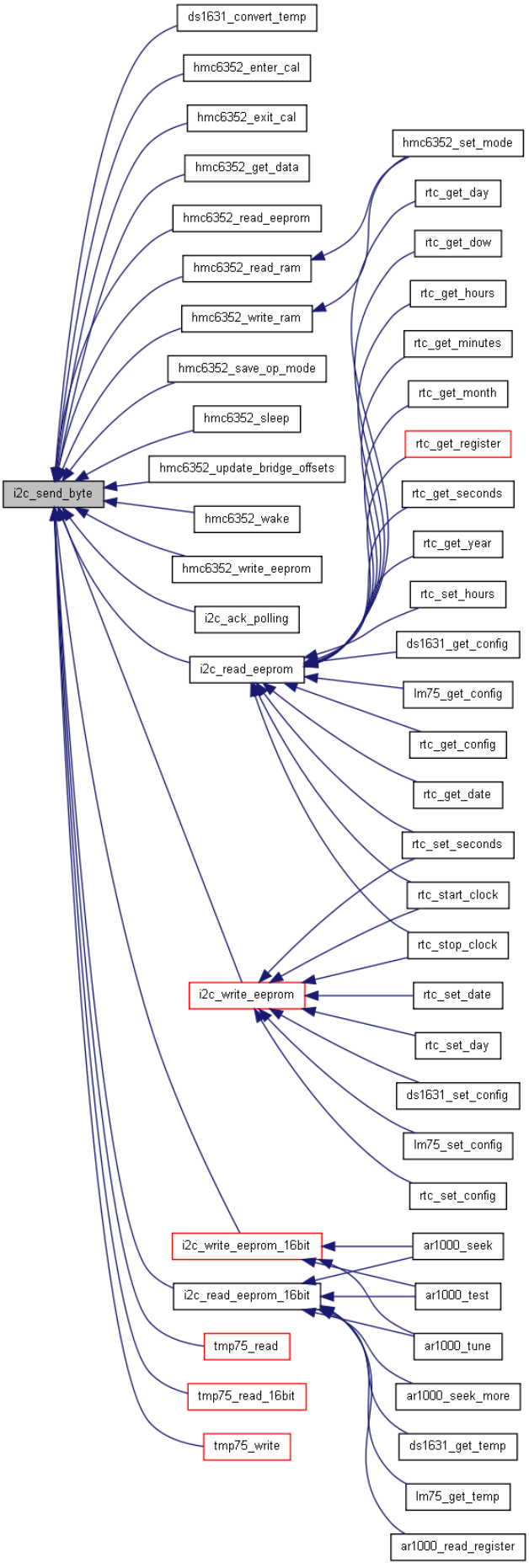
Clock out a byte from the I2C buss. An acknowledge is "received" (although ignored).

Definition at line 206 of file i2c.c.

References change_pin, clear_pin, DELAY_AMOUNT, i2c_read_sda, i2c_write_sda, set_pin, and uns8.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_send_byte
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
i2c_write_eeprom
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

### void i2c_setup_io ()

Set port and pins correctly for I2C communication

Definition at line 231 of file i2c.c.

### void i2c_start (void)

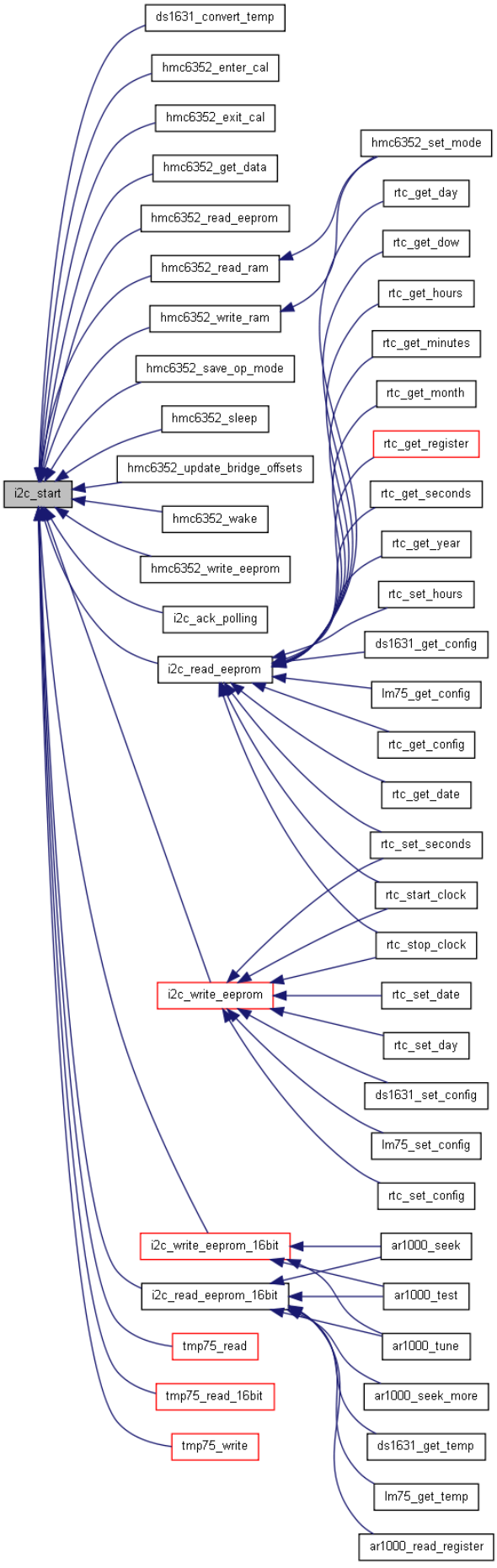
Signals a start condition on the I2C buss.

Definition at line 155 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_start
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
i2c_write_eeprom
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

**void i2c_stop (void)**

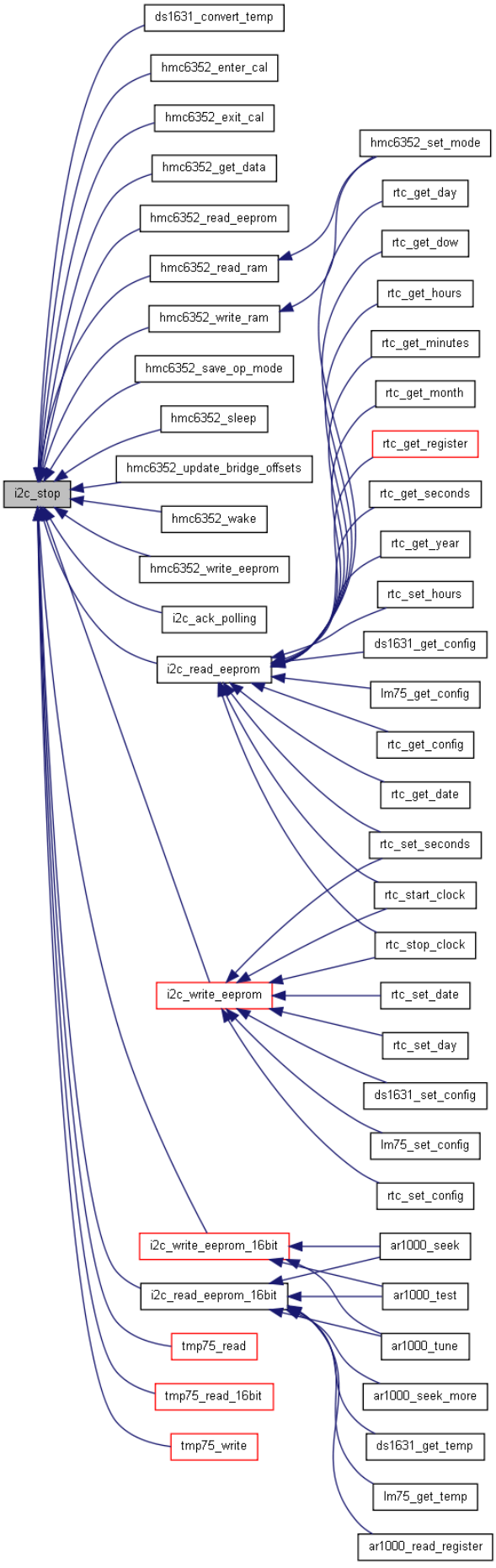
Signals a stop condition on the I2C buss.

Definition at line 171 of file i2c.c.

References clear_pin, DELAY_AMOUNT, i2c_write_sda, and set_pin.

Referenced by ds1631_convert_temp(), hmc6352_enter_cal(), hmc6352_exit_cal(), hmc6352_get_data(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_save_op_mode(), hmc6352_sleep(), hmc6352_update_bridge_offsets(), hmc6352_wake(), hmc6352_write_eeprom(), hmc6352_write_ram(), i2c_ack_polling(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_write_eeprom(), i2c_write_eeprom_16bit(), tmp75_read(), tmp75_read_16bit(), and tmp75_write().

Here is the caller graph for this function:


ds1631_convert_temp
hmc6352_enter_cal
hmc6352_exit_cal
hmc6352_get_data
hmc6352_read_eeprom
hmc6352_read_ram
hmc6352_write_ram
hmc6352_save_op_mode
hmc6352_sleep
hmc6352_update_bridge_offsets
i2c_stop
hmc6352_wake
hmc6352_write_eeprom
i2c_ack_polling
i2c_read_eeprom
i2c_write_eeprom
i2c_write_eeprom_16bit
i2c_read_eeprom_16bit
tmp75_read
tmp75_read_16bit
tmp75_write
hmc6352_set_mode
rtc_get_day
rtc_get_dow
rtc_get_hours
rtc_get_minutes
rtc_get_month
rtc_get_register
rtc_get_seconds
rtc_get_year
rtc_set_hours
ds1631_get_config
lm75_get_config
rtc_get_config
rtc_get_date
rtc_set_seconds
rtc_start_clock
rtc_stop_clock
rtc_set_date
rtc_set_day
ds1631_set_config
lm75_set_config
rtc_set_config
ar1000_seek
ar1000_test
ar1000_tune
ar1000_seek_more
ds1631_get_temp
lm75_get_temp
ar1000_read_register

**void i2c_write_eeprom (uns8 *device_address*, uns8 *mem_address*, uns8 *data*)**

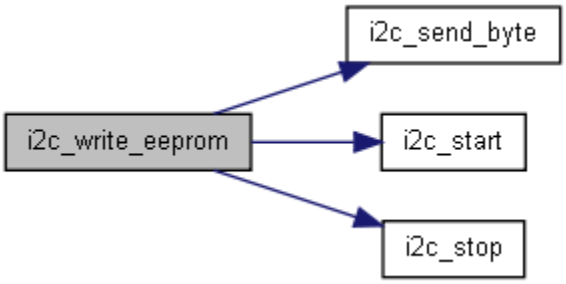
Write a byte to a given device address at the memory address

Definition at line 51 of file i2c.c.

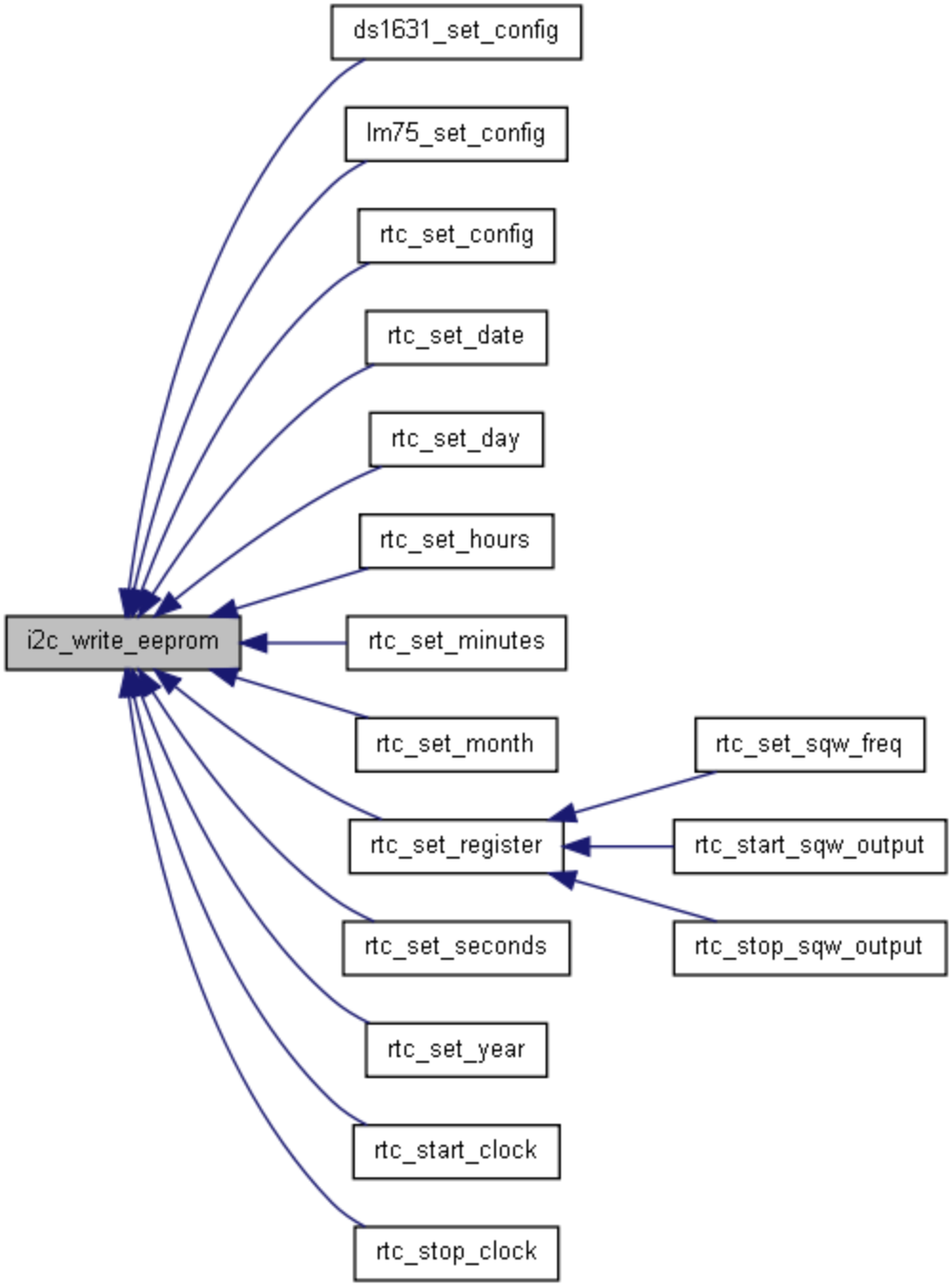
References i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by ds1631_set_config(), lm75_set_config(), rtc_set_config(), rtc_set_date(), rtc_set_day(), rtc_set_hours(), rtc_set_minutes(), rtc_set_month(), rtc_set_register(), rtc_set_seconds(), rtc_set_year(), rtc_start_clock(), and rtc_stop_clock().

Here is the call graph for this function:




Here is the caller graph for this function:




**void i2c_write_eeprom_16bit (uns8 *device_address*, uns8 *mem_address*, uns16 *data*)**

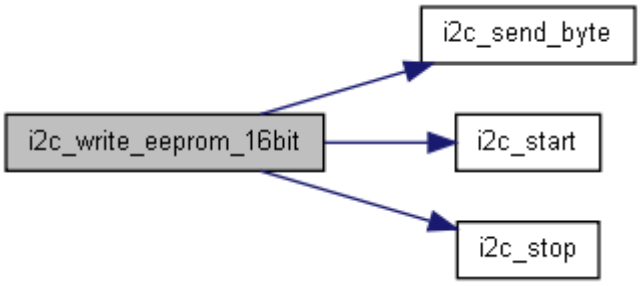
Write a byte to a given device address at the memory address

Definition at line 63 of file i2c.c.

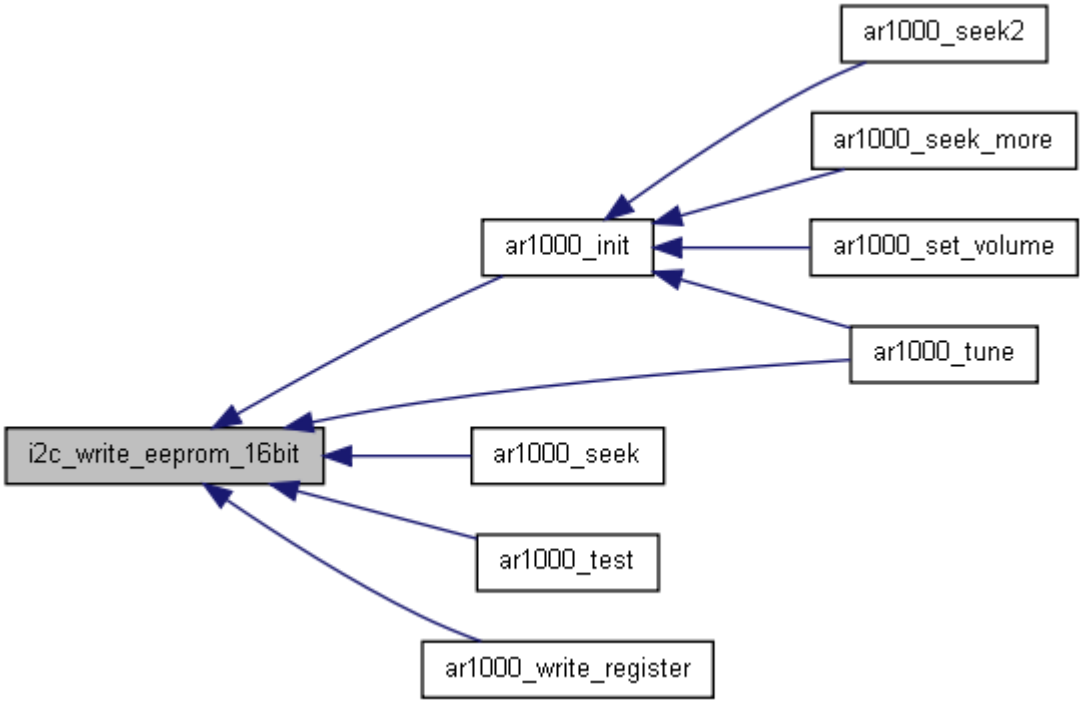
References i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by ar1000_init(), ar1000_seek(), ar1000_test(), ar1000_tune(), and ar1000_write_register().

Here is the call graph for this function:




Here is the caller graph for this function:

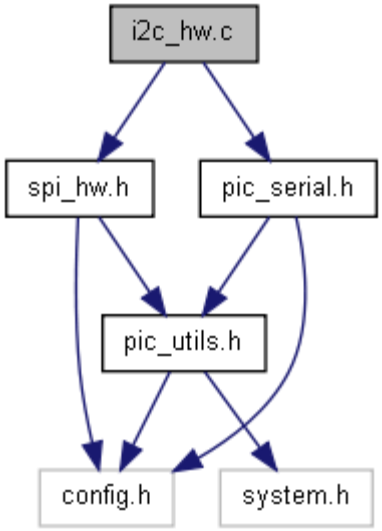



---

## i2c_hw.c File Reference

```
#include "spi_hw.h"
#include "pic_serial.h"
```

Include dependency graph for i2c_hw.c:




## Functions

- void spi_hw_init ()
- uns8 spi_hw_receive ()
- void spi_hw_setup_io ()
- void spi_hw_transmit (uns8 data)

---

## Function Documentation

### void spi_hw_init ()

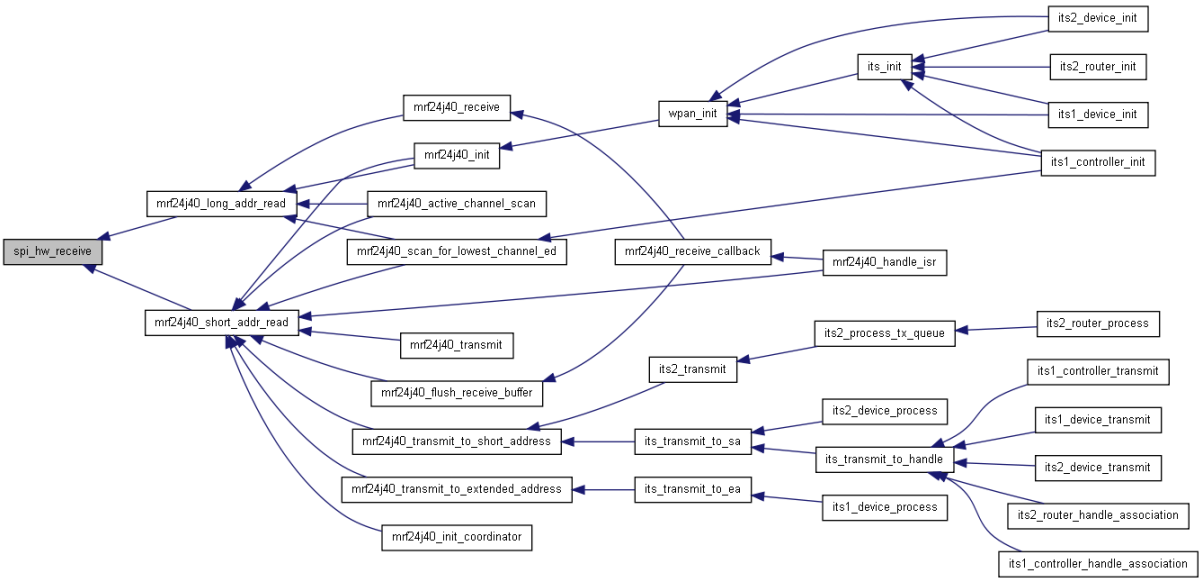
Definition at line 82 of file i2c_hw.c.

### uns8 spi_hw_receive ()

Definition at line 106 of file i2c_hw.c.

Referenced by mrf24j40_long_addr_read(), and mrf24j40_short_addr_read().

Here is the caller graph for this function:




### void spi_hw_setup_io ()

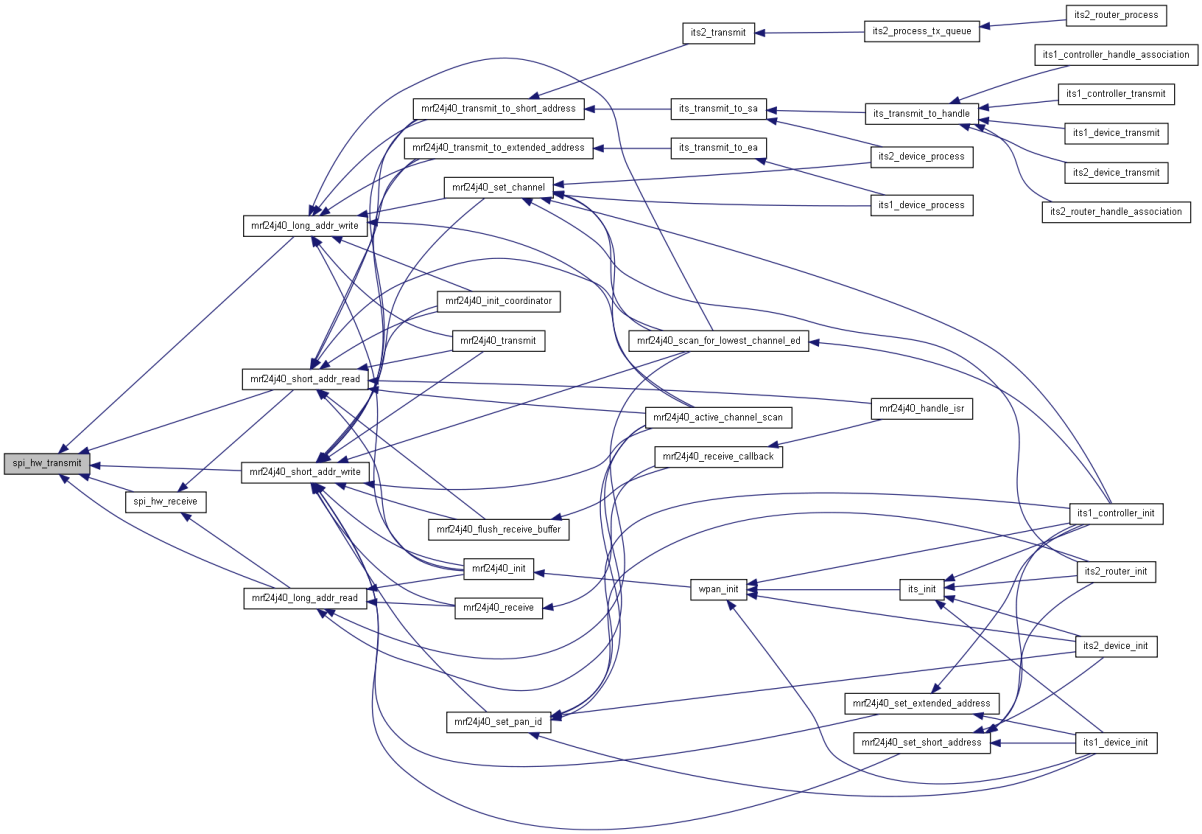
Definition at line 40 of file i2c_hw.c.

### void spi_hw_transmit (uns8 *data*)

Definition at line 91 of file i2c_hw.c.

Referenced by mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), and spi_hw_receive().

Here is the caller graph for this function:



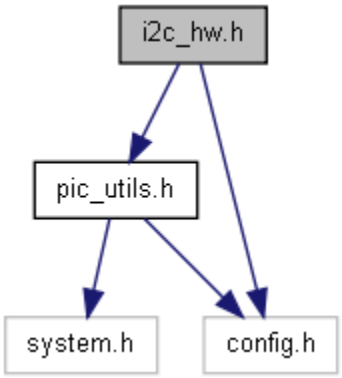

---

## i2c_hw.h File Reference

Outputs i2c interfaces (clock+data).

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

Include dependency graph for i2c_hw.h:



## Functions

- void i2c_pulse_0 ()
- *i2c test routine* void i2c_pulse_1 ()
- *i2c test routine* void i2c_setup_io ()
- *Setup ports and pins for i2c output.* void i2c_write (uns8 data)
- *Send a byte of data using software i2c.* void i2c_write_lsb (uns8 data)

*Send a byte of data using software i2c.*

---

## Detailed Description

Covers standard i2c-like interfaces (clock + data) and Sure Electronics displays which are a little different

Definition in file i2c_hw.h.

---

## Function Documentation

### void i2c_pulse_0 ()

### void i2c_pulse_1 ()

### void i2c_setup_io ()

Setup ports and pins for i2c output

Setup ports and pins for i2c output.

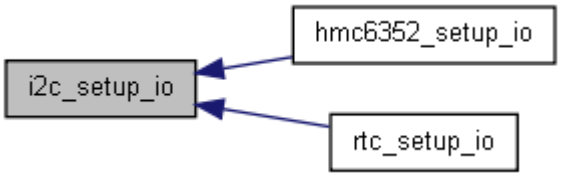
Set port and pins correctly for I2C communication

Definition at line 231 of file i2c.c.

References make_input, and make_output.

Referenced by hmc6352_setup_io(), and rtc_setup_io().

Here is the caller graph for this function:




### void i2c_write (uns8 *data*)

Sends a byte of data MSB first, data only changes on clock low

### void i2c_write_lsb (uns8 *data*)

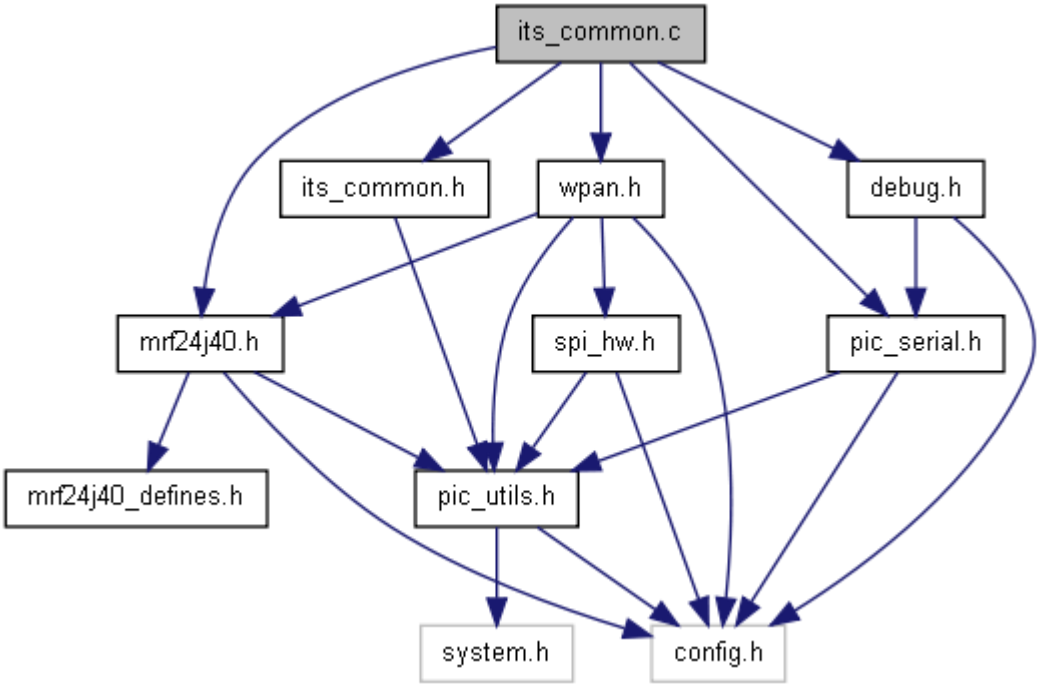
Sends a byte of data LSB first, data only changes on clock low

---

# its_common.c File Reference

```
#include "its_common.h"
#include "mrf24j40.h"
#include "pic_serial.h"
#include "wpan.h"
#include "debug.h"
```

Include dependency graph for its_common.c:




## Functions

- its_device_handle its_add_local_device (uns16 device_id, uns16 pan_id, uns16 short_address)
- its_device_handle its_add_net_device (uns16 device_id, uns16 previous_hop)
- its_device_handle its_get_device_handle (uns16 device_id)
- uns16 its_get_device_id ()
- its_device_info * its_get_device_info (uns8 handle)
- uns16 its_get_network_id ()
- *Retrieve the current network ID.* uns8 its_get_next_sequence ()
- void its_init ()
- *Initialise ITS and lower layers.* void its_print_devices ()
- *Print devices currently known to this one.* void its_set_device_id (uns16 device_id)
- *Set the ITS device ID.* void its_set_network_id (uns16 network_id)
- *Set the ITS network ID.* void its_transmit_to_ea (uns8 *dest_ea, uns16 dest_its_device_id, uns8 packet_type, uns8 *data, uns8 data_length)
- void its_transmit_to_handle (its_device_handle handle, uns8 packet_type, uns8 *data, uns8 data_length)
- void its_transmit_to_sa (uns16 dest_pan_id, uns16 dest_sa, uns16 dest_device_id, uns8 packet_type, uns8 *data, uns8 data_length)

## Variables

- uns16 its_device_id
- its_device_info its_devices [ITS_MAX_KNOWN_DEVICES]

- uns16 its_network_id
- uns8 its_sequence = 0

---

## Function Documentation

### its_device_handle its_add_local_device (uns16 *device_id*, uns16 *pan_id*, uns16 *short_address*)

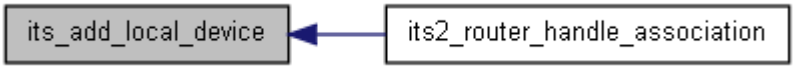
Definition at line 127 of file its_common.c.

References its_device_info::addr, debug_int_hex_16bit, debug_str, its_device_info::its_device_id, its_device_id, ITS_DEVICE_NONE, its_print_devices(), its_address::local, local_address::pan_id, local_address::short_address, and uns8.

Referenced by its2_router_handle_association().

Here is the call graph for this function:


Here is the caller graph for this function:




### its_device_handle its_add_net_device (uns16 *device_id*, uns16 *previous_hop*)

Definition at line 158 of file its_common.c.

References its_device_info::addr, debug_int_hex_16bit, debug_str, its_device_info::its_device_id, its_device_id, ITS_DEVICE_NONE, its_print_devices(), remote_address::prior_device_id, its_address::remote, remote_address::remote_indicator, and uns8.

Here is the call graph for this function:

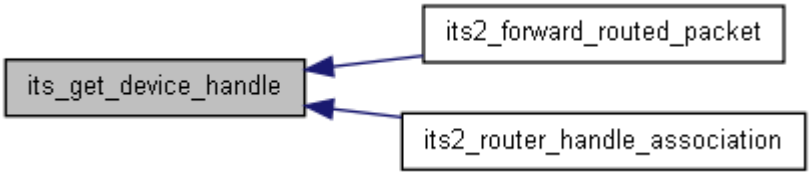

### its_device_handle its_get_device_handle (uns16 *device_id*)

Definition at line 108 of file its_common.c.

References its_device_id, ITS_DEVICE_NONE, and uns8.

Referenced by its2_forward_routed_packet(), and its2_router_handle_association().

Here is the caller graph for this function:



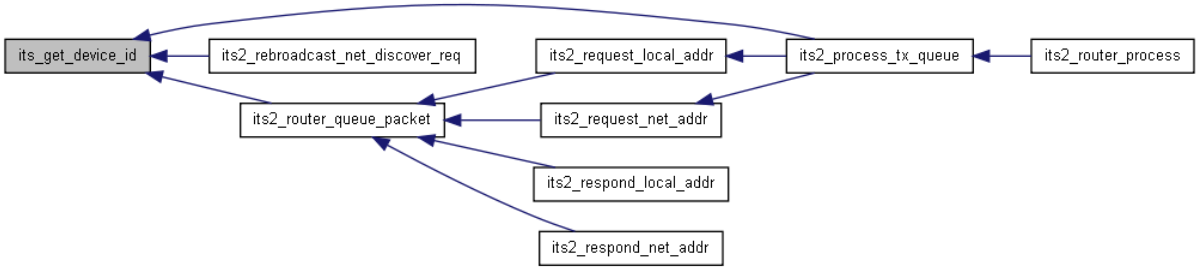
### uns16 its_get_device_id ()

Definition at line 89 of file its_common.c.

References its_device_id.

Referenced by its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

Here is the caller graph for this function:

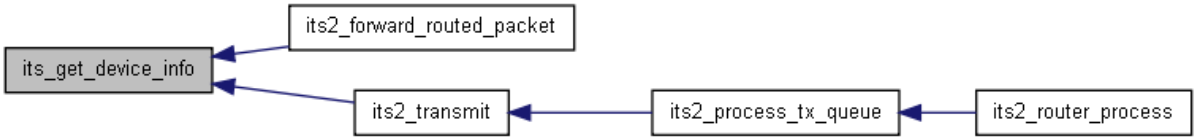



### its_device_info* its_get_device_info (uns8 *handle*)

Definition at line 100 of file its_common.c.

Referenced by its2_forward_routed_packet(), and its2_transmit().

Here is the caller graph for this function:



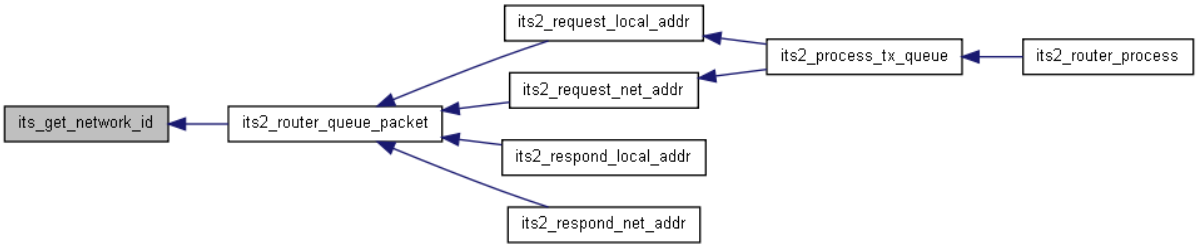

### uns16 its_get_network_id ()

Definition at line 81 of file its_common.c.

References its_network_id.

Referenced by its2_router_queue_packet().

Here is the caller graph for this function:



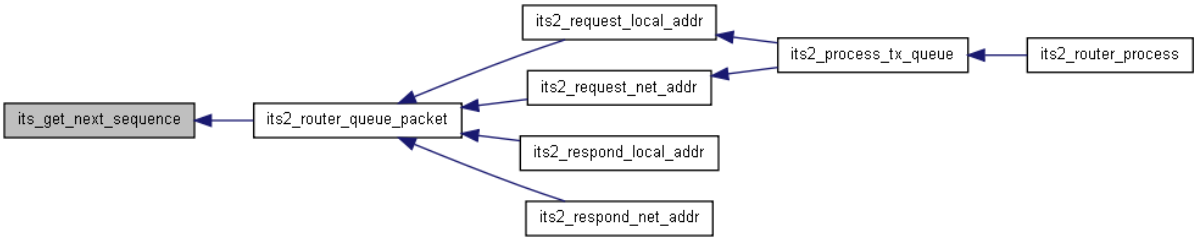

### uns8 its_get_next_sequence ()

Definition at line 72 of file its_common.c.

References its_sequence.

Referenced by its2_router_queue_packet().

Here is the caller graph for this function:




**void its_init ()**

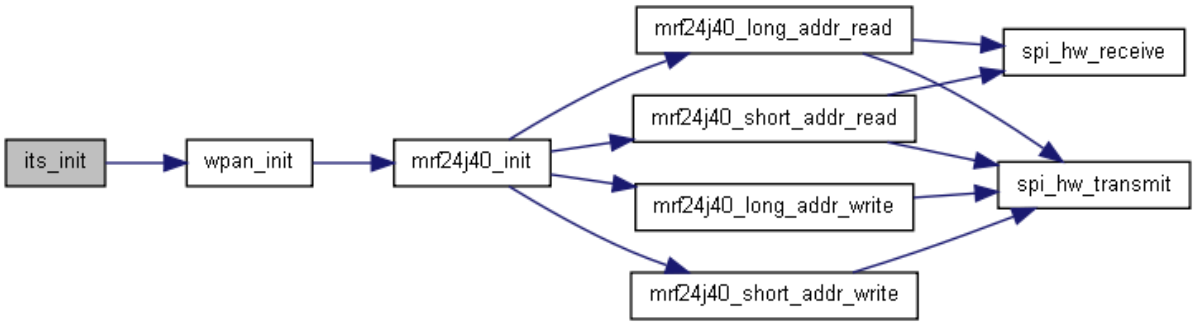
Call before using any ITS functionality

Definition at line 93 of file its_common.c.

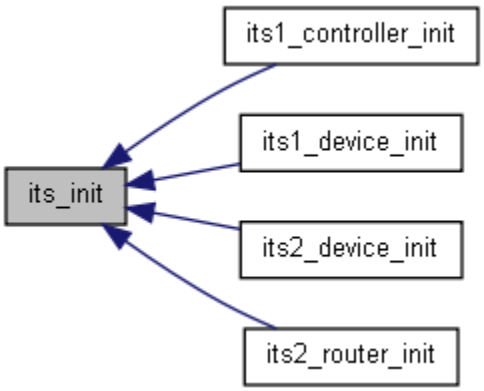
References its_device_info::its_device_id, uns8, and wpan_init().

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the call graph for this function:




Here is the caller graph for this function:



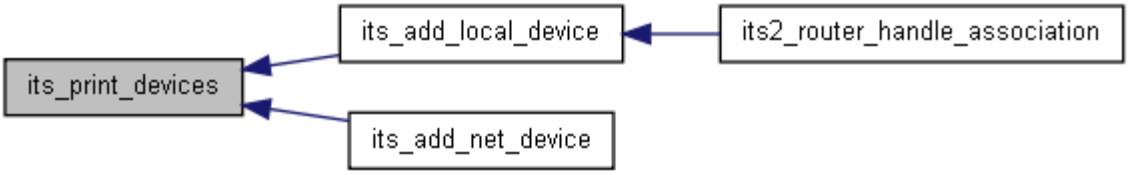

**void its_print_devices ()**

Definition at line 51 of file its_common.c.

References debug_int_hex_16bit, debug_nl, debug_str, its_device_id, and uns8.

Referenced by its_add_local_device(), and its_add_net_device().

Here is the caller graph for this function:




**void its_set_device_id (uns16 *device_id*)**

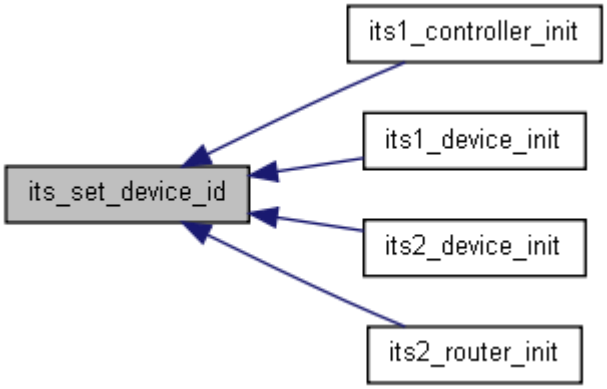
Set the ITS device ID before sending packets. See also: its_set_network_id(uns16 network_id);

Definition at line 85 of file its_common.c.

References its_device_id.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the caller graph for this function:




**void its_set_network_id (uns16 *network_id*)**

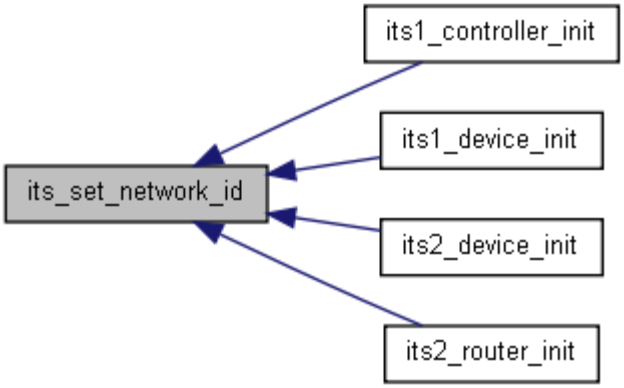
Set the ITS network ID before sending packets See also: its_set_device_id(uns16 device_id);

Definition at line 76 of file its_common.c.

References its_network_id.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the caller graph for this function:



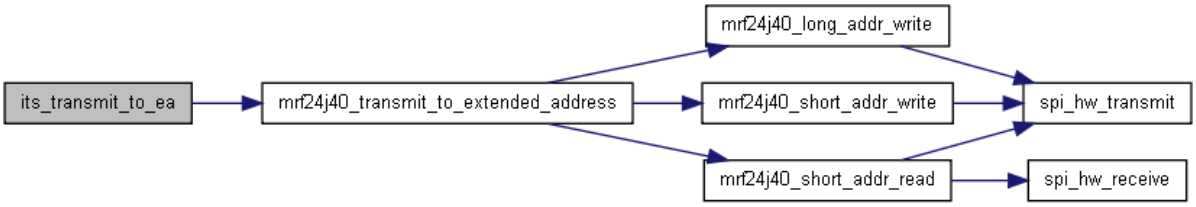

**void its_transmit_to_ea (uns8 * *dest_ea*, uns16 *dest_its_device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 232 of file its_common.c.

References FRAME_TYPE_DATA, its_device_id, its_network_id, its_sequence, mrf24j40_transmit_to_extended_address(), MRF_NO_ACK, and uns8.

Referenced by its1_device_process().

Here is the call graph for this function:




Here is the caller graph for this function:

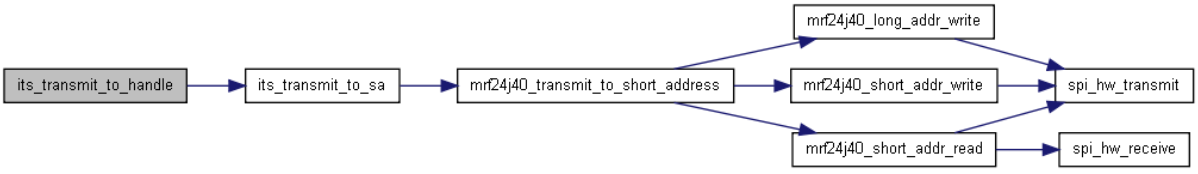

**void its_transmit_to_handle (its_device_handle *handle*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 185 of file its_common.c.

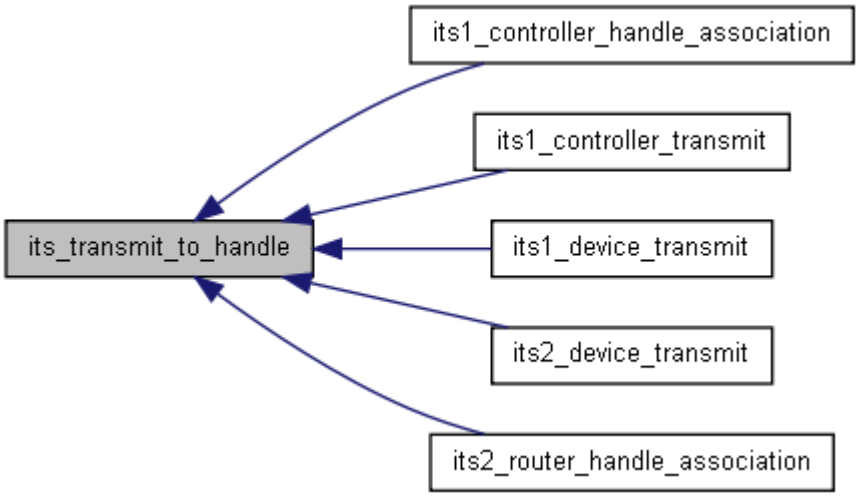
References its_device_info::addr, its_device_info::its_device_id, its_transmit_to_sa(), its_address::local, and local_address::short_address.

Referenced by its1_controller_handle_association(), its1_controller_transmit(), its1_device_transmit(), its2_device_transmit(), and its2_router_handle_association().

Here is the call graph for this function:



Here is the caller graph for this function:



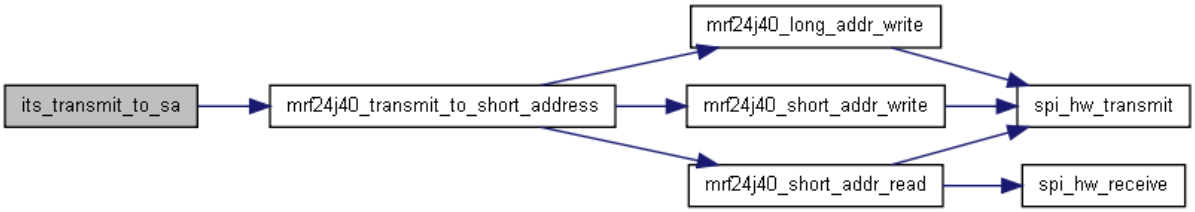

**void its_transmit_to_sa (uns16 *dest_pan_id*, uns16 *dest_sa*, uns16 *dest_device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 195 of file its_common.c.

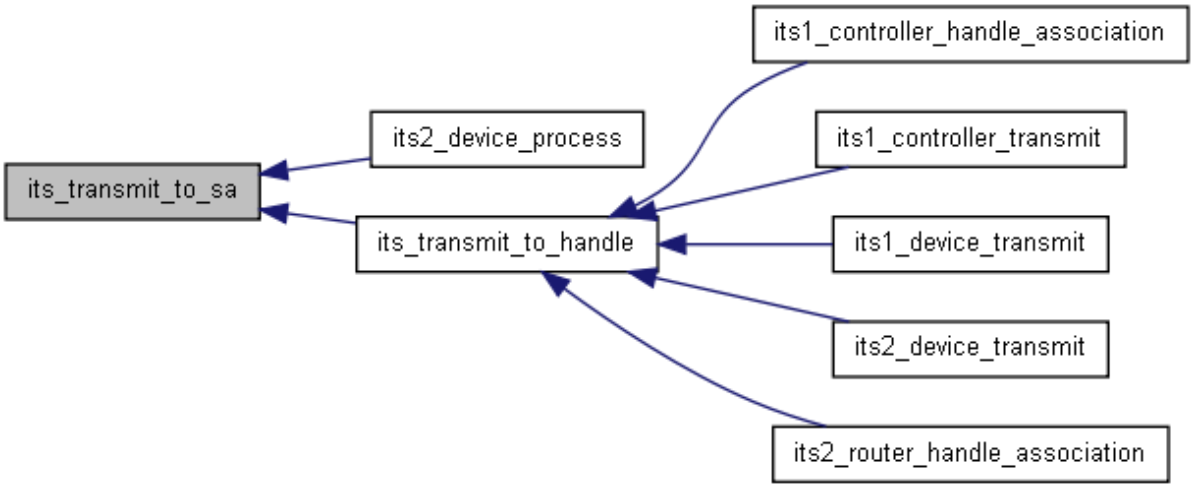
References FRAME_TYPE_DATA, its_device_id, its_network_id, its_sequence, mrf24j40_transmit_to_short_address(), MRF_ACK, uns16, and uns8.

Referenced by its2_device_process(), and its_transmit_to_handle().

Here is the call graph for this function:




Here is the caller graph for this function:




---

### Variable Documentation

**uns16 its_device_id**

Definition at line 47 of file its_common.c.

Referenced by its_add_local_device(), its_add_net_device(), its_get_device_handle(), its_get_device_id(), its_print_devices(), its_set_device_id(), its_transmit_to_ea(), and its_transmit_to_sa().

**its_device_info its_devices[ITS_MAX_KNOWN_DEVICES]**

Definition at line 44 of file its_common.c.

**uns16 its_network_id**

Definition at line 46 of file its_common.c.

Referenced by its_get_network_id(), its_set_network_id(), its_transmit_to_ea(), and its_transmit_to_sa().

**uns8 its_sequence = 0**

Definition at line 49 of file its_common.c.

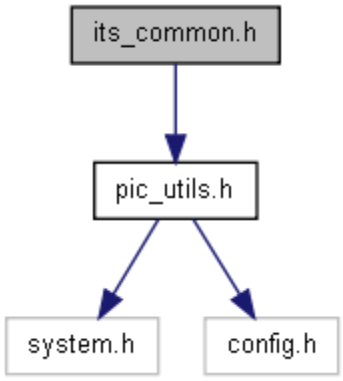
Referenced by its_get_next_sequence(), its_transmit_to_ea(), and its_transmit_to_sa().

---

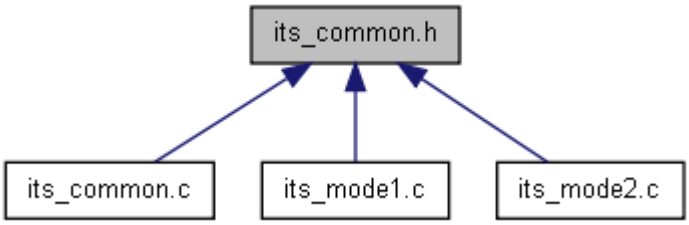
## its_common.h File Reference

```
#include "pic_utils.h"
```

Include dependency graph for its_common.h:




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




## Data Structures

- union its_address
- struct its_device_info
- struct local_address
- struct remote_address

## Defines

- #define ITS_ACK 0x03
- #define ITS_APP_DATA 0x02
- #define ITS_ASSOC_REQ 0x00
- #define ITS_ASSOC_RES 0x01
- #define ITS_DEVICE_NONE 0xff
- #define ITS_ENDPOINT_DATA 0x0c
- #define ITS_ENDPOINT_REQ 0x0a
- #define ITS_ENDPOINT_RES 0x0b
- #define ITS_GENERIC_DATA 0x09
- #define ITS_LOCAL_DISCOVER_REQ 0x05
- #define ITS_LOCAL_DISCOVER_RES 0x06
- #define ITS_NET_DISCOVER_REQ 0x07
- #define ITS_NET_DISCOVER_RES 0x08
- #define ITS_PENDING_DATA_REQ 0x04
- #define ITS_ROUTE_FAILURE 0x0d

## Typedefs

- typedef uns8 its_device_handle

## Functions

- its_device_handle its_add_local_device (uns16 device_id, uns16 pan_id, uns16 short_address)
- its_device_handle its_add_net_device (uns16 device_id, uns16 previous_hop)
- its_device_handle its_get_device_handle (uns16 device_id)
- uns16 its_get_device_id ()
- its_device_info * its_get_device_info (uns8 handle)
- uns16 its_get_network_id ()
- *Retrieve the current network ID.* uns8 its_get_next_sequence ()
- void its_init ()
- *Initialise ITS and lower layers.* void its_print_devices ()
- *Print devices currently known to this one.* void its_set_device_id (uns16 device_id)
- *Set the ITS device ID.* void its_set_network_id (uns16 network_id)
- *Set the ITS network ID.* void its_transmit_to_ea (uns8 *dest_ea, uns16 dest_its_device_id, uns8 packet_type, uns8 *data, uns8 data_length)
- void its_transmit_to_handle (its_device_handle handle, uns8 packet_type, uns8 *data, uns8 data_length)
- void its_transmit_to_sa (uns16 dest_pan_id, uns16 dest_sa, uns16 dest_device_id, uns8 packet_type, uns8 *data, uns8 data_length)

---

## Define Documentation

### #define ITS_ACK 0x03

Definition at line 62 of file its_common.h.

### #define ITS_APP_DATA 0x02

Definition at line 61 of file its_common.h.

### #define ITS_ASSOC_REQ 0x00

Definition at line 59 of file its_common.h.

Referenced by its1_device_process(), and its2_device_process().

### #define ITS_ASSOC_RES 0x01

Definition at line 60 of file its_common.h.

Referenced by its1_controller_handle_association(), its2_router_handle_association(), and wpan_data_received_callback().

### #define ITS_DEVICE_NONE 0xff

Definition at line 107 of file its_common.h.

Referenced by its1_controller_handle_association(), its2_forward_routed_packet(), its2_rebroadcast_net_discover_req(), its2_router_handle_association(), its2_transmit(), its_add_local_device(), its_add_net_device(), its_get_device_handle(), and wpan_data_received_callback().

**#define ITS_ENDPOINT_DATA 0x0c**

Definition at line 74 of file its_common.h.

**#define ITS_ENDPOINT_REQ 0x0a**

Definition at line 69 of file its_common.h.

**#define ITS_ENDPOINT_RES 0x0b**

Definition at line 72 of file its_common.h.

**#define ITS_GENERIC_DATA 0x09**

Definition at line 68 of file its_common.h.

Referenced by its1_controller_transmit(), its1_device_transmit(), its2_device_transmit(), and wpan_data_received_callback().

**#define ITS_LOCAL_DISCOVER_REQ 0x05**

Definition at line 64 of file its_common.h.

Referenced by its2_request_local_addr(), and its2_router_queue_packet().

**#define ITS_LOCAL_DISCOVER_RES 0x06**

Definition at line 65 of file its_common.h.

Referenced by its2_respond_local_addr().

**#define ITS_NET_DISCOVER_REQ 0x07**

Definition at line 66 of file its_common.h.

Referenced by its2_request_net_addr(), and its2_router_queue_packet().

**#define ITS_NET_DISCOVER_RES 0x08**

Definition at line 67 of file its_common.h.

Referenced by its2_respond_net_addr().

**#define ITS_PENDING_DATA_REQ 0x04**

Definition at line 63 of file its_common.h.

**#define ITS_ROUTE_FAILURE 0x0d**

Definition at line 77 of file its_common.h.

---

## Typedef Documentation

### typedef uns8 its_device_handle

Definition of its_device_handle

Definition at line 105 of file its_common.h.

---

## Function Documentation

### its_device_handle its_add_local_device (uns16 *device_id*, uns16 *pan_id*, uns16 *short_address*)

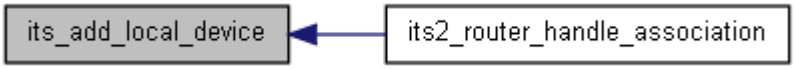
Definition at line 127 of file its_common.c.

References its_device_info::addr, debug_int_hex_16bit, debug_str, its_device_info::its_device_id, its_device_id, ITS_DEVICE_NONE, its_print_devices(), its_address::local, local_address::pan_id, local_address::short_address, and uns8.

Referenced by its2_router_handle_association().

Here is the call graph for this function:


Here is the caller graph for this function:




### its_device_handle its_add_net_device (uns16 *device_id*, uns16 *previous_hop*)

Definition at line 158 of file its_common.c.

References its_device_info::addr, debug_int_hex_16bit, debug_str, its_device_info::its_device_id, its_device_id, ITS_DEVICE_NONE, its_print_devices(), remote_address::prior_device_id, its_address::remote, remote_address::remote_indicator, and uns8.

Here is the call graph for this function:

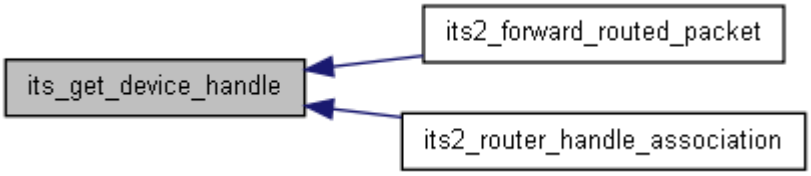

### its_device_handle its_get_device_handle (uns16 *device_id*)

Definition at line 108 of file its_common.c.

References its_device_id, ITS_DEVICE_NONE, and uns8.

Referenced by its2_forward_routed_packet(), and its2_router_handle_association().

Here is the caller graph for this function:



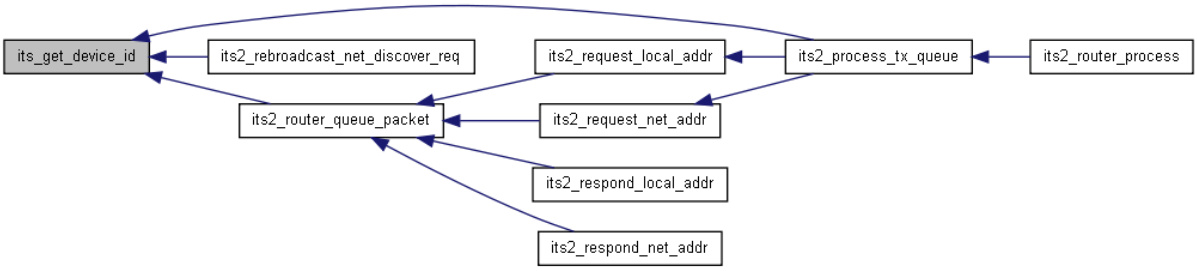

### uns16 its_get_device_id ()

Definition at line 89 of file its_common.c.

References its_device_id.

Referenced by its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), and its2_router_queue_packet().

Here is the caller graph for this function:

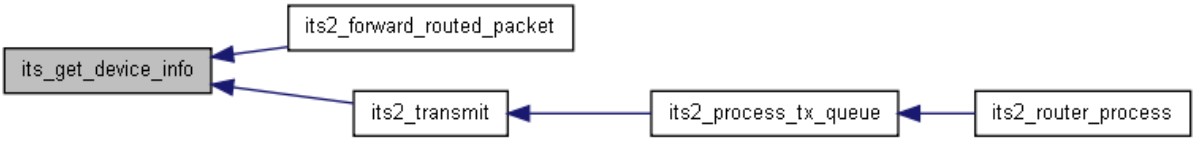



### its_device_info* its_get_device_info (uns8 *handle*)

Definition at line 100 of file its_common.c.

Referenced by its2_forward_routed_packet(), and its2_transmit().

Here is the caller graph for this function:



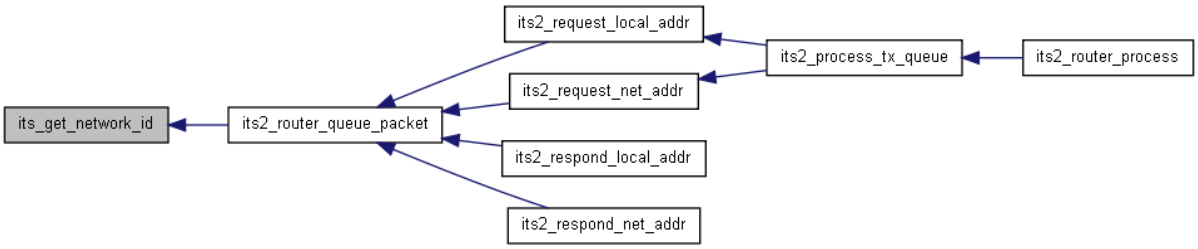

### uns16 its_get_network_id ()

Definition at line 81 of file its_common.c.

References its_network_id.

Referenced by its2_router_queue_packet().

Here is the caller graph for this function:



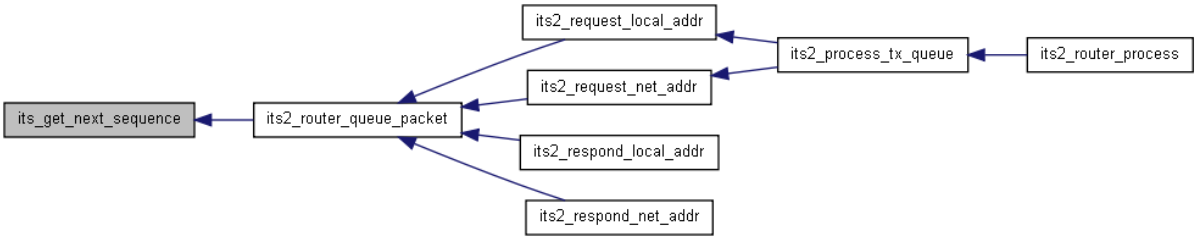

### uns8 its_get_next_sequence ()

Definition at line 72 of file its_common.c.

References its_sequence.

Referenced by its2_router_queue_packet().

Here is the caller graph for this function:




### void its_init ()

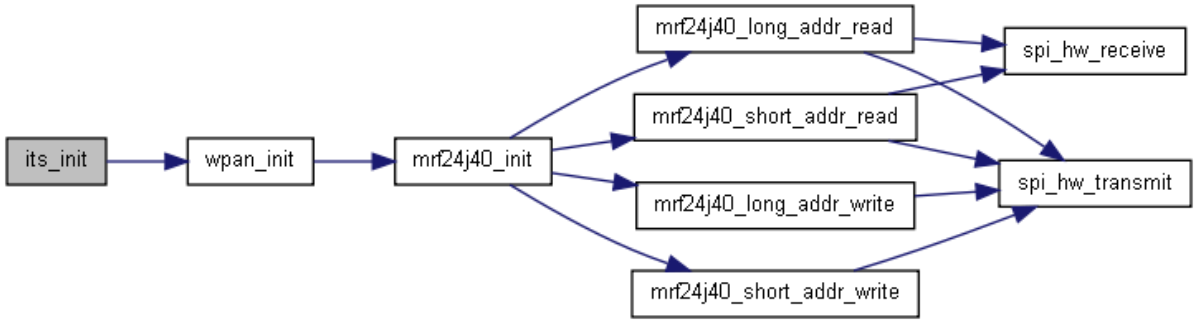
Call before using any ITS functionality

Definition at line 93 of file its_common.c.

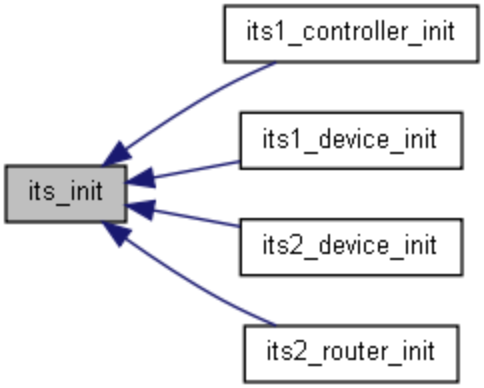
References its_device_info::its_device_id, uns8, and wpan_init().

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the call graph for this function:




Here is the caller graph for this function:



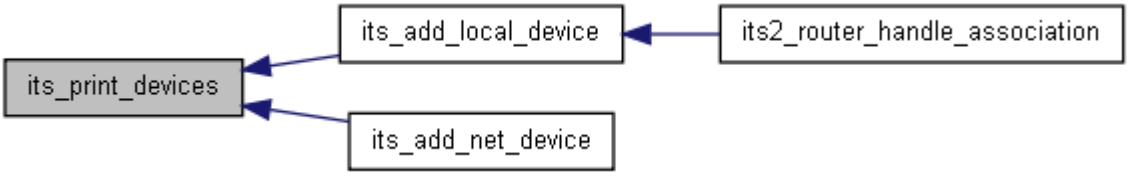

**void its_print_devices ()**

Definition at line 51 of file its_common.c.

References debug_int_hex_16bit, debug_nl, debug_str, its_device_id, and uns8.

Referenced by its_add_local_device(), and its_add_net_device().

Here is the caller graph for this function:




**void its_set_device_id (uns16 *device_id*)**

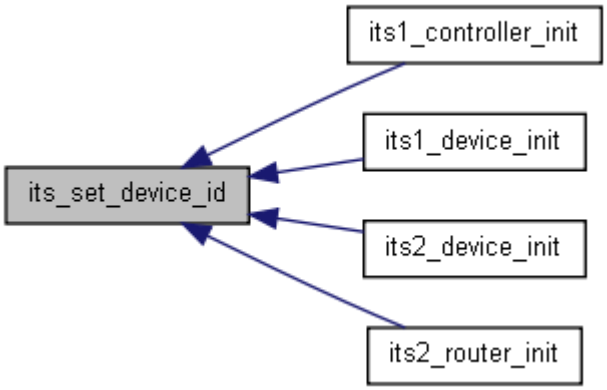
Set the ITS device ID before sending packets. See also: its_set_network_id(uns16 network_id);

Definition at line 85 of file its_common.c.

References its_device_id.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the caller graph for this function:



**void its_set_network_id (uns16 *network_id*)**

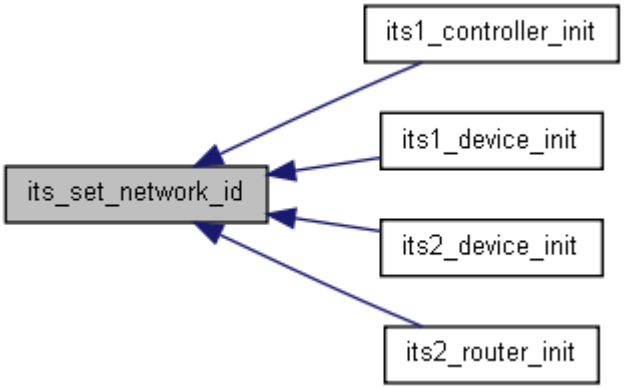
Set the ITS network ID before sending packets See also: its_set_device_id(uns16 device_id);

Definition at line 76 of file its_common.c.

References its_network_id.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the caller graph for this function:



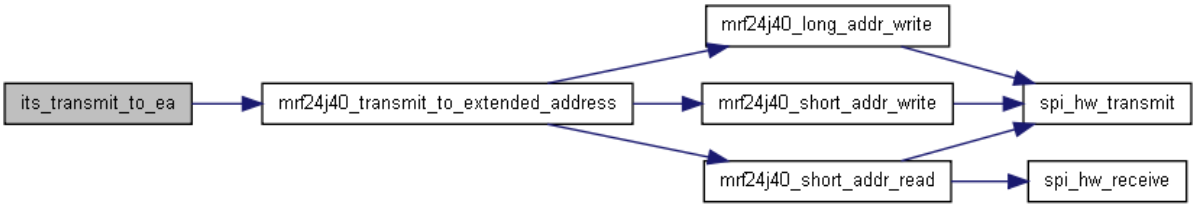

**void its_transmit_to_ea (uns8 * *dest_ea*, uns16 *dest_its_device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 232 of file its_common.c.

References FRAME_TYPE_DATA, its_device_id, its_network_id, its_sequence, mrf24j40_transmit_to_extended_address(), MRF_NO_ACK, and uns8.

Referenced by its1_device_process().

Here is the call graph for this function:




Here is the caller graph for this function:

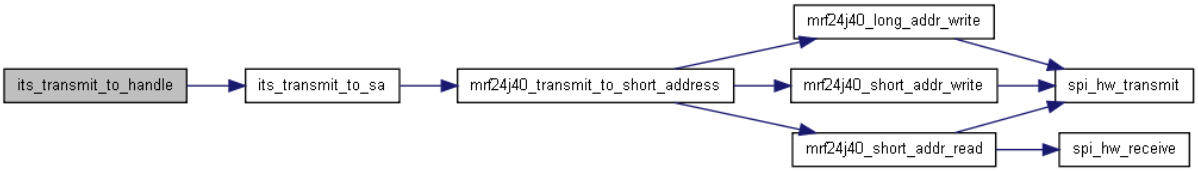

**void its_transmit_to_handle (its_device_handle *handle*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 185 of file its_common.c.

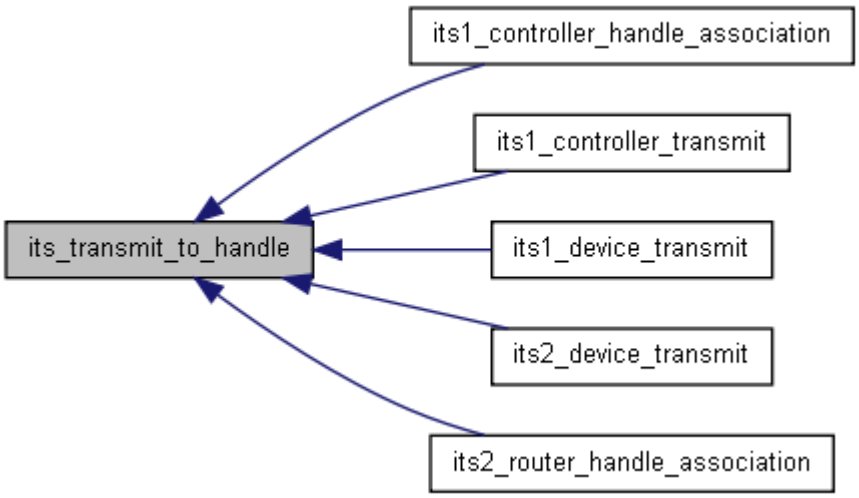
References its_device_info::addr, its_device_info::its_device_id, its_transmit_to_sa(), its_address::local, and local_address::short_address.

Referenced by its1_controller_handle_association(), its1_controller_transmit(), its1_device_transmit(), its2_device_transmit(), and its2_router_handle_association().

Here is the call graph for this function:




Here is the caller graph for this function:



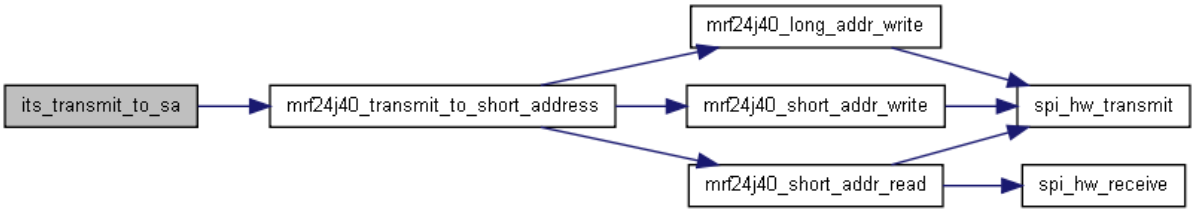

**void its_transmit_to_sa (uns16 *dest_pan_id*, uns16 *dest_sa*, uns16 *dest_device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*)**

Definition at line 195 of file its_common.c.

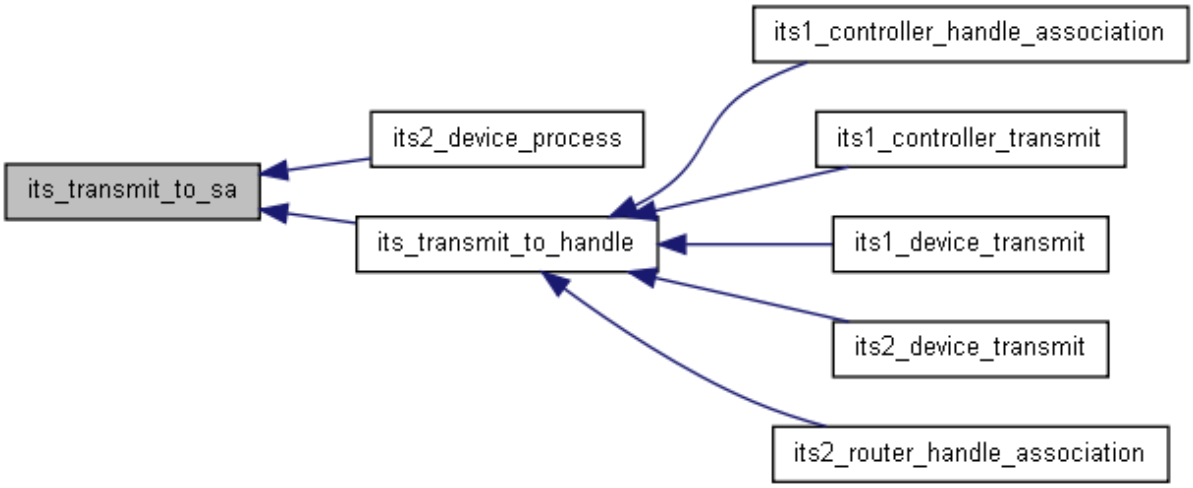
References FRAME_TYPE_DATA, its_device_id, its_network_id, its_sequence, mrf24j40_transmit_to_short_address(), MRF_ACK, uns16, and uns8.

Referenced by its2_device_process(), and its_transmit_to_handle().

Here is the call graph for this function:




Here is the caller graph for this function:

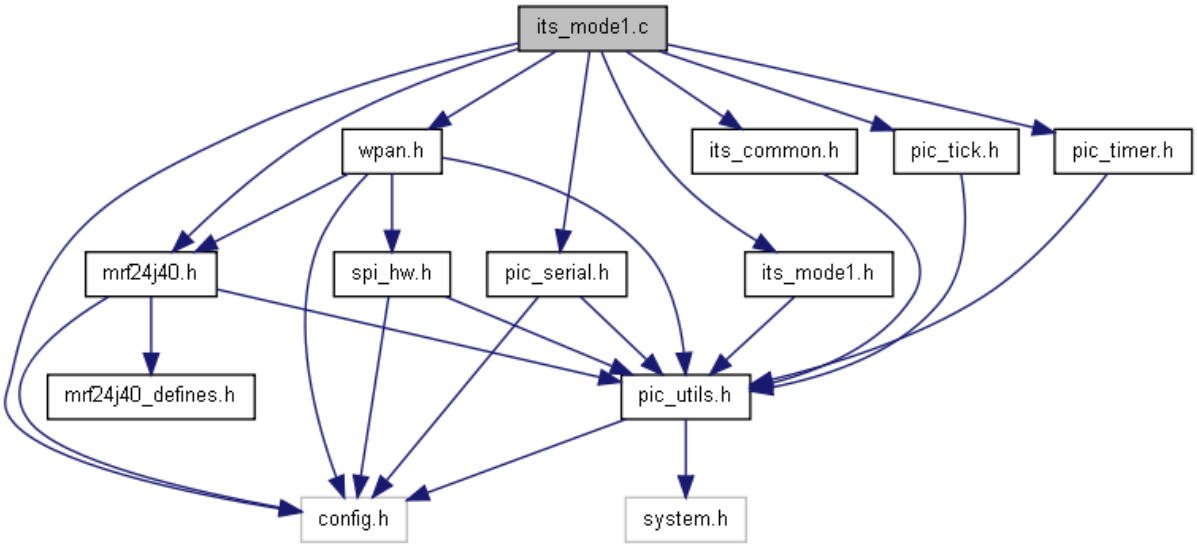



---

## its_mode1.c File Reference

```
#include "its_mode1.h"
#include "its_common.h"
#include "wpan.h"
#include "mrf24j40.h"
#include "config.h"
#include "pic_serial.h"
#include "pic_tick.h"
#include "pic_timer.h"
```

Include dependency graph for its_mode1.c:



## Functions

- its1_result its1_controller_handle_association (uns16 pan_id, uns16 its_device_id)
- uns8 its1_controller_init (uns16 my_device_id, uns16 network_id)
- void its1_controller_process ()
- void its1_controller_receive_callback (uns16 device_id, uns8 *data, uns8 data_length)
- its1_result its1_controller_transmit (uns16 device_id, uns8 *data, uns8 data_length)
- its1_result its1_device_init (uns16 my_device_id, uns16 network_id)
- void its1_device_process ()
- its1_result its1_device_transmit (uns8 *data, uns8 data_length)
- its1_result its1_find_controller ()
- void its1_setup_io ()
- void wpan_data_received_callback (wpan_address *addr, uns8 *data, uns8 data_size)

## Variables

- uns8 channel
- uns8 controller_handle
- its1_state state = STATE_STARTUP
- uns16 tick_marker

---

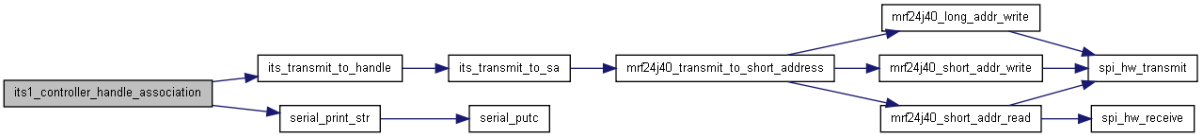
## Function Documentation

### its1_result its1_controller_handle_association (uns16 *pan_id*, uns16 *its_device_id*)

Definition at line 62 of file its_mode1.c.

References ITS_ASSOC_RES, ITS_DEVICE_NONE, its_transmit_to_handle(), RESULT_FAILED, RESULT_SUCCESSFUL, and serial_print_str().

Here is the call graph for this function:

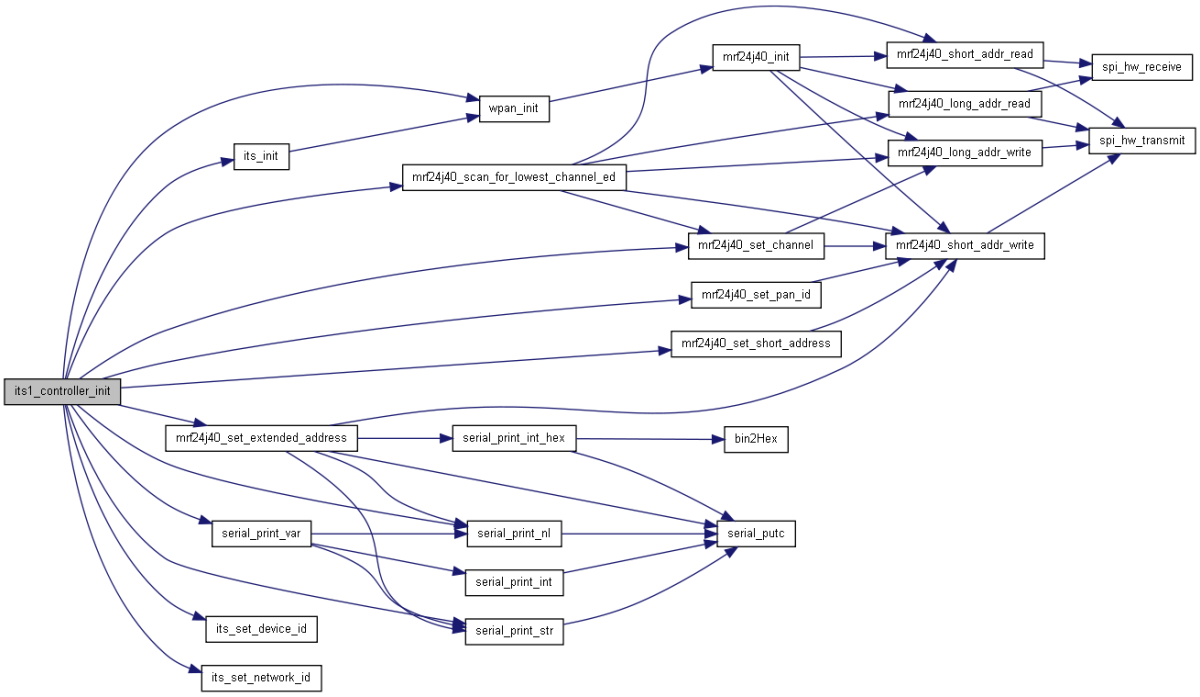



### uns8 its1_controller_init (uns16 *my_device_id*, uns16 *network_id*)

Definition at line 235 of file its_mode1.c.

References its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), serial_print_nl(), serial_print_str(), serial_print_var(), uns8, and wpan_init().

Here is the call graph for this function:




**void its1_controller_process ()**

Definition at line 279 of file its_mode1.c.

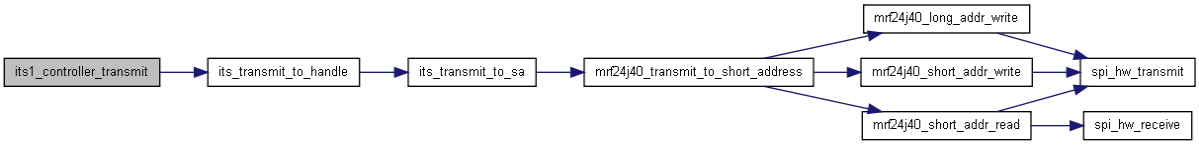
**void its1_controller_receive_callback (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

**its1_result its1_controller_transmit (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

Definition at line 265 of file its_mode1.c.

References ITS_GENERIC_DATA, and its_transmit_to_handle().

Here is the call graph for this function:

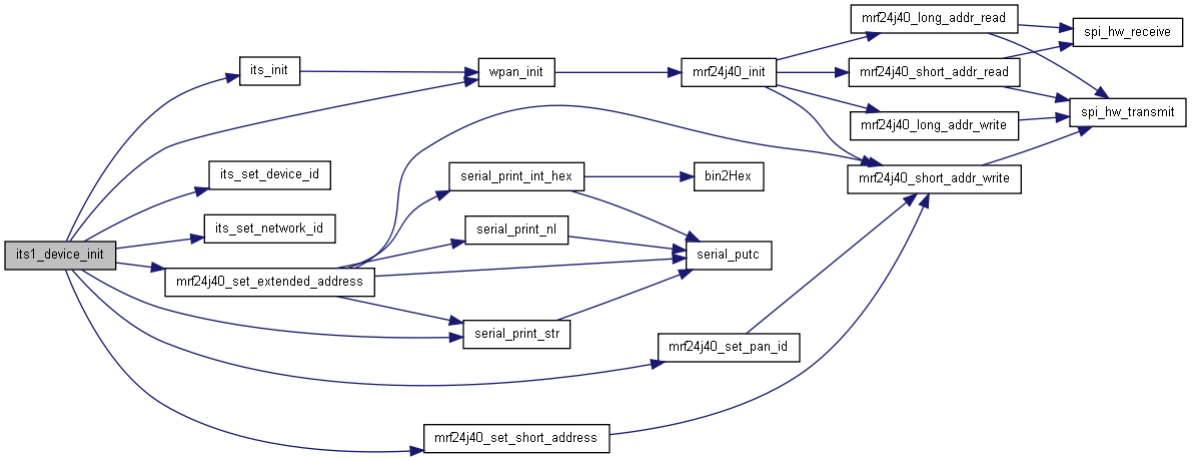



**its1_result its1_device_init (uns16 *my_device_id*, uns16 *network_id*)**

Definition at line 289 of file its_mode1.c.

References its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), serial_print_str(), state, STATE_UNASSOCIATED, uns8, and wpan_init().

Here is the call graph for this function:

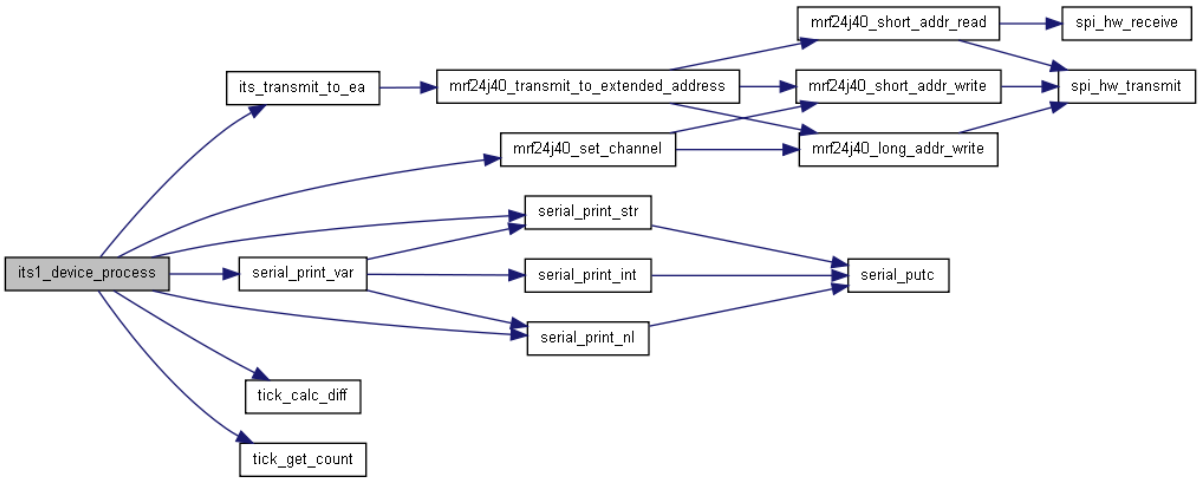



**void its1_device_process ()**

Definition at line 316 of file its_mode1.c.

References channel, ITS_ASSOC_REQ, its_transmit_to_ea(), mrf24j40_set_channel(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, serial_print_nl(), serial_print_str(), serial_print_var(), state, STATE_SEARCHING, STATE_UNASSOCIATED, tick_calc_diff(), tick_get_count(), tick_marker, uns16, and uns8.

Here is the call graph for this function:

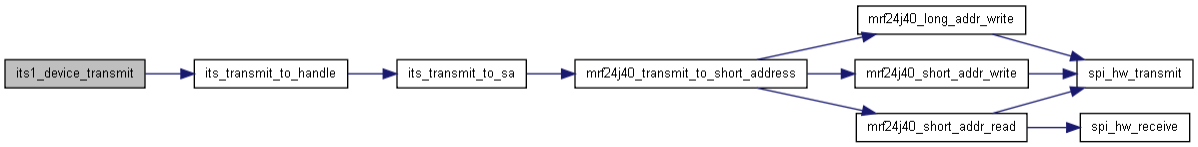



**its1_result its1_device_transmit (uns8 * *data*, uns8 *data_length*)**

Definition at line 352 of file its_mode1.c.

References controller_handle, ITS_GENERIC_DATA, and its_transmit_to_handle().

Here is the call graph for this function:



### its1_result its1_find_controller ()

Definition at line 310 of file its_mode1.c.

References channel, state, STATE_SEARCHING, tick_get_count(), and tick_marker.

Here is the call graph for this function:


### void its1_setup_io ()

Definition at line 57 of file its_mode1.c.

References wpan_setup_io().

Here is the call graph for this function:

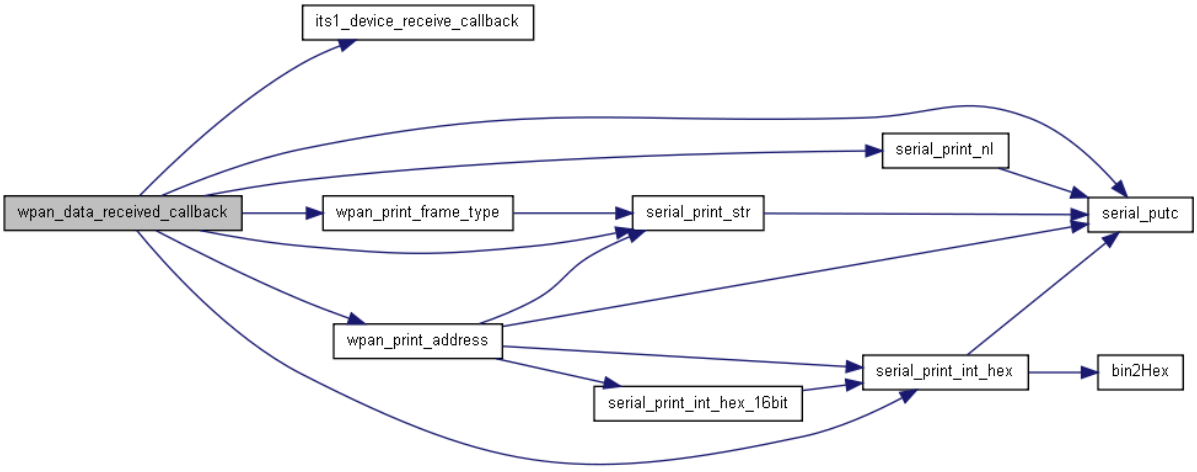

### void wpan_data_received_callback (wpan_address * *addr*, uns8 * *data*, uns8 *data_size*)

Definition at line 155 of file its_mode1.c.

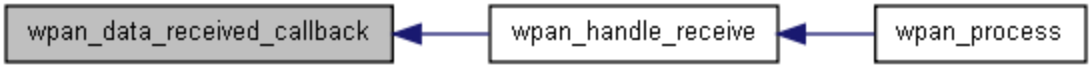
References controller_handle, its1_device_receive_callback(), ITS_ASSOC_RES, ITS_DEVICE_NONE, ITS_GENERIC_DATA, serial_print_int_hex(), serial_print_nl(), serial_print_str(), serial_putc(), wpan_address::source_pan_id, state, STATE_ASSOCIATED, STATE_SEARCHING, STATE_UNASSOCIATED, uns16, uns8, wpan_print_address(), and wpan_print_frame_type().

Referenced by wpan_handle_receive().

Here is the call graph for this function:




Here is the caller graph for this function:




---

### Variable Documentation

#### uns8 channel

Definition at line 51 of file its_mode1.c.

Referenced by ar1000_seek_more(), its1_device_process(), its1_find_controller(), its2_device_process(), its2_find_controller(), mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

#### uns8 controller_handle

Definition at line 49 of file its_mode1.c.

Referenced by its1_device_transmit(), its2_device_transmit(), and wpan_data_received_callback().

#### its1_state state = STATE_STARTUP

Definition at line 47 of file its_mode1.c.

Referenced by its1_device_init(), its1_device_process(), its1_find_controller(), its2_device_init(), its2_device_process(), its2_find_controller(), and wpan_data_received_callback().

#### uns16 tick_marker

Definition at line 50 of file its_mode1.c.

Referenced by its1_device_process(), its1_find_controller(), its2_device_process(), its2_find_controller(), and its2_router_process().

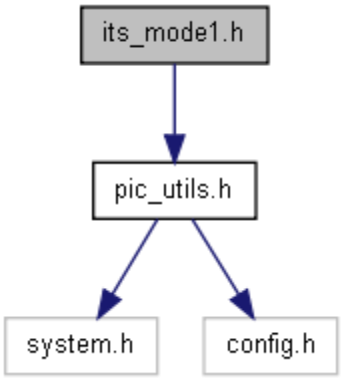
---

## its_mode1.h File Reference

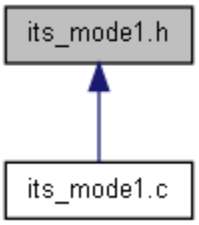
ITS networking mode 1.

```
#include "pic_utils.h"
```

Include dependency graph for its_mode1.h:




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




## Typedefs

- typedef enum _its1_result its1_result
- typedef enum _its1_state its1_state

## Enumerations

- enum _its1_result { RESULT_SUCCESSFUL, RESULT_FAILED }
- enum _its1_state { STATE_STARTUP, STATE_RUNNING, STATE_SEARCHING, STATE_ASSOCIATED, STATE_UNASSOCIATED }

## Functions

- its1_result its1_controller_handle_association (uns16 pan_id, uns16 its_device_id)
- uns8 its1_controller_init (uns16 my_device_id, uns16 network_id)
- void its1_controller_process ()
- void its1_controller_receive_callback (uns16 device_id, uns8 *data, uns8 data_length)
- its1_result its1_controller_transmit (uns16 device_id, uns8 *data, uns8 data_length)
- its1_result its1_device_init (uns16 my_device_id, uns16 network_id)
- void its1_device_process ()
- void its1_device_receive_callback (uns8 *data, uns8 data_length)
- its1_result its1_device_transmit (uns8 *data, uns8 data_length)
- its1_result its1_find_controller ()
- void its1_setup_io ()

## Variables

- its1_state state

## Detailed Description

Definition in file its_mode1.h.

---

## Typedef Documentation

### typedef enum _its1_result its1_result

Definition at line 59 of file its_mode1.h.

### typedef enum _its1_state its1_state

Definition at line 63 of file its_mode1.h.

---

## Enumeration Type Documentation

### enum _its1_result

**Enumerator:**

***RESULT_SUCCESSFUL***
***RESULT_FAILED***

Definition at line 58 of file its_mode1.h.

### enum _its1_state

**Enumerator:**

***STATE_STARTUP***
***STATE_RUNNING***
***STATE_SEARCHING***
***STATE_ASSOCIATED***
***STATE_UNASSOCIATED***

Definition at line 61 of file its_mode1.h.

---

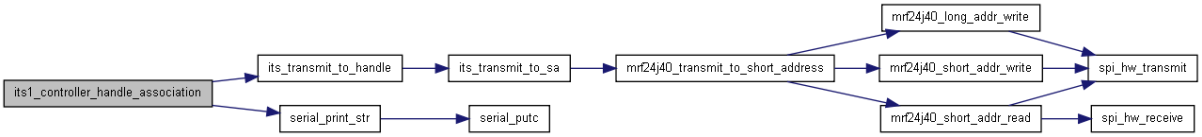
## Function Documentation

### its1_result its1_controller_handle_association (uns16 *pan_id*, uns16 *its_device_id*)

Definition at line 62 of file its_mode1.c.

References ITS_ASSOC_RES, ITS_DEVICE_NONE, its_transmit_to_handle(), RESULT_FAILED, RESULT_SUCCESSFUL, and serial_print_str().

Here is the call graph for this function:

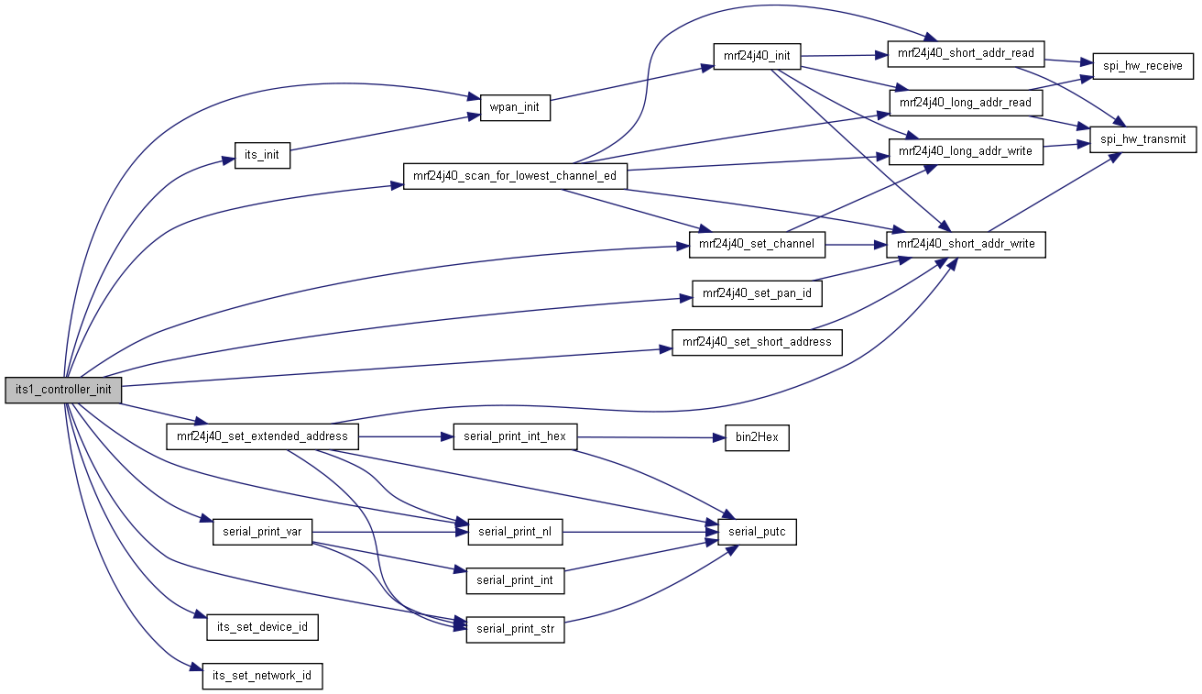



**uns8 its1_controller_init (uns16 *my_device_id*, uns16 *network_id*)**

Definition at line 235 of file its_mode1.c.

References its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), serial_print_nl(), serial_print_str(), serial_print_var(), uns8, and wpan_init().

Here is the call graph for this function:




**void its1_controller_process ()**

Definition at line 279 of file its_mode1.c.

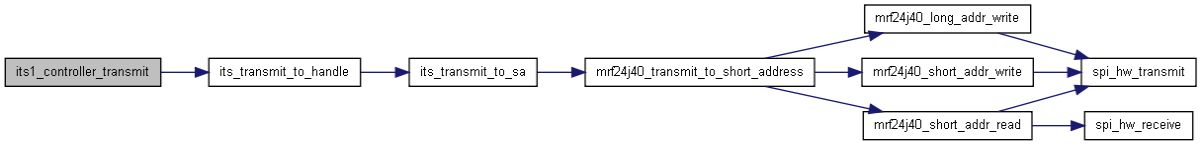
**void its1_controller_receive_callback (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

**its1_result its1_controller_transmit (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

Definition at line 265 of file its_mode1.c.

References ITS_GENERIC_DATA, and its_transmit_to_handle().

Here is the call graph for this function:

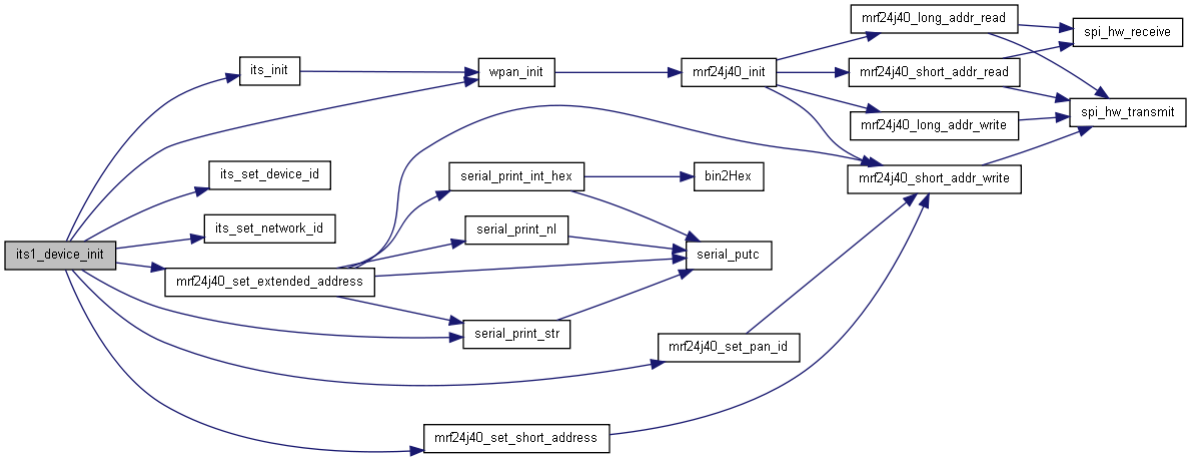



### its1_result its1_device_init (uns16 *my_device_id*, uns16 *network_id*)

Definition at line 289 of file its_mode1.c.

References its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), serial_print_str(), state, STATE_UNASSOCIATED, uns8, and wpan_init().

Here is the call graph for this function:

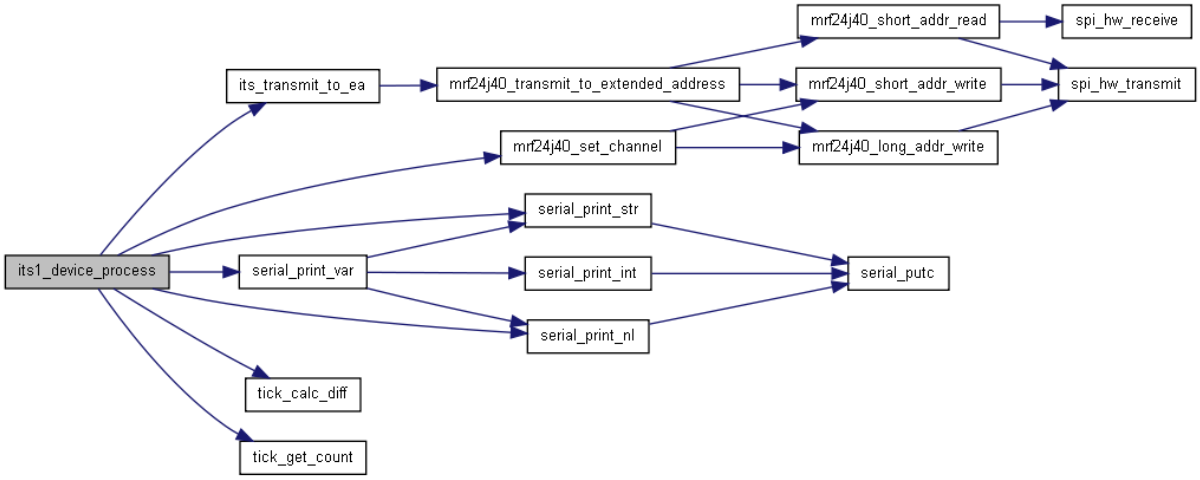



### void its1_device_process ()

Definition at line 316 of file its_mode1.c.

References channel, ITS_ASSOC_REQ, its_transmit_to_ea(), mrf24j40_set_channel(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, serial_print_nl(), serial_print_str(), serial_print_var(), state, STATE_SEARCHING, STATE_UNASSOCIATED, tick_calc_diff(), tick_get_count(), tick_marker, uns16, and uns8.

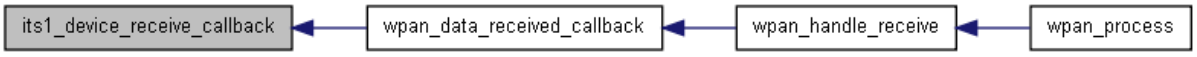
Here is the call graph for this function:




**void its1_device_receive_callback (uns8 * *data*, uns8 *data_length*)**

Referenced by wpan_data_received_callback().

Here is the caller graph for this function:

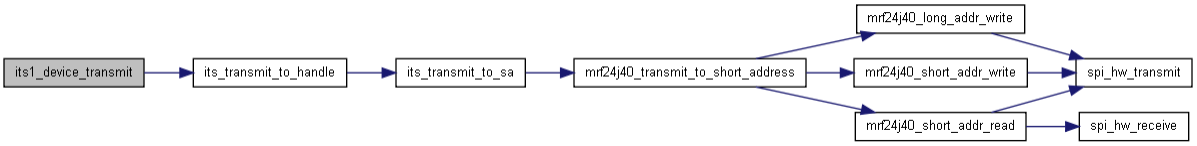



**its1_result it1_device_transmit (uns8 * *data*, uns8 *data_length*)**

Definition at line 352 of file its_mode1.c.

References controller_handle, ITS_GENERIC_DATA, and its_transmit_to_handle().

Here is the call graph for this function:




**its1_result it1_find_controller ()**

Definition at line 310 of file its_mode1.c.

References channel, state, STATE_SEARCHING, tick_get_count(), and tick_marker.

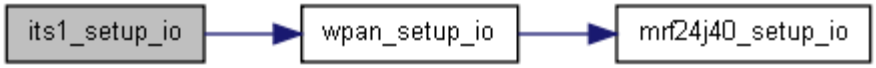
Here is the call graph for this function:


**void its1_setup_io ()**

Definition at line 57 of file its_mode1.c.

References wpan_setup_io().

Here is the call graph for this function:




## Variable Documentation

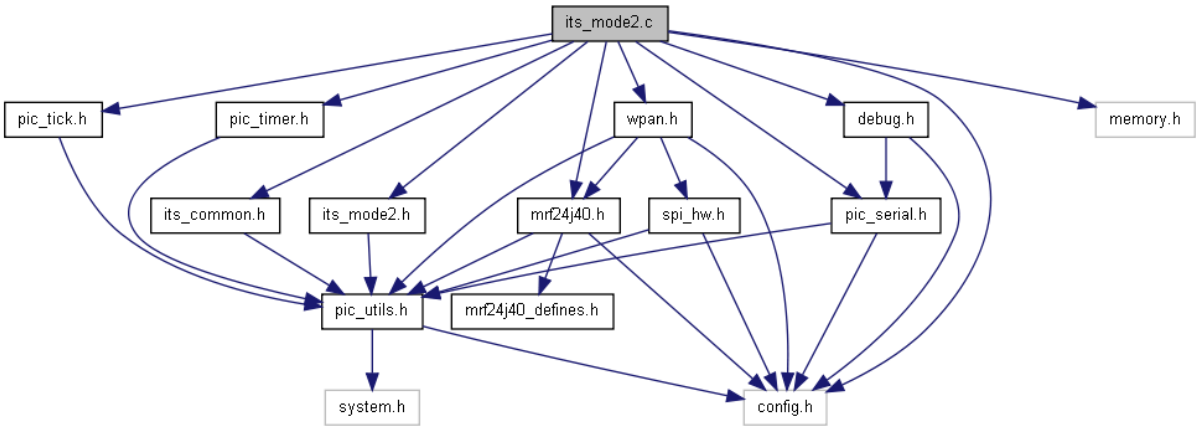
### its1_state state

Definition at line 47 of file its_mode1.c.

# its_mode2.c File Reference

```
#include "its_mode2.h"
#include "its_common.h"
#include "wpan.h"
#include "mrf24j40.h"
#include "config.h"
#include "pic_serial.h"
#include "pic_tick.h"
#include "pic_timer.h"
#include "memory.h"
#include "debug.h"
```

Include dependency graph for its_mode2.c:




## Functions

- void its2_delete_item_from_queue (queued_item *item)
- its2_result its2_device_init (uns16 my_device_id, uns16 network_id)
- void its2_device_process ()

- its2_result its2_device_transmit (uns8 *data, uns8 data_length)
- its2_result its2_find_controller ()
- uns8 its2_find_free_queue_slot ()
- its2_result its2_forward_routed_packet (its2_packet *pkt, uns8 *data, uns8 data_length)
- void its2_print_packet (its2_packet *pkt)
- void its2_print_queue ()
- void its2_process_tx_queue ()
- void its2_rebroadcast_net_addr_req ()
- its2_result its2_rebroadcast_net_discover_req (its2_packet *pkt)
- void its2_request_local_addr (uns16 device_id)
- void its2_request_net_addr (uns16 device_id)
- void its2_respond_local_addr (uns16 device_id)
- void its2_respond_net_addr (uns16 device_id)
- its2_result its2_router_handle_association (uns16 pan_id, uns16 its_device_id)
- uns8 its2_router_init (uns16 my_device_id, uns16 network_id)
- void its2_router_process (queued_item *item)
- void its2_router_process ()
- its2_result its2_router_queue_packet (uns16 device_id, uns8 packet_type, uns8 *data, uns8 data_length, uns8 ack)
- void its2_setup_io ()
- void its2_transmit (queued_item *item)
- void turn_off_mrf_interrupts ()
- void turn_on_mrf_interrupts ()
- void wpan_data_received_callback (wpan_address *addr, uns8 *data, uns8 data_size)
- void wpan_data_transmitted_callback (uns8 status, uns8 retries, uns8 channel_busy)

### *Callback for data transmitted.* Variables

- uns8 channel
- uns8 controller_handle
- bit debug_module = 0
- uns8 its2_seen_index
- static seen_packet its2_seen_list [ITS2_SEEN_LIST_SIZE]
- bit its2_transmitting = 0
- static queued_item its2_tx_queue [ITS2_TX_QUEUE_SIZE]
- bit queue_processing = 0
- its2_state state = STATE_STARTUP
- uns16 state_timeout
- uns16 tick_marker

---

## Function Documentation

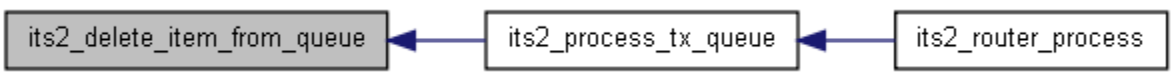
#### void its2_delete_item_from_queue (queued_item * *item*)

Definition at line 1025 of file its_mode2.c.

References queued_item::data, queued_item::data_length, queued_item::flag, and ITS2_FLAG_DELETED.

Referenced by its2_process_tx_queue().

Here is the caller graph for this function:

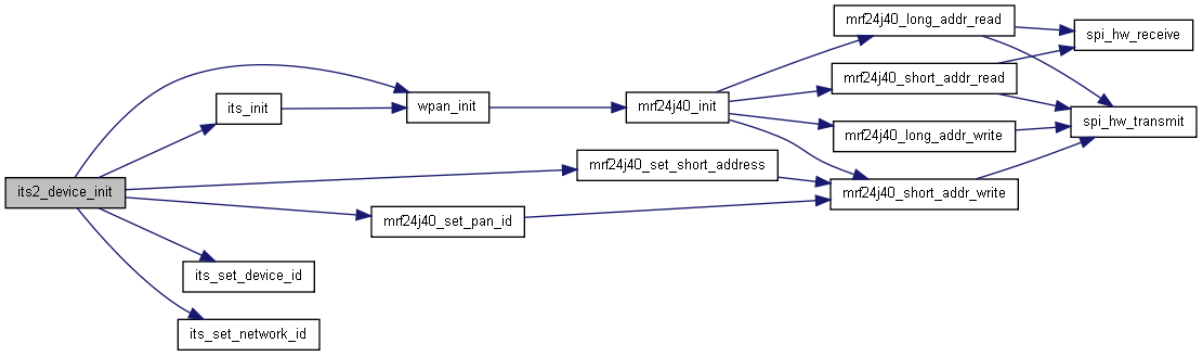


### its2_result its2_device_init (uns16 *my_device_id*, uns16 *network_id*)

Definition at line 1136 of file its_mode2.c.

References debug_str, its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), state, STATE_UNASSOCIATED, and wpan_init().

Here is the call graph for this function:

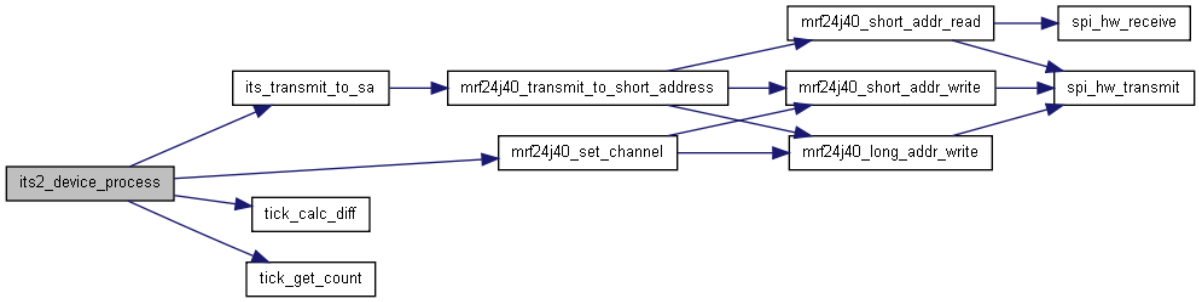



### void its2_device_process ()

Definition at line 1163 of file its_mode2.c.

References channel, debug_nl, debug_str, debug_var, ITS_ASSOC_REQ, its_transmit_to_sa(), mrf24j40_set_channel(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, state, STATE_SEARCHING, STATE_UNASSOCIATED, tick_calc_diff(), tick_get_count(), tick_marker, and uns16.

Here is the call graph for this function:

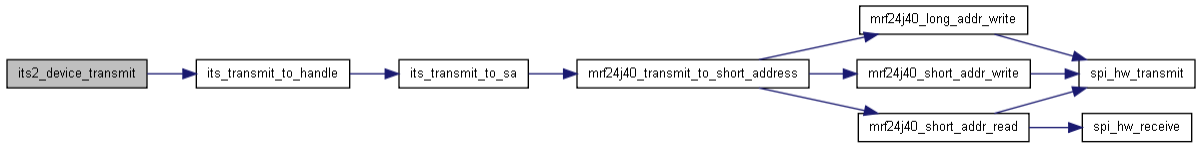



### its2_result its2_device_transmit (uns8 * *data*, uns8 *data_length*)

Definition at line 1199 of file its_mode2.c.

References controller_handle, ITS_GENERIC_DATA, and its_transmit_to_handle().

Here is the call graph for this function:



### its2_result its2_find_controller ()

Definition at line 1157 of file its_mode2.c.

References channel, state, STATE_SEARCHING, tick_get_count(), and tick_marker.

Here is the call graph for this function:

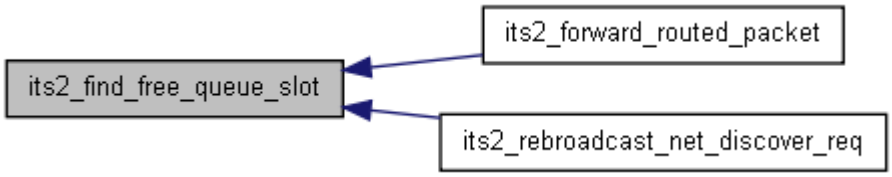

### uns8 its2_find_free_queue_slot ()

Definition at line 99 of file its_mode2.c.

References ITS2_FLAG_DELETED, and uns8.

Referenced by its2_forward_routed_packet(), and its2_rebroadcast_net_discover_req().

Here is the caller graph for this function:

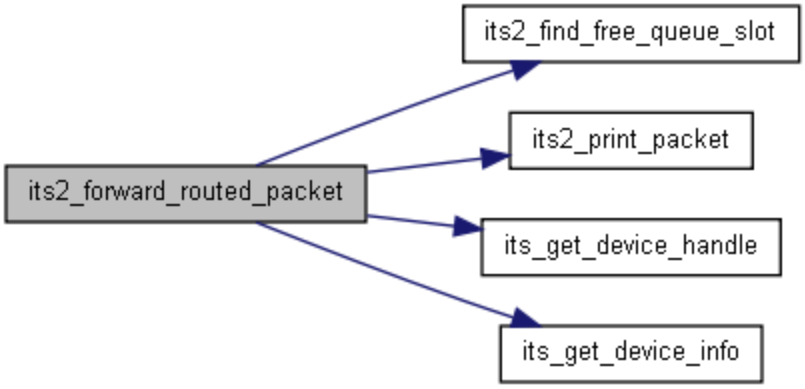



### its2_result its2_forward_routed_packet (its2_packet * *pkt*, uns8 * *data*, uns8 *data_length*)

Definition at line 653 of file its_mode2.c.

References its_device_info::addr, queued_item::data, queued_item::data_length, debug_int, debug_int_hex_16bit, debug_spc, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, its2_find_free_queue_slot(), ITS2_FLAG_DELETED, ITS2_FLAG_NO_ACK, ITS2_NO_AVAILABLE_SLOTS, its2_print_packet(), its2_packet::its_dest_id, ITS_DEVICE_NONE, its_get_device_handle(), its_get_device_info(), NEXT_HOP_NOT_LOCAL, its2_packet::num_routes, queued_item::packet, QS_READY_TO_SEND, QS_WAITING_ON_LOCAL_ADDR, QUEUE_FULL, its_address::remote, remote_address::remote_indicator, its2_packet::routers, queued_item::sent_count, queued_item::status, and uns8.

Here is the call graph for this function:



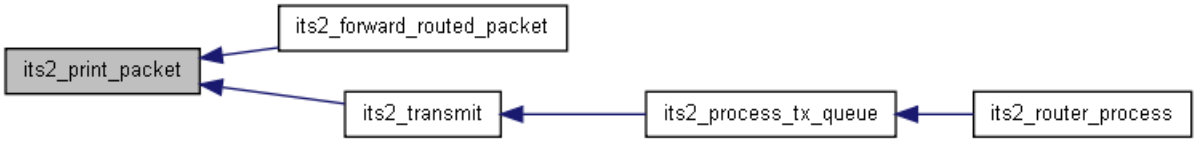

### void its2_print_packet (its2_packet * *pkt*)

Definition at line 580 of file its_mode2.c.

References debug_int, debug_int_hex_16bit, debug_putc, debug_str, its2_packet::hop_count, its2_packet::its_dest_id, its2_packet::its_network_id, its2_packet::its_source_id, its2_packet::max_hop_count, its2_packet::num_routes, its2_packet::packet_type, its2_packet::routers, its2_packet::sequence, and uns8.

Referenced by its2_forward_routed_packet(), and its2_transmit().

Here is the caller graph for this function:




### void its2_print_queue ()

Definition at line 1106 of file its_mode2.c.

References debug_nl, debug_var, queued_item::flag, ITS2_FLAG_DELETED, its2_packet::its_dest_id, its2_packet::its_source_id, queued_item::packet, queued_item::sent_count, queued_item::status, tick_get_count(), queued_item::tick_sent, and uns8.

Here is the call graph for this function:


### void its2_process_tx_queue ()

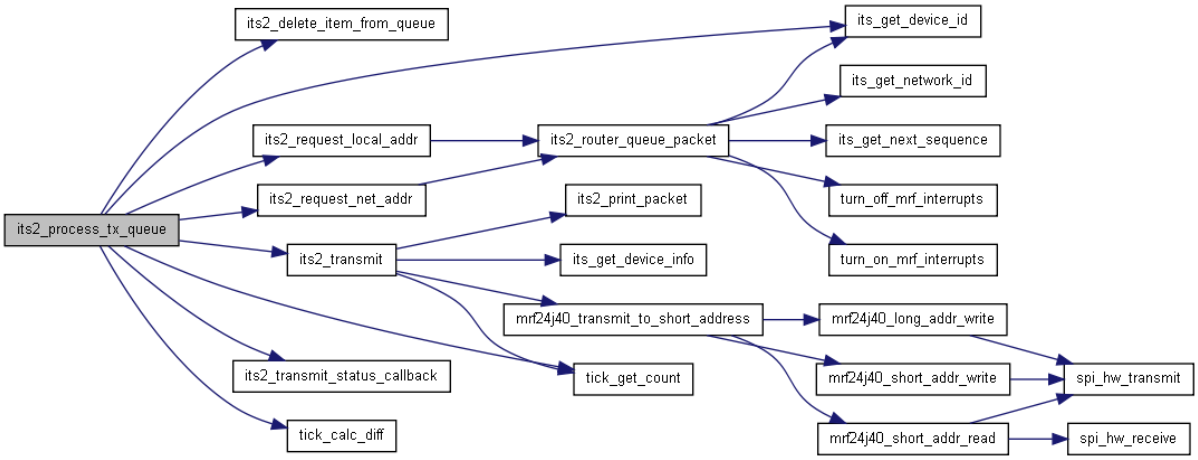
Definition at line 904 of file its_mode2.c.

References debug_int, debug_int_hex_16bit, debug_str, queued_item::dest_its_device_id, queued_item::flag, its2_delete_item_from_queue(), ITS2_FLAG_DELETED, its2_request_local_addr(), its2_request_net_addr(), its2_transmit(), its2_transmit_status_callback(), its2_transmitting, ITS2_TX_STATUS_NEXT_HOP_UNKNOWN, ITS2_TX_STATUS_NO_ACK, ITS2_TX_STATUS_NO_ROUTE, ITS2_TX_STATUS_SUCCESS, its_get_device_id(), its2_packet::its_source_id, queued_item::packet, QS_ACK_RECEIVED, QS_READY_TO_SEND,

QS_ROUTING_FAILED, QS_SENT, QS_WAITING_ON_ACK, QS_WAITING_ON_LOCAL_ADDR, QS_WAITING_ON_NETWORK_ADDR, queued_item::sent_count, queued_item::status, tick_calc_diff(), tick_get_count(), queued_item::tick_sent, uns16, and uns8.

Referenced by its2_router_process().

Here is the call graph for this function:




Here is the caller graph for this function:


### void its2_rebroadcast_net_addr_req ()

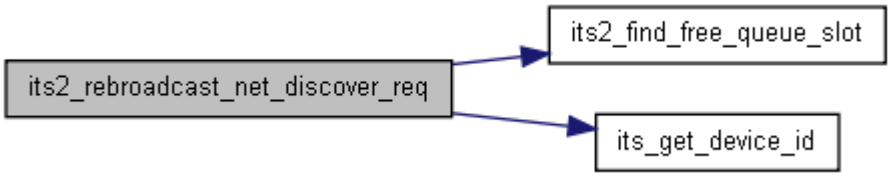
Definition at line 800 of file its_mode2.c.

### its2_result its2_rebroadcast_net_discover_req (its2_packet * *pkt*)

Definition at line 613 of file its_mode2.c.

References queued_item::data, queued_item::data_length, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, its2_find_free_queue_slot(), ITS2_FLAG_NO_ACK, ITS2_NO_AVAILABLE_SLOTS, ITS_DEVICE_NONE, its_get_device_id(), its2_packet::its_source_id, its2_packet::max_hop_count, its2_packet::num_routes, queued_item::packet, QS_READY_TO_SEND, QUEUE_FULL, its2_packet::routers, ROUTING_TOO_MANY_HOPS, queued_item::sent_count, its2_packet::sequence, queued_item::status, and uns8.

Here is the call graph for this function:



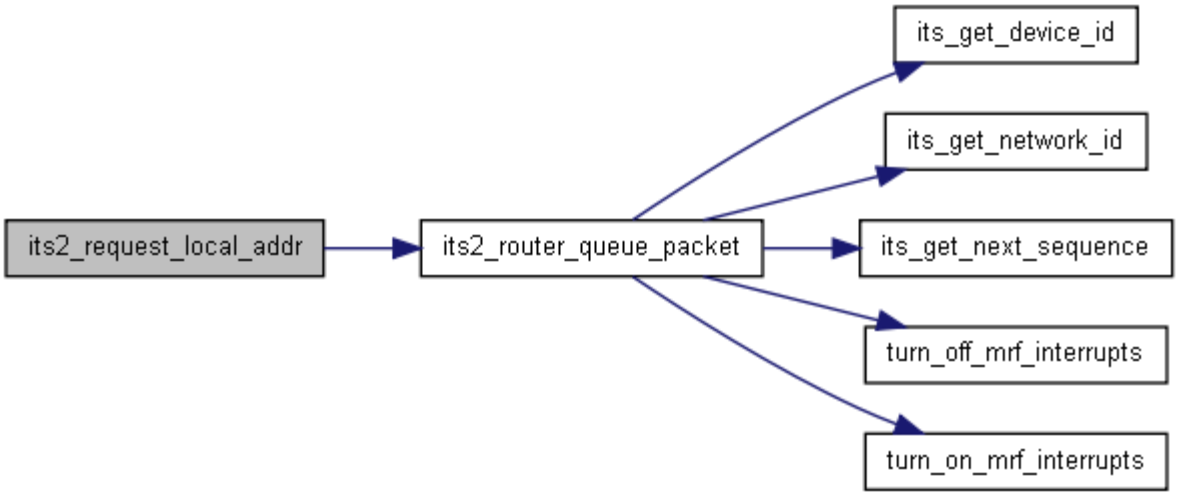
**void its2_request_local_addr (uns16 *device_id*)**

Definition at line 779 of file its_mode2.c.

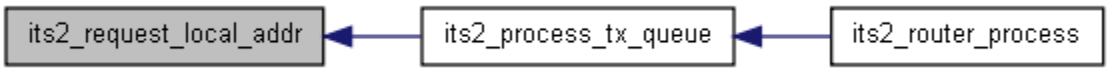
References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_LOCAL_DISCOVER_REQ.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:




Here is the caller graph for this function:



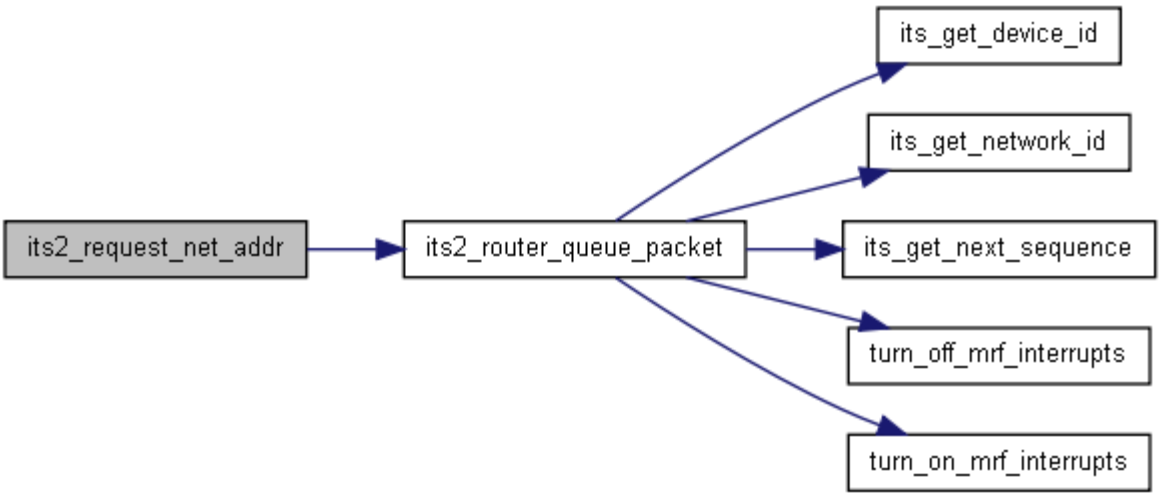

**void its2_request_net_addr (uns16 *device_id*)**

Definition at line 792 of file its_mode2.c.

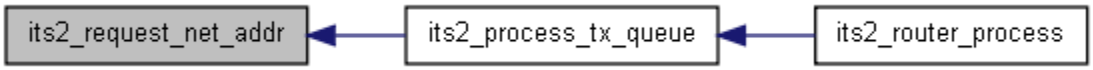
References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_NET_DISCOVER_REQ.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:



Here is the caller graph for this function:

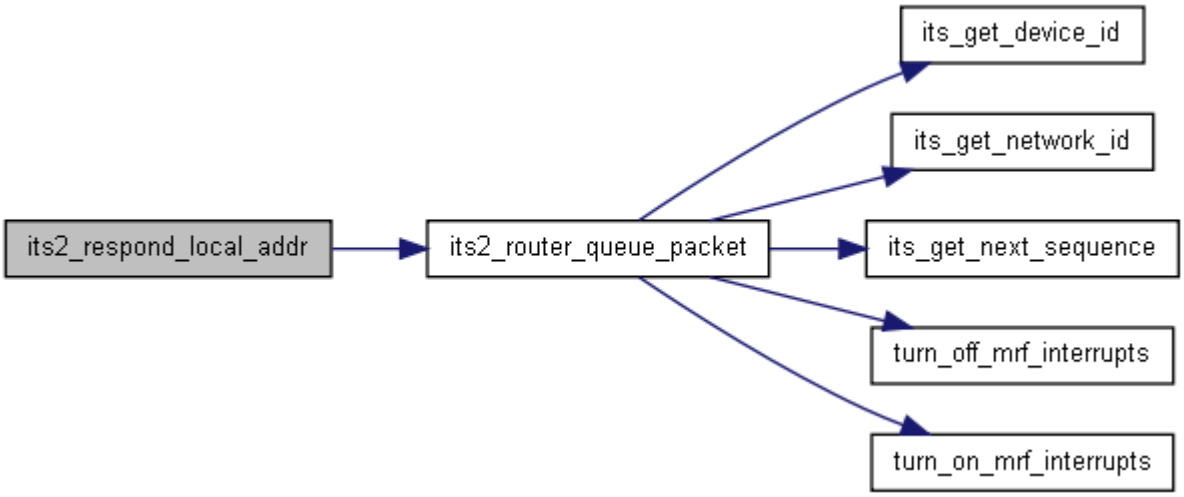



### void its2_respond_local_addr (uns16 *device_id*)

Definition at line 783 of file its_mode2.c.

References debug_module, debug_str, ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_LOCAL_DISCOVER_RES.

Here is the call graph for this function:

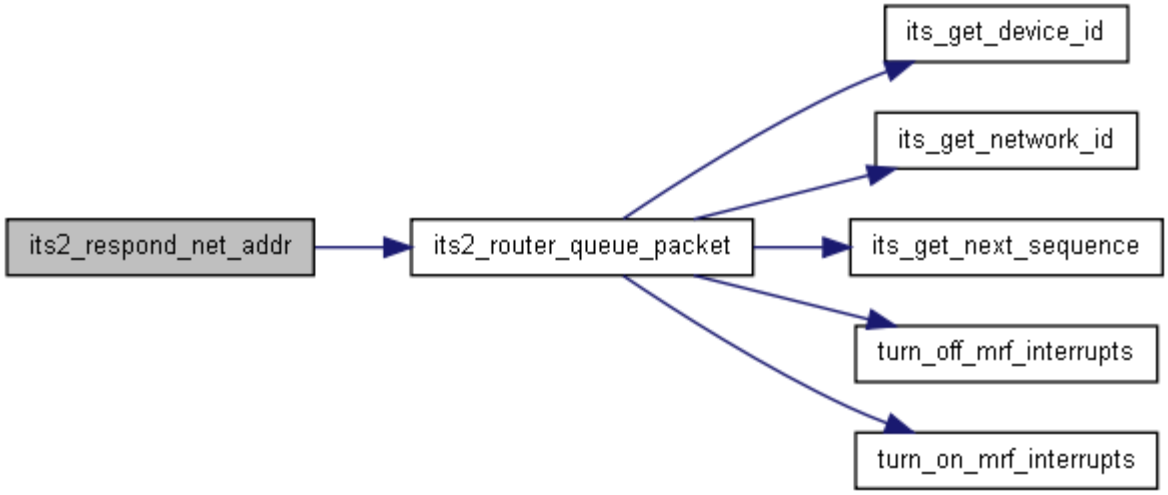



### void its2_respond_net_addr (uns16 *device_id*)

Definition at line 796 of file its_mode2.c.

References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_NET_DISCOVER_RES.

Here is the call graph for this function:

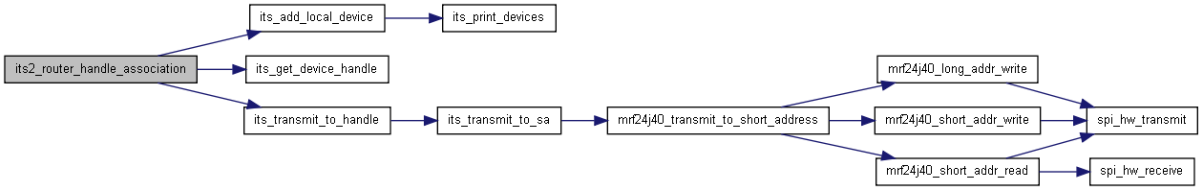



**its2_result its2_router_handle_association (uns16 *pan_id*, uns16 *its_device_id*)**

Definition at line 118 of file its_mode2.c.

References debug_str, its_add_local_device(), ITS_ASSOC_RES, ITS_DEVICE_NONE, its_get_device_handle(), its_transmit_to_handle(), RESULT_FAILED, and RESULT_SUCCESSFUL.

Here is the call graph for this function:

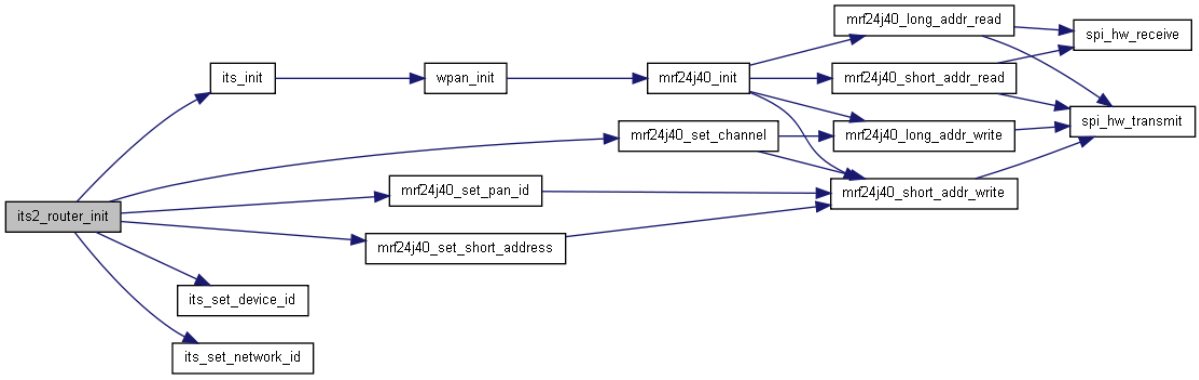



**uns8 its2_router_init (uns16 *my_device_id*, uns16 *network_id*)**

Definition at line 743 of file its_mode2.c.

References debug_nl, debug_str, debug_var, queued_item::flag, ITS2_FLAG_DELETED, its2_seen_index, its_init(), its_set_device_id(), its_set_network_id(), seen_packet::its_source_id, mrf24j40_set_channel(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), and uns8.

Here is the call graph for this function:



**void its2_router_process (queued_item * *item*)**

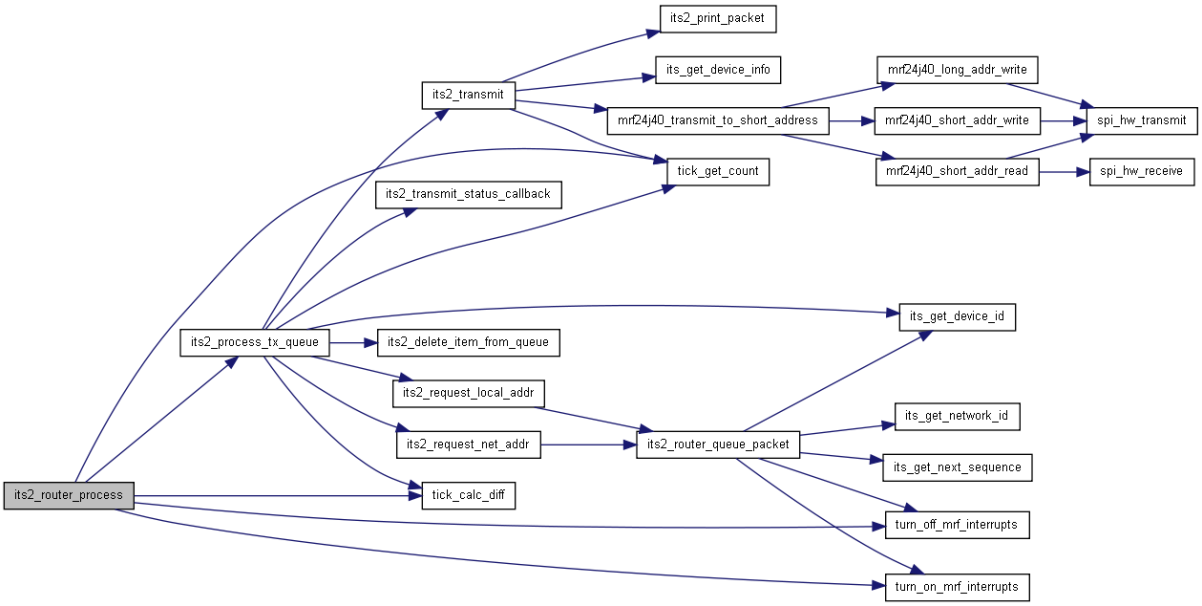
Definition at line 1102 of file its_mode2.c.

**void its2_router_process ()**

Definition at line 808 of file its_mode2.c.

References its2_process_tx_queue(), state_timeout, tick_calc_diff(), tick_get_count(), tick_marker, turn_off_mrf_interrupts(), turn_on_mrf_interrupts(), and uns16.

Here is the call graph for this function:



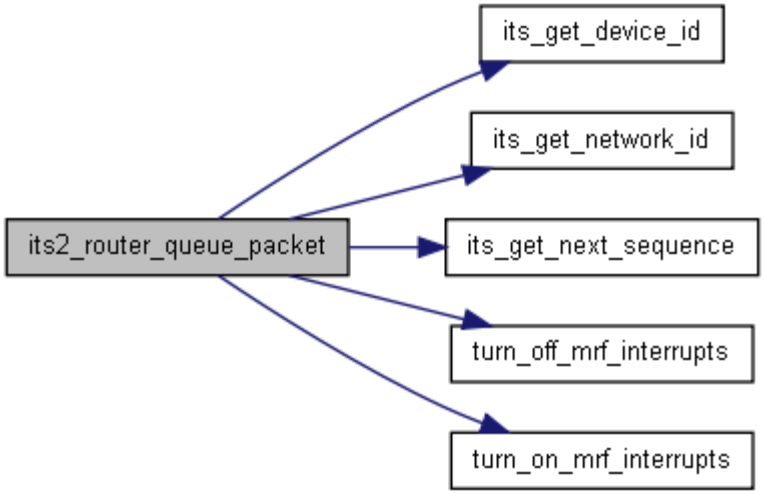

**its2_result its2_router_queue_packet (uns16 *device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*, uns8 *ack*)**

Definition at line 823 of file its_mode2.c.

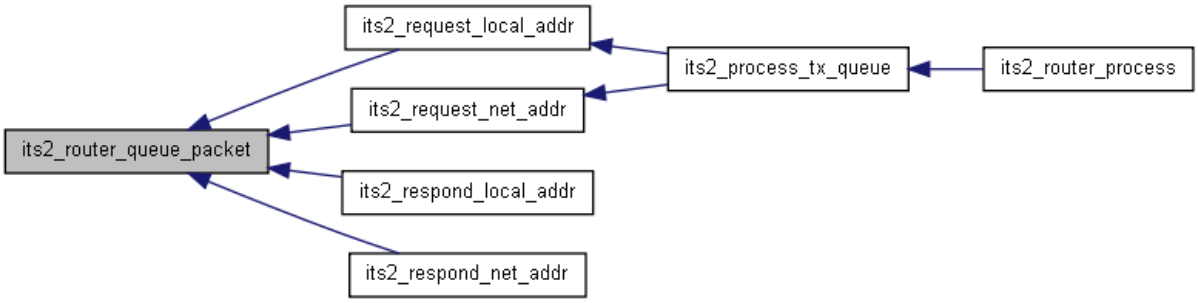
References queued_item::data, queued_item::data_length, debug_str, debug_var, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, ITS2_FLAG_DELETED, ITS2_UPDATE_ROUTE_FAIL, its2_packet::its_dest_id, its_get_device_id(), its_get_network_id(), its_get_next_sequence(), ITS_LOCAL_DISCOVER_REQ, ITS_NET_DISCOVER_REQ, its2_packet::its_network_id, its2_packet::its_source_id, its2_packet::max_hop_count, queued_item::packet, its2_packet::packet_type, QUEUE_FULL, ROUTING_TOO_MANY_HOPS, queued_item::sent_count, its2_packet::sequence, turn_off_mrf_interrupts(), turn_on_mrf_interrupts(), and uns8.

Referenced by its2_request_local_addr(), its2_request_net_addr(), its2_respond_local_addr(), and its2_respond_net_addr().

Here is the call graph for this function:




Here is the caller graph for this function:




### void its2_setup_io ()

Definition at line 95 of file its_mode2.c.

References wpan_setup_io().

Here is the call graph for this function:

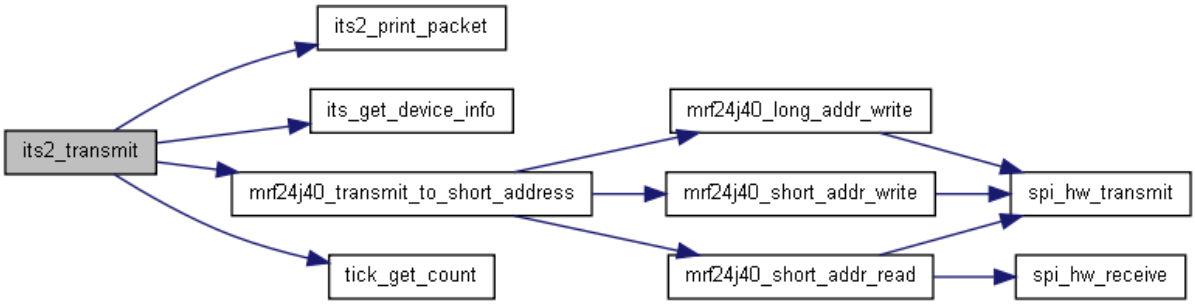

### void its2_transmit (queued_item * *item*)

Definition at line 1035 of file its_mode2.c.

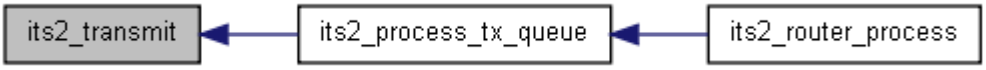
References its_device_info::addr, queued_item::data, queued_item::data_length, debug_int, debug_int_hex_16bit, debug_putc, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, FRAME_TYPE_DATA, ITS2_FLAG_ACK, its2_print_packet(), its2_transmitting, ITS_DEVICE_NONE, its_get_device_info(), its_address::local, mrf24j40_transmit_to_short_address(), MRF_ACK, MRF_NO_ACK, its2_packet::num_routes, queued_item::packet, local_address::pan_id, QS_SENT, QS_WAITING_ON_ACK, its2_packet::routers, queued_item::sent_count, local_address::short_address, queued_item::status, tick_get_count(), queued_item::tick_sent, and uns8.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:




Here is the caller graph for this function:



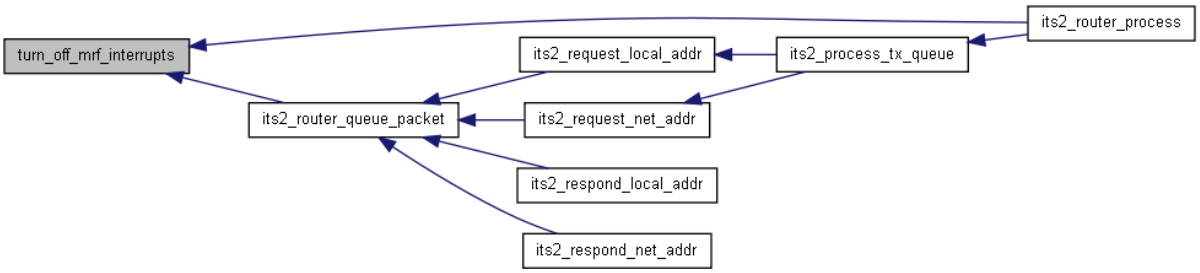

**void turn_off_mrf_interrupts ()**

!clear_bit(intcon, INT0IE);

Definition at line 82 of file its_mode2.c.

Referenced by its2_router_process(), and its2_router_queue_packet().

Here is the caller graph for this function:



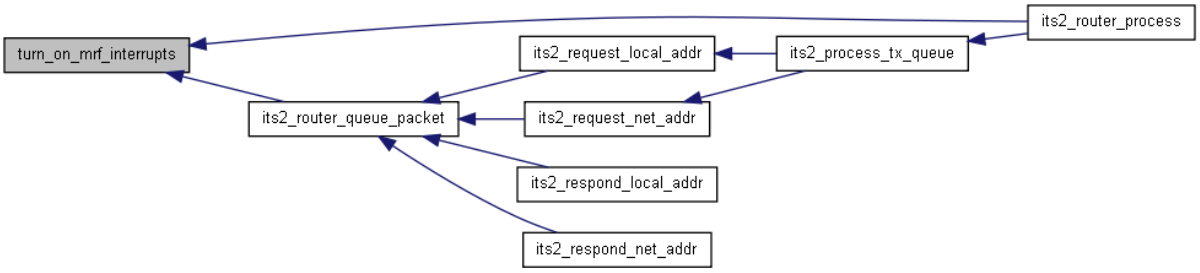

**void turn_on_mrf_interrupts ()**

!set_bit(intcon, INT0IE);

Definition at line 90 of file its_mode2.c.

Referenced by its2_router_process(), and its2_router_queue_packet().

Here is the caller graph for this function:

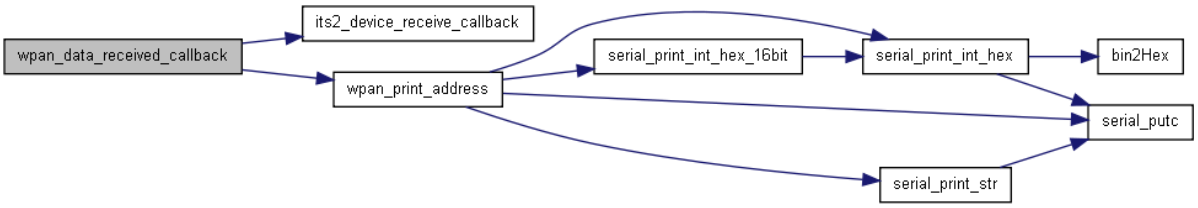



**void wpan_data_received_callback (wpan_address * *addr*, uns8 * *data*, uns8 *data_size*)**

Definition at line 508 of file its_mode2.c.

References controller_handle, debug_int_hex, debug_nl, debug_putc, debug_str, its2_device_receive_callback(), ITS_ASSOC_RES, ITS_DEVICE_NONE, ITS_GENERIC_DATA, wpan_address::source_pan_id, state, STATE_ASSOCIATED, STATE_SEARCHING, STATE_UNASSOCIATED, uns16, uns8, and wpan_print_address().

Here is the call graph for this function:




**void wpan_data_transmitted_callback (uns8 *status*, uns8 *retries*, uns8 *channel_busy*)**

Once the lower layers have attempted to transmit the packet, the results will be presented to the callback.

**Parameters:**

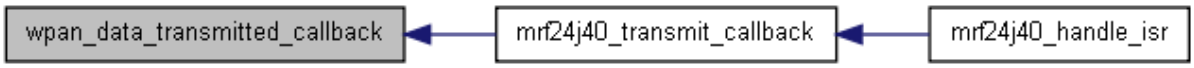
*status* Set to 0 for success or 1 for failure
*retries* Set to the number of retries
*channel_busy* Set to 1 if failure was due to the channel being busy.

Definition at line 804 of file its_mode2.c.

References its2_transmitting.

Referenced by mrf24j40_transmit_callback().

Here is the caller graph for this function:




---

## Variable Documentation

**uns8 channel**

Definition at line 71 of file its_mode2.c.

**uns8 controller_handle**

Definition at line 67 of file its_mode2.c.

**bit debug_module = 0**

Definition at line 64 of file its_mode2.c.
Referenced by its2_respond_local_addr().

**uns8 its2_seen_index**

Definition at line 73 of file its_mode2.c.
Referenced by its2_router_init().

**seen_packet its2_seen_list[ITS2_SEEN_LIST_SIZE]** `[static]`

Definition at line 75 of file its_mode2.c.

**bit its2_transmitting = 0**

Definition at line 72 of file its_mode2.c.
Referenced by its2_process_tx_queue(), its2_transmit(), and wpan_data_transmitted_callback().

**queued_item its2_tx_queue[ITS2_TX_QUEUE_SIZE]** `[static]`

Definition at line 76 of file its_mode2.c.

**bit queue_processing = 0**

Definition at line 65 of file its_mode2.c.

**its2_state state = STATE_STARTUP**

Definition at line 62 of file its_mode2.c.

**uns16 state_timeout**

Definition at line 69 of file its_mode2.c.
Referenced by its2_router_process().

**uns16 tick_marker**

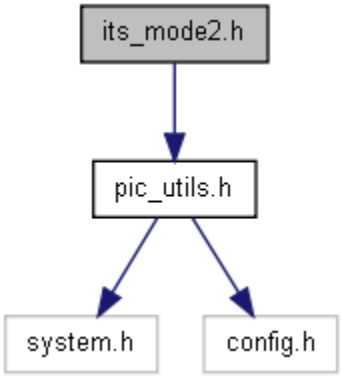
Definition at line 68 of file its_mode2.c.

# its_mode2.h File Reference

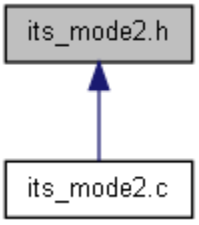
ITS Mesh networking mode 2.

```
#include "pic_utils.h"
```

Include dependency graph for its_mode2.h:




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




## Data Structures

- struct its2_packet
- struct queued_item
- struct seen_packet

## Defines

- #define ITS2_FLAG_ACK 0x01
- #define ITS2_FLAG_DELETED 0xff
- #define ITS2_FLAG_NO_ACK 0x02
- #define ITS2_NO_AVAILABLE_SLOTS 0xff
- #define ITS2_STATUS_QUEUED 0x00
- #define ITS2_STATUS_TX_QUEUE_FULL 0x01
- #define ITS2_TX_STATUS_NEXT_HOP_UNKNOWN 0x04
- #define ITS2_TX_STATUS_NO_ACK 0x02
- #define ITS2_TX_STATUS_NO_ROUTE 0x01
- #define ITS2_TX_STATUS_REMOTE_NO_ACK 0x03
- #define ITS2_TX_STATUS_SUCCESS 0x00
- #define ITS2_UPDATE_ROUTE_FAIL 0x01
- #define ITS2_UPDATE_ROUTE_SUCCESS 0x00

- #define POS_DEST_H 0x0a
- #define POS_DEST_L 0x09
- #define POS_HOP_COUNT 0x0d
- #define POS_KEY1 0x00
- *!! look at the adding in routes bit...* #define POS_KEY2 0x01
- #define POS_LENGTH_HEADER 0x02
- #define POS_MAX_HOP_COUNT 0x0b
- #define POS_NETWORK_H 0x06
- #define POS_NETWORK_L 0x05
- #define POS_NUM_ROUTES 0x0c
- #define POS_PKT_TYPE 0x03
- #define POS_ROUTE_START 0x0e
- #define POS_SEQUENCE 0x04
- #define POS_SOURCE_H 0x08
- #define POS_SOURCE_L 0x07
- #define QS_ACK_RECEIVED 0x04
- #define QS_READY_TO_SEND 0x02
- #define QS_ROUTING_FAILED 0x06
- #define QS_SENT 0x05
- #define QS_WAITING_ON_ACK 0x03
- #define QS_WAITING_ON_LOCAL_ADDR 0x00
- #define QS_WAITING_ON_NETWORK_ADDR 0x01
- #define REMOTE_DEVICE 0xffff

## Typedefs

- typedef enum _its2_result its2_result
- typedef enum _its2_state its2_state

## Enumerations

- enum _its2_result { RESULT_SUCCESSFUL, RESULT_FAILED, ITEM_QUEUED, QUEUE_FULL, ROUTING_TOO_MANY_HOPS, NEXT_HOP_NOT_LOCAL }
- enum _its2_state { STATE_STARTUP, STATE_RUNNING, STATE_SEARCHING, STATE_ASSOCIATED, STATE_UNASSOCIATED }

## Functions

- void its2_delete_item_from_queue (queued_item *item)
- its2_result its2_device_init (uns16 my_device_id, uns16 network_id)
- void its2_device_process ()
- void its2_device_receive_callback (uns8 *data, uns8 data_length)
- its2_result its2_device_transmit (uns16 device_id, uns8 *data, uns8 data_length)
- its2_result its2_find_coordinator ()
- its2_result its2_forward_routed_packet (its2_packet *pkt, uns8 *data, uns8 data_length)
- void its2_print_packet (its2_packet *pkt)
- void its2_print_queue ()
- void its2_process_tx_queue ()
- its2_result its2_rebroadcast_net_discover_req (its2_packet *pkt)
- void its2_request_local_addr (uns16 device_id)
- void its2_request_net_addr (uns16 device_id)
- void its2_respond_local_addr (uns16 device_id)
- void its2_respond_net_addr (uns16 device_id)
- its2_result its2_router_handle_association (uns16 pan_id, uns16 its_device_id)
- uns8 its2_router_init (uns16 my_device_id, uns16 network_id)

- void its2_router_process ()
- its2_result its2_router_queue_packet (uns16 device_id, uns8 packet_type, uns8 *data, uns8 data_length, uns8 ack)
- void its2_router_receive_callback (uns16 device_id, uns8 *data, uns8 data_length)
- void its2_setup_io ()
- void its2_transmit (queued_item *item)
- void its2_transmit_status_callback (its2_packet *pkt, uns8 status)

## Variables

- bit debug_module
- its2_state state

---

## Detailed Description

Definition in file its_mode2.h.

---

## Define Documentation

### #define ITS2_FLAG_ACK 0x01

Definition at line 112 of file its_mode2.h.

Referenced by its2_transmit().

### #define ITS2_FLAG_DELETED 0xff

Definition at line 114 of file its_mode2.h.

Referenced by its2_delete_item_from_queue(), its2_find_free_queue_slot(), its2_forward_routed_packet(), its2_print_queue(), its2_process_tx_queue(), its2_router_init(), and its2_router_queue_packet().

### #define ITS2_FLAG_NO_ACK 0x02

Definition at line 113 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_rebroadcast_net_discover_req(), its2_request_local_addr(), its2_request_net_addr(), its2_respond_local_addr(), and its2_respond_net_addr().

### #define ITS2_NO_AVAILABLE_SLOTS 0xff

Definition at line 109 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), and its2_rebroadcast_net_discover_req().

### #define ITS2_STATUS_QUEUED 0x00

Definition at line 107 of file its_mode2.h.

**#define ITS2_STATUS_TX_QUEUE_FULL 0x01**

Definition at line 108 of file its_mode2.h.

**#define ITS2_TX_STATUS_NEXT_HOP_UNKNOWN 0x04**

Definition at line 100 of file its_mode2.h.
Referenced by its2_process_tx_queue().

**#define ITS2_TX_STATUS_NO_ACK 0x02**

Definition at line 98 of file its_mode2.h.
Referenced by its2_process_tx_queue().

**#define ITS2_TX_STATUS_NO_ROUTE 0x01**

Definition at line 97 of file its_mode2.h.
Referenced by its2_process_tx_queue().

**#define ITS2_TX_STATUS_REMOTE_NO_ACK 0x03**

Definition at line 99 of file its_mode2.h.

**#define ITS2_TX_STATUS_SUCCESS 0x00**

Definition at line 96 of file its_mode2.h.
Referenced by its2_process_tx_queue().

**#define ITS2_UPDATE_ROUTE_FAIL 0x01**

Definition at line 104 of file its_mode2.h.
Referenced by its2_router_queue_packet().

**#define ITS2_UPDATE_ROUTE_SUCCESS 0x00**

Definition at line 103 of file its_mode2.h.

**#define POS_DEST_H 0x0a**

Definition at line 78 of file its_mode2.h.

**#define POS_DEST_L 0x09**

Definition at line 77 of file its_mode2.h.

**#define POS_HOP_COUNT 0x0d**

Definition at line 81 of file its_mode2.h.

**#define POS_KEY1 0x00**

Definition at line 68 of file its_mode2.h.

**#define POS_KEY2 0x01**

Definition at line 69 of file its_mode2.h.

**#define POS_LENGTH_HEADER 0x02**

Definition at line 70 of file its_mode2.h.

**#define POS_MAX_HOP_COUNT 0x0b**

Definition at line 79 of file its_mode2.h.

**#define POS_NETWORK_H 0x06**

Definition at line 74 of file its_mode2.h.

**#define POS_NETWORK_L 0x05**

Definition at line 73 of file its_mode2.h.

**#define POS_NUM_ROUTES 0x0c**

Definition at line 80 of file its_mode2.h.

**#define POS_PKT_TYPE 0x03**

Definition at line 71 of file its_mode2.h.

**#define POS_ROUTE_START 0x0e**

Definition at line 82 of file its_mode2.h.

**#define POS_SEQUENCE 0x04**

Definition at line 72 of file its_mode2.h.

**#define POS_SOURCE_H 0x08**

Definition at line 76 of file its_mode2.h.

**#define POS_SOURCE_L 0x07**

Definition at line 75 of file its_mode2.h.

**#define QS_ACK_RECEIVED 0x04**

Definition at line 89 of file its_mode2.h.

Referenced by its2_process_tx_queue().

**#define QS_READY_TO_SEND 0x02**

Definition at line 87 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), its2_process_tx_queue(), and its2_rebroadcast_net_discover_req().

**#define QS_ROUTING_FAILED 0x06**

Definition at line 91 of file its_mode2.h.

Referenced by its2_process_tx_queue().

**#define QS_SENT 0x05**

Definition at line 90 of file its_mode2.h.

Referenced by its2_process_tx_queue(), and its2_transmit().

**#define QS_WAITING_ON_ACK 0x03**

Definition at line 88 of file its_mode2.h.

Referenced by its2_process_tx_queue(), and its2_transmit().

**#define QS_WAITING_ON_LOCAL_ADDR 0x00**

Definition at line 85 of file its_mode2.h.

Referenced by its2_forward_routed_packet(), and its2_process_tx_queue().

**#define QS_WAITING_ON_NETWORK_ADDR 0x01**

Definition at line 86 of file its_mode2.h.

Referenced by its2_process_tx_queue().

#### #define REMOTE_DEVICE 0xffff

Definition at line 94 of file its_mode2.h.

---

## Typedef Documentation

#### typedef enum _its2_result its2_result

Definition at line 152 of file its_mode2.h.

#### typedef enum _its2_state its2_state

Definition at line 162 of file its_mode2.h.

---

## Enumeration Type Documentation

#### enum _its2_result

**Enumerator:**

***RESULT_SUCCESSFUL***
***RESULT_FAILED***
***ITEM_QUEUED***
***QUEUE_FULL***
***ROUTING_TOO_MANY_HOPS***
***NEXT_HOP_NOT_LOCAL***

Definition at line 150 of file its_mode2.h.

#### enum _its2_state

**Enumerator:**

***STATE_STARTUP***
***STATE_RUNNING***
***STATE_SEARCHING***
***STATE_ASSOCIATED***
***STATE_UNASSOCIATED***

Definition at line 154 of file its_mode2.h.

---

## Function Documentation

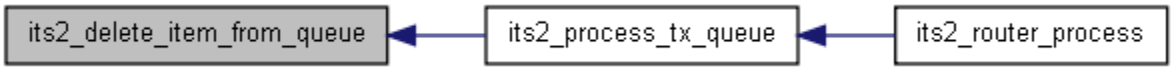
### void its2_delete_item_from_queue (queued_item * *item*)

Definition at line 1025 of file its_mode2.c.

References queued_item::data, queued_item::data_length, queued_item::flag, and ITS2_FLAG_DELETED.

Referenced by its2_process_tx_queue().

Here is the caller graph for this function:

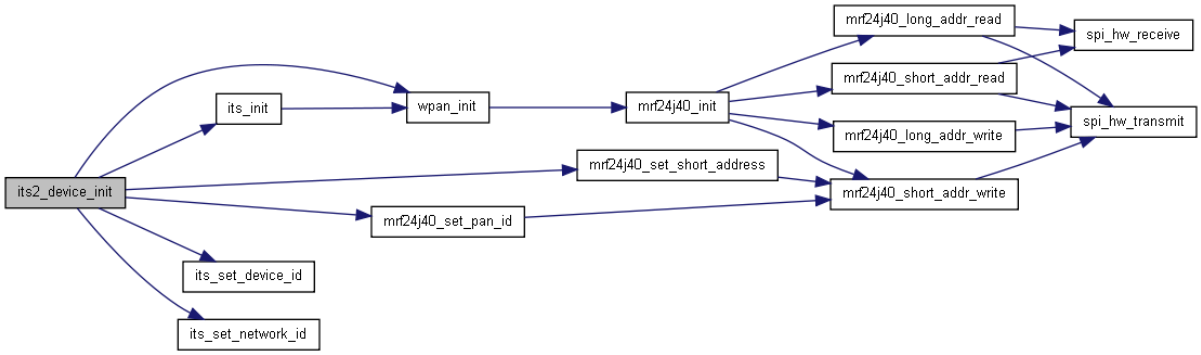



### its2_result its2_device_init (uns16 *my_device_id*, uns16 *network_id*)

Definition at line 1136 of file its_mode2.c.

References debug_str, its_init(), its_set_device_id(), its_set_network_id(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), state, STATE_UNASSOCIATED, and wpan_init().

Here is the call graph for this function:

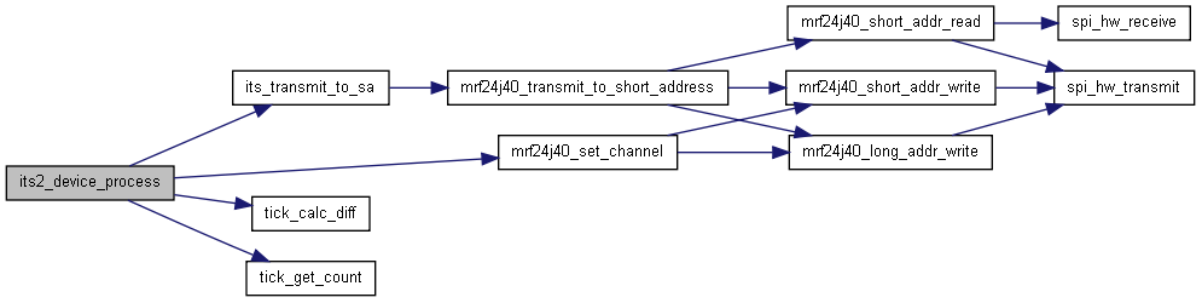



### void its2_device_process ()

Definition at line 1163 of file its_mode2.c.

References channel, debug_nl, debug_str, debug_var, ITS_ASSOC_REQ, its_transmit_to_sa(), mrf24j40_set_channel(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, state, STATE_SEARCHING, STATE_UNASSOCIATED, tick_calc_diff(), tick_get_count(), tick_marker, and uns16.

Here is the call graph for this function:



**void its2_device_receive_callback (uns8 * *data*, uns8 *data_length*)**

Referenced by wpan_data_received_callback().

Here is the caller graph for this function:


**its2_result its2_device_transmit (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

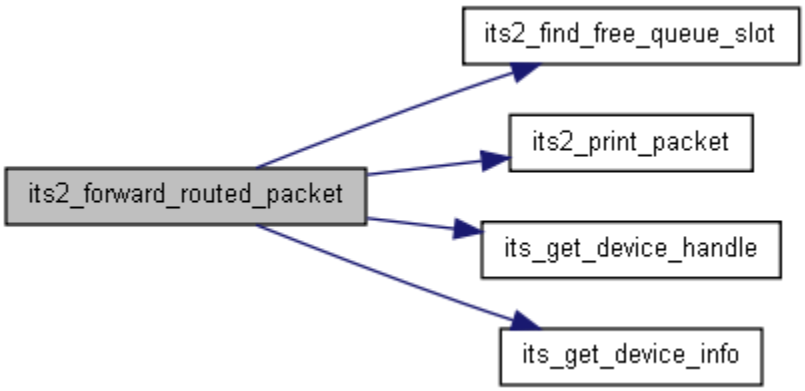
**its2_result its2_find_coordinator ()**

**its2_result its2_forward_routed_packet (its2_packet * *pkt*, uns8 * *data*, uns8 *data_length*)**

Definition at line 653 of file its_mode2.c.

References its_device_info::addr, queued_item::data, queued_item::data_length, debug_int, debug_int_hex_16bit, debug_spc, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, its2_find_free_queue_slot(), ITS2_FLAG_DELETED, ITS2_FLAG_NO_ACK, ITS2_NO_AVAILABLE_SLOTS, its2_print_packet(), its2_packet::its_dest_id, ITS_DEVICE_NONE, its_get_device_handle(), its_get_device_info(), NEXT_HOP_NOT_LOCAL, its2_packet::num_routes, queued_item::packet, QS_READY_TO_SEND, QS_WAITING_ON_LOCAL_ADDR, QUEUE_FULL, its_address::remote, remote_address::remote_indicator, its2_packet::routers, queued_item::sent_count, queued_item::status, and uns8.

Here is the call graph for this function:



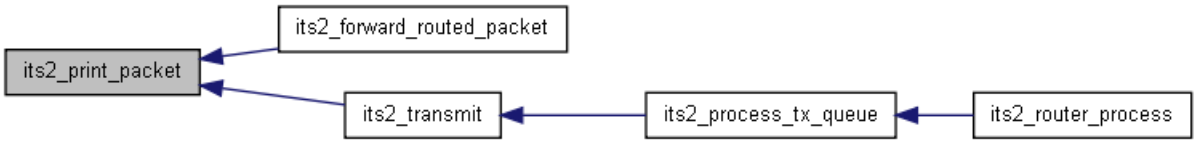

**void its2_print_packet (its2_packet * *pkt*)**

Definition at line 580 of file its_mode2.c.

References debug_int, debug_int_hex_16bit, debug_putc, debug_str, its2_packet::hop_count, its2_packet::its_dest_id, its2_packet::its_network_id, its2_packet::its_source_id, its2_packet::max_hop_count, its2_packet::num_routes, its2_packet::packet_type, its2_packet::routers, its2_packet::sequence, and uns8.

Referenced by its2_forward_routed_packet(), and its2_transmit().

Here is the caller graph for this function:




### void its2_print_queue ()

Definition at line 1106 of file its_mode2.c.

References debug_nl, debug_var, queued_item::flag, ITS2_FLAG_DELETED, its2_packet::its_dest_id, its2_packet::its_source_id, queued_item::packet, queued_item::sent_count, queued_item::status, tick_get_count(), queued_item::tick_sent, and uns8.

Here is the call graph for this function:

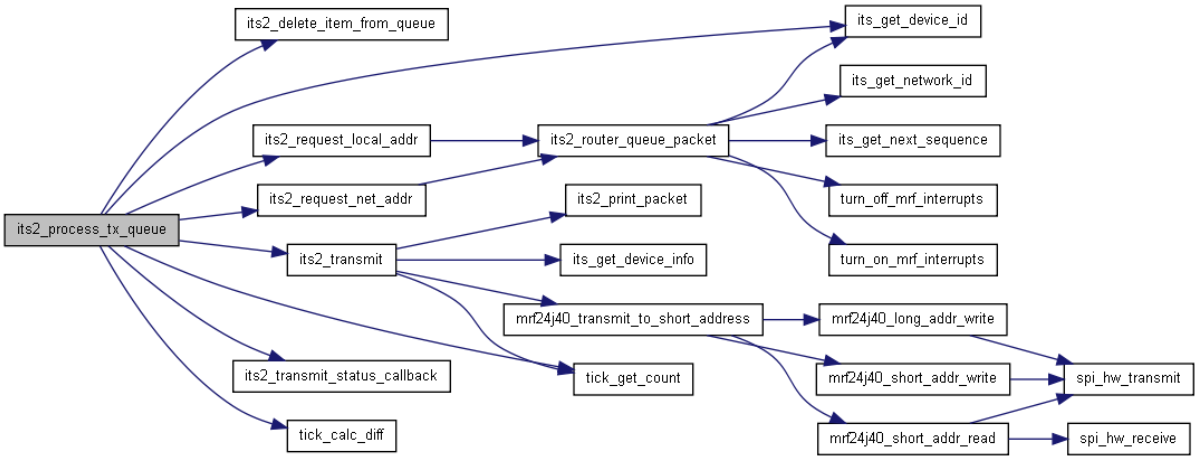

### void its2_process_tx_queue ()

Definition at line 904 of file its_mode2.c.

References debug_int, debug_int_hex_16bit, debug_str, queued_item::dest_its_device_id, queued_item::flag, its2_delete_item_from_queue(), ITS2_FLAG_DELETED, its2_request_local_addr(), its2_request_net_addr(), its2_transmit(), its2_transmit_status_callback(), its2_transmitting, ITS2_TX_STATUS_NEXT_HOP_UNKNOWN, ITS2_TX_STATUS_NO_ACK, ITS2_TX_STATUS_NO_ROUTE, ITS2_TX_STATUS_SUCCESS, its_get_device_id(), its2_packet::its_source_id, queued_item::packet, QS_ACK_RECEIVED, QS_READY_TO_SEND, QS_ROUTING_FAILED, QS_SENT, QS_WAITING_ON_ACK, QS_WAITING_ON_LOCAL_ADDR, QS_WAITING_ON_NETWORK_ADDR, queued_item::sent_count, queued_item::status, tick_calc_diff(), tick_get_count(), queued_item::tick_sent, uns16, and uns8.

Referenced by its2_router_process().

Here is the call graph for this function:



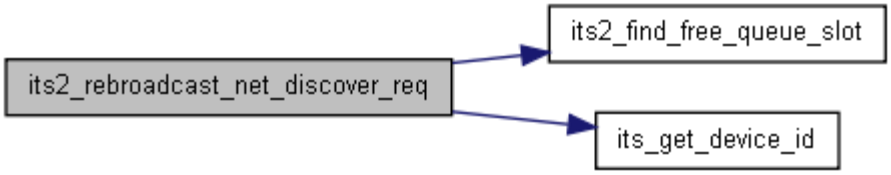

Here is the caller graph for this function:


**its2_result its2_rebroadcast_net_discover_req (its2_packet * *pkt*)**

Definition at line 613 of file its_mode2.c.

References queued_item::data, queued_item::data_length, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, its2_find_free_queue_slot(), ITS2_FLAG_NO_ACK, ITS2_NO_AVAILABLE_SLOTS, ITS_DEVICE_NONE, its_get_device_id(), its2_packet::its_source_id, its2_packet::max_hop_count, its2_packet::num_routes, queued_item::packet, QS_READY_TO_SEND, QUEUE_FULL, its2_packet::routers, ROUTING_TOO_MANY_HOPS, queued_item::sent_count, its2_packet::sequence, queued_item::status, and uns8.

Here is the call graph for this function:



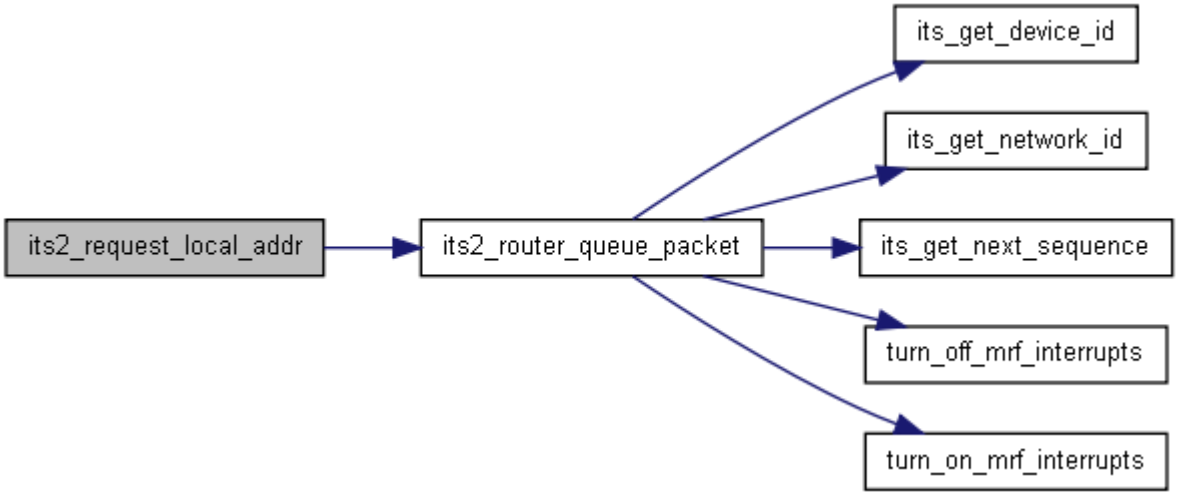

**void its2_request_local_addr (uns16 *device_id*)**

Definition at line 779 of file its_mode2.c.

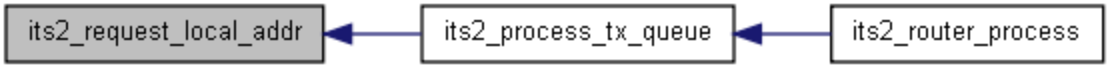
References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_LOCAL_DISCOVER_REQ.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:




Here is the caller graph for this function:



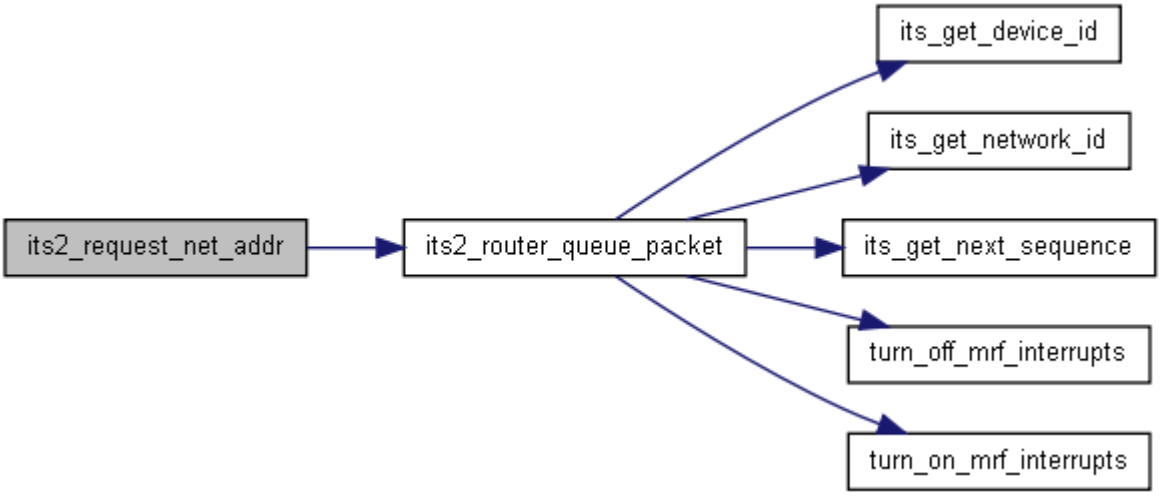
### void its2_request_net_addr (uns16 *device_id*)

Definition at line 792 of file its_mode2.c.

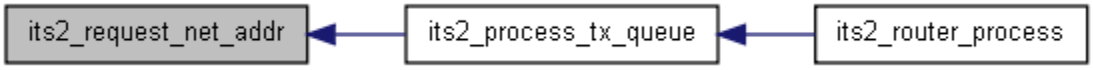
References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_NET_DISCOVER_REQ.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:




Here is the caller graph for this function:

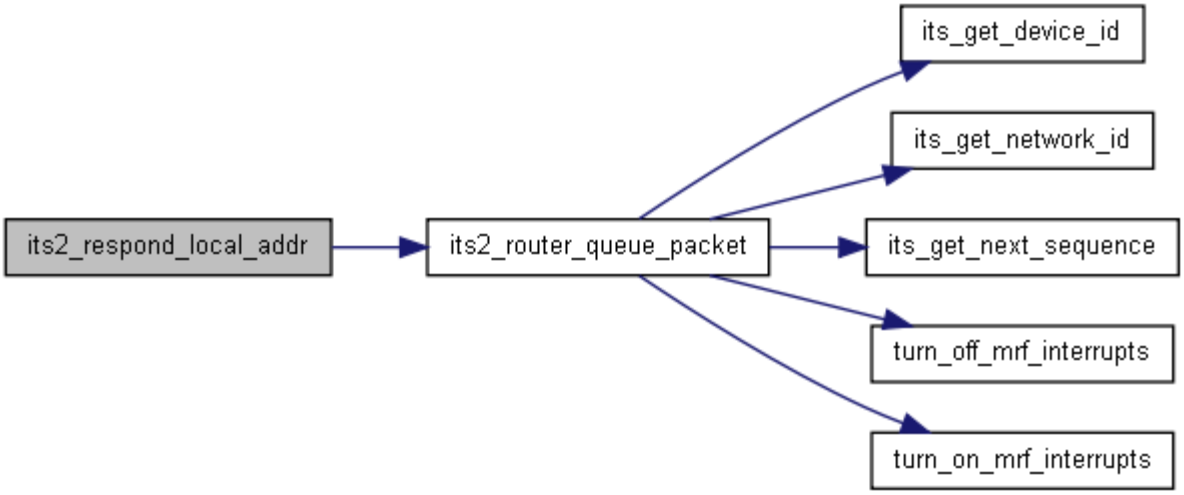



### void its2_respond_local_addr (uns16 *device_id*)

Definition at line 783 of file its_mode2.c.

References debug_module, debug_str, ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_LOCAL_DISCOVER_RES.

Here is the call graph for this function:

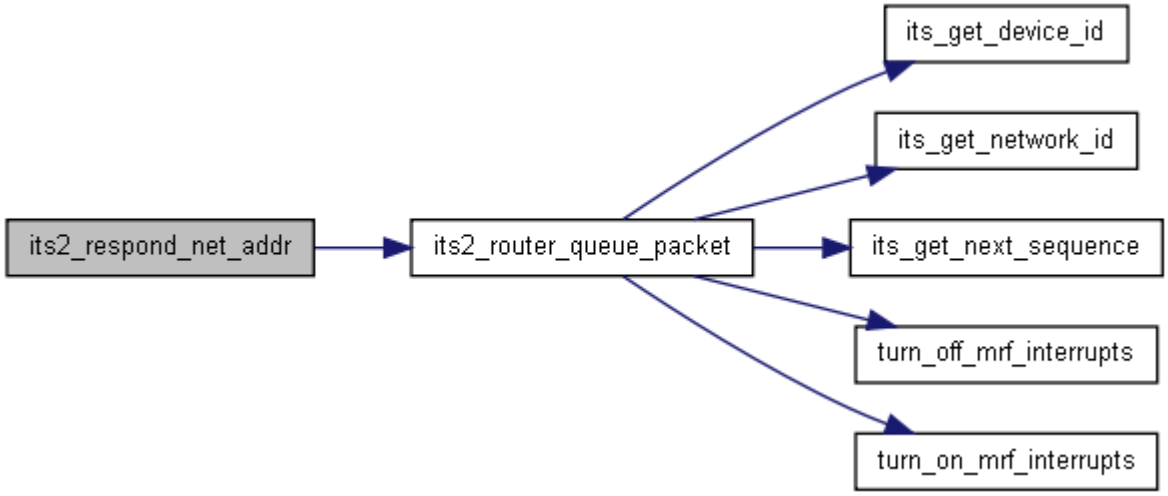



**void its2_respond_net_addr (uns16 *device_id*)**

Definition at line 796 of file its_mode2.c.

References ITS2_FLAG_NO_ACK, its2_router_queue_packet(), and ITS_NET_DISCOVER_RES.

Here is the call graph for this function:

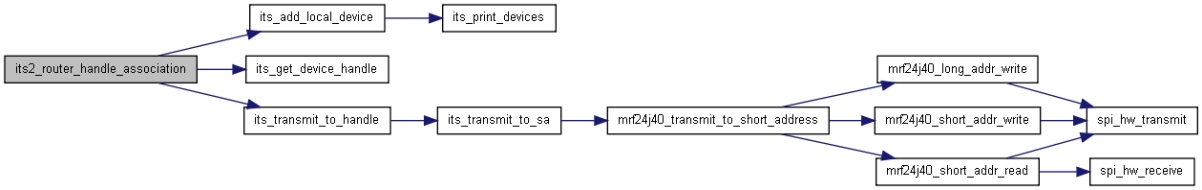



**its2_result it2_router_handle_association (uns16 *pan_id*, uns16 *its_device_id*)**

Definition at line 118 of file its_mode2.c.

References debug_str, its_add_local_device(), ITS_ASSOC_RES, ITS_DEVICE_NONE, its_get_device_handle(), its_transmit_to_handle(), RESULT_FAILED, and RESULT_SUCCESSFUL.

Here is the call graph for this function:

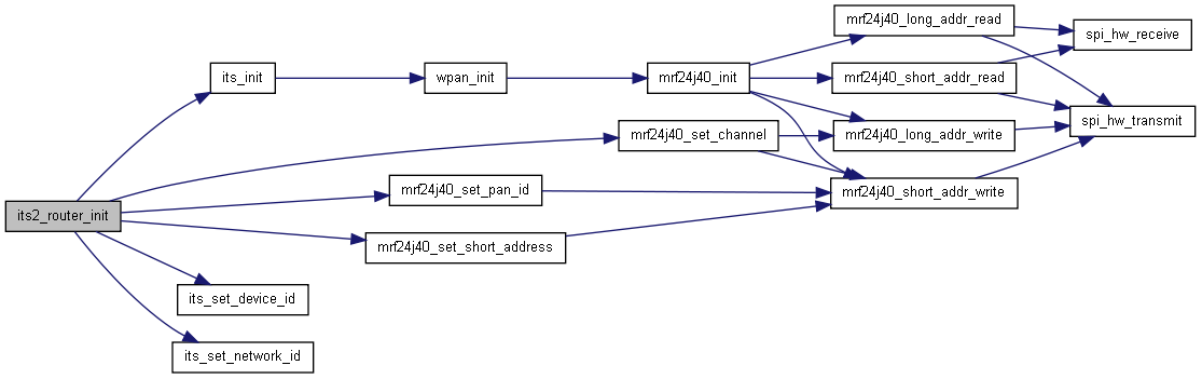



### uns8 its2_router_init (uns16 *my_device_id*, uns16 *network_id*)

Definition at line 743 of file its_mode2.c.

References debug_nl, debug_str, debug_var, queued_item::flag, ITS2_FLAG_DELETED, its2_seen_index, its_init(), its_set_device_id(), its_set_network_id(), seen_packet::its_source_id, mrf24j40_set_channel(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), and uns8.

Here is the call graph for this function:

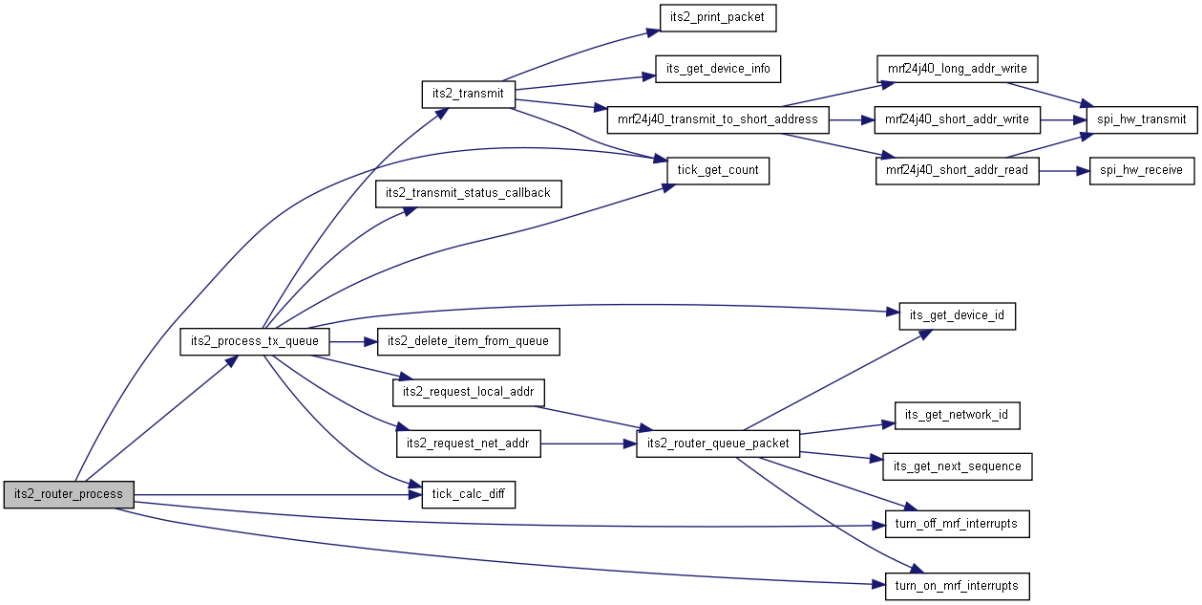



### void its2_router_process ()

Definition at line 808 of file its_mode2.c.

References its2_process_tx_queue(), state_timeout, tick_calc_diff(), tick_get_count(), tick_marker, turn_off_mrf_interrupts(), turn_on_mrf_interrupts(), and uns16.

Here is the call graph for this function:



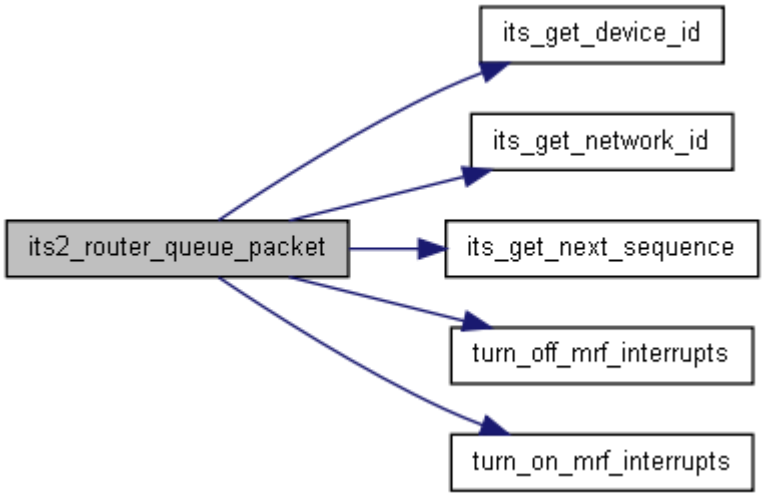

**its2_result its2_router_queue_packet (uns16 *device_id*, uns8 *packet_type*, uns8 * *data*, uns8 *data_length*, uns8 *ack*)**

Definition at line 823 of file its_mode2.c.

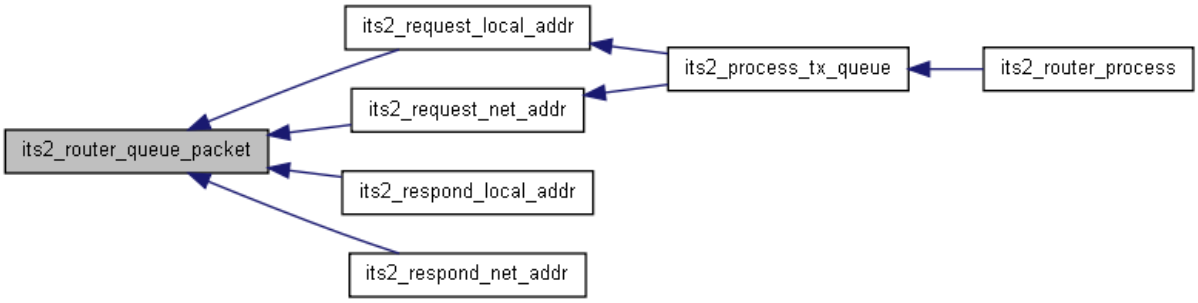
References queued_item::data, queued_item::data_length, debug_str, debug_var, queued_item::dest_its_device_id, queued_item::flag, its2_packet::hop_count, ITEM_QUEUED, ITS2_FLAG_DELETED, ITS2_UPDATE_ROUTE_FAIL, its2_packet::its_dest_id, its_get_device_id(), its_get_network_id(), its_get_next_sequence(), ITS_LOCAL_DISCOVER_REQ, ITS_NET_DISCOVER_REQ, its2_packet::its_network_id, its2_packet::its_source_id, its2_packet::max_hop_count, queued_item::packet, its2_packet::packet_type, QUEUE_FULL, ROUTING_TOO_MANY_HOPS, queued_item::sent_count, its2_packet::sequence, turn_off_mrf_interrupts(), turn_on_mrf_interrupts(), and uns8.

Referenced by its2_request_local_addr(), its2_request_net_addr(), its2_respond_local_addr(), and its2_respond_net_addr().

Here is the call graph for this function:



Here is the caller graph for this function:



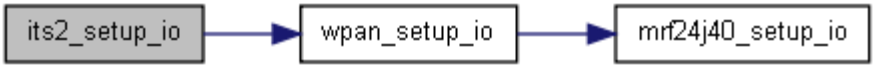

**void its2_router_receive_callback (uns16 *device_id*, uns8 * *data*, uns8 *data_length*)**

**void its2_setup_io ()**

Definition at line 95 of file its_mode2.c.

References wpan_setup_io().

Here is the call graph for this function:



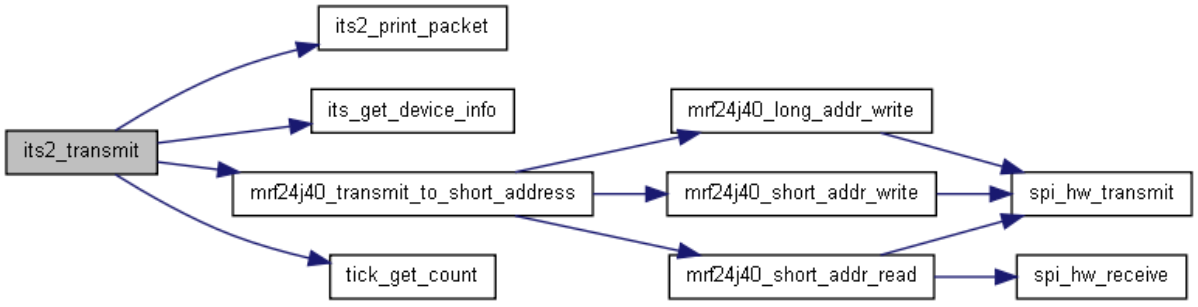

**void its2_transmit (queued_item * *item*)**

Definition at line 1035 of file its_mode2.c.

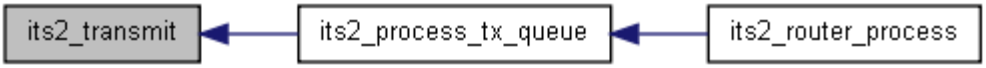
References its_device_info::addr, queued_item::data, queued_item::data_length, debug_int, debug_int_hex_16bit, debug_putc, debug_str, debug_var, queued_item::dest_device_handle, queued_item::dest_its_device_id, queued_item::flag, FRAME_TYPE_DATA, ITS2_FLAG_ACK, its2_print_packet(), its2_transmitting, ITS_DEVICE_NONE, its_get_device_info(), its_address::local, mrf24j40_transmit_to_short_address(), MRF_ACK, MRF_NO_ACK, its2_packet::num_routes, queued_item::packet, local_address::pan_id, QS_SENT, QS_WAITING_ON_ACK, its2_packet::routers, queued_item::sent_count, local_address::short_address, queued_item::status, tick_get_count(), queued_item::tick_sent, and uns8.

Referenced by its2_process_tx_queue().

Here is the call graph for this function:



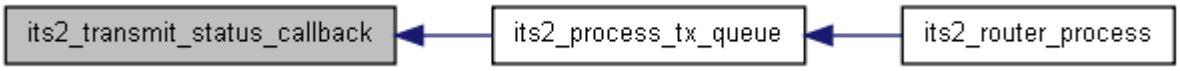

Here is the caller graph for this function:




**void its2_transmit_status_callback (its2_packet * *pkt*, uns8 *status*)**

Referenced by its2_process_tx_queue().

Here is the caller graph for this function:




---

## Variable Documentation

**bit debug_module**

Definition at line 64 of file its_mode2.c.

Referenced by its2_respond_local_addr().

**its2_state state**

Definition at line 47 of file its_mode1.c.

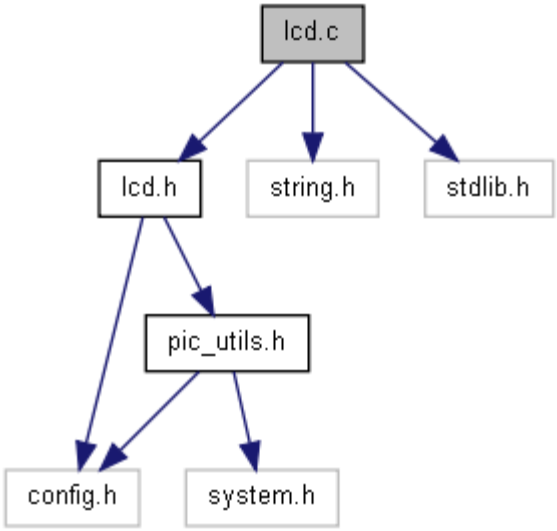
Referenced by its1_device_init(), its1_device_process(), its1_find_controller(), its2_device_init(), its2_device_process(), its2_find_controller(), and wpan_data_received_callback().

---

# lcd.c File Reference

```
#include "lcd.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for lcd.c:




## Functions

- void lcd_cursor_home ()
- void lcd_init ()
- *Initialise LCD ready for display.* void lcd_set_cgram_pos (uns8 x)
- void lcd_set_ddram_pos (uns8 x)
- void lcd_setup ()
- *Setup port and pins to talk to LCD.* void lcd_toggle_e ()
- void lcd_wait_busy ()
- *Wait while LCD is busy.* void lcd_write_byte (uns8 data)
- void lcd_write_command (uns8 data)
- *Sends a command to the LCD.* void lcd_write_data (uns8 data)
- *Send one byte of data to the LCD.* void lcd_write_data_int (uns16 i)
- *Print a 16 bit integer the the LCD.* void lcd_write_data_str (char *str)
- *Print a string to the LCD.* void lcd_write_nibble (uns8 data)

---

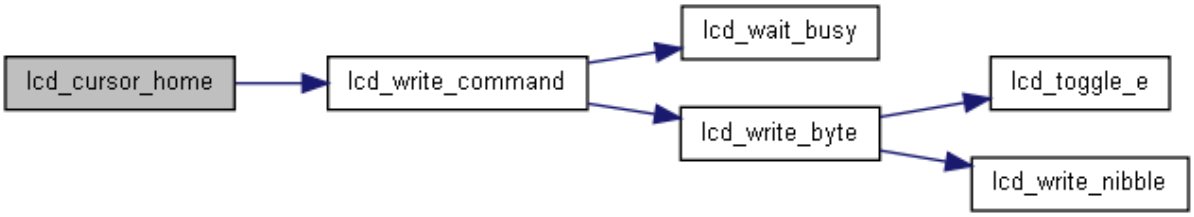
## Function Documentation

### void lcd_cursor_home ()

Definition at line 204 of file lcd.c.

References LCD_CLEAR_DISP, and lcd_write_command().

Here is the call graph for this function:



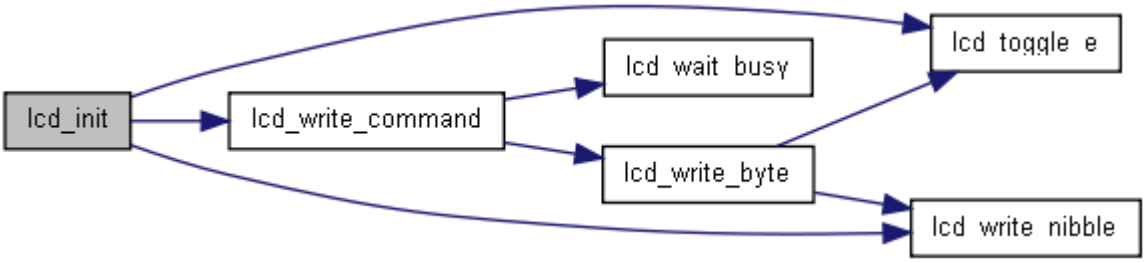
**void lcd_init ()**

Configures LCD for 4 bit operation and gets ready for displaying text

Definition at line 84 of file lcd.c.

References LCD_CLEAR_DISP, LCD_RETURN_HOME, lcd_toggle_e(), lcd_write_command(), and lcd_write_nibble().

Here is the call graph for this function:

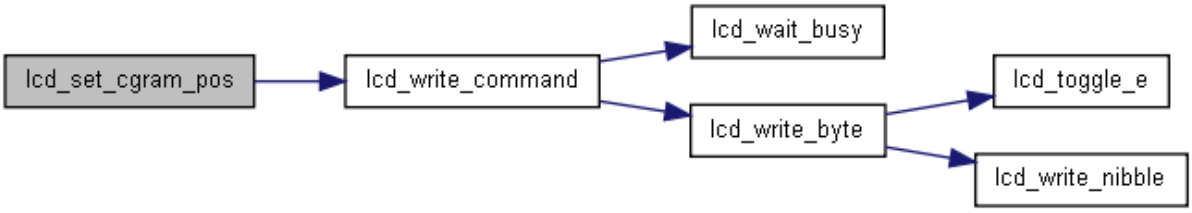



**void lcd_set_cgram_pos (uns8 *x*)**

Definition at line 198 of file lcd.c.

References lcd_write_command().

Here is the call graph for this function:

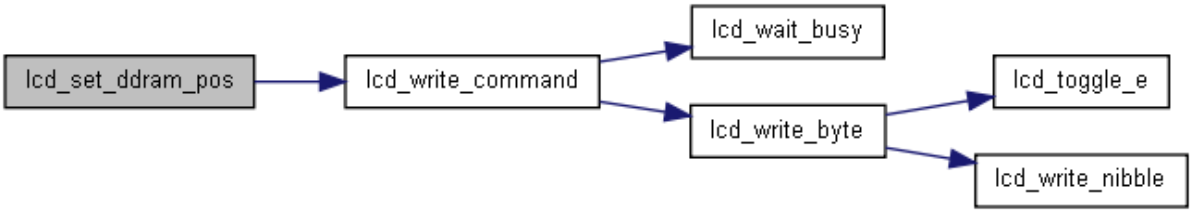



**void lcd_set_ddram_pos (uns8 *x*)**

Definition at line 193 of file lcd.c.

References lcd_write_command().

Here is the call graph for this function:




**void lcd_setup ()**

Call this routine first, to set up tris bits correctly to talk to the LCD

Definition at line 67 of file lcd.c.

References clear_pin, and make_output.

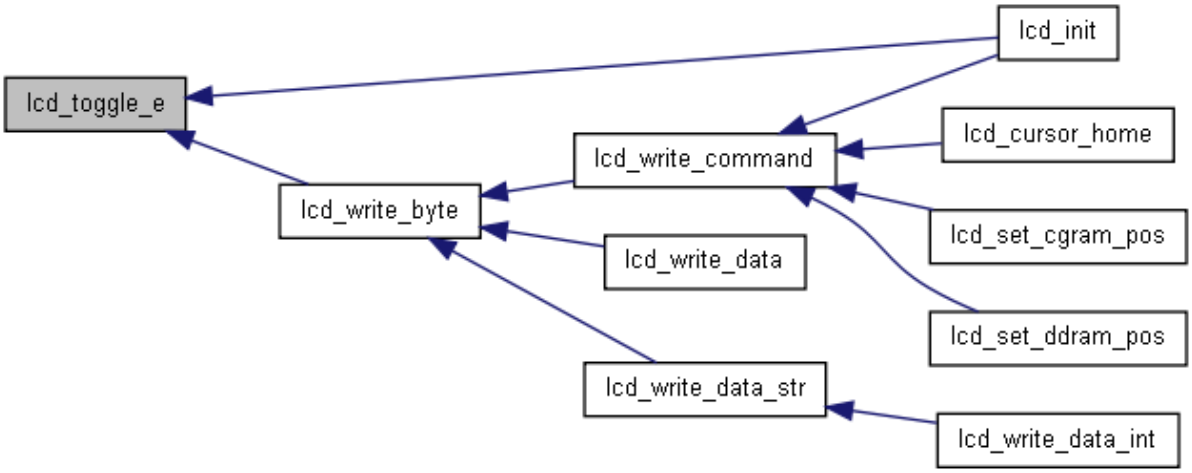
**void lcd_toggle_e ()**

Definition at line 40 of file lcd.c.

References clear_pin, and set_pin.

Referenced by lcd_init(), and lcd_write_byte().

Here is the caller graph for this function:




**void lcd_wait_busy ()**

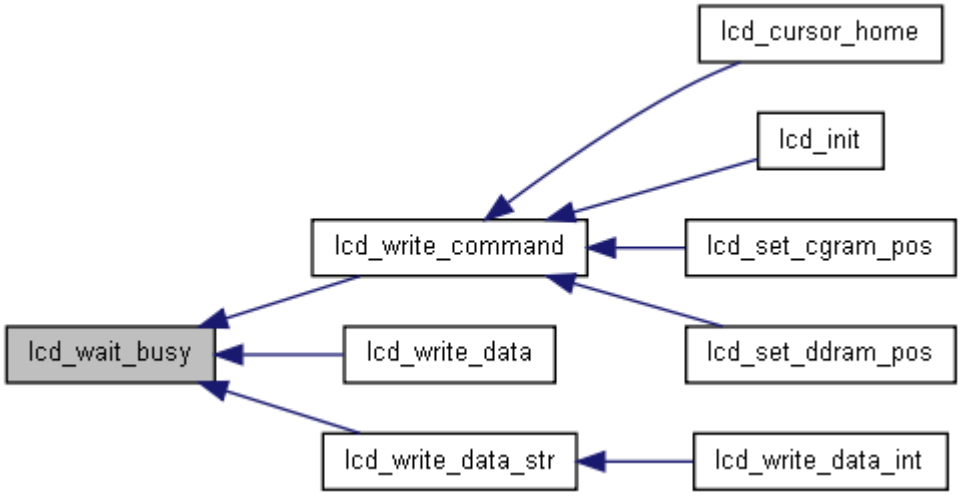
Internal routine to wait while the LCD is busy and unable to accept more data

Definition at line 154 of file lcd.c.

References clear_pin, set_pin, and test_pin.

Referenced by lcd_write_command(), lcd_write_data(), and lcd_write_data_str().

Here is the caller graph for this function:



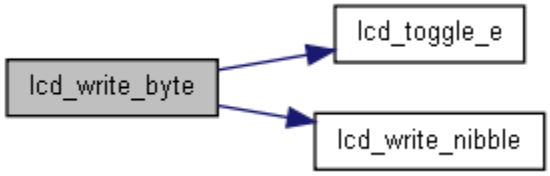

**void lcd_write_byte (uns8 *data*)**

Definition at line 58 of file lcd.c.

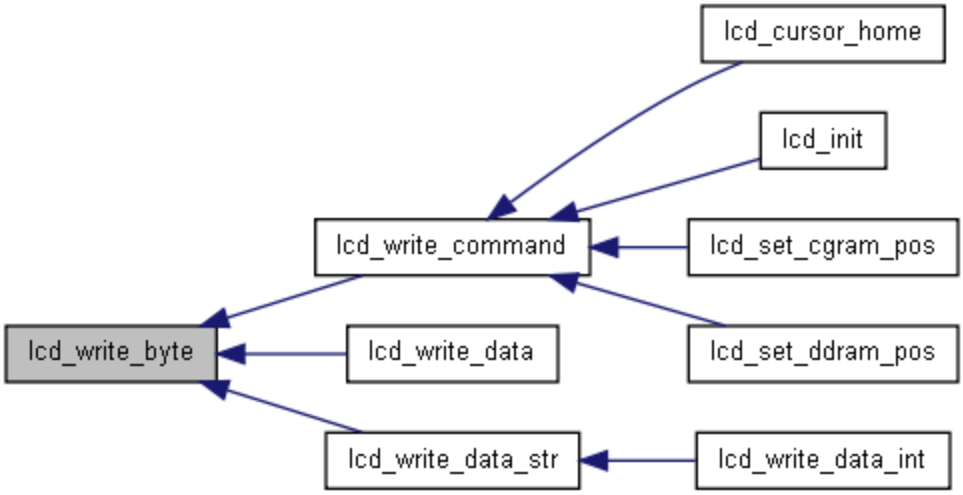
References lcd_toggle_e(), and lcd_write_nibble().

Referenced by lcd_write_command(), lcd_write_data(), and lcd_write_data_str().

Here is the call graph for this function:




Here is the caller graph for this function:




**void lcd_write_command (uns8 *data*)**

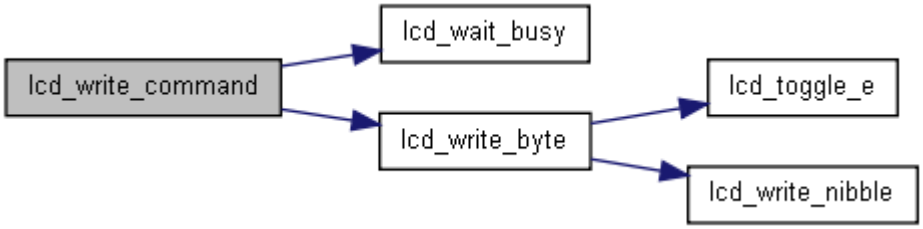
Use this to send commands to the LCD, eg, changing cursor position

Definition at line 113 of file lcd.c.

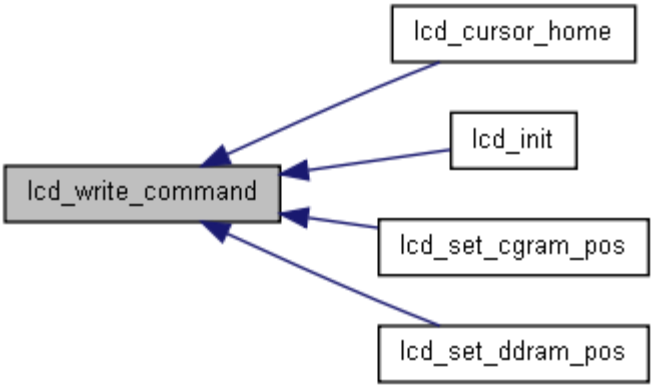
References clear_pin, lcd_wait_busy(), and lcd_write_byte().

Referenced by lcd_cursor_home(), lcd_init(), lcd_set_cgram_pos(), and lcd_set_ddram_pos().

Here is the call graph for this function:




Here is the caller graph for this function:

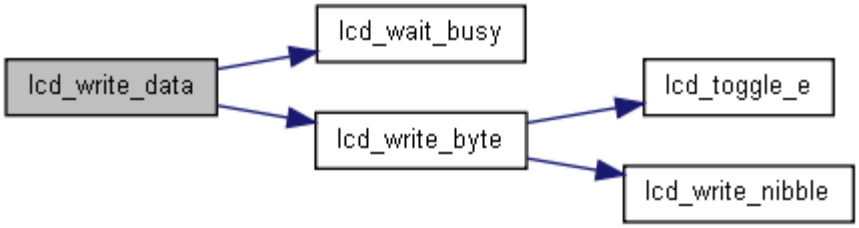



**void lcd_write_data (uns8 *data*)**

Definition at line 123 of file lcd.c.

References clear_pin, lcd_wait_busy(), lcd_write_byte(), and set_pin.

Here is the call graph for this function:



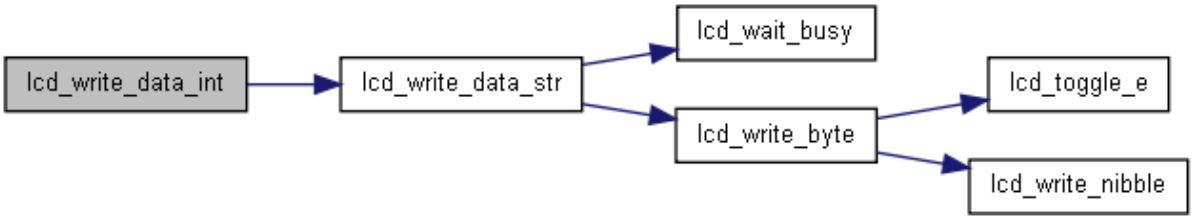

**void lcd_write_data_int (uns16 *i*)**

Displays an unsigned 16 bit integer on the LCD

Definition at line 145 of file lcd.c.

References lcd_write_data_str().

Here is the call graph for this function:




**void lcd_write_data_str (char * *str*)**

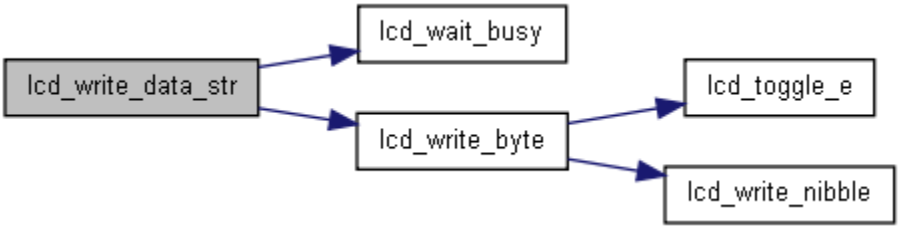
Display the string on the LCD from the current cursor position

Definition at line 132 of file lcd.c.

References clear_pin, lcd_wait_busy(), lcd_write_byte(), and set_pin.

Referenced by lcd_write_data_int().

Here is the call graph for this function:




Here is the caller graph for this function:

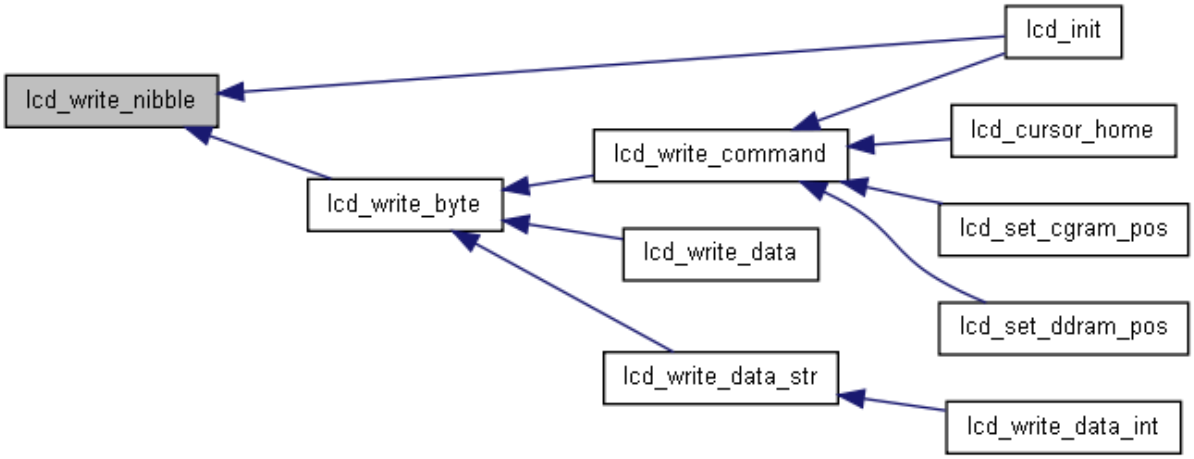

### void lcd_write_nibble (uns8 *data*)

Definition at line 47 of file lcd.c.

References change_pin.

Referenced by lcd_init(), and lcd_write_byte().

Here is the caller graph for this function:



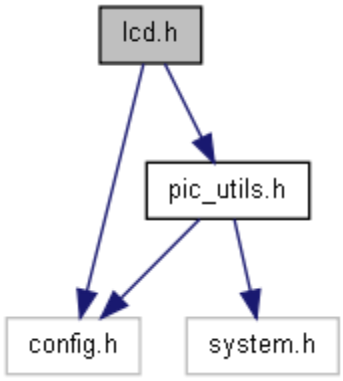

---

## lcd.h File Reference

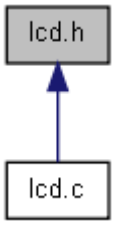
LCD routines.

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

Include dependency graph for lcd.h:




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




## Defines

- #define __lcd_h include
- #define LCD_CLEAR_DISP 0b00000001
- #define lcd_clear_display() lcd_write_command(LCD_CLEAR_DISP);
- *Clear LCD display.* #define LCD_LINE1 0b10000000
- #define LCD_LINE2 0b11000000
- #define LCD_LINE3 0b10010100
- #define LCD_LINE4 0b11010100
- #define lcd_return_home() lcd_write_command(LCD_RETURN_HOME);
- *Return cursor home.* #define LCD_RETURN_HOME 0b00000010
- #define LCD_SET_DRAM_ADDR 0b10000000

## Functions

- void lcd_init ()
- *Initialise LCD ready for display.* void lcd_setup ()
- *Setup port and pins to talk to LCD.* void lcd_wait_busy ()
- *Wait while LCD is busy.* void lcd_write_command (uns8 data)
- *Sends a command to the LCD.* void lcd_write_data (uns8 data)
- *Send one byte of data to the LCD.* void lcd_write_data_int (uns16 i)
- *Print a 16 bit integer the the LCD.* void lcd_write_data_str (char *str)

*Print a string to the LCD.*

---

## Detailed Description

This module contains routines to communicate with an LCD via the 4 bit parallel mode. All pins are protected from read-before-write problems with picpack's port latch simulation. Reads ready flag to check LCD isn't busy before sending more data.

Definition in file lcd.h.

## Define Documentation

**#define __lcd_h  include**

Definition at line 47 of file lcd.h.

**#define LCD_CLEAR_DISP  0b00000001**

Clear LCD display

Definition at line 77 of file lcd.h.

Referenced by lcd_cursor_home(), and lcd_init().

**#define lcd_clear_display()  lcd_write_command(LCD_CLEAR_DISP);**

Definition at line 140 of file lcd.h.

**#define LCD_LINE1  0b10000000**

Move cursor to line 1

Definition at line 83 of file lcd.h.

**#define LCD_LINE2  0b11000000**

Move cursor to line 2

Definition at line 85 of file lcd.h.

**#define LCD_LINE3  0b10010100**

Move cursor to line 3

Definition at line 87 of file lcd.h.

**#define LCD_LINE4  0b11010100**

Move cursor to line 4

Definition at line 89 of file lcd.h.

**#define lcd_return_home()  lcd_write_command(LCD_RETURN_HOME);**

Definition at line 142 of file lcd.h.

**#define LCD_RETURN_HOME  0b00000010**

Move cursor to top left position

Definition at line 79 of file lcd.h.

Referenced by lcd_init().

#### #define LCD_SET_DRAM_ADDR 0b10000000

Move to DRAM address (cursor position)

Definition at line 81 of file lcd.h.

---

## Function Documentation

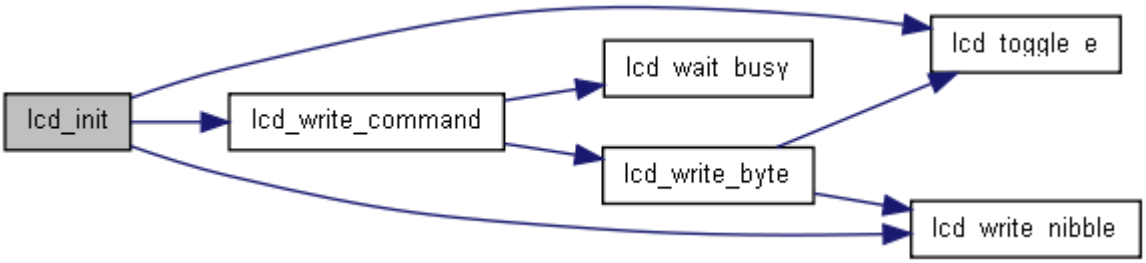
### void lcd_init ()

Configures LCD for 4 bit operation and gets ready for displaying text

Definition at line 84 of file lcd.c.

References LCD_CLEAR_DISP, LCD_RETURN_HOME, lcd_toggle_e(), lcd_write_command(), and lcd_write_nibble().

Here is the call graph for this function:




### void lcd_setup ()

Call this routine first, to set up tris bits correctly to talk to the LCD

Definition at line 67 of file lcd.c.

References clear_pin, and make_output.

### void lcd_wait_busy ()

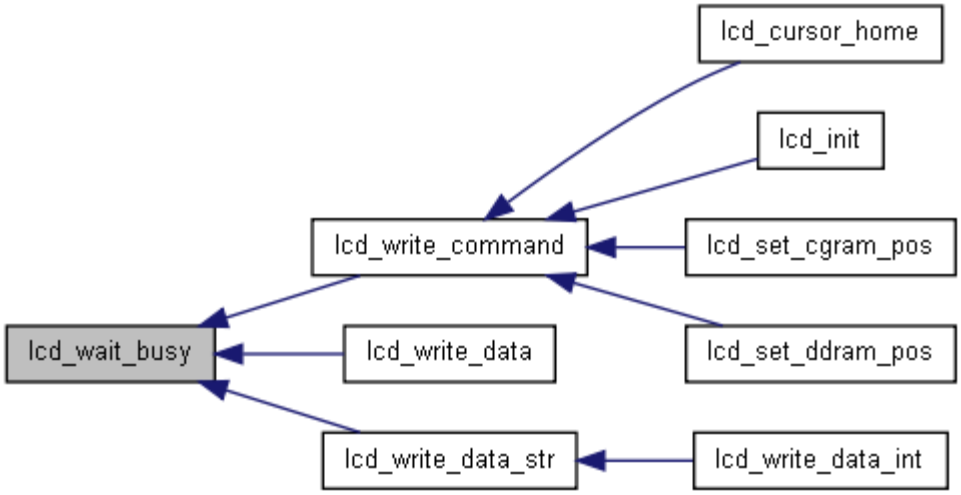
Internal routine to wait while the LCD is busy and unable to accept more data

Definition at line 154 of file lcd.c.

References clear_pin, set_pin, and test_pin.

Referenced by lcd_write_command(), lcd_write_data(), and lcd_write_data_str().

Here is the caller graph for this function:




**void lcd_write_command (uns8 *data*)**

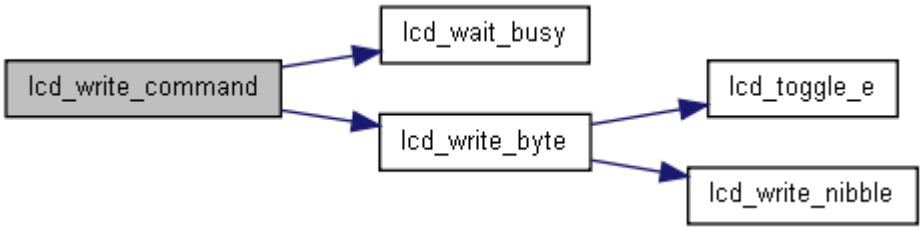
Use this to send commands to the LCD, eg, changing cursor position

Definition at line 113 of file lcd.c.

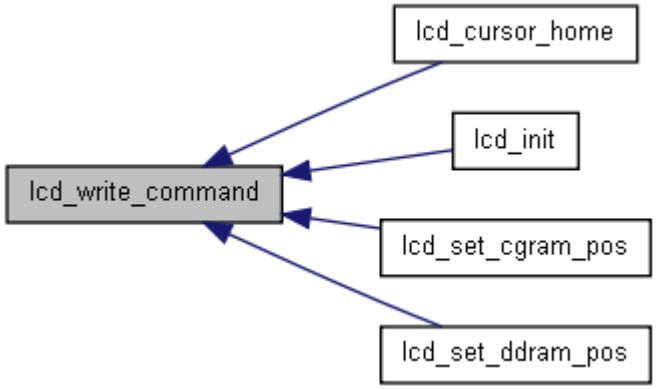
References clear_pin, lcd_wait_busy(), and lcd_write_byte().

Referenced by lcd_cursor_home(), lcd_init(), lcd_set_cgram_pos(), and lcd_set_ddram_pos().

Here is the call graph for this function:




Here is the caller graph for this function:

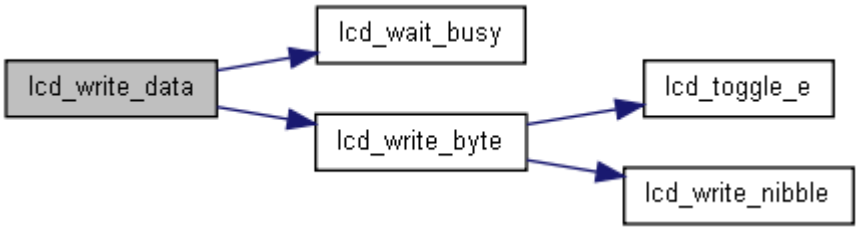



**void lcd_write_data (uns8 *data*)**

Definition at line 123 of file lcd.c.

References clear_pin, lcd_wait_busy(), lcd_write_byte(), and set_pin.

Here is the call graph for this function:



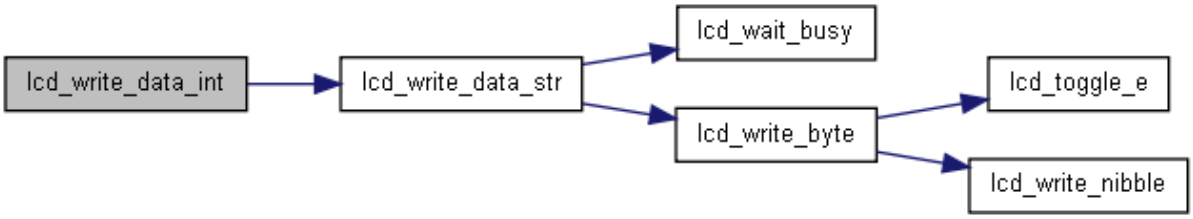

### void lcd_write_data_int (uns16 *i*)

Displays an unsigned 16 bit integer on the LCD

Definition at line 145 of file lcd.c.

References lcd_write_data_str().

Here is the call graph for this function:




### void lcd_write_data_str (char * *str*)

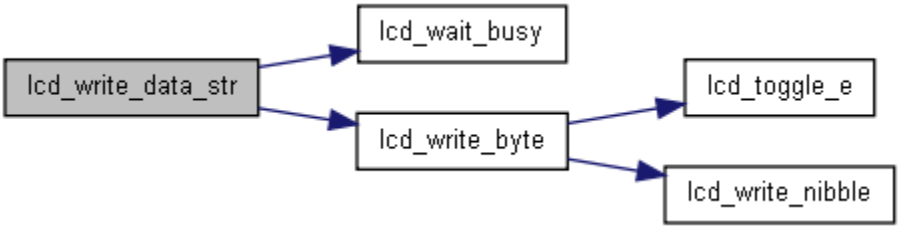
Display the string on the LCD from the current cursor position

Definition at line 132 of file lcd.c.

References clear_pin, lcd_wait_busy(), lcd_write_byte(), and set_pin.

Referenced by lcd_write_data_int().

Here is the call graph for this function:

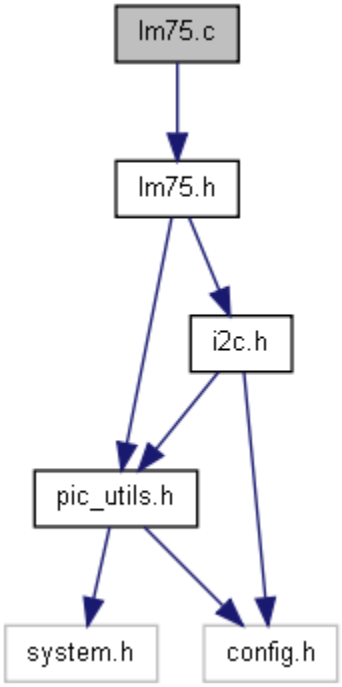



Here is the caller graph for this function:

# lm75.c File Reference

```
#include "lm75.h"
```

Include dependency graph for lm75.c:




## Functions

- uns8 lm75_get_config (uns8 addr)
- *Get LM75 config register.* uns16 lm75_get_temp (uns8 addr)
- *Request temperature from LM75.* void lm75_set_config (uns8 addr, uns8 config)
- *Set LM75 config register.* void lm75_setup ()

*Setup lm75 ports and pins.*

---

## Function Documentation

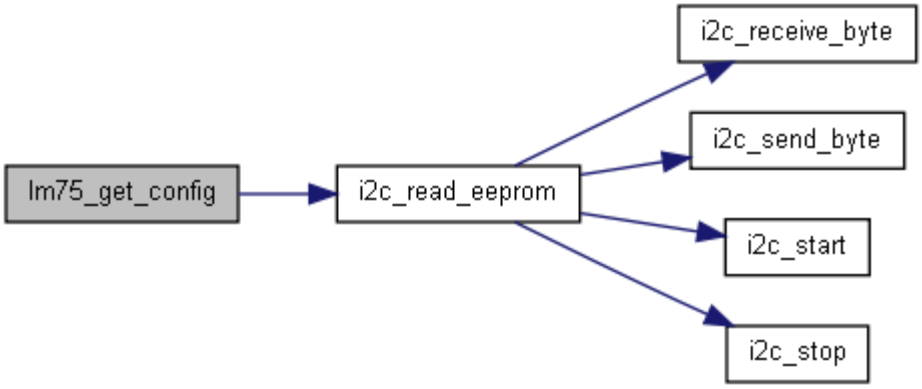
### uns8 lm75_get_config (uns8 *addr*)

Gets the LM75 config register (memory location 0x01)

Definition at line 48 of file lm75.c.

References i2c_read_eeprom().

Here is the call graph for this function:



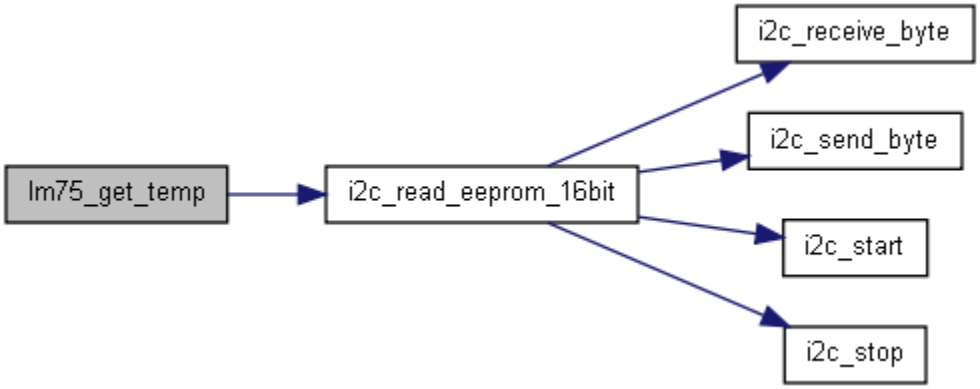

**uns16 lm75_get_temp (uns8 *addr*)**

Returns 16bit raw temperature register from LM75

Definition at line 54 of file lm75.c.

References i2c_read_eeprom_16bit().

Here is the call graph for this function:



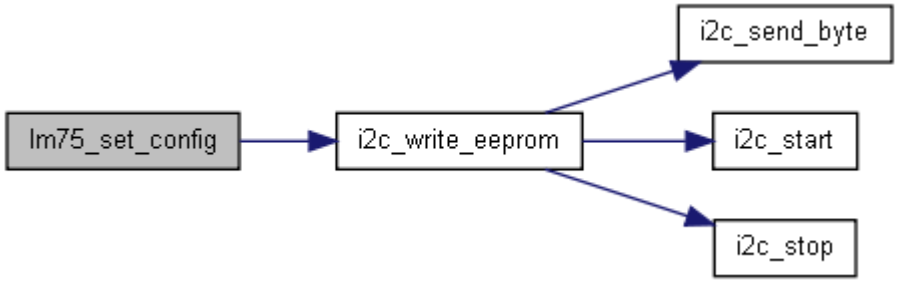

**void lm75_set_config (uns8 *addr*, uns8 *config*)**

Sets the LM75 config register (memory location 0x01)

Definition at line 43 of file lm75.c.

References i2c_write_eeprom().

Here is the call graph for this function:



**void lm75_setup (void)**

Definition at line 39 of file lm75.c.

References i2c_setup.

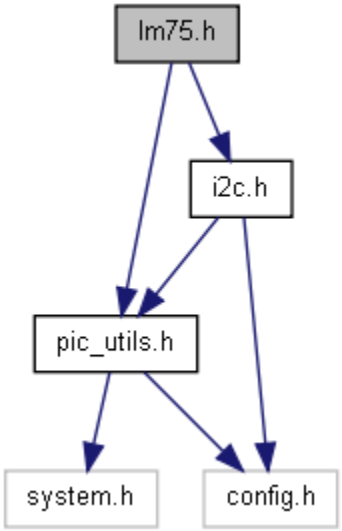
---

# lm75.h File Reference

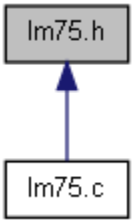
LM75 temperature sensor routines.

```
#include "pic_utils.h"
#include "i2c.h"
```

Include dependency graph for lm75.h:




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




## Defines

- #define LM75_NORMAL 0
- #define LM75_SHUTDOWN 1

## Functions

- uns8 lm75_get_config (uns8 addr)
- *Get LM75 config register.* uns16 lm75_get_temp (uns8 addr)

- *Request temperature from LM75.* void lm75_set_config (uns8 addr, uns8 config)
- *Set LM75 config register.* void lm75_setup (void)

*Setup lm75 ports and pins.*

---

## Detailed Description

A library to communicate with the LM75 sensor

Definition in file lm75.h.

---

## Define Documentation

### #define LM75_NORMAL 0

config define for normal mode

Definition at line 49 of file lm75.h.

### #define LM75_SHUTDOWN 1

Config define for low power mode

Definition at line 47 of file lm75.h.

---

## Function Documentation

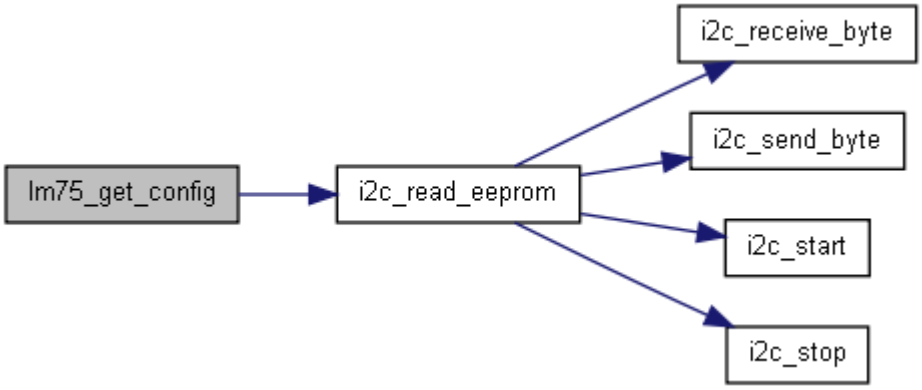
### uns8 lm75_get_config (uns8 *addr*)

Gets the LM75 config register (memory location 0x01)

Definition at line 48 of file lm75.c.

References i2c_read_eeprom().

Here is the call graph for this function:



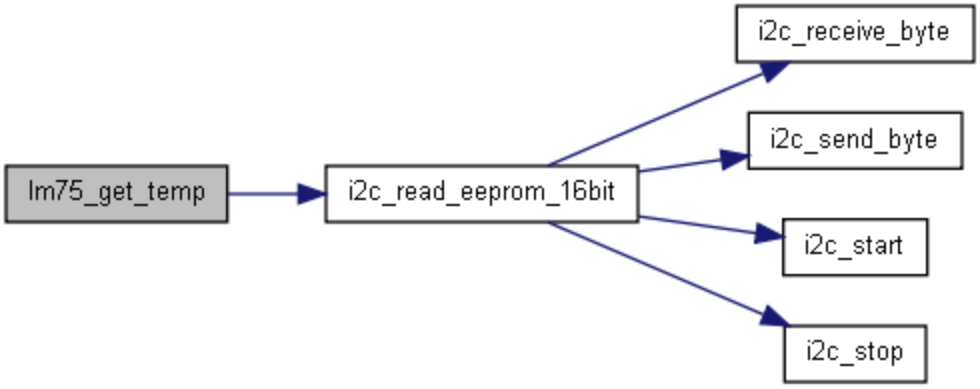

### uns16 lm75_get_temp (uns8 *addr*)

Returns 16bit raw temperature register from LM75

Definition at line 54 of file lm75.c.

References i2c_read_eeprom_16bit().

Here is the call graph for this function:



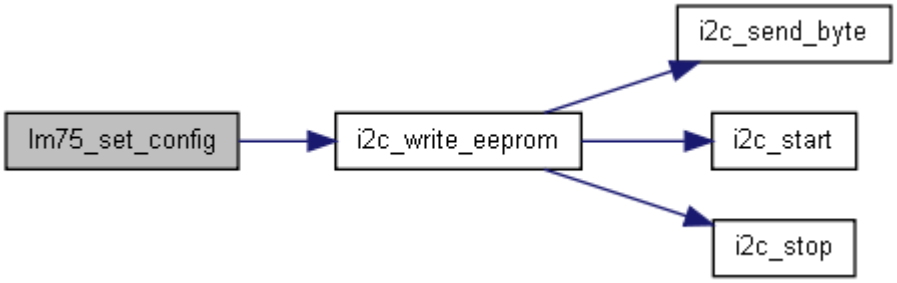

**void lm75_set_config (uns8 *addr*, uns8 *config*)**

Sets the LM75 config register (memory location 0x01)

Definition at line 43 of file lm75.c.

References i2c_write_eeprom().

Here is the call graph for this function:




**void lm75_setup (void)**

Definition at line 39 of file lm75.c.

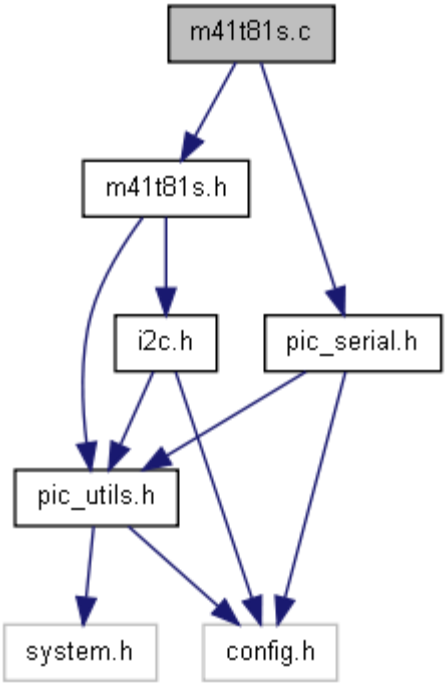
References i2c_setup.

---

## m41t81s.c File Reference

```
#include "m41t81s.h"
#include "pic_serial.h"
```

Include dependency graph for m41t81s.c:




## Functions

- uns8 bcd_to_dec (uns8 bcd)
- uns8 dec_to_bcd (uns8 dec)
- uns8 rtc_get_date ()
- *Get the date register from the ds1307.* uns8 rtc_get_dow ()
- *Get the day register from the m41t81s.* uns8 rtc_get_hours ()
- *Get the decoded hours register from the ds1307.* uns8 rtc_get_minutes ()
- *Get the decoded minutes register from the ds1307.* uns8 rtc_get_month ()
- *Get the month register from the ds1307.* uns8 rtc_get_register (uns8 reg)
- uns8 rtc_get_seconds ()
- *Get the decoded seconds register from the ds1307.* uns8 rtc_get_year ()
- *Get the year register from the ds1307.* void rtc_set_date (uns8 date)
- *Set the date register from the ds1307.* void rtc_set_day (uns8 day)
- *Set the day of the week register from the ds1307.* void rtc_set_hours (uns8 hours)
- *Set the hours register in the ds1307.* void rtc_set_minutes (uns8 minutes)
- *Set the minutes register from the m41t81s.* void rtc_set_month (uns8 month)
- *Set the month register in the ds1307.* void rtc_set_register (uns8 reg, uns8 data)
- void rtc_set_seconds (uns8 seconds)
- *Set the seconds register in the ds1307.* void rtc_set_sqw_freq (uns8 freq)
- *Set the frequence of the square wave output pin.* void rtc_set_year (uns8 year)
- *Set the year register from the m41t81s.* void rtc_setup_io ()
- *Setup ports and pins for use in the ds1307.* void rtc_start_clock ()
- *Starts the clock in the ds1307.* void rtc_start_sqw_output ()
- *Start pulsing on square wave output pin.* void rtc_stop_clock ()
- *Stop the clock in the ds1307.* void rtc_stop_sqw_output ()

*Stop pulsing on square wave output pin.*

## Function Documentation

### uns8 bcd_to_dec (uns8 *bcd*)

Definition at line 40 of file m41t81s.c.

### uns8 dec_to_bcd (uns8 *dec*)

Definition at line 44 of file m41t81s.c.

### uns8 rtc_get_date ()

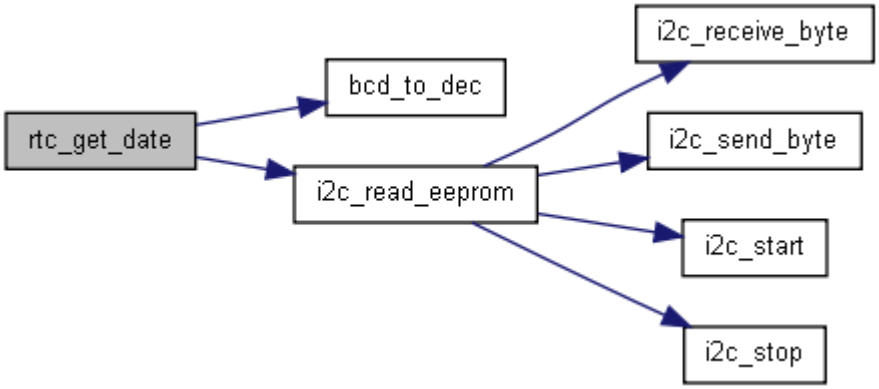
Get the date register from the m41t81s.

Returns the date in month from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 28/29/30/31 depending on month

Definition at line 66 of file m41t81s.c.

References bcd_to_dec(), ds1307_date_register, ds1307_device, i2c_read_eeprom(), m41t81s_date_reg, and m41t81s_device_addr.

Here is the call graph for this function:



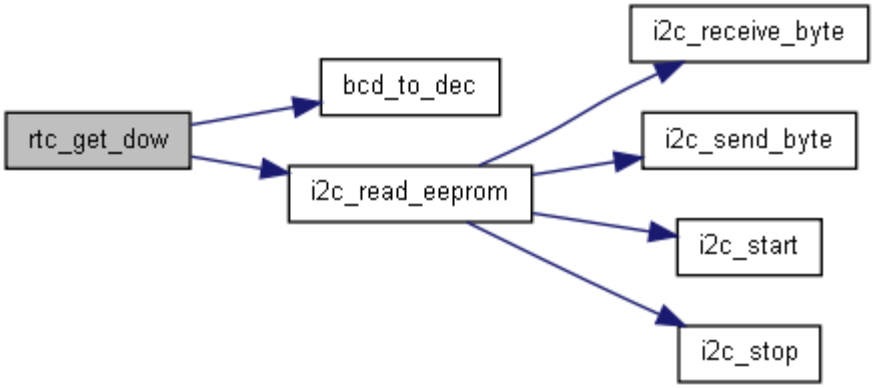

### uns8 rtc_get_dow ()

Returns the day of the week from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range - 1 through 7

Definition at line 62 of file m41t81s.c.

References bcd_to_dec(), i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_dow_reg.

Here is the call graph for this function:




**uns8 rtc_get_hours ()**

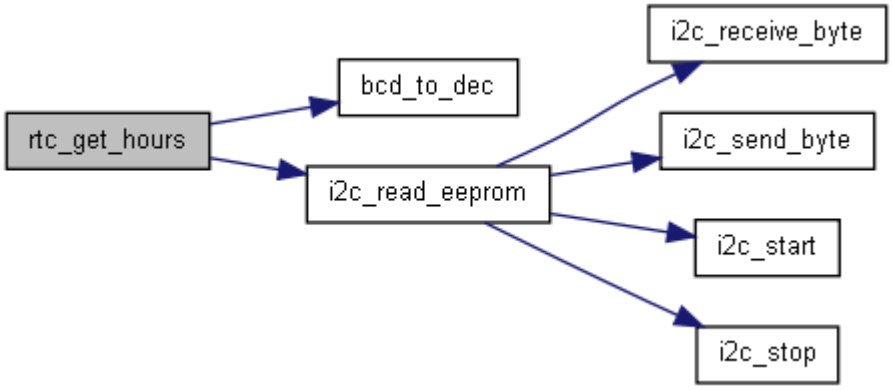
Get the decoded hours register from the m41t81s.

Returns hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. These routines assume the ds1307 is running in 24 hour mode. Range - 0 through 23

Definition at line 51 of file m41t81s.c.

References bcd_to_dec(), ds1307_device, ds1307_hours_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_hours_reg.

Here is the call graph for this function:




**uns8 rtc_get_minutes ()**

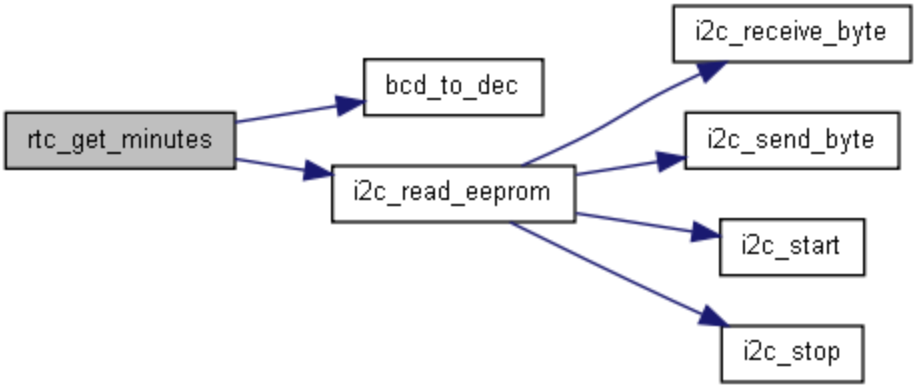
Get the decoded minutes register from the m41t81s.

Returns the number of minutes past the hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 48 of file m41t81s.c.

References bcd_to_dec(), ds1307_device, ds1307_minutes_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_minutes_reg.

Here is the call graph for this function:




**uns8 rtc_get_month ()**

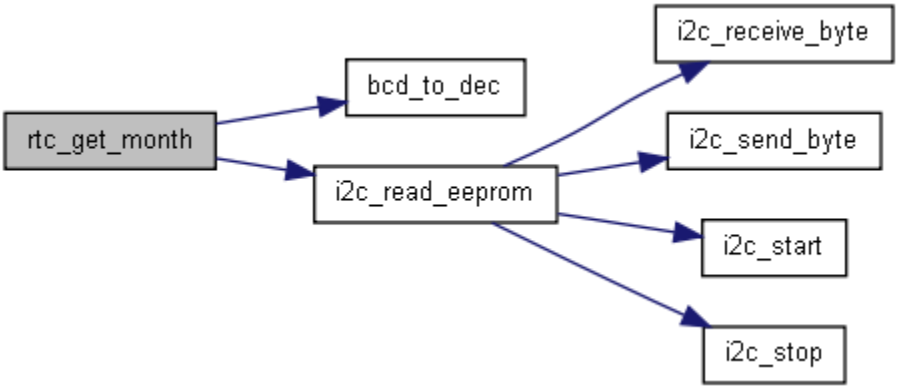
Get the month register from the m41t81s.

Returns the month of the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 12

Definition at line 70 of file m41t81s.c.

References bcd_to_dec(), ds1307_device, ds1307_month_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_month_reg.

Here is the call graph for this function:



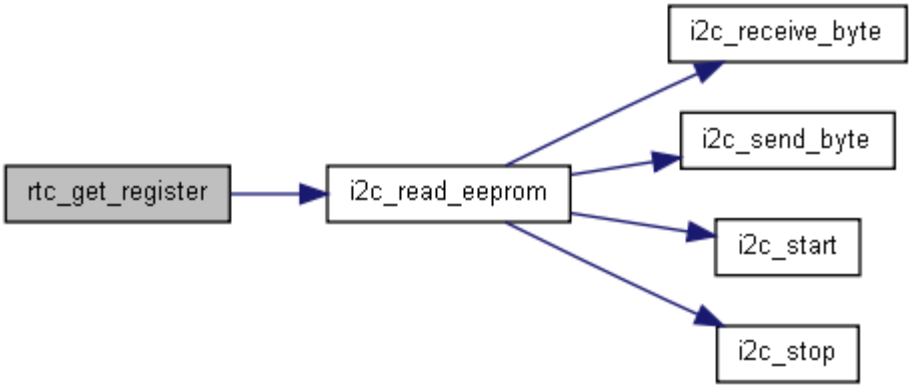

**uns8 rtc_get_register (uns8 *reg*)**

Definition at line 78 of file m41t81s.c.

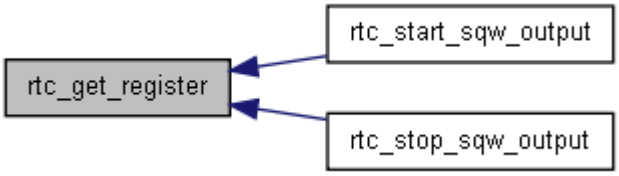
References i2c_read_eeprom(), and m41t81s_device_addr.

Referenced by rtc_start_sqw_output(), and rtc_stop_sqw_output().

Here is the call graph for this function:




Here is the caller graph for this function:




**uns8 rtc_get_seconds ()**

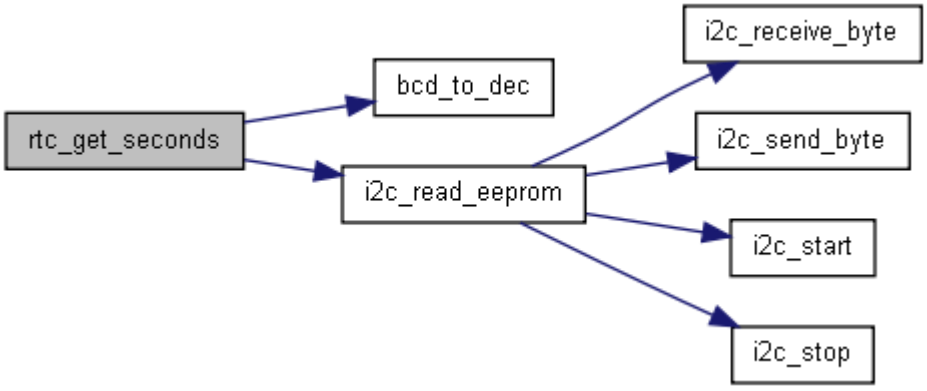
Get the decoded seconds register from the m41t81s.

Returns seconds from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 58 of file m41t81s.c.

References bcd_to_dec(), ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:




**uns8 rtc_get_year ()**

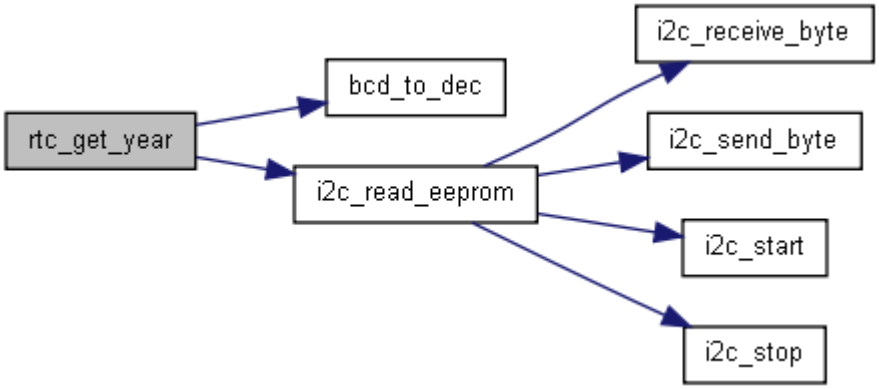
Get the year register from the m41t81s.

Returns the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 0 through 99

Definition at line 74 of file m41t81s.c.

References bcd_to_dec(), ds1307_device, ds1307_year_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_year_reg.

Here is the call graph for this function:




### void rtc_set_date (uns8 *date*)

Set the date register from the m41t81s.

Changes the date in the ds1307.

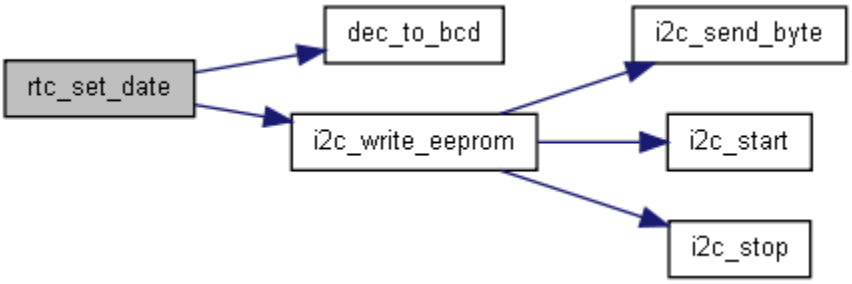
**Parameters:**

*seconds* Value to set date to

Definition at line 107 of file m41t81s.c.

References dec_to_bcd(), ds1307_date_register, ds1307_device, i2c_write_eeprom(), m41t81s_date_reg, and m41t81s_device_addr.

Here is the call graph for this function:




### void rtc_set_day (uns8 *day*)

Set the day of the week register from the m41t81s.

Changes the day of the week in the ds1307.

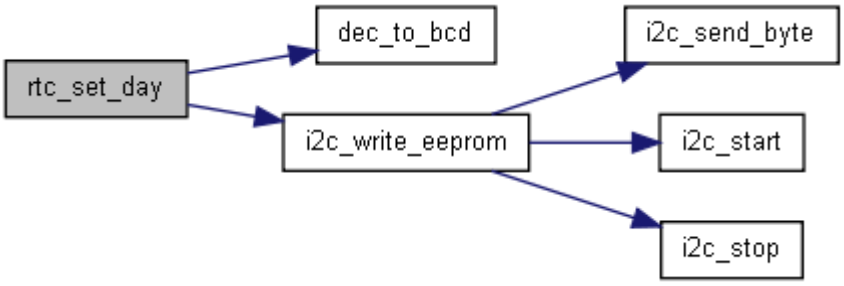
**Parameters:**

*seconds* Value to set day to

Definition at line 104 of file m41t81s.c.

References dec_to_bcd(), ds1307_day_register, ds1307_device, i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_dow_reg.

Here is the call graph for this function:




**void rtc_set_hours (uns8 *hours*)**

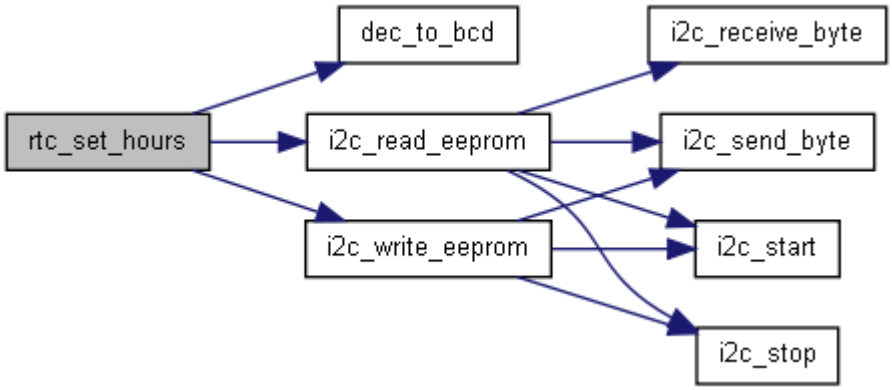
Set the hours register in the m41t81s.

Changes the hours in the ds1307. Forces the ds1307 into 24 hour mode.

Definition at line 116 of file m41t81s.c.

References dec_to_bcd(), ds1307_device, ds1307_hours_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_hours_reg.

Here is the call graph for this function:




**void rtc_set_minutes (uns8 *minutes*)**

Changes the minutes in the m41t81s.

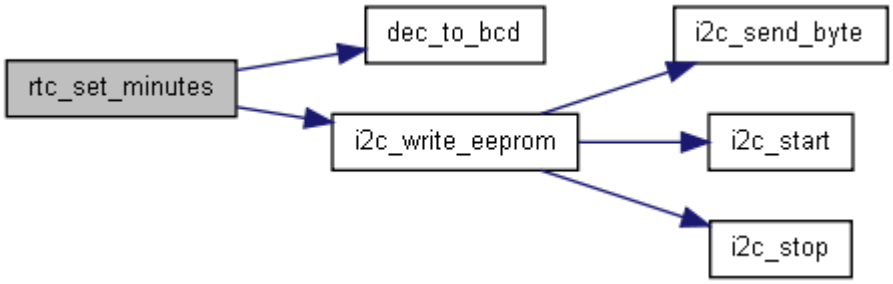
**Parameters:**

*seconds* Value to set minutes to

Definition at line 101 of file m41t81s.c.

References dec_to_bcd(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_minutes_reg.

Here is the call graph for this function:



**void rtc_set_month (uns8 *month*)**

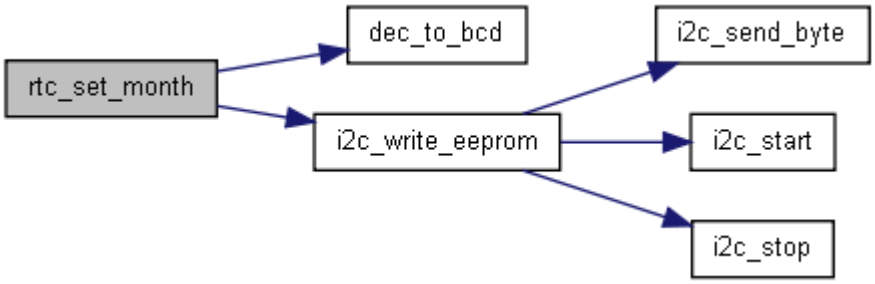
Set the month register in the m41t81s.

Changes the month in the ds1307.

Definition at line 121 of file m41t81s.c.

References dec_to_bcd(), ds1307_device, ds1307_month_register, i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_month_reg.

Here is the call graph for this function:



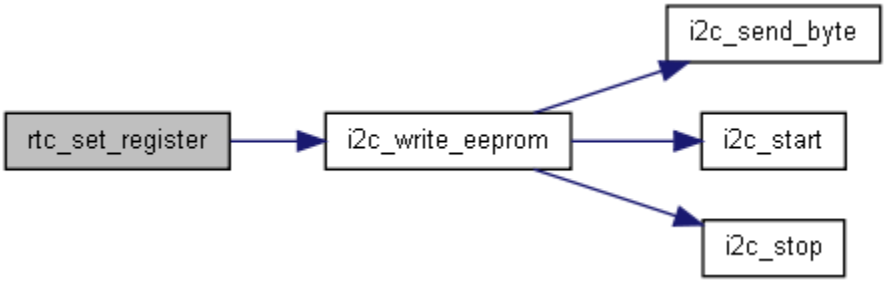

**void rtc_set_register (uns8 *reg*, uns8 *data*)**

Definition at line 82 of file m41t81s.c.

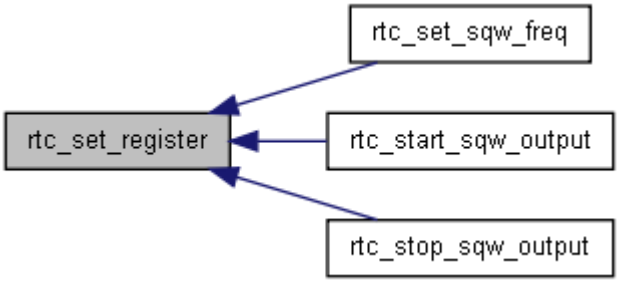
References i2c_write_eeprom(), and m41t81s_device_addr.

Referenced by rtc_set_sqw_freq(), rtc_start_sqw_output(), and rtc_stop_sqw_output().

Here is the call graph for this function:




Here is the caller graph for this function:




**void rtc_set_seconds (uns8 *seconds*)**

Set the seconds register in the m41t81s.

Changes the seconds in the ds1307.

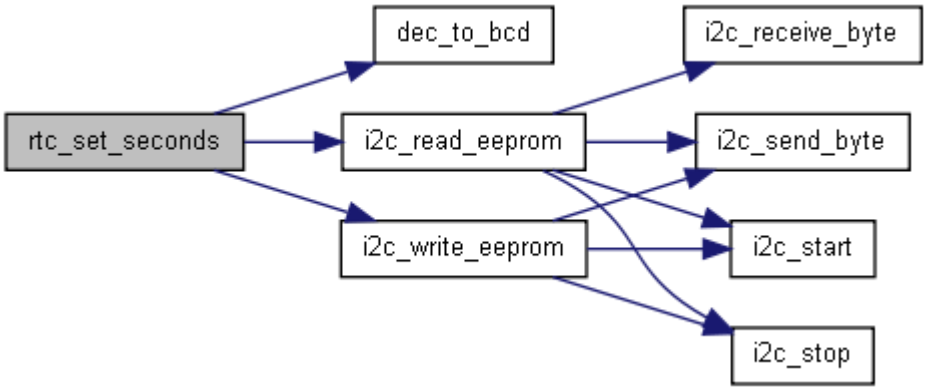
**Parameters:**

*seconds* Value to set seconds to

Definition at line 111 of file m41t81s.c.

References dec_to_bcd(), ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:




**void rtc_set_sqw_freq (uns8 *freq*)**

Use one of the following self explanitory defines:

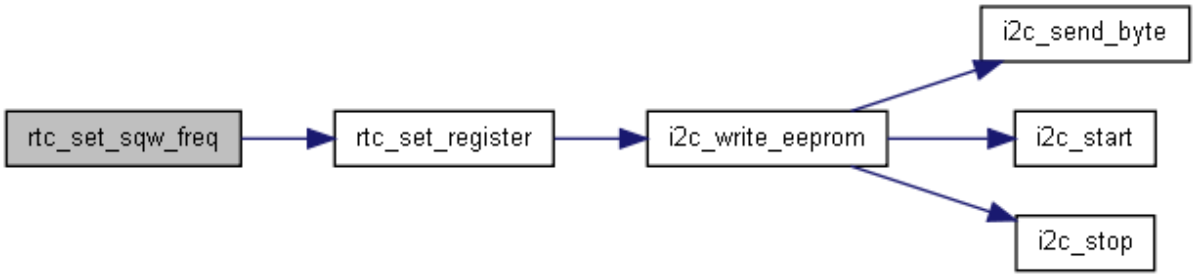
rtc_sqw_freq_32768Hz rtc_sqw_freq_8192Hz rtc_sqw_freq_4096Hz rtc_sqw_freq_2048Hz rtc_sqw_freq_1024Hz rtc_sqw_freq_512Hz rtc_sqw_freq_256Hz rtc_sqw_freq_128Hz rtc_sqw_freq_64Hz rtc_sqw_freq_32Hz rtc_sqw_freq_16Hz rtc_sqw_freq_8Hz rtc_sqw_freq_4Hz rtc_sqw_freq_2Hz rtc_sqw_freq_1Hz

Note that on the m41t81s 18384Hz is not available.

Definition at line 125 of file m41t81s.c.

References m41t81s_sqw_reg, and rtc_set_register().

Here is the call graph for this function:



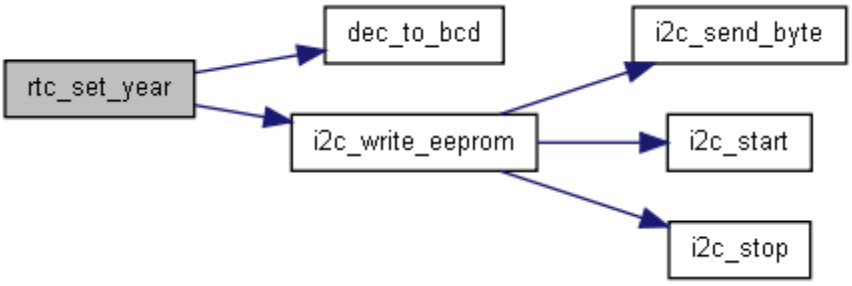

**void rtc_set_year (uns8 *year*)**

Changes the year in the m41t81s.

Definition at line 98 of file m41t81s.c.

References dec_to_bcd(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_year_reg.

Here is the call graph for this function:




**void rtc_setup_io ()**

Setup ports and pins for use in the m41t81s.

Calls i2c_setup() to configure ports and pins ready for use

Definition at line 143 of file m41t81s.c.

References i2c_setup_io().

Here is the call graph for this function:


**void rtc_start_clock ()**

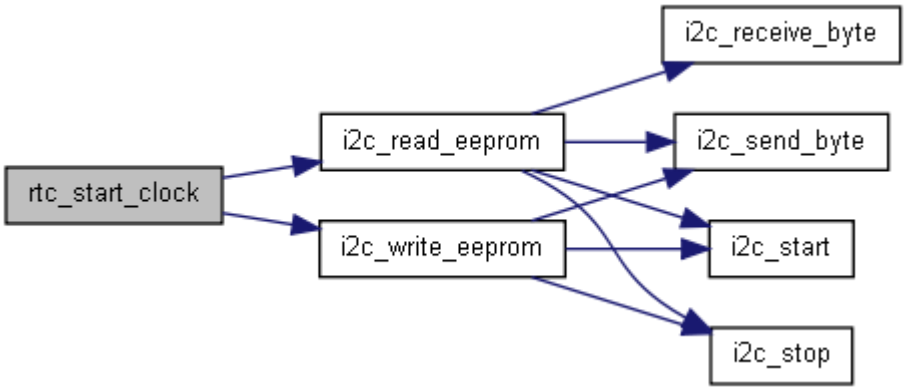
Starts the clock in the m41t81s.

Resume time in the ds1307

Definition at line 91 of file m41t81s.c.

References ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_alarm_hour_reg, m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:



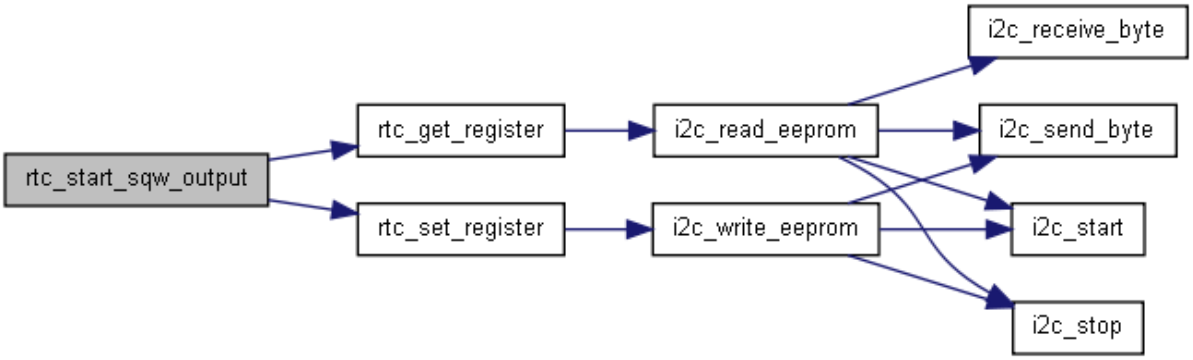

**void rtc_start_sqw_output ()**

Outputs desired frequency on the SQW output pin. To set the frequency, see rtc_set_sqw_freq(uns8 freq);

Definition at line 131 of file m41t81s.c.

References m41t81s_alarm_month_reg, rtc_get_register(), and rtc_set_register().

Here is the call graph for this function:




**void rtc_stop_clock ()**

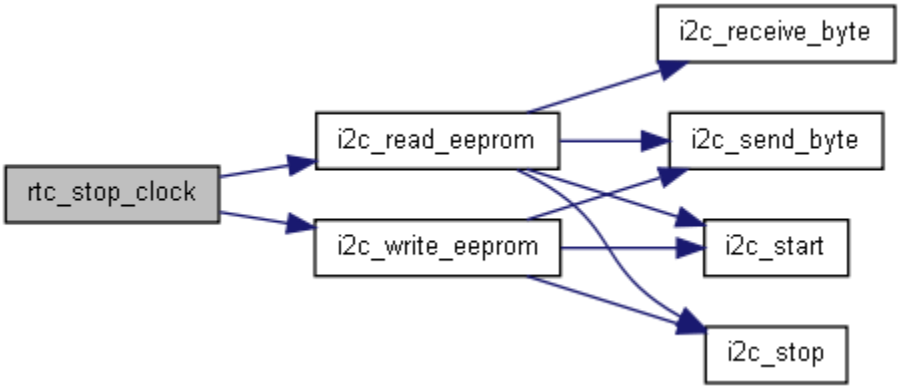
Stop the clock in the m41t81s.

Pauses time in the ds1307

Definition at line 87 of file m41t81s.c.

References ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:



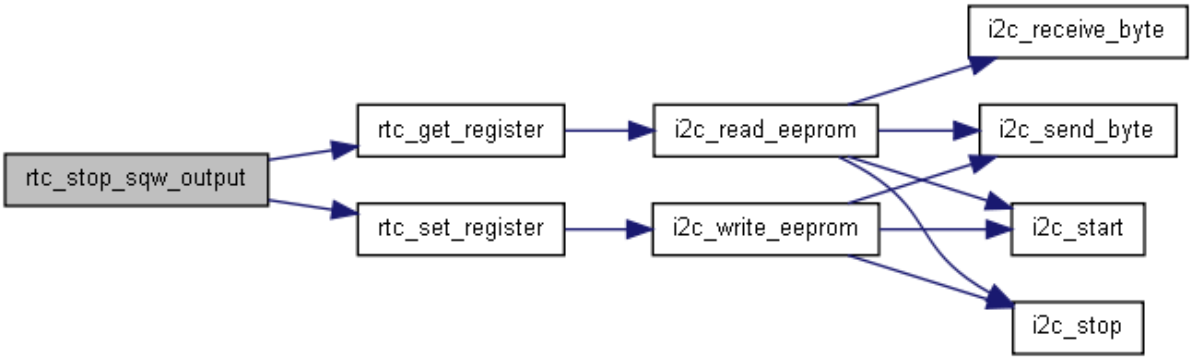

**void rtc_stop_sqw_output ()**

Stops square wave output

Definition at line 136 of file m41t81s.c.

References m41t81s_alarm_month_reg, rtc_get_register(), and rtc_set_register().

Here is the call graph for this function:



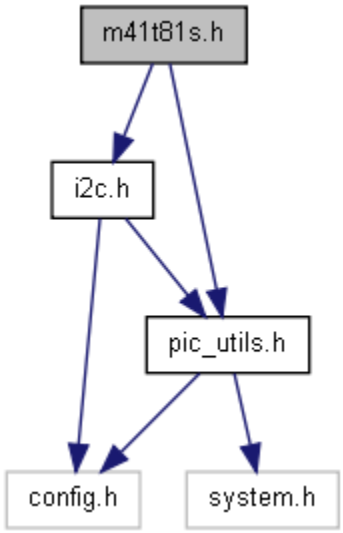

---

## m41t81s.h File Reference

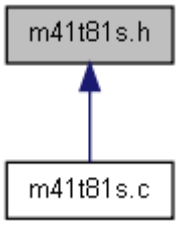
Routines for communicating with the m41t81s real time clock.

```
#include "i2c.h"
#include "pic_utils.h"
```

Include dependency graph for m41t81s.h:




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



## Defines

- #define __m41t81s_h defined
- #define m41t81s_alarm_date_reg 0x0B
- #define m41t81s_alarm_hour_reg 0x0C
- #define m41t81s_alarm_min_reg 0x0D
- #define m41t81s_alarm_month_reg 0x0A
- #define m41t81s_alarm_seconds_reg 0x0E
- #define m41t81s_calibration_reg 0x08
- #define m41t81s_date_reg 0x05
- #define m41t81s_device_addr 0xD0
- #define m41t81s_dow_reg 0x04
- #define m41t81s_flags_reg 0x0F
- #define m41t81s_hours_reg 0x03
- #define m41t81s_minutes_reg 0x02
- #define m41t81s_month_reg 0x06
- #define m41t81s_part_seconds_reg 0x00
- #define m41t81s_reserved1_reg 0x10
- #define m41t81s_reserved2_reg 0x11
- #define m41t81s_reserved3_reg 0x12
- #define m41t81s_seconds_reg 0x01
- #define m41t81s_sqw_reg 0x13
- #define m41t81s_watchdog_reg 0x09
- #define m41t81s_year_reg 0x07
- #define rtc_setup() rtc_setup_io()
- #define rtc_sqw_freq_1024Hz 0b00000101
- #define rtc_sqw_freq_128Hz 0b00001000
- #define rtc_sqw_freq_16Hz 0b00001011
- #define rtc_sqw_freq_1Hz 0b00001111
- #define rtc_sqw_freq_2048Hz 0b00000100
- #define rtc_sqw_freq_256Hz 0b00000111
- #define rtc_sqw_freq_2Hz 0b00001110
- #define rtc_sqw_freq_32768Hz 0b00000001
- #define rtc_sqw_freq_32Hz 0b00001010
- #define rtc_sqw_freq_4096Hz 0b00000011
- #define rtc_sqw_freq_4Hz 0b00001101
- #define rtc_sqw_freq_512Hz 0b00000110
- #define rtc_sqw_freq_64Hz 0b00001001
- #define rtc_sqw_freq_8192Hz 0b00000010
- #define rtc_sqw_freq_8Hz 0b00001100

## Functions

- uns8 rtc_get_date ()
- *Get the date register from the m41t81s.* uns8 rtc_get_dow ()
- *Get the day register from the m41t81s.* uns8 rtc_get_hours ()
- *Get the decoded hours register from the m41t81s.* uns8 rtc_get_minutes ()
- *Get the decoded minutes register from the m41t81s.* uns8 rtc_get_month ()
- *Get the month register from the m41t81s.* uns8 rtc_get_register (uns8 reg)
- uns8 rtc_get_seconds ()
- *Get the decoded seconds register from the m41t81s.* uns8 rtc_get_year ()
- *Get the year register from the m41t81s.* uns8 rtc_set_config (uns8 config)
- *Set the config register in the m41t81s.* void rtc_set_date (uns8 date)
- *Set the date register from the m41t81s.* void rtc_set_day (uns8 day)

- *Set the day of the week register from the m41t81s.* void rtc_set_hours (uns8 hours)
- *Set the hours register in the m41t81s.* void rtc_set_minutes (uns8 minutes)
- *Set the minutes register from the m41t81s.* void rtc_set_month (uns8 month)
- *Set the month register in the m41t81s.* void rtc_set_register (uns8 reg, uns8 data)
- void rtc_set_seconds (uns8 seconds)
- *Set the seconds register in the m41t81s.* void rtc_set_sqw_freq (uns8 freq)
- *Set the frequence of the square wave output pin.* void rtc_set_year (uns8 year)
- *Set the year register from the m41t81s.* void rtc_setup_io ()
- *Setup ports and pins for use in the m41t81s.* void rtc_start_clock ()
- *Starts the clock in the m41t81s.* void rtc_start_sqw_output ()
- *Start pulsing on square wave output pin.* void rtc_stop_clock ()
- *Stop the clock in the m41t81s.* void rtc_stop_sqw_output ()

*Stop pulsing on square wave output pin.*

---

## Detailed Description

Definition in file m41t81s.h.

---

## Define Documentation

### #define __m41t81s_h defined

Definition at line 44 of file m41t81s.h.

### #define m41t81s_alarm_date_reg 0x0B

m41t81s alarm date register (D7=RPT4, D6=RPT5, D5-D4=ABE, D4=AL 10M, D3-D0=Alarm month)

Definition at line 95 of file m41t81s.h.

### #define m41t81s_alarm_hour_reg 0x0C

m41t81s alarm hour register (D7=RPT3, D6=HT, D5-D4=Alarm 10 Hour, D3-D0=Alarm Hour)

Definition at line 97 of file m41t81s.h.

Referenced by rtc_start_clock().

### #define m41t81s_alarm_min_reg 0x0D

m41t81s alarm min register (D7=RPT2, D6-D4=Alarm 10 Minutes, D3-D0=Alarm Minutes)

Definition at line 99 of file m41t81s.h.

### #define m41t81s_alarm_month_reg 0x0A

m41t81s alarm month register (D7=AFE, D6=SQWE, D5=ABE, D4=AL 10M, D3-D0=Alarm month)

Definition at line 93 of file m41t81s.h.

Referenced by rtc_start_sqw_output(), and rtc_stop_sqw_output().

**#define m41t81s_alarm_seconds_reg 0x0E**

m41t81s alarm seconds register (D7=RPT1, D6-D4=Alarm 10 Seconds, D3-D0=Alarm Seconds)

Definition at line 101 of file m41t81s.h.

**#define m41t81s_calibration_reg 0x08**

m41t81s calibration register (D7=OUT, D6=FT, D5=S D4-D0=Calibration)

Definition at line 89 of file m41t81s.h.

**#define m41t81s_date_reg 0x05**

m41t81s date in month register

Definition at line 83 of file m41t81s.h.

Referenced by rtc_get_date(), and rtc_set_date().

**#define m41t81s_device_addr 0xD0**

The m41t81s device address

Definition at line 70 of file m41t81s.h.

Referenced by rtc_get_date(), rtc_get_dow(), rtc_get_hours(), rtc_get_minutes(), rtc_get_month(), rtc_get_register(), rtc_get_seconds(), rtc_get_year(), rtc_set_date(), rtc_set_day(), rtc_set_hours(), rtc_set_minutes(), rtc_set_month(), rtc_set_register(), rtc_set_seconds(), rtc_set_year(), rtc_start_clock(), and rtc_stop_clock().

**#define m41t81s_dow_reg 0x04**

m41t81s day of week register

Definition at line 81 of file m41t81s.h.

Referenced by rtc_get_dow(), and rtc_set_day().

**#define m41t81s_flags_reg 0x0F**

m41t81s flags register (D7=WDF, D6=AF, D5=0, D4=BL, D3=0, D2=OF, D1=0, D0=0 -D4=Alarm 10 Seconds, D3-D0=Alarm Seconds)

Definition at line 103 of file m41t81s.h.

**#define m41t81s_hours_reg 0x03**

m41t81s hours register (D7=CEB, D6=CB)

Definition at line 79 of file m41t81s.h.

Referenced by rtc_get_hours(), and rtc_set_hours().

**#define m41t81s_minutes_reg 0x02**

m41t81s minutes register

Definition at line 77 of file m41t81s.h.

Referenced by rtc_get_minutes(), and rtc_set_minutes().

**#define m41t81s_month_reg 0x06**

m41t81s month register

Definition at line 85 of file m41t81s.h.

Referenced by rtc_get_month(), and rtc_set_month().

**#define m41t81s_part_seconds_reg 0x00**

m41t81s tenths and hundreths of seconds register

Definition at line 73 of file m41t81s.h.

**#define m41t81s_reserved1_reg 0x10**

m41t81s reserved register

Definition at line 105 of file m41t81s.h.

**#define m41t81s_reserved2_reg 0x11**

m41t81s reserved register

Definition at line 107 of file m41t81s.h.

**#define m41t81s_reserved3_reg 0x12**

m41t81s reserved register

Definition at line 109 of file m41t81s.h.

**#define m41t81s_seconds_reg 0x01**

m41t81s seconds register (D7=ST)

Definition at line 75 of file m41t81s.h.

Referenced by rtc_get_seconds(), rtc_set_seconds(), rtc_start_clock(), and rtc_stop_clock().

**#define m41t81s_sqw_reg 0x13**

m41t81s SQW register (D7-D4=RS3-RS0)

Definition at line 111 of file m41t81s.h.

Referenced by rtc_set_sqw_freq().

**#define m41t81s_watchdog_reg 0x09**

m41t81s watchdog register (D7=OFIE, D6-D2=BMB, D1-D0=RB

Definition at line 91 of file m41t81s.h.

**#define m41t81s_year_reg 0x07**

m41t81s year register

Definition at line 87 of file m41t81s.h.

Referenced by rtc_get_year(), and rtc_set_year().

**#define rtc_setup() rtc_setup_io()**

Definition at line 356 of file m41t81s.h.

**#define rtc_sqw_freq_1024Hz 0b00000101**

Definition at line 56 of file m41t81s.h.

**#define rtc_sqw_freq_128Hz 0b00001000**

Definition at line 59 of file m41t81s.h.

**#define rtc_sqw_freq_16Hz 0b00001011**

Definition at line 62 of file m41t81s.h.

**#define rtc_sqw_freq_1Hz 0b00001111**

Definition at line 66 of file m41t81s.h.

**#define rtc_sqw_freq_2048Hz 0b00000100**

Definition at line 55 of file m41t81s.h.

**#define rtc_sqw_freq_256Hz 0b00000111**

Definition at line 58 of file m41t81s.h.

**#define rtc_sqw_freq_2Hz 0b00001110**

Definition at line 65 of file m41t81s.h.

**#define rtc_sqw_freq_32768Hz 0b00000001**

Definition at line 51 of file m41t81s.h.

**#define rtc_sqw_freq_32Hz 0b00001010**

Definition at line 61 of file m41t81s.h.

**#define rtc_sqw_freq_4096Hz 0b00000011**

Definition at line 54 of file m41t81s.h.

**#define rtc_sqw_freq_4Hz 0b00001101**

Definition at line 64 of file m41t81s.h.

**#define rtc_sqw_freq_512Hz 0b00000110**

Definition at line 57 of file m41t81s.h.

**#define rtc_sqw_freq_64Hz 0b00001001**

Definition at line 60 of file m41t81s.h.

**#define rtc_sqw_freq_8192Hz 0b00000010**

Definition at line 53 of file m41t81s.h.

**#define rtc_sqw_freq_8Hz 0b00001100**

Definition at line 63 of file m41t81s.h.

---

## Function Documentation

**uns8 rtc_get_date ()**

Returns the date in month from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range 1 through 28/29/30/31 depending on month

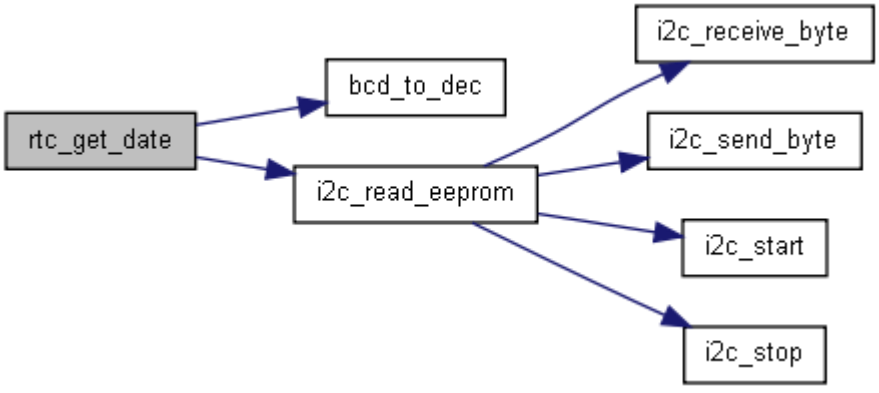
Get the date register from the m41t81s.

Returns the date in month from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 28/29/30/31 depending on month

Definition at line 65 of file ds1307.c.

References bcd_to_dec(), ds1307_date_register, ds1307_device, i2c_read_eeprom(), m41t81s_date_reg, and m41t81s_device_addr.

Here is the call graph for this function:



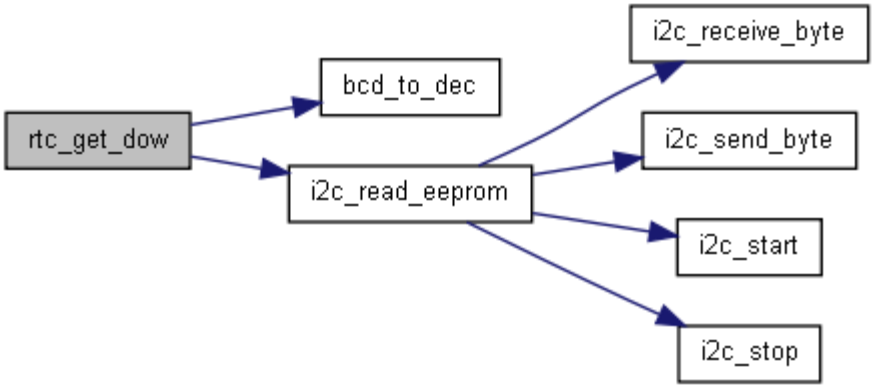

**uns8 rtc_get_dow ()**

Returns the day of the week from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range - 1 through 7

Definition at line 62 of file m41t81s.c.

References bcd_to_dec(), i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_dow_reg.

Here is the call graph for this function:




### uns8 rtc_get_hours ()

Returns hour from the m41t81s. The result is coverted to decimal from BCD and is ready to use. These routines assume the m41t81s is running in 24 hour mode. Range - 0 through 23

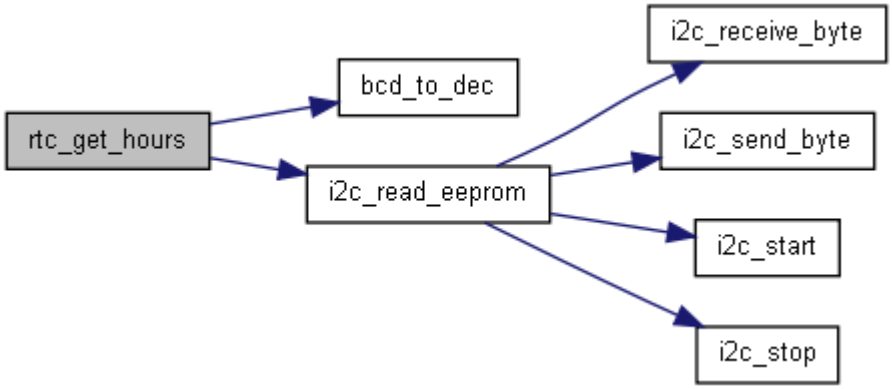
Get the decoded hours register from the m41t81s.

Returns hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. These routines assume the ds1307 is running in 24 hour mode. Range - 0 through 23

Definition at line 50 of file ds1307.c.

References bcd_to_dec(), ds1307_device, ds1307_hours_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_hours_reg.

Here is the call graph for this function:




### uns8 rtc_get_minutes ()

Returns the number of minutes past the hour from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

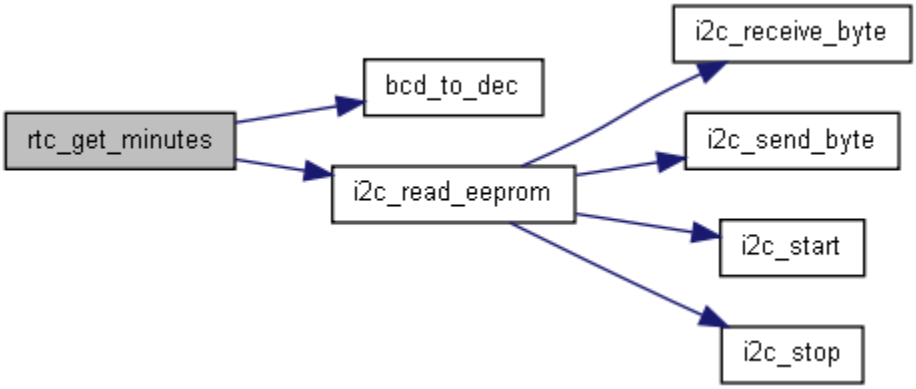
Get the decoded minutes register from the m41t81s.

Returns the number of minutes past the hour from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 47 of file ds1307.c.

References bcd_to_dec(), ds1307_device, ds1307_minutes_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_minutes_reg.

Here is the call graph for this function:




**uns8 rtc_get_month ()**

Returns the month of the year from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range 1 through 12

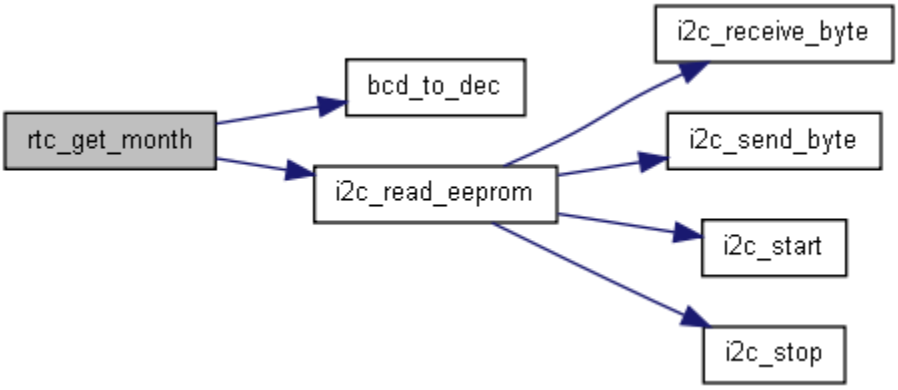
Get the month register from the m41t81s.

Returns the month of the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 1 through 12

Definition at line 69 of file ds1307.c.

References bcd_to_dec(), ds1307_device, ds1307_month_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_month_reg.

Here is the call graph for this function:



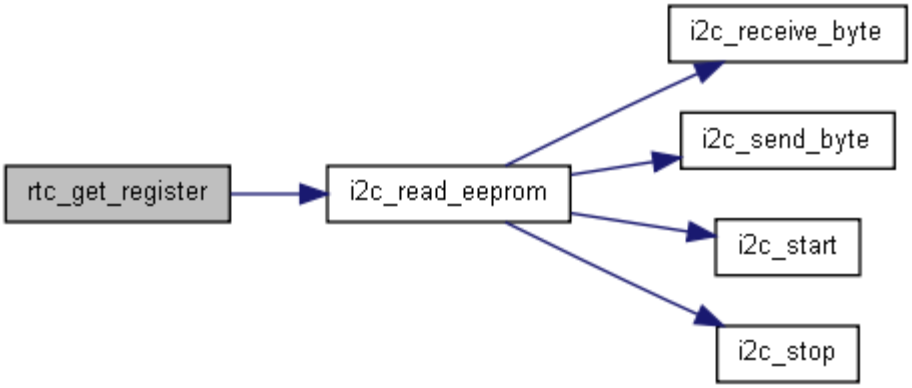

**uns8 rtc_get_register (uns8 *reg*)**

Definition at line 78 of file m41t81s.c.

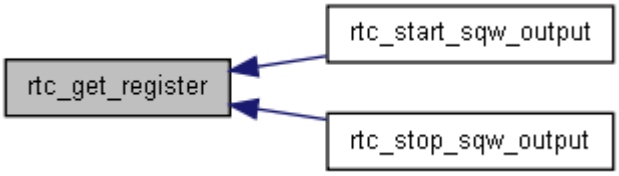
References i2c_read_eeprom(), and m41t81s_device_addr.

Referenced by rtc_start_sqw_output(), and rtc_stop_sqw_output().

Here is the call graph for this function:




Here is the caller graph for this function:




**uns8 rtc_get_seconds ()**

Returns seconds from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

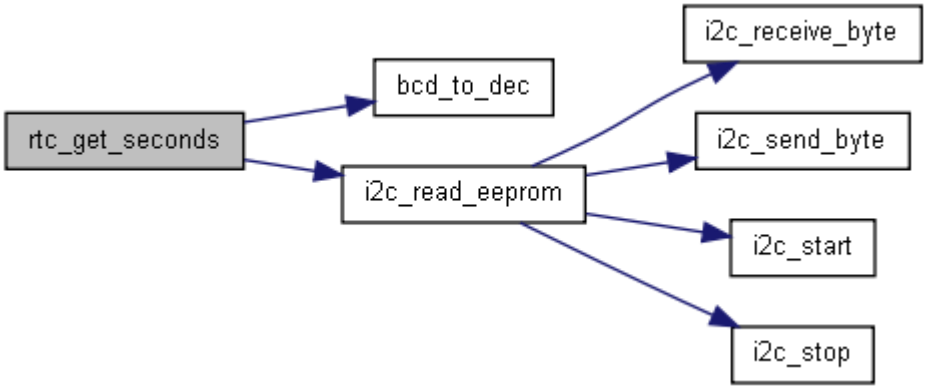
Get the decoded seconds register from the m41t81s.

Returns seconds from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range - 0 through 59

Definition at line 57 of file ds1307.c.

References bcd_to_dec(), ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:




**uns8 rtc_get_year ()**

Returns the year from the m41t81s. The result is coverted to decimal from BCD and is ready to use. Range 0 through 99

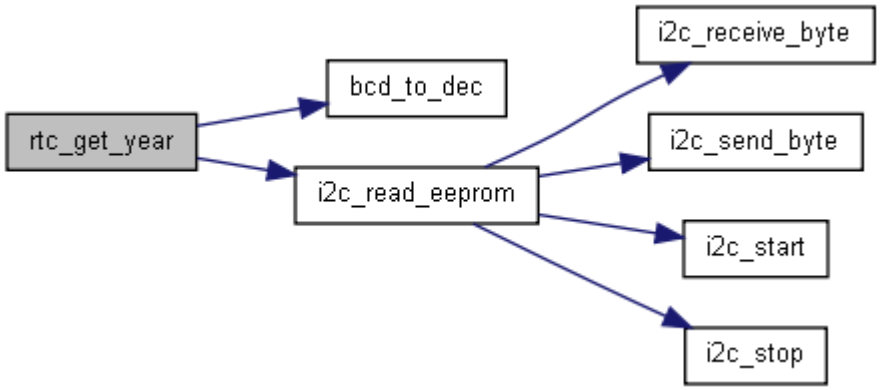
Get the year register from the m41t81s.

Returns the year from the ds1307. The result is coverted to decimal from BCD and is ready to use. Range 0 through 99

Definition at line 72 of file ds1307.c.

References bcd_to_dec(), ds1307_device, ds1307_year_register, i2c_read_eeprom(), m41t81s_device_addr, and m41t81s_year_reg.

Here is the call graph for this function:




**uns8 rtc_set_config (uns8 *config*)**

**Parameters:**

*config* Value to set the config register to

Set the config register in the m41t81s.

Sets the config register in the ds1307.

Bit 7 - Out - Value on SQWE pin if not outputting square wave Bit 6 - 0 Bit 5 - 0 Bit 4 - SQWE - Enable square wave output Bit 3 - 0 Bit 2 - 0 Bit 1 - RS1 Bit 0 - RS0

RS1/0 determin the speed of the square wave output. Set to 0/0 for 1 Hz.

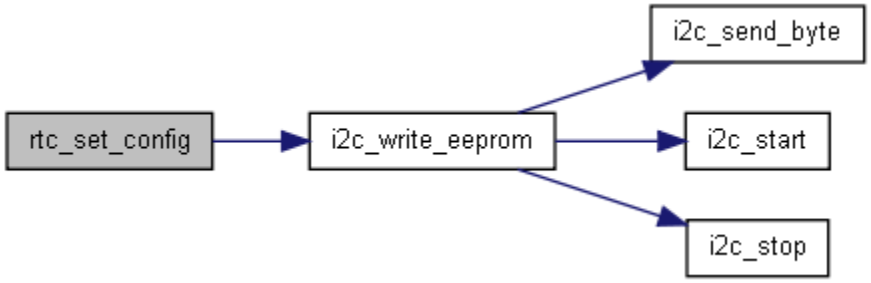
**Parameters:**

*config* Value to set the config register to

Definition at line 80 of file ds1307.c.

References ds1307_control_register, ds1307_device, and i2c_write_eeprom().

Here is the call graph for this function:



### void rtc_set_date (uns8 *date*)

Changes the date in the m41t81s.

**Parameters:**

*seconds* Value to set date to

Set the date register from the m41t81s.

Changes the date in the ds1307.

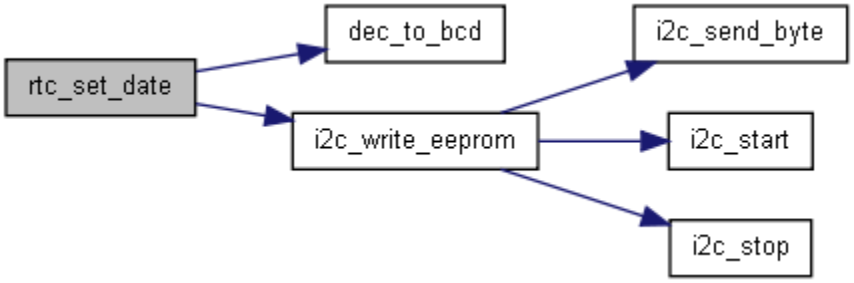
**Parameters:**

*seconds* Value to set date to

Definition at line 102 of file ds1307.c.

References dec_to_bcd(), ds1307_date_register, ds1307_device, i2c_write_eeprom(), m41t81s_date_reg, and m41t81s_device_addr.

Here is the call graph for this function:




### void rtc_set_day (uns8 *day*)

Changes the day of the week in the m41t81s.

**Parameters:**

*seconds* Value to set day to

Set the day of the week register from the m41t81s.

Changes the day of the week in the ds1307.

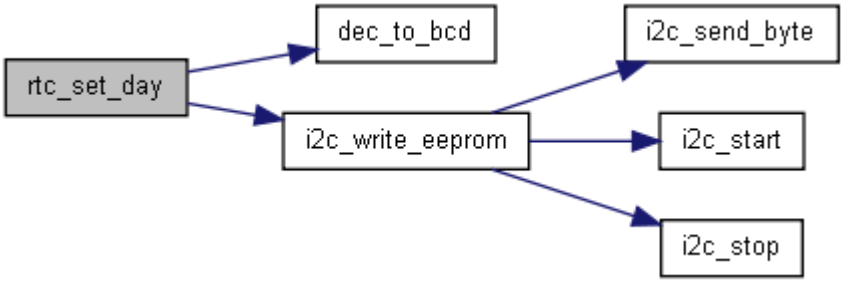
**Parameters:**

*seconds* Value to set day to

Definition at line 99 of file ds1307.c.

References dec_to_bcd(), ds1307_day_register, ds1307_device, i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_dow_reg.

Here is the call graph for this function:



**void rtc_set_hours (uns8 *hours*)**

Changes the hours in the m41t81s. Forces the m41t81s into 24 hour mode.

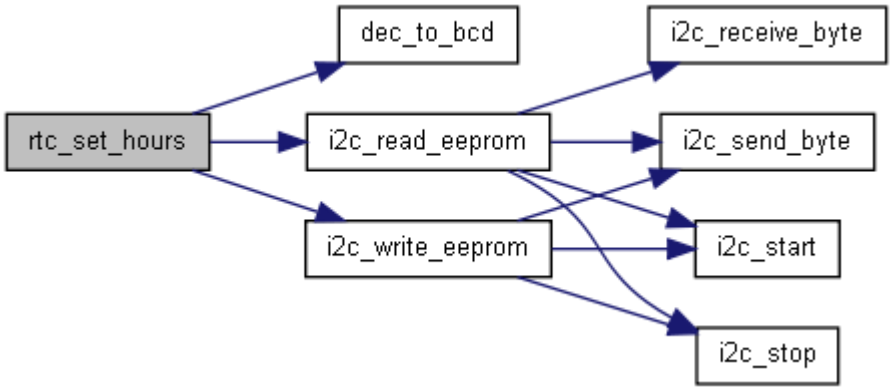
Set the hours register in the m41t81s.

Changes the hours in the ds1307. Forces the ds1307 into 24 hour mode.

Definition at line 110 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_hours_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_hours_reg.

Here is the call graph for this function:




**void rtc_set_minutes (uns8 *minutes*)**

Changes the minutes in the m41t81s.

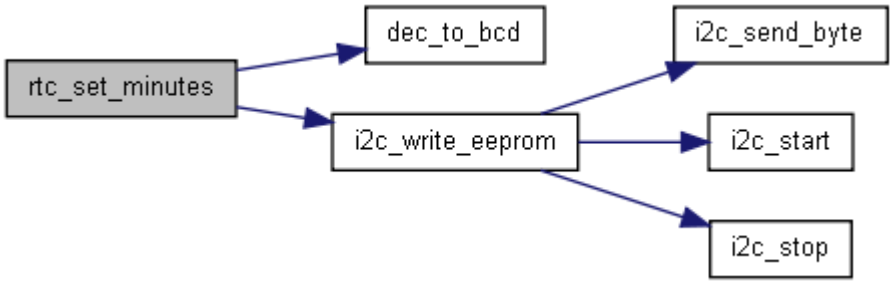
**Parameters:**

*seconds* Value to set minutes to

Definition at line 101 of file m41t81s.c.

References dec_to_bcd(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_minutes_reg.

Here is the call graph for this function:




**void rtc_set_month (uns8 *month*)**

Changes the month in the m41t81s.

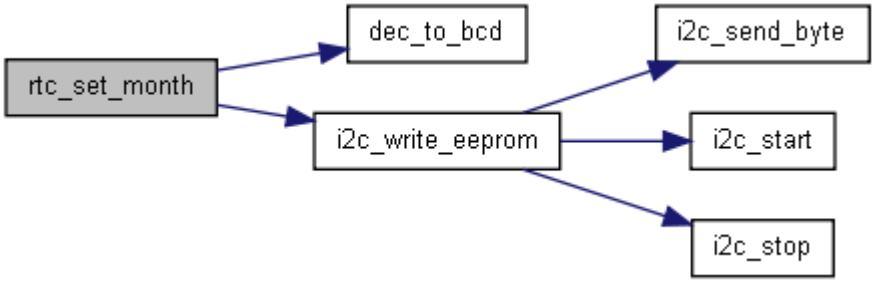
Set the month register in the m41t81s.

Changes the month in the ds1307.

Definition at line 115 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_month_register, i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_month_reg.

Here is the call graph for this function:



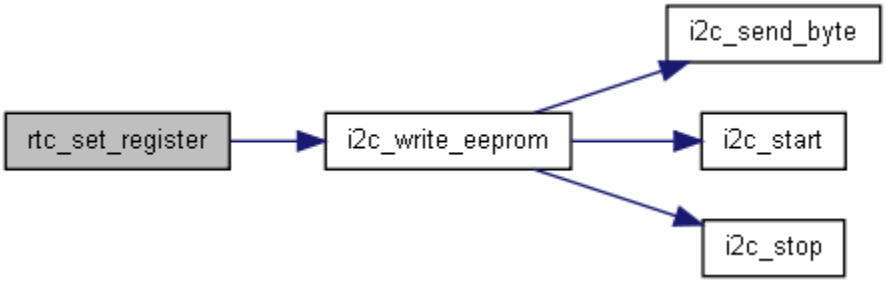

**void rtc_set_register (uns8 *reg*, uns8 *data*)**

Definition at line 82 of file m41t81s.c.

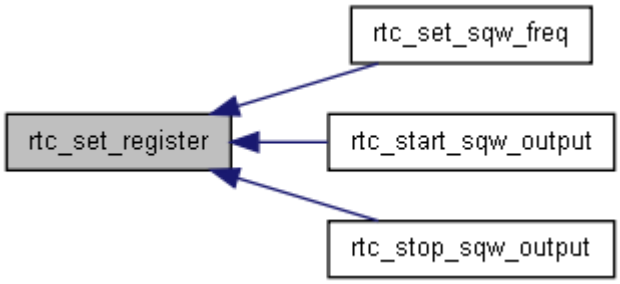
References i2c_write_eeprom(), and m41t81s_device_addr.

Referenced by rtc_set_sqw_freq(), rtc_start_sqw_output(), and rtc_stop_sqw_output().

Here is the call graph for this function:




Here is the caller graph for this function:




**void rtc_set_seconds (uns8 *seconds*)**

Changes the seconds in the m41t81s.

**Parameters:**

*seconds* Value to set seconds to

Set the seconds register in the m41t81s.

Changes the seconds in the ds1307.

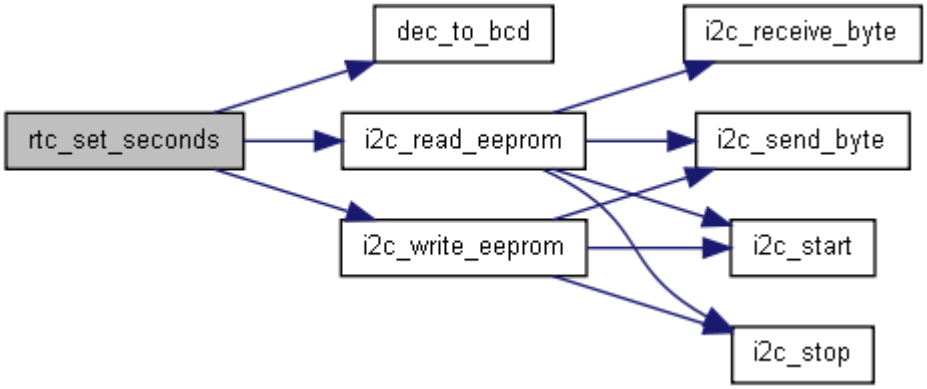
**Parameters:**

*seconds* Value to set seconds to

Definition at line 106 of file ds1307.c.

References dec_to_bcd(), ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:




**void rtc_set_sqw_freq (uns8 *freq*)**

Use one of the following self explanitory defines:

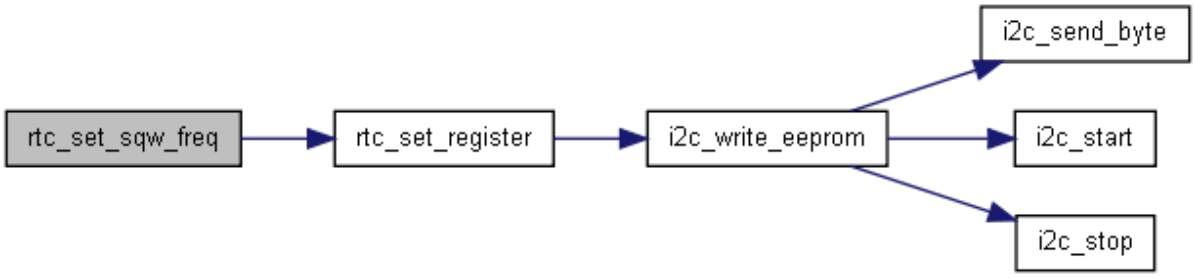
rtc_sqw_freq_32768Hz rtc_sqw_freq_8192Hz rtc_sqw_freq_4096Hz rtc_sqw_freq_2048Hz rtc_sqw_freq_1024Hz rtc_sqw_freq_512Hz rtc_sqw_freq_256Hz rtc_sqw_freq_128Hz rtc_sqw_freq_64Hz rtc_sqw_freq_32Hz rtc_sqw_freq_16Hz rtc_sqw_freq_8Hz rtc_sqw_freq_4Hz rtc_sqw_freq_2Hz rtc_sqw_freq_1Hz

Note that on the m41t81s 18384Hz is not available.

Definition at line 125 of file m41t81s.c.

References m41t81s_sqw_reg, and rtc_set_register().

Here is the call graph for this function:



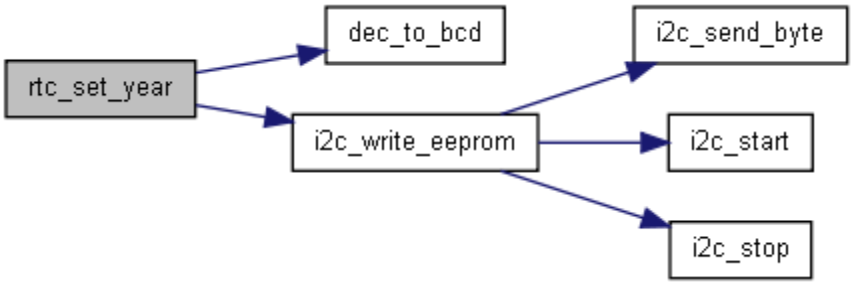

**void rtc_set_year (uns8 *year*)**

Changes the year in the m41t81s.

Definition at line 98 of file m41t81s.c.

References dec_to_bcd(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_year_reg.

Here is the call graph for this function:




**void rtc_setup_io ()**

Calls i2c_setup() to configure ports and pins ready for use

Setup ports and pins for use in the m41t81s.

Calls i2c_setup() to configure ports and pins ready for use

Definition at line 119 of file ds1307.c.

References i2c_setup_io().

Here is the call graph for this function:


**void rtc_start_clock ()**

Resume time in the m41t81s. Also resumes the paused time that happens upon non-battery backup start up (which allows you to read the time before "resuming" so you know how long the clock has been running on battery back up).

If you want to do this, read the time etc before calling rtc_start_clock();

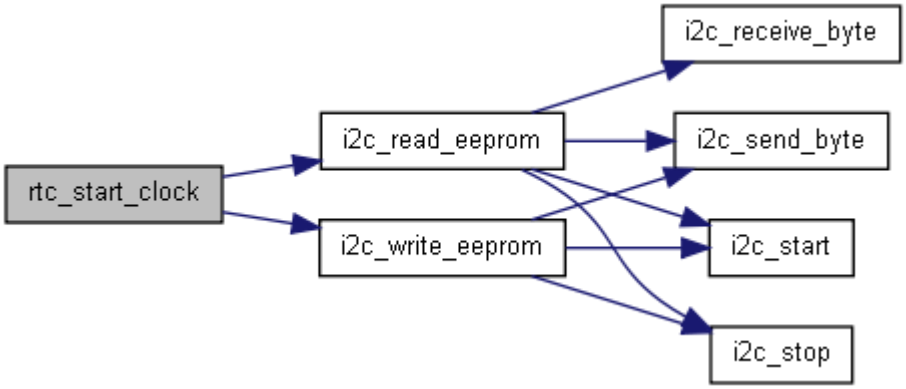
Starts the clock in the m41t81s.

Resume time in the ds1307

Definition at line 89 of file ds1307.c.

References ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_alarm_hour_reg, m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:



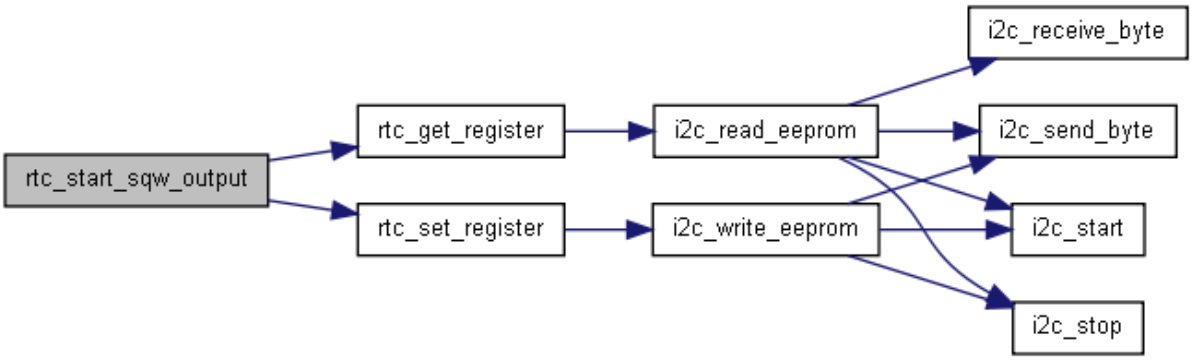
**void rtc_start_sqw_output ()**

Outputs desired frequency on the SQW output pin. To set the frequency, see rtc_set_sqw_freq(uns8 freq);

Definition at line 131 of file m41t81s.c.

References m41t81s_alarm_month_reg, rtc_get_register(), and rtc_set_register().

Here is the call graph for this function:




**void rtc_stop_clock ()**

Pauses time in the m41t81s

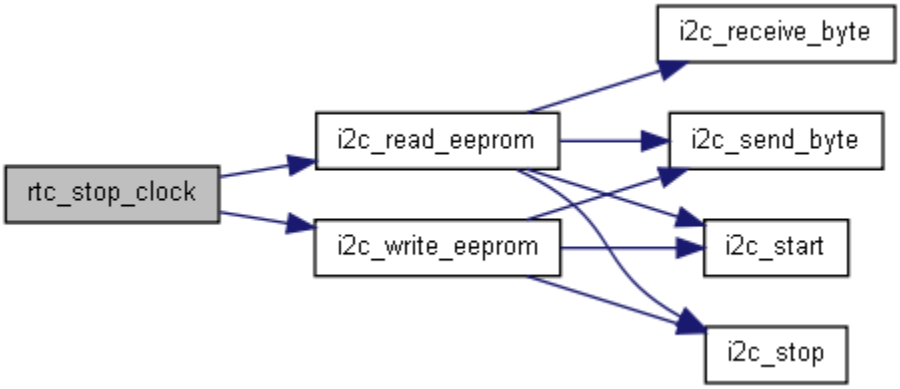
Stop the clock in the m41t81s.

Pauses time in the ds1307

Definition at line 85 of file ds1307.c.

References ds1307_device, ds1307_seconds_register, i2c_read_eeprom(), i2c_write_eeprom(), m41t81s_device_addr, and m41t81s_seconds_reg.

Here is the call graph for this function:



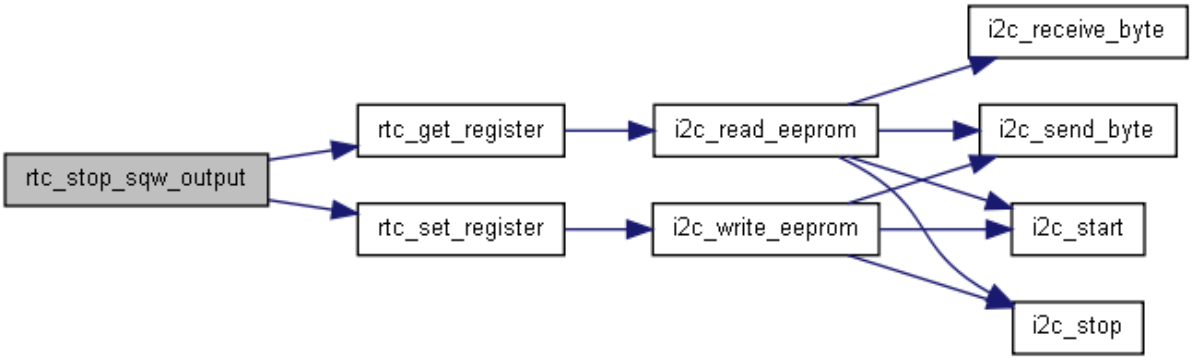

**void rtc_stop_sqw_output ()**

Stops square wave output

Definition at line 136 of file m41t81s.c.

References m41t81s_alarm_month_reg, rtc_get_register(), and rtc_set_register().

Here is the call graph for this function:

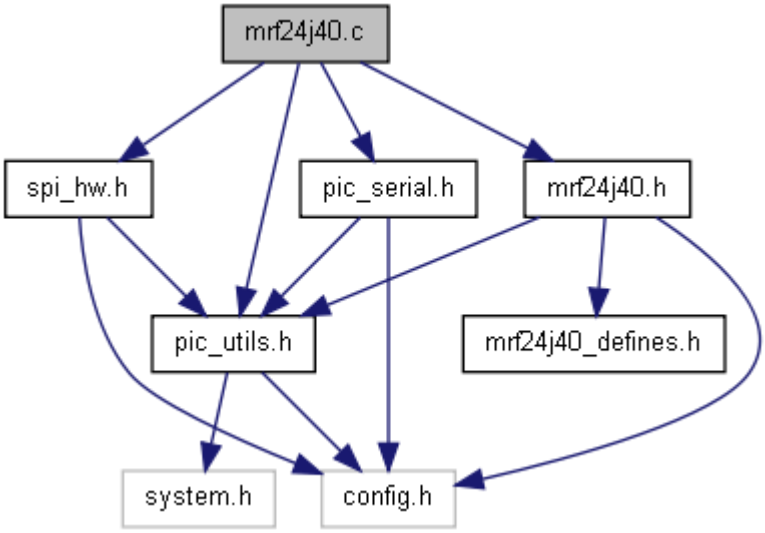



---

## mrf24j40.c File Reference

```
#include "mrf24j40.h"
#include "pic_utils.h"
#include "spi_hw.h"
#include "pic_serial.h"
```

Include dependency graph for mrf24j40.c:




### Functions

- void mrf24h40_pan_association_requested ()
- void mrf24j40_active_channel_scan ()
- void mrf24j40_associate_to_pan ()
- void mrf24j40_flush_receive_buffer ()
- *Flush receive buffer of mrf24j40.* void mrf24j40_handle_isr ()
- *Interrupt service routine for mrf24j40 chip.* void mrf24j40_init ()
- *Initialises mrf24j40 chip ready for use.* void mrf24j40_init_coordinator ()
- uns8 mrf24j40_long_addr_read (uns16 addr)

- *Read data from long address of memory location in mrf24j40.* void mrf24j40_long_addr_write (uns16 addr, uns8 data)
- *Write data to long address memory location.* void mrf24j40_orphan_channel_scan ()
- void mrf24j40_realign_pan ()
- uns8 mrf24j40_receive (uns8 *data, uns8 bytes_to_receive)
- *Pull received data from buffer.* uns8 mrf24j40_scan_for_lowest_channel_ed ()
- *Scan all channels for lowest RF energy.* void mrf24j40_set_channel (uns8 channel)
- *Change channels.* void mrf24j40_set_extended_address (uns8 *_extended_address)
- *Set extended address.* void mrf24j40_set_pan_id (uns16 _pan_id)
- *Set PAN id.* void mrf24j40_set_short_address (uns16 _short_address)
- *Set short address.* void mrf24j40_setup_io ()
- *Setup ports/pins as inputs/outputs ready for use.* uns8 mrf24j40_short_addr_read (uns8 addr)
- *Read data from short address of memory location in mrf24j40.* void mrf24j40_short_addr_write (uns8 addr, uns8 data)
- *Write data to short address memory location.* void mrf24j40_start_pan ()
- void mrf24j40_transmit (uns8 *data, uns8 bytes_to_transmit)
- *Transmit raw data.* void mrf24j40_transmit_to_extended_address (uns8 frame_type, uns16 dest_pan_id, uns8 *dest_extended_address, uns8 *data, uns8 data_length, uns8 ack)
- *Transmit packet to extended address.* void mrf24j40_transmit_to_short_address (uns8 frame_type, uns16 dest_pan_id, uns16 dest_short_address, uns8 *data, uns8 bytes_to_transmit, uns8 ack)

## *Transmit packet to short address.* Variables

- uns8 current_channel = 0
- uns8 data_sequence_number
- uns8 extended_address [8] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff}
- uns16 pan_id = 0xffff
- uns16 short_address = 0xffff

---

## Function Documentation

### void mrf24h40_pan_association_requested ()

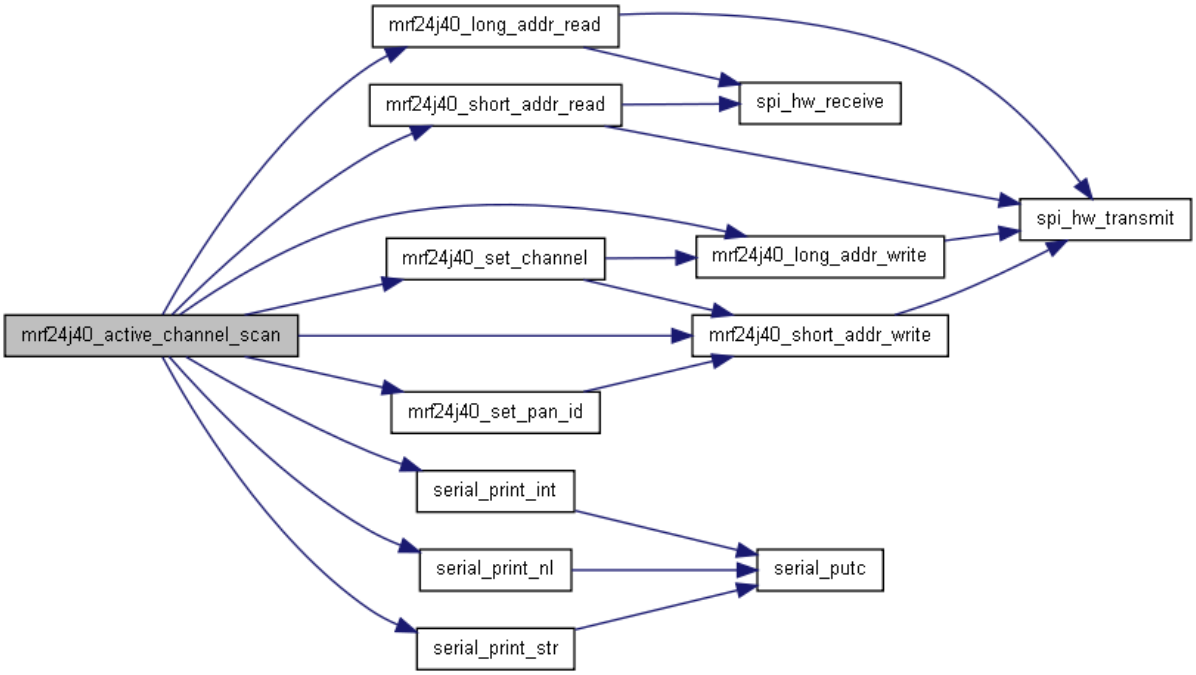
Definition at line 242 of file mrf24j40.c.

### void mrf24j40_active_channel_scan ()

Definition at line 134 of file mrf24j40.c.

References BBREG6, BBREG6_RSSIMODE1, BBREG6_RSSIRDY, channel, data_sequence_number, mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_set_channel(), mrf24j40_set_pan_id(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, pan_id, RSSI, RXFLUSH, RXFLUSH_BCNONLY, serial_print_int(), serial_print_nl(), serial_print_str(), uns16, and uns8.

Here is the call graph for this function:




**void mrf24j40_associate_to_pan ()**

Definition at line 236 of file mrf24j40.c.

**void mrf24j40_flush_receive_buffer ()**

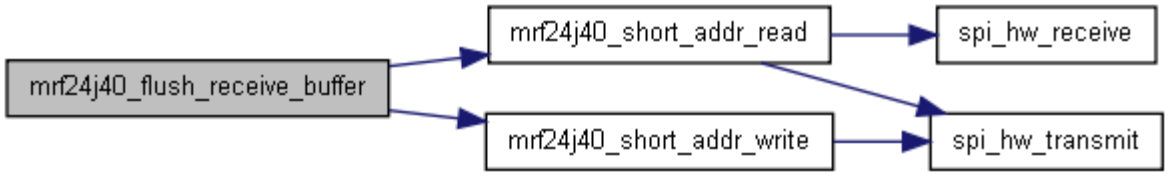
No need to call this routine normally, except according to mrf24j40 errata (promiscuous mode)

Definition at line 52 of file mrf24j40.c.

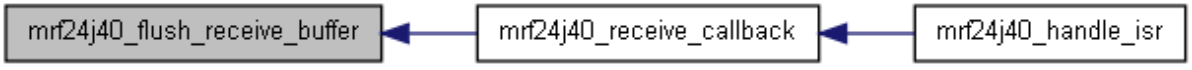
References mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), RXFLUSH, RXFLUSH_RXFLUSH, and uns8.

Referenced by mrf24j40_receive_callback().

Here is the call graph for this function:




Here is the caller graph for this function:



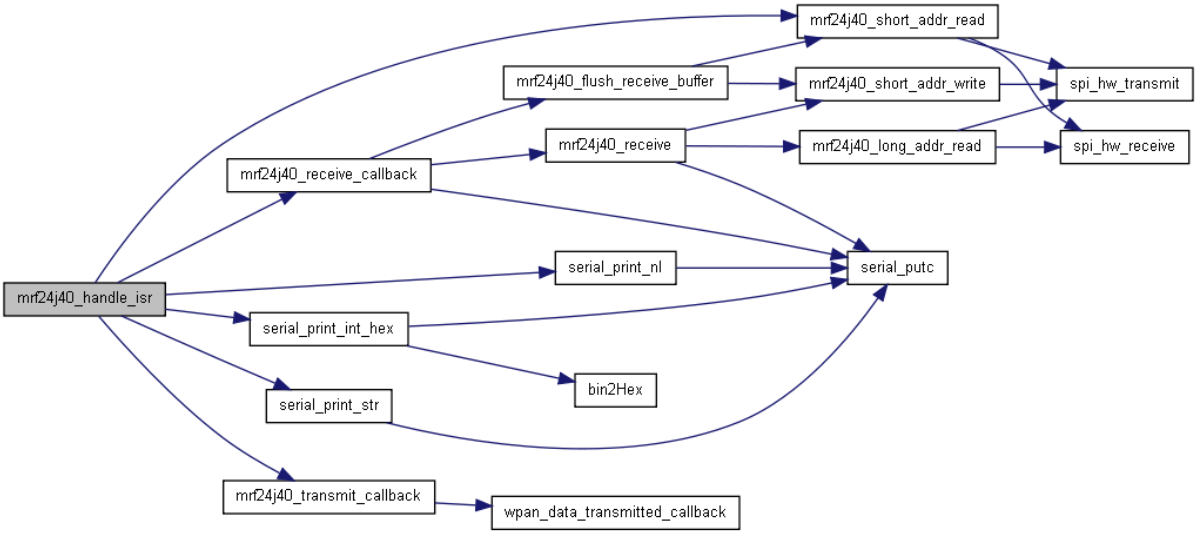

**void mrf24j40_handle_isr ()**

Call this routine when the mrf24j40 indicates an interrupt condition, or called regularly.

Definition at line 287 of file mrf24j40.c.

References INTSTAT, INTSTAT_RXIF, INTSTAT_TXNIF, mrf24j40_receive_callback(), mrf24j40_short_addr_read(), mrf24j40_transmit_callback(), serial_print_int_hex(), serial_print_nl(), serial_print_str(), TXSTAT, TXSTAT_CCAFAIL, and uns8.

Here is the call graph for this function:




### void mrf24j40_init ()

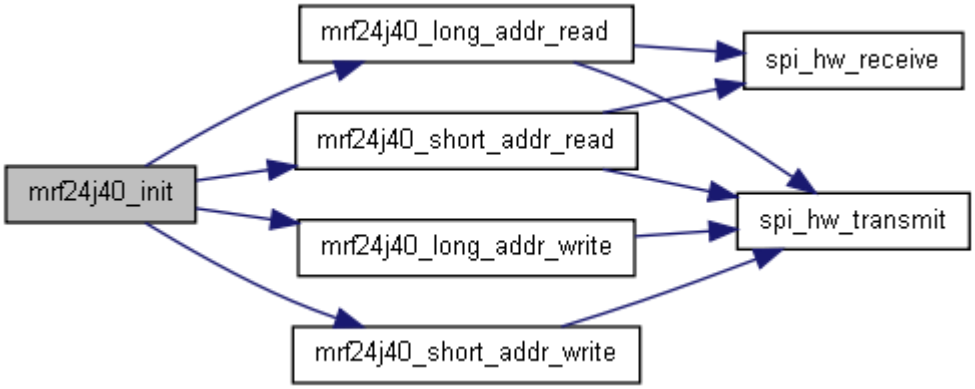
Sets int pin as input and cs pin as output.

Definition at line 499 of file mrf24j40.c.

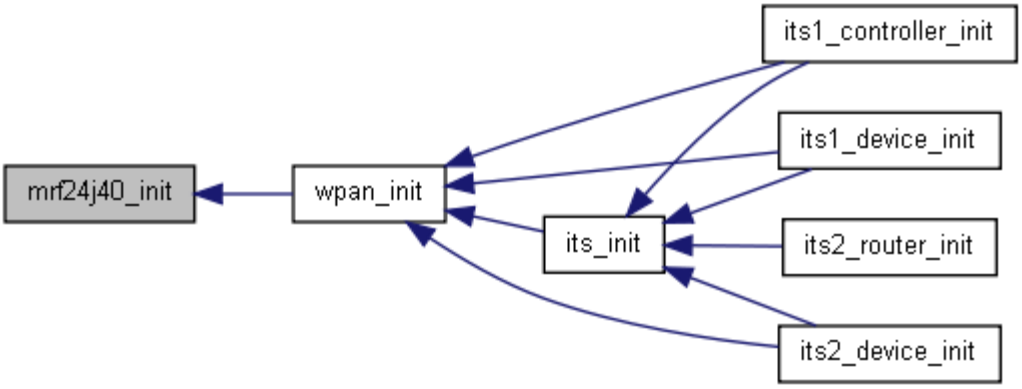
References BBREG2, BBREG6, BBREG6_RSSIMODE2, CCAEDTH, data_sequence_number, mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_INTCON, MRF_INTCON_RXIE, MRF_INTCON_TXNIE, PACON2, RFCON0, RFCON1, RFCON2, RFCON3, RFCON6, RFCON7, RFCON8, RFCTL, RFSTATE, SLPCON1, SOFTRST, TESTMODE, TXSTBL, and uns8.

Referenced by wpan_init().

Here is the call graph for this function:




Here is the caller graph for this function:

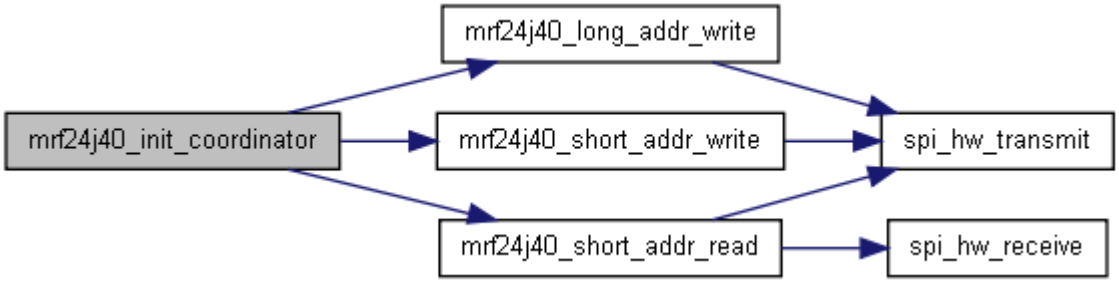



**void mrf24j40_init_coordinator ()**

Definition at line 606 of file mrf24j40.c.

References BBREG2, BBREG6, CCAEDTH, data_sequence_number, extended_address, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_INTCON, MRF_INTCON_RXIE, MRF_INTCON_TXNIE, PACON2, RFCON0, RFCON1, RFCON2, RFCON6, RFCON7, RFCON8, RFCTL, RXMCR, RXMCR_PANCOORD, SLPCON1, SOFTRST, TXMCR, TXMCR_SLOTTED, TXSTBL, and uns8.

Here is the call graph for this function:




**uns8 mrf24j40_long_addr_read (uns16 *addr*)**

Returns the value in the memory location specified.

**Parameters:**

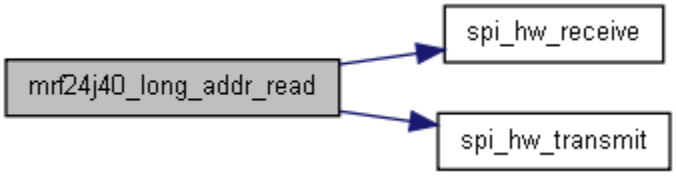
*addr* Long address memory location (see mrf24h40_defines.h)

Definition at line 749 of file mrf24j40.c.

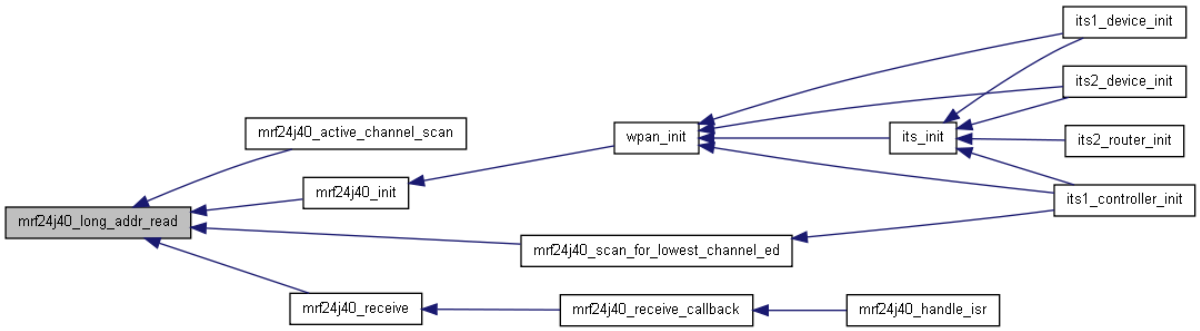
References clear_pin, set_pin, spi_hw_receive(), spi_hw_transmit(), and uns8.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_receive(), and mrf24j40_scan_for_lowest_channel_ed().

Here is the call graph for this function:



Here is the caller graph for this function:




### void mrf24j40_long_addr_write (uns16 *addr*, uns8 *data*)

Sets the memory location to the value specified

**Parameters:**

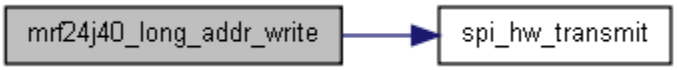
*addr* Long address memory location (see mrf24h40_defines.h)
*data* Value that the memory location should be set to

Definition at line 767 of file mrf24j40.c.

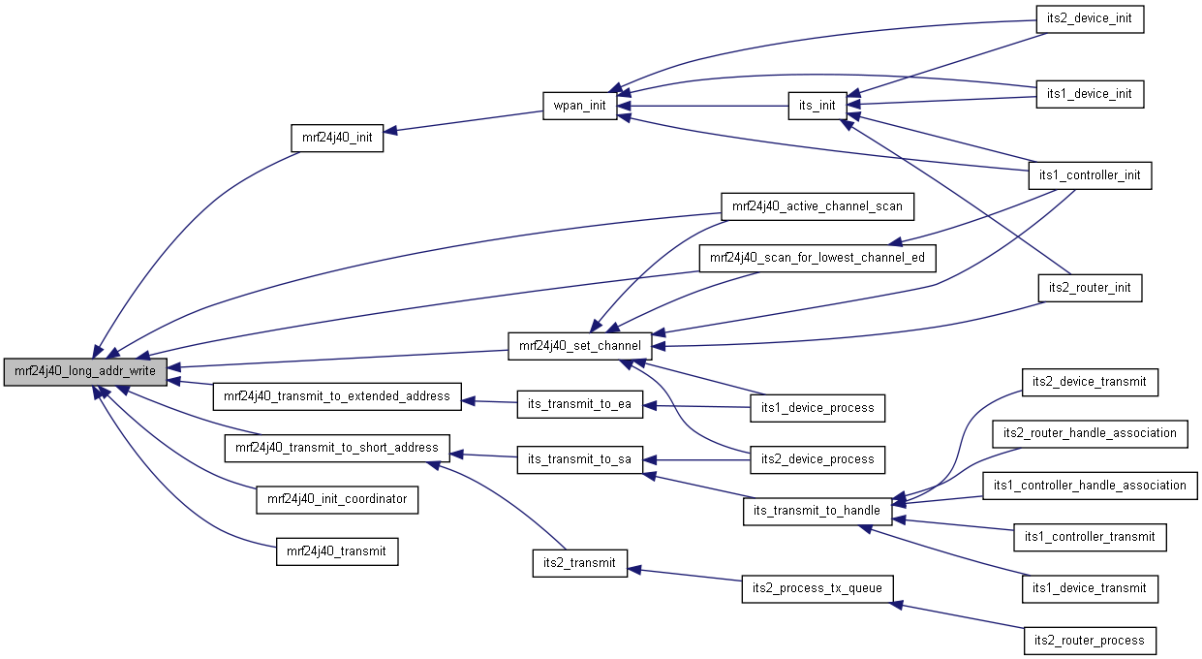
References clear_pin, set_pin, and spi_hw_transmit().

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:



### void mrf24j40_orphan_channel_scan ()

Definition at line 226 of file mrf24j40.c.

### void mrf24j40_realign_pan ()

Definition at line 233 of file mrf24j40.c.

### uns8 mrf24j40_receive (uns8 * *data*, uns8 *bytes_to_receive*)

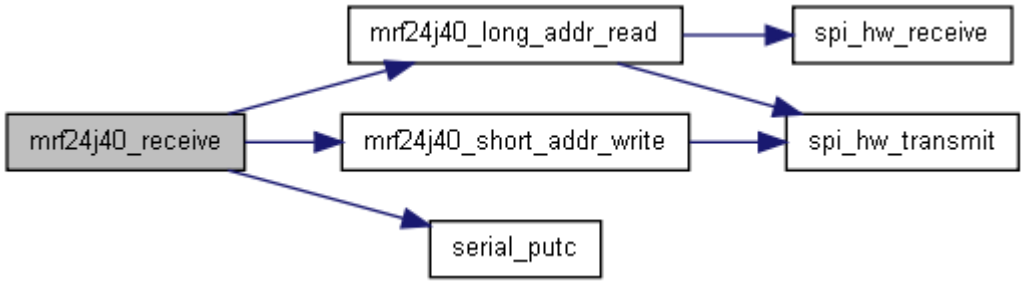
Once an interrupt has occured and we know it is because a packet has been received, this routine will pull the data from the mrf receive buffer. This needs to be done quickly so that the next packet is not lost.

Definition at line 312 of file mrf24j40.c.

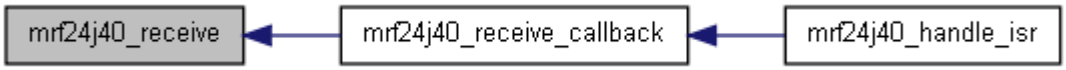
References BBREG1, BBREG1_RXDECINV, mrf24j40_long_addr_read(), mrf24j40_short_addr_write(), serial_putc(), uns16, and uns8.

Referenced by mrf24j40_receive_callback().

Here is the call graph for this function:




Here is the caller graph for this function:




### uns8 mrf24j40_scan_for_lowest_channel_ed ()

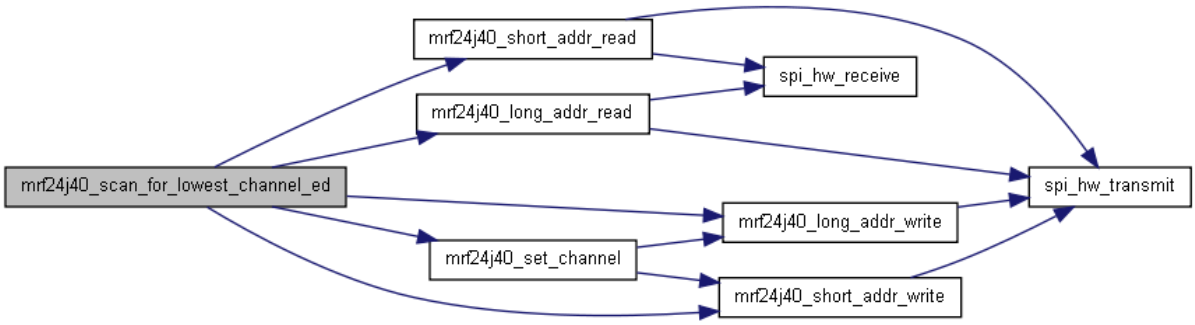
Scans through all channels and reports the channel with the lowest Energy Detection level.

Definition at line 77 of file mrf24j40.c.

References BBREG6, BBREG6_RSSIMODE1, BBREG6_RSSIMODE2, BBREG6_RSSIRDY, channel, GPIO, mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_set_channel(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, RSSI, TESTMODE, TRISGPIO, uns16, and uns8.

Referenced by its1_controller_init().

Here is the call graph for this function:




Here is the caller graph for this function:


### void mrf24j40_set_channel (uns8 *channel*)

Change to the specified channel, in the range MRF_FIRST_CHANNEL to MRF_LAST_CHANNEL.

**Parameters:**

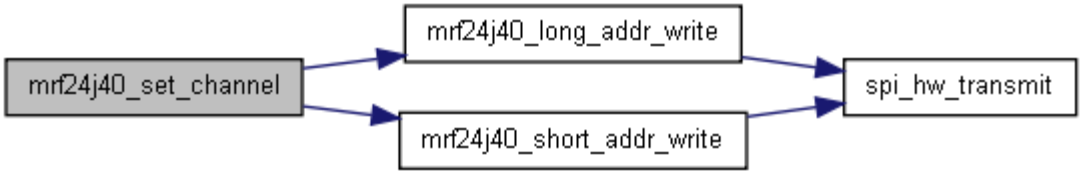
*channel* Channel to change to.

Definition at line 63 of file mrf24j40.c.

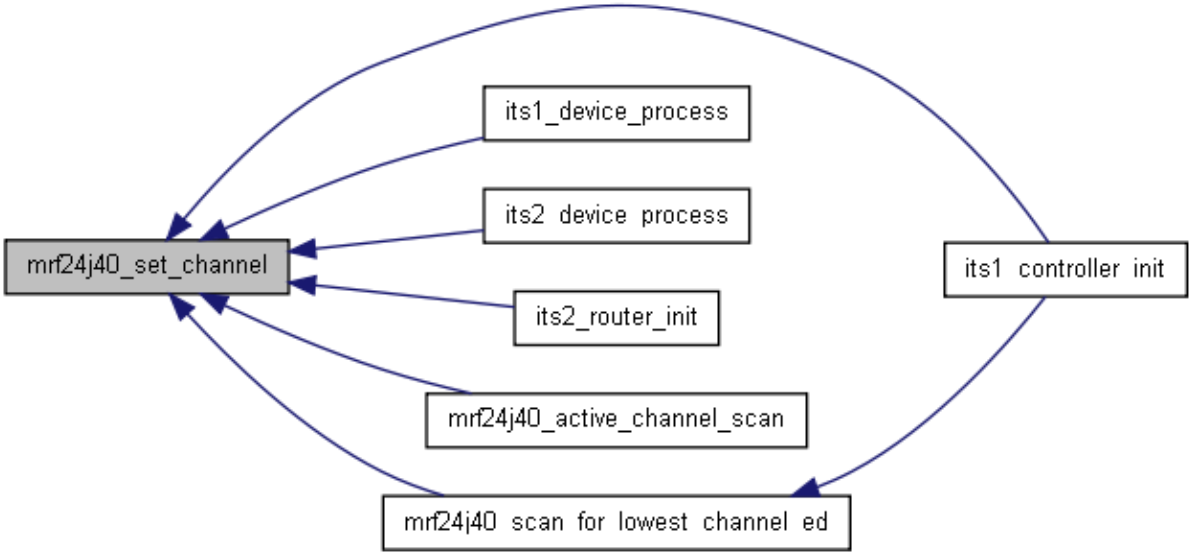
References current_channel, mrf24j40_long_addr_write(), mrf24j40_short_addr_write(), RFCON0, and RFCTL.

Referenced by its1_controller_init(), its1_device_process(), its2_device_process(), its2_router_init(), mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_extended_address (uns8 * *_extended_address*)**

Set IEEE 802.15.4 extended address.

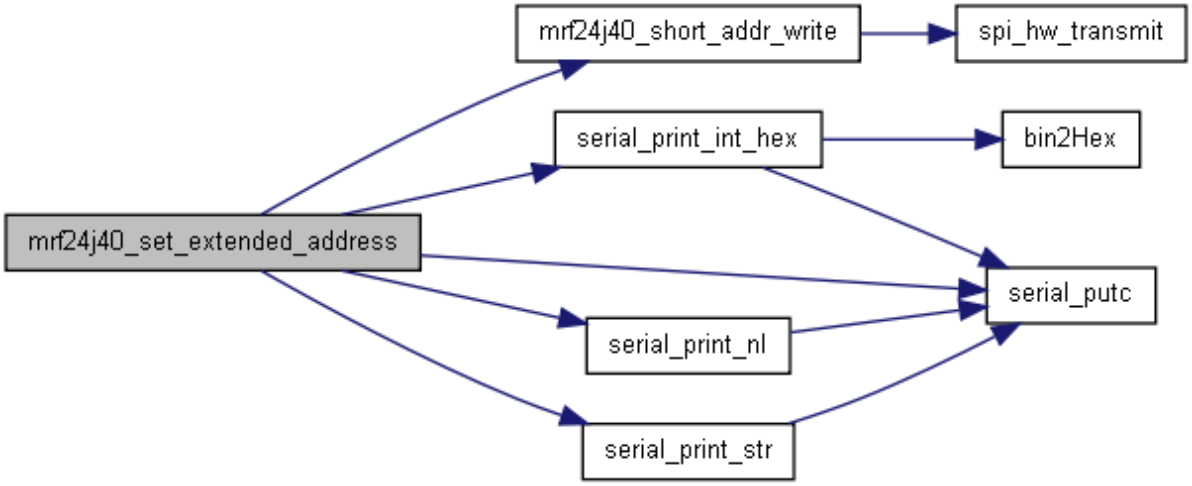
Sets the 64 bit extended address of the module. Pass a pointer to an 8 byte uns8 array containing the address.

Definition at line 259 of file mrf24j40.c.

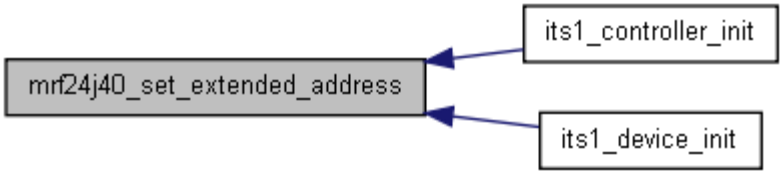
References EADR7, extended_address, mrf24j40_short_addr_write(), serial_print_int_hex(), serial_print_nl(), serial_print_str(), serial_putc(), and uns8.

Referenced by its1_controller_init(), and its1_device_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_pan_id (uns16 *_pan_id*)**

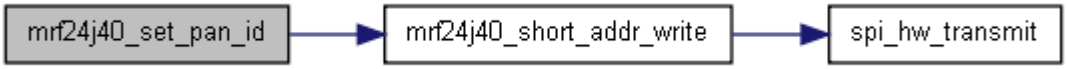
Sets the 16 bit PAN id of the module.

Definition at line 252 of file mrf24j40.c.

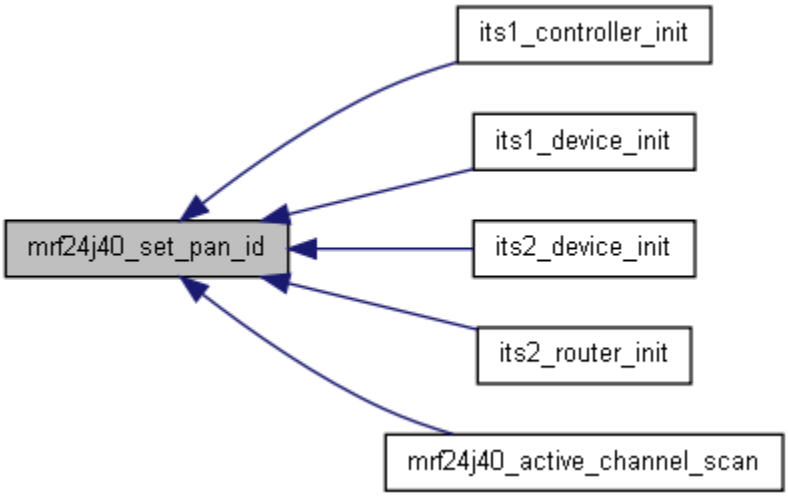
References mrf24j40_short_addr_write(), pan_id, PANIDH, and PANIDL.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), its2_router_init(), and mrf24j40_active_channel_scan().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_short_address (uns16 *_short_address*)**

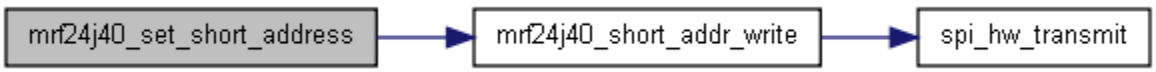
Sets the 16 bit short address of the module.

Definition at line 273 of file mrf24j40.c.

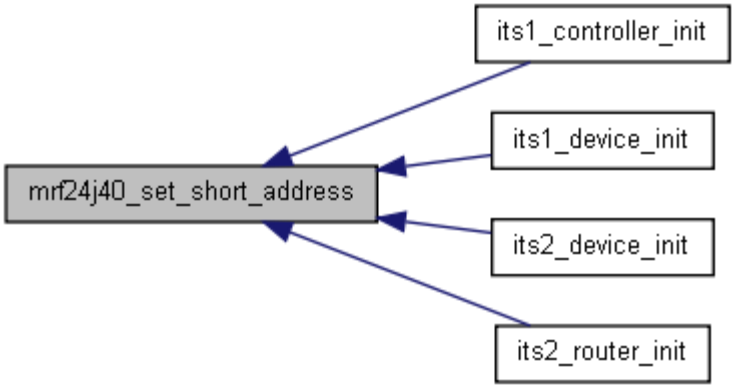
References mrf24j40_short_addr_write(), SADRH, SADRL, and short_address.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_setup_io ()**

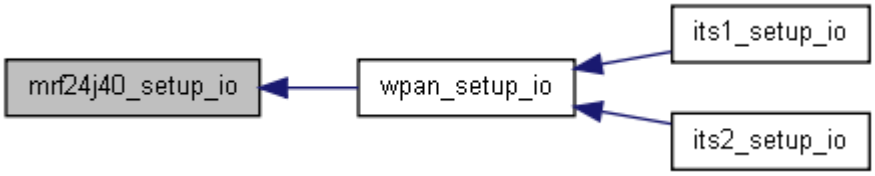
Sets int pin as input and cs pin as output.

Definition at line 785 of file mrf24j40.c.

References make_input, make_output, and set_pin.

Referenced by wpan_setup_io().

Here is the caller graph for this function:




**uns8 mrf24j40_short_addr_read (uns8 *addr*)**

Returns the value in the memory location specified.

**Parameters:**

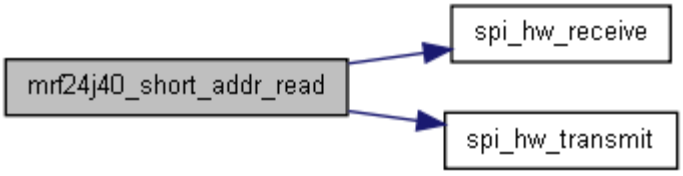
*addr* Short address memory location (see mrf24h40_defines.h)

Definition at line 725 of file mrf24j40.c.

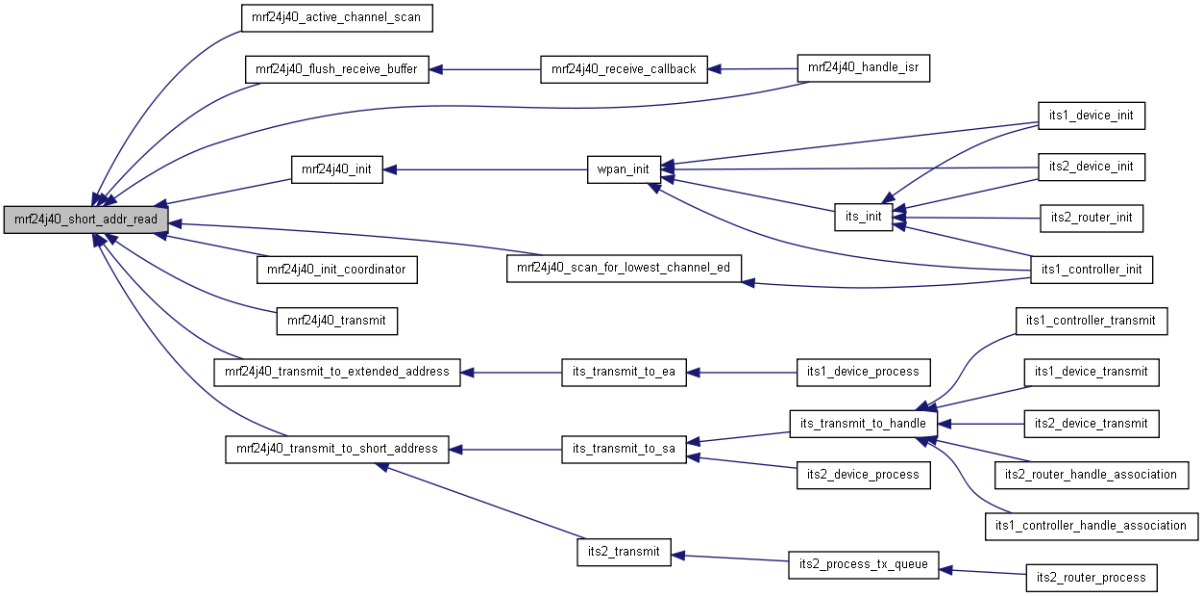
References clear_pin, set_pin, spi_hw_receive(), spi_hw_transmit(), and uns8.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_flush_receive_buffer(), mrf24j40_handle_isr(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_short_addr_write (uns8 *addr*, uns8 *data*)**

Sets the memory location to the value specified

**Parameters:**

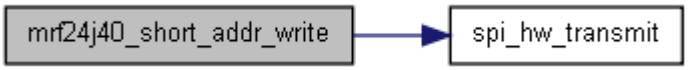
*addr* Short address memory location (see mrf24h40_defines.h)
*data* Value that the memory location should be set to

Definition at line 737 of file mrf24j40.c.

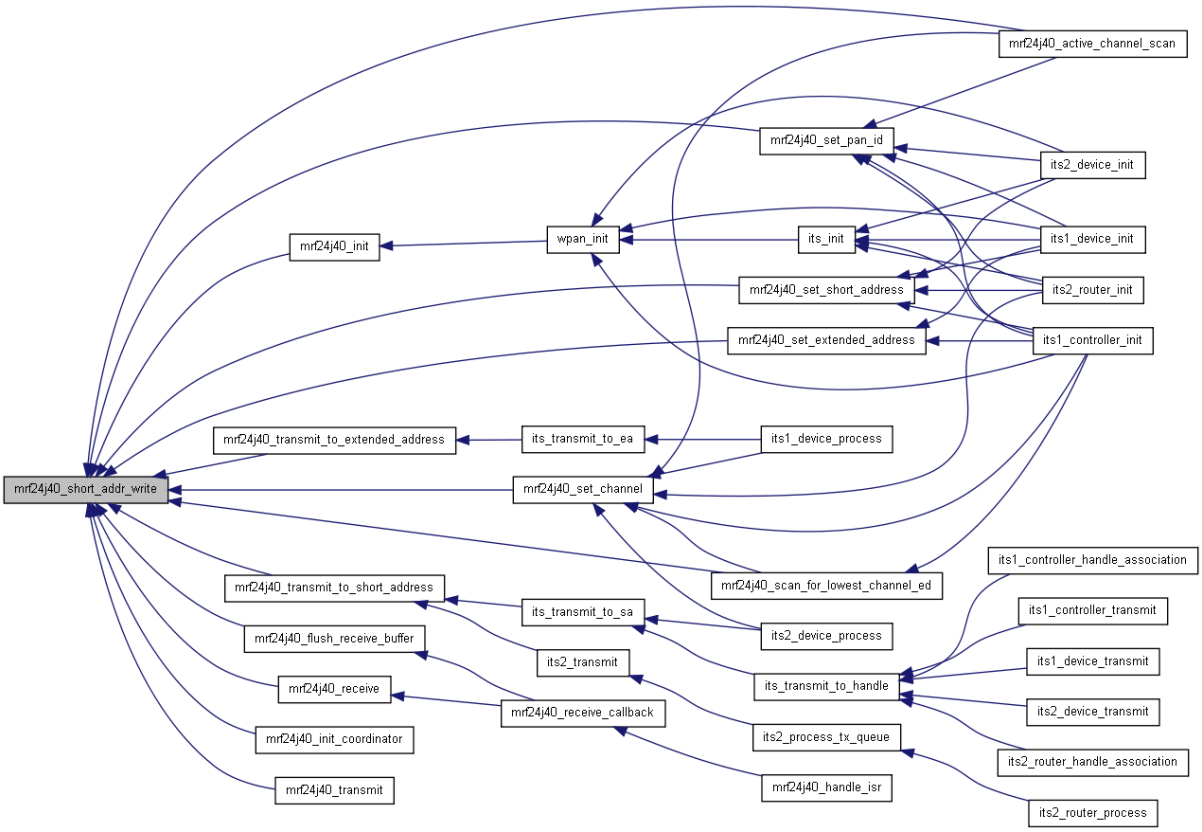
References clear_pin, set_pin, and spi_hw_transmit().

Referenced by mrf24j40_active_channel_scan(), mrf24j40_flush_receive_buffer(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_receive(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:




### void mrf24j40_start_pan ()

Definition at line 229 of file mrf24j40.c.

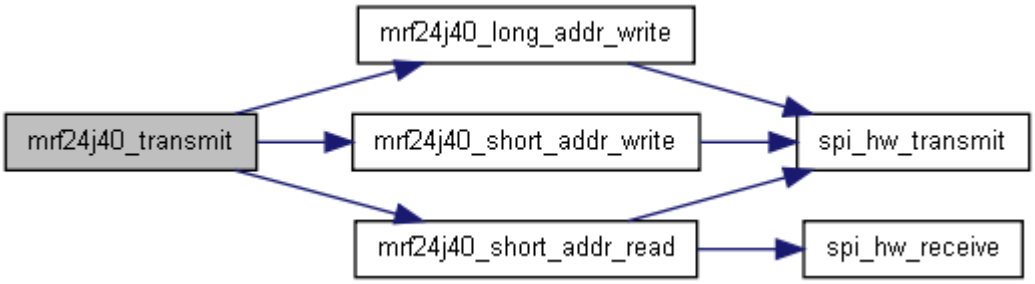
### void mrf24j40_transmit (uns8 * *data*, uns8 *bytes_to_transmit*)

Transmit a raw packet - this assumes you have already created an 802.15.4 compatible packet. Normally you would use transmit_to_extended_adddress or transmit_to_short_address instead.

Definition at line 470 of file mrf24j40.c.

References data_sequence_number, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), TXNCON, TXNCON_TXNTRIG, and uns8.

Here is the call graph for this function:



**void mrf24j40_transmit_to_extended_address (uns8 *frame_type*, uns16 *dest_pan_id*, uns8 * *dest_extended_address*, uns8 * *data*, uns8 *data_length*, uns8 *ack*)**

Requests that the mrf24j40 transmit the packet to the specified extended address

**Parameters:**

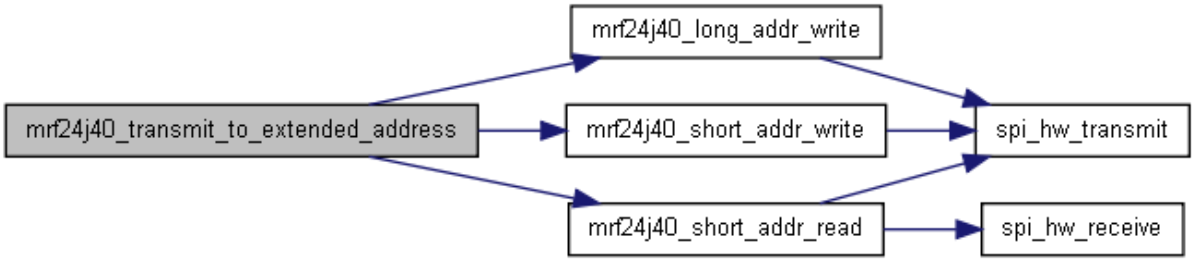
*frame_type* 802.15.4 frame type
*dest_pan_id* PAN id of destination
*dest_extended_address* Pointer to uns8 array indicating extended address of destination
*data* Pointer to uns8 array of bytes to transmit
*bytes_to_transmit*
*ack* Either MRF_ACK or MRF_NO_ACK depending if you want hardware acknowledgement

Definition at line 350 of file mrf24j40.c.

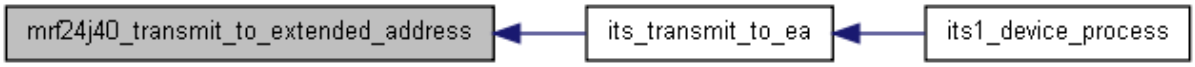
References data_sequence_number, extended_address, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), pan_id, TXNCON, TXNCON_TXNACKREQ, TXNCON_TXNTRIG, and uns8.

Referenced by its_transmit_to_ea().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_transmit_to_short_address (uns8 *frame_type*, uns16 *dest_pan_id*, uns16 *dest_short_address*, uns8 * *data*, uns8 *bytes_to_transmit*, uns8 *ack*)**

Requests that the mrf24j40 transmit the packet to the specified short address

**Parameters:**

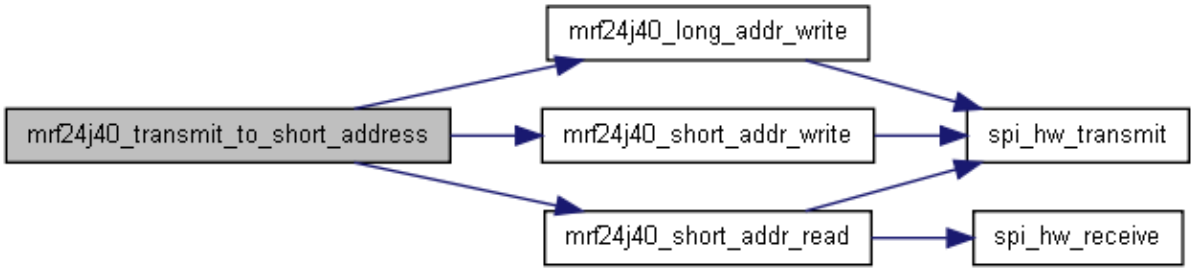
*frame_type* 802.15.4 frame type
*dest_pan_id* PAN id of destination
*dest_short_address* 16 bit short address of destination
*data* Pointer to uns8 array of bytes to transmit
*bytes_to_transmit*
*ack* Either MRF_ACK or MRF_NO_ACK depending if you want hardware acknowledgement

Definition at line 416 of file mrf24j40.c.

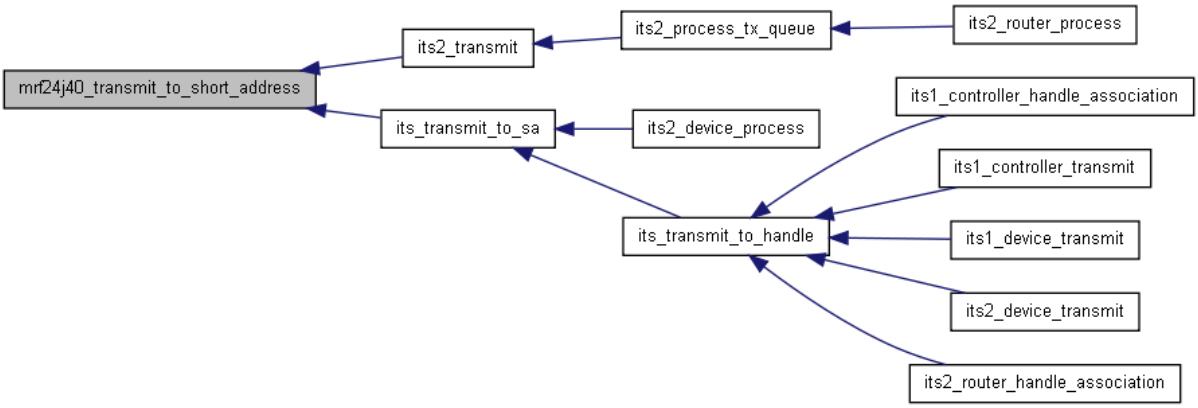
References data_sequence_number, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), pan_id, short_address, TXNCON, TXNCON_TXNACKREQ, TXNCON_TXNTRIG, and uns8.

Referenced by its2_transmit(), and its_transmit_to_sa().

Here is the call graph for this function:




Here is the caller graph for this function:




---

## Variable Documentation

### uns8 current_channel = 0

Definition at line 49 of file mrf24j40.c.

Referenced by mrf24j40_set_channel().

### uns8 data_sequence_number

Definition at line 45 of file mrf24j40.c.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

### uns8 extended_address[8] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff}

Definition at line 47 of file mrf24j40.c.

Referenced by mrf24j40_init_coordinator(), mrf24j40_set_extended_address(), and mrf24j40_transmit_to_extended_address().

### uns16 pan_id = 0xffff

Definition at line 46 of file mrf24j40.c.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_set_pan_id(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

**uns16 short_address = 0xffff**

Definition at line 48 of file mrf24j40.c.

Referenced by mrf24j40_set_short_address(), and mrf24j40_transmit_to_short_address().

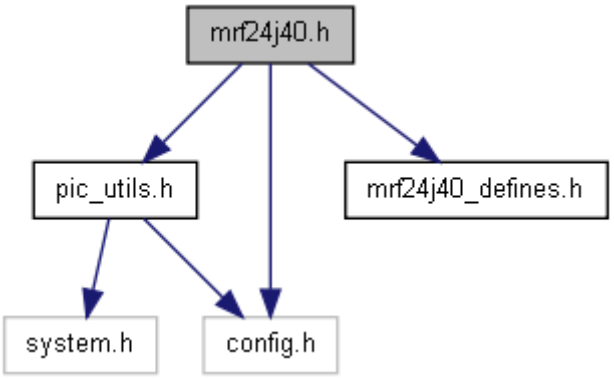
---

# mrf24j40.h File Reference

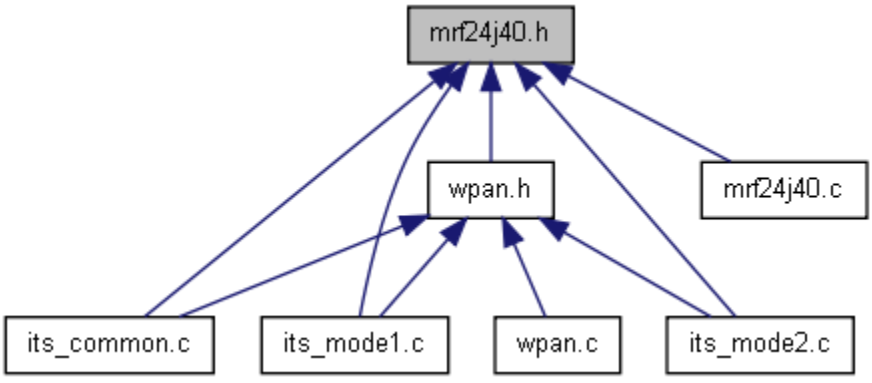
Microchip mrf24j40 IEEE 802.15.4 module routines.

```
#include "pic_utils.h"
#include "config.h"
#include "mrf24j40_defines.h"
```

Include dependency graph for mrf24j40.h:




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




## Defines

- #define LOC_CANADA 0x02
- #define LOC_EUROPE 0x03

- #define LOC_UNDEFINED 0x00
- #define LOC_UNITED_STATES 0x01
- #define MRF_ACK 1
- #define MRF_FIRST_CHANNEL 11
- #define MRF_LAST_CHANNEL 26
- #define MRF_NO_ACK 0

## Functions

- void mrf24j40_flush_receive_buffer ()
- *Flush receive buffer of mrf24j40.* void mrf24j40_handle_isr ()
- *Interrupt service routine for mrf24j40 chip.* void mrf24j40_init ()
- *Initialises mrf24j40 chip ready for use.* uns8 mrf24j40_long_addr_read (uns16 addr)
- *Read data from long address of memory location in mrf24j40.* void mrf24j40_long_addr_write (uns16 addr, uns8 data)
- *Write data to long address memory location.* uns8 mrf24j40_receive (uns8 *data, uns8 bytes_to_receive)
- *Pull received data from buffer.* void mrf24j40_receive_callback ()
- *Callback is actioned when mrf24j40 has a packet received off air.* uns8 mrf24j40_scan_for_lowest_channel_ed ()
- *Scan all channels for lowest RF energy.* void mrf24j40_set_channel (uns8 channel)
- *Change channels.* void mrf24j40_set_extended_address (uns8 *_extended_address)
- *Set IEEE 802.15.4 extended address.* void mrf24j40_set_pan_id (uns16 _pan_id)
- *Set PAN id.* void mrf24j40_set_short_address (uns16 _short_address)
- *Set short address.* void mrf24j40_setup_io ()
- *Setup ports/pins as inputs/outputs ready for use.* uns8 mrf24j40_short_addr_read (uns8 addr)
- *Read data from short address of memory location in mrf24j40.* void mrf24j40_short_addr_write (uns8 addr, uns8 data)
- *Write data to short address memory location.* void mrf24j40_transmit (uns8 *data, uns8 bytes_to_transmit)
- *Transmit raw data.* void mrf24j40_transmit_callback (uns8 status, uns8 retries, uns8 channel_busy)
- *Callback is actioned when mrf24j40 has finished transmitting a packet, or failed to do so.* void mrf24j40_transmit_to_extended_address (uns8 frame_type, uns16 dest_pan_id, uns8 *dest_extended_address, uns8 *data, uns8 data_length, uns8 ack)
- *Transmit packet to extended address.* void mrf24j40_transmit_to_short_address (uns8 frame_type, uns16 dest_pan_id, uns16 dest_short_address, uns8 *data, uns8 bytes_to_transmit, uns8 ack)

*Transmit packet to short address.*

---

## Detailed Description

Definition in file mrf24j40.h.

---

## Define Documentation

### #define LOC_CANADA 0x02

Module located in Canada (?? power)

Definition at line 99 of file mrf24j40.h.

### #define LOC_EUROPE 0x03

Module located in Europe (-14.9dB power)

Definition at line 101 of file mrf24j40.h.

#### #define LOC_UNDEFINED 0x00

Module located in undefined location (full power)

Definition at line 95 of file mrf24j40.h.

#### #define LOC_UNITED_STATES 0x01

Module located in United States (?? power)

Definition at line 97 of file mrf24j40.h.

#### #define MRF_ACK 1

Send mrf packet and request acknowledgement

Definition at line 90 of file mrf24j40.h.

Referenced by its2_transmit(), and its_transmit_to_sa().

#### #define MRF_FIRST_CHANNEL 11

First available mrf channel

Definition at line 85 of file mrf24j40.h.

Referenced by its1_device_process(), its2_device_process(), mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

#### #define MRF_LAST_CHANNEL 26

Last availablre mrf channel

Definition at line 87 of file mrf24j40.h.

Referenced by its1_device_process(), its2_device_process(), mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

#### #define MRF_NO_ACK 0

Send mrf packet without acknowledgement

Definition at line 92 of file mrf24j40.h.

Referenced by its2_transmit(), and its_transmit_to_ea().

---

## Function Documentation

#### void mrf24j40_flush_receive_buffer ()

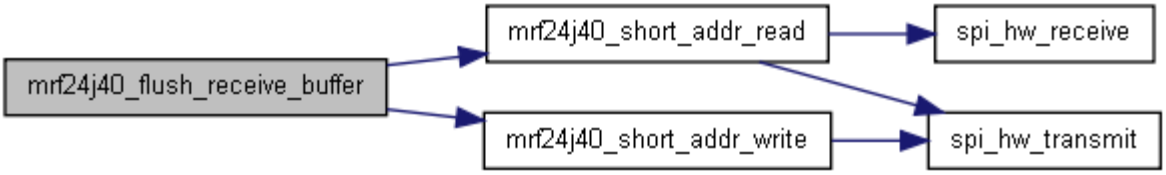
No need to call this routine normally, except according to mrf24j40 errata (promiscuous mode)

Definition at line 52 of file mrf24j40.c.

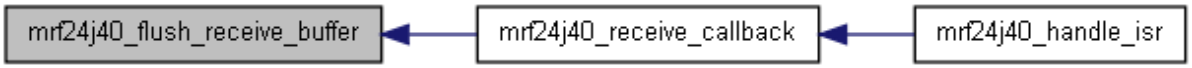
References mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), RXFLUSH, RXFLUSH_RXFLUSH, and uns8.

Referenced by mrf24j40_receive_callback().

Here is the call graph for this function:




Here is the caller graph for this function:



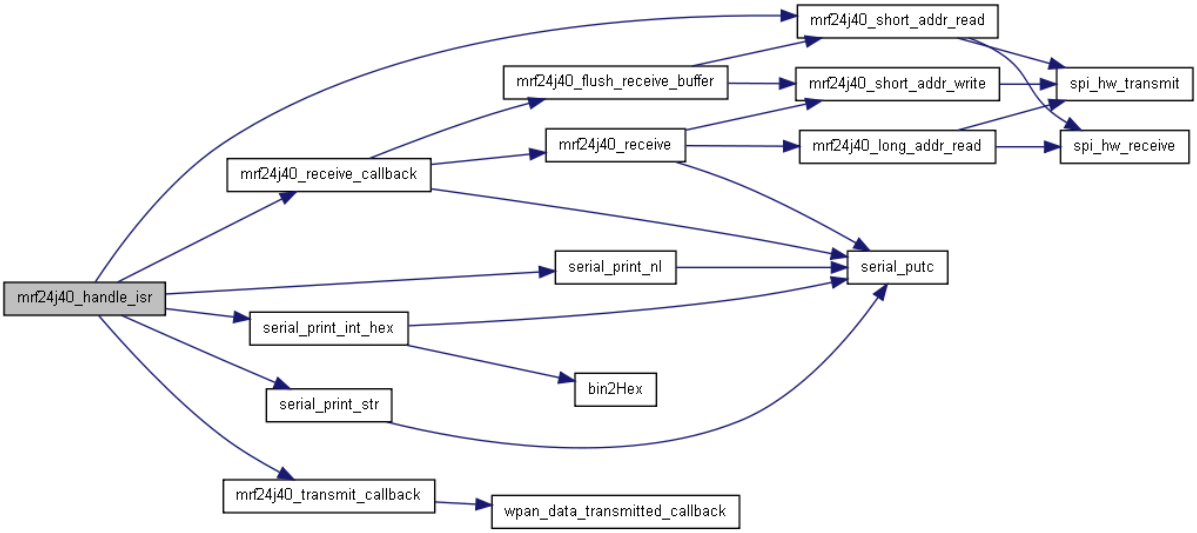

### void mrf24j40_handle_isr ()

Call this routine when the mrf24j40 indicates an interrupt condition, or called regularly.

Definition at line 287 of file mrf24j40.c.

References INTSTAT, INTSTAT_RXIF, INTSTAT_TXNIF, mrf24j40_receive_callback(), mrf24j40_short_addr_read(), mrf24j40_transmit_callback(), serial_print_int_hex(), serial_print_nl(), serial_print_str(), TXSTAT, TXSTAT_CCAFAIL, and uns8.

Here is the call graph for this function:




### void mrf24j40_init ()

Sets int pin as input and cs pin as output.

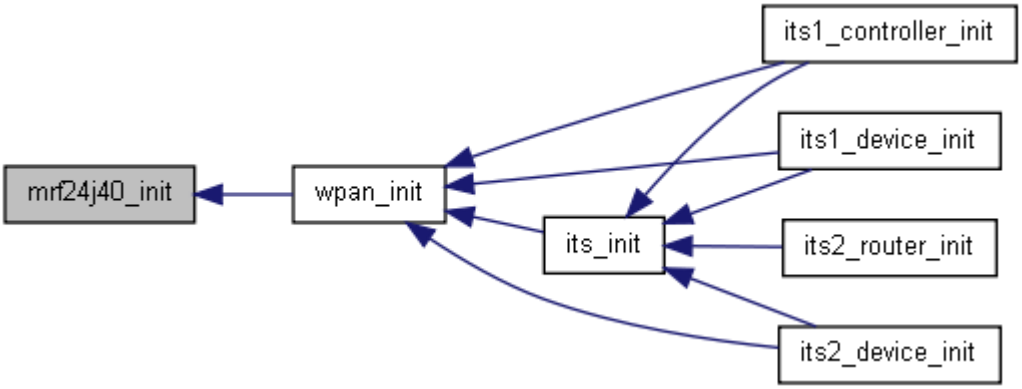
Definition at line 499 of file mrf24j40.c.

References BBREG2, BBREG6, BBREG6_RSSIMODE2, CCAEDTH, data_sequence_number, mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_INTCON, MRF_INTCON_RXIE, MRF_INTCON_TXNIE, PACON2, RFCON0, RFCON1, RFCON2, RFCON3, RFCON6, RFCON7, RFCON8, RFCTL, RFSTATE, SLPCON1, SOFTRST, TESTMODE, TXSTBL, and uns8.

Referenced by wpan_init().

Here is the call graph for this function:


Here is the caller graph for this function:




**uns8 mrf24j40_long_addr_read (uns16 *addr*)**

Returns the value in the memory location specified.

**Parameters:**

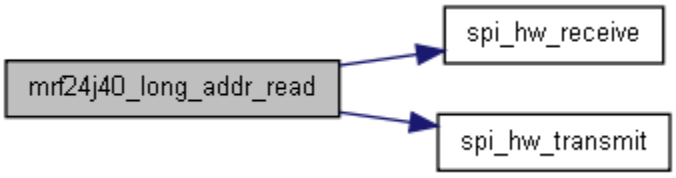
*addr* Long address memory location (see mrf24h40_defines.h)

Definition at line 749 of file mrf24j40.c.

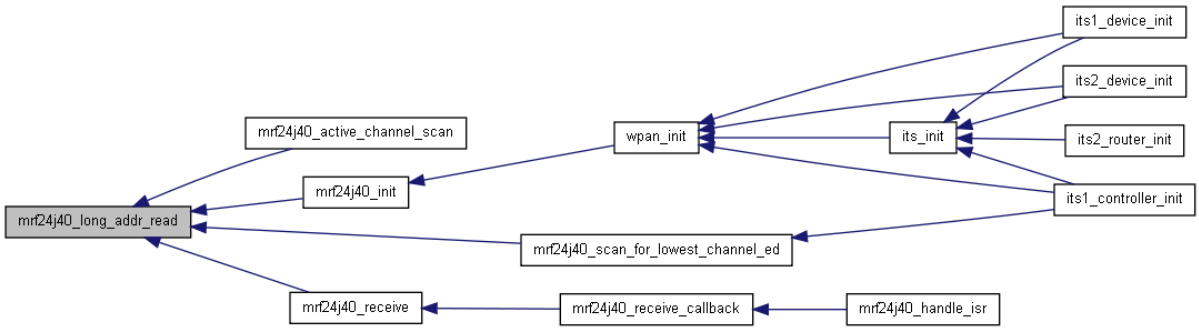
References clear_pin, set_pin, spi_hw_receive(), spi_hw_transmit(), and uns8.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_receive(), and mrf24j40_scan_for_lowest_channel_ed().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_long_addr_write (uns16 *addr*, uns8 *data*)**

Sets the memory location to the value specified

**Parameters:**

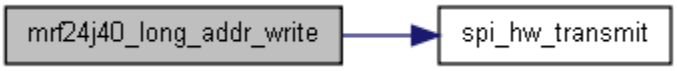
*addr* Long address memory location (see mrf24h40_defines.h)
*data* Value that the memory location should be set to

Definition at line 767 of file mrf24j40.c.

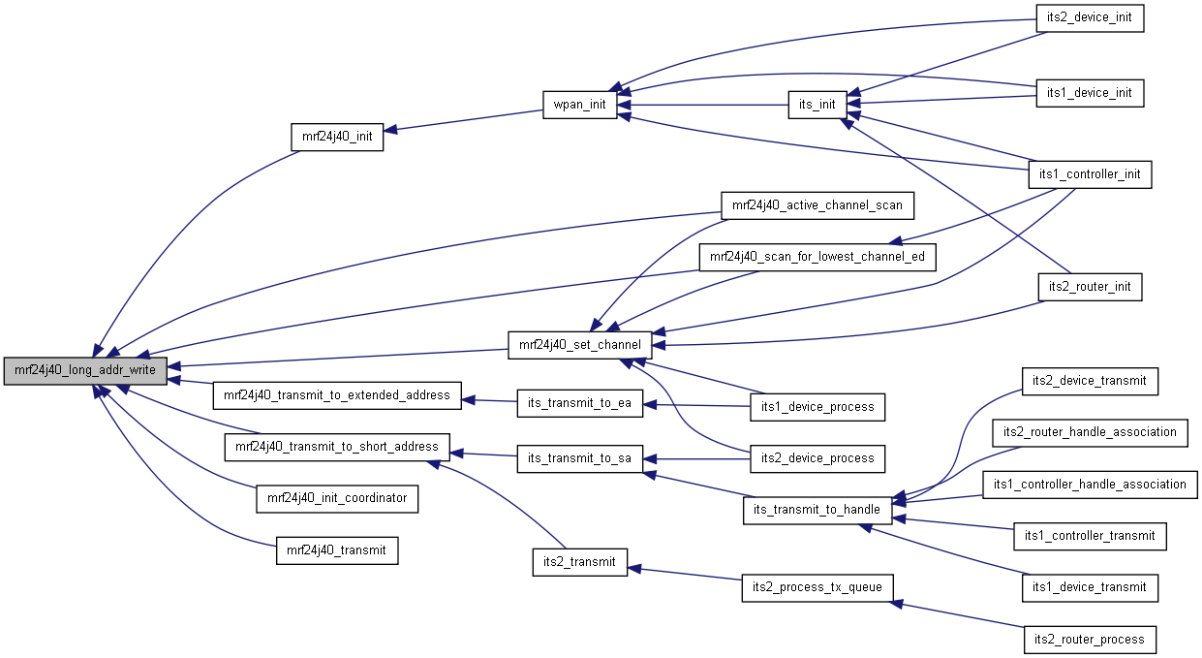
References clear_pin, set_pin, and spi_hw_transmit().

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:



**uns8 mrf24j40_receive (uns8 * *data*, uns8 *bytes_to_receive*)**

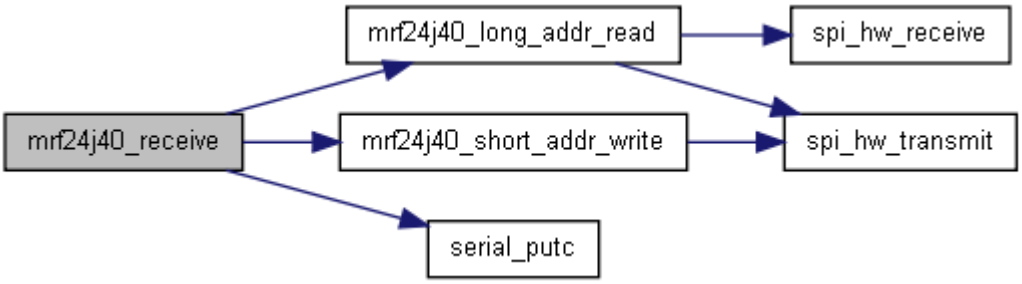
Once an interrupt has occured and we know it is because a packet has been received, this routine will pull the data from the mrf receive buffer. This needs to be done quickly so that the next packet is not lost.

Definition at line 312 of file mrf24j40.c.

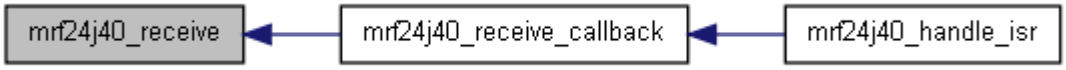
References BBREG1, BBREG1_RXDECINV, mrf24j40_long_addr_read(), mrf24j40_short_addr_write(), serial_putc(), uns16, and uns8.

Referenced by mrf24j40_receive_callback().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_receive_callback ()**

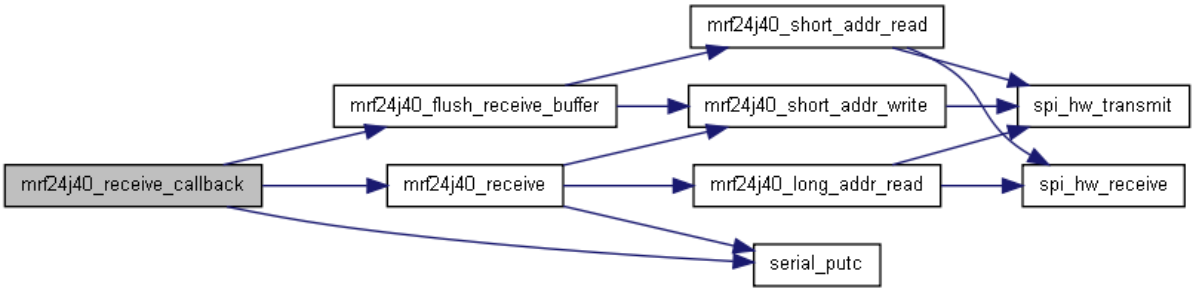
Callback indicating the mrf24j40 has a packet ready.

Definition at line 56 of file wpan.c.

References debug_int, debug_str, mrf24j40_flush_receive_buffer(), mrf24j40_receive(), pkt_received, receive_lost, serial_putc(), wpan_rx_buffer, and wpan_rx_count.

Referenced by mrf24j40_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:

### uns8 mrf24j40_scan_for_lowest_channel_ed ()

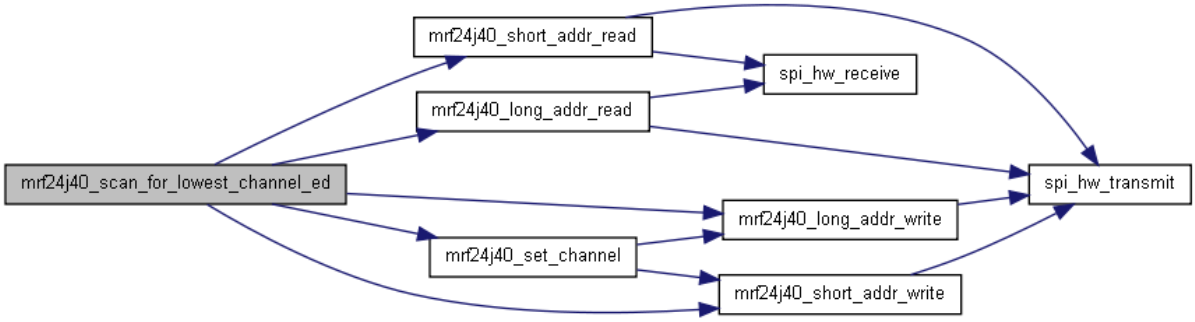
Scans through all channels and reports the channel with the lowest Energy Detection level.

Definition at line 77 of file mrf24j40.c.

References BBREG6, BBREG6_RSSIMODE1, BBREG6_RSSIMODE2, BBREG6_RSSIRDY, channel, GPIO, mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_set_channel(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), MRF_FIRST_CHANNEL, MRF_LAST_CHANNEL, RSSI, TESTMODE, TRISGPIO, uns16, and uns8.

Referenced by its1_controller_init().

Here is the call graph for this function:




Here is the caller graph for this function:


### void mrf24j40_set_channel (uns8 *channel*)

Change to the specified channel, in the range MRF_FIRST_CHANNEL to MRF_LAST_CHANNEL.

**Parameters:**

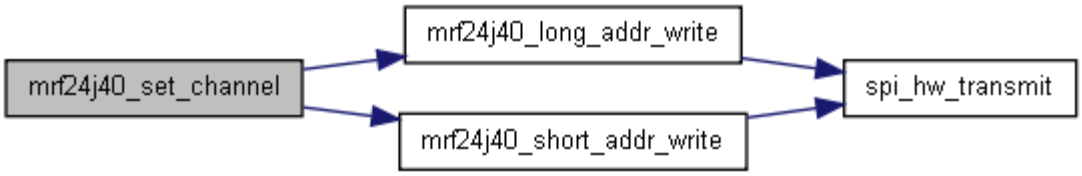
*channel* Channel to change to.

Definition at line 63 of file mrf24j40.c.

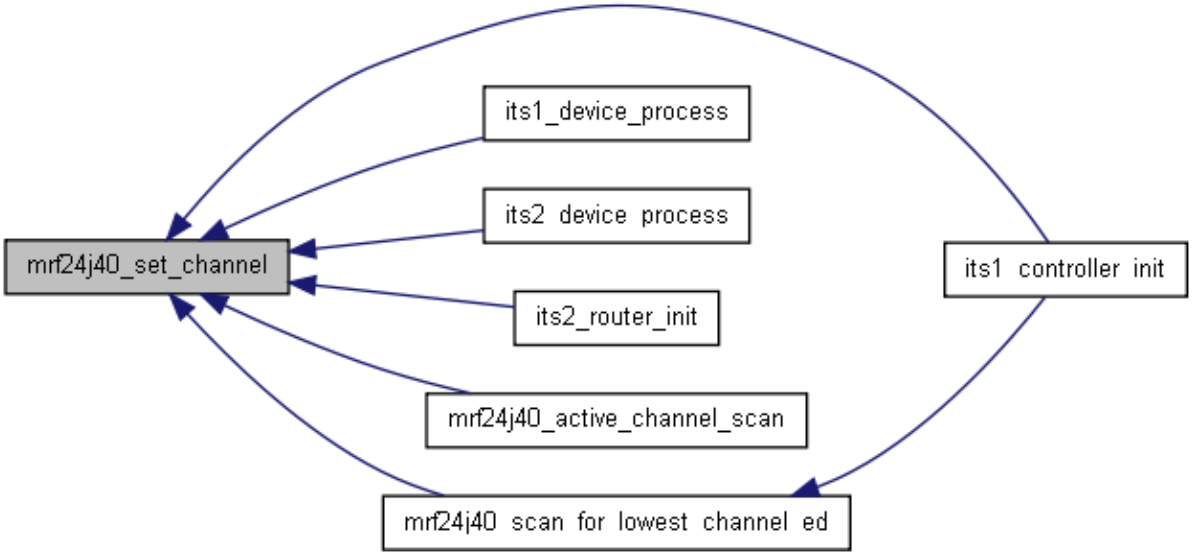
References current_channel, mrf24j40_long_addr_write(), mrf24j40_short_addr_write(), RFCON0, and RFCTL.

Referenced by its1_controller_init(), its1_device_process(), its2_device_process(), its2_router_init(), mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_extended_address (uns8 * *_extended_address*)**

Set extended address.

Pass a pointer to an 8 byte uns8 array with the address embedded as MSB leftmost byte and LSB rightmost byte, eg

uns8 EA_3[8] = { 0, 0, 0, 0, 0, 0, 0, 3 };

Set extended address to 0x0000000000000003 (64 bit address) mrf24j40_set_extended_address(&EA_3);

Sets the 64 bit extended address of the module. Pass a pointer to an 8 byte uns8 array containing the address.

Set IEEE 802.15.4 extended address.

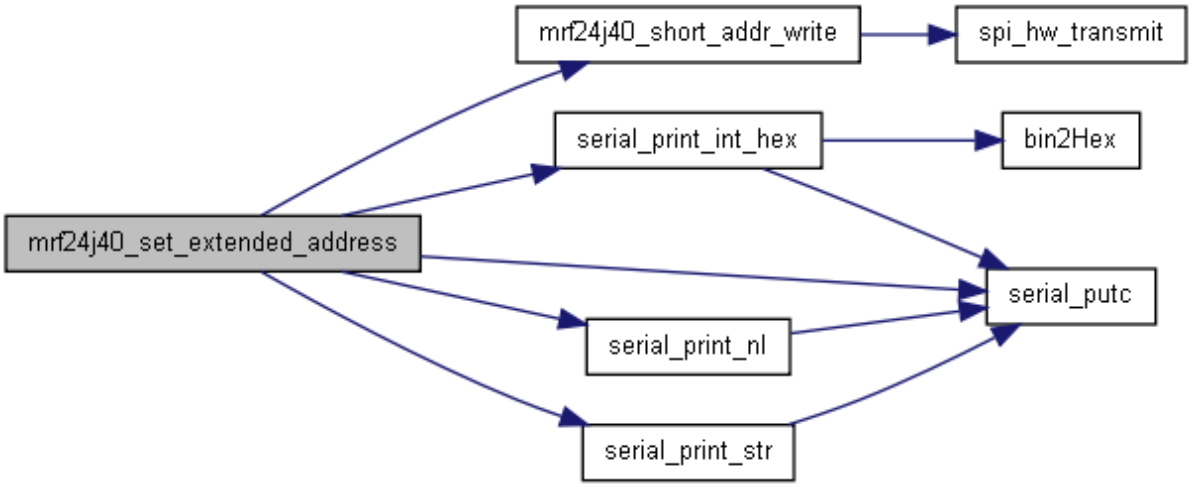
Sets the 64 bit extended address of the module. Pass a pointer to an 8 byte uns8 array containing the address.

Definition at line 259 of file mrf24j40.c.

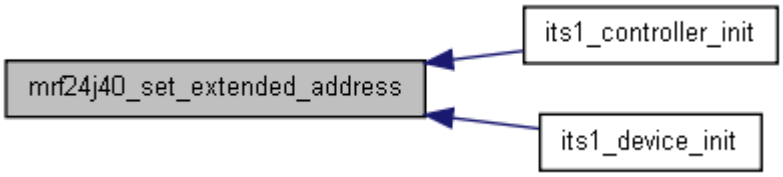
References EADR7, extended_address, mrf24j40_short_addr_write(), serial_print_int_hex(), serial_print_nl(), serial_print_str(), serial_putc(), and uns8.

Referenced by its1_controller_init(), and its1_device_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_pan_id (uns16 *_pan_id*)**

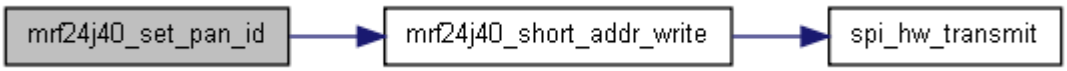
Sets the 16 bit PAN id of the module.

Definition at line 252 of file mrf24j40.c.

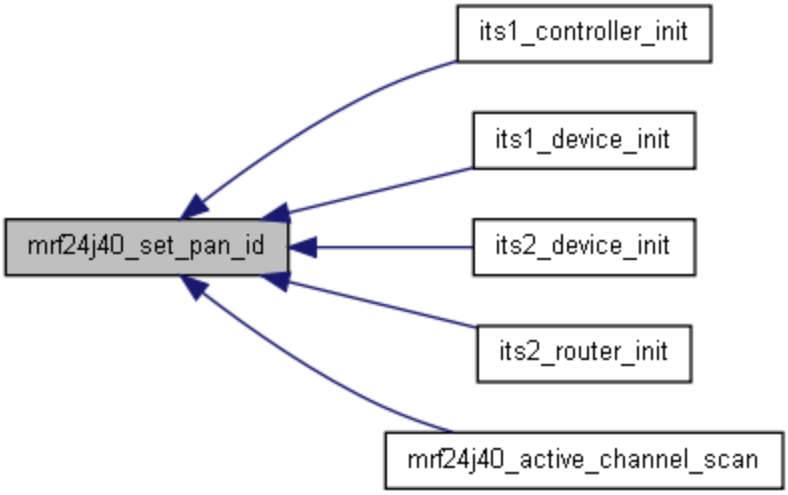
References mrf24j40_short_addr_write(), pan_id, PANIDH, and PANIDL.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), its2_router_init(), and mrf24j40_active_channel_scan().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_set_short_address (uns16 *_short_address*)**

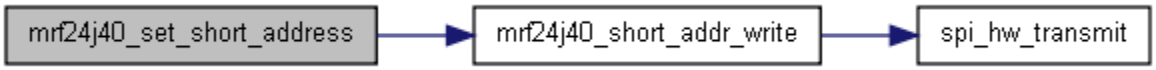
Sets the 16 bit short address of the module.

Definition at line 273 of file mrf24j40.c.

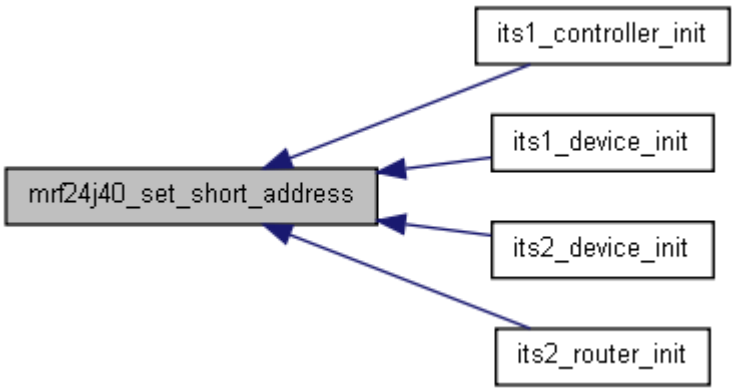
References mrf24j40_short_addr_write(), SADRH, SADRL, and short_address.

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its2_router_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_setup_io ()**

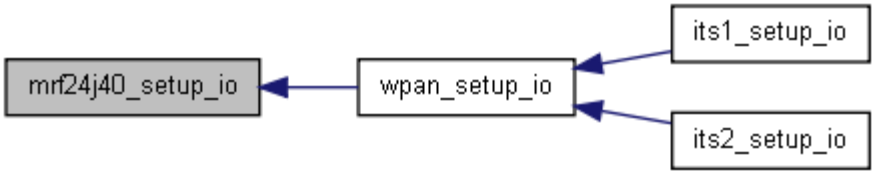
Sets int pin as input and cs pin as output.

Definition at line 785 of file mrf24j40.c.

References make_input, make_output, and set_pin.

Referenced by wpan_setup_io().

Here is the caller graph for this function:




**uns8 mrf24j40_short_addr_read (uns8 *addr*)**

Returns the value in the memory location specified.

**Parameters:**

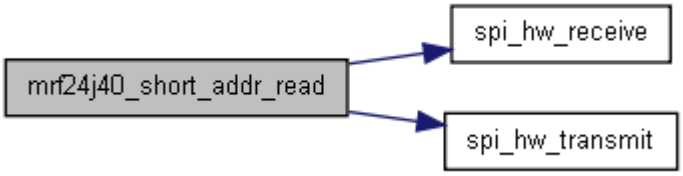
*addr* Short address memory location (see mrf24h40_defines.h)

Definition at line 725 of file mrf24j40.c.

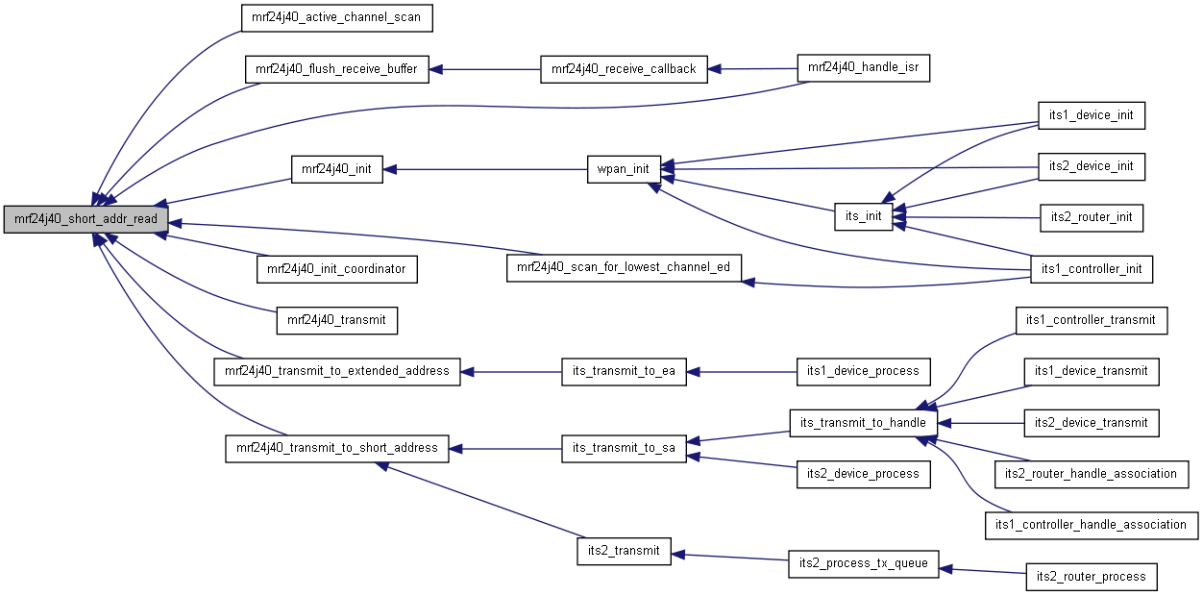
References clear_pin, set_pin, spi_hw_receive(), spi_hw_transmit(), and uns8.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_flush_receive_buffer(), mrf24j40_handle_isr(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_short_addr_write (uns8 *addr*, uns8 *data*)**

Sets the memory location to the value specified

**Parameters:**

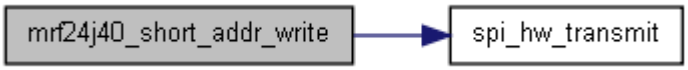
*addr* Short address memory location (see mrf24h40_defines.h)
*data* Value that the memory location should be set to

Definition at line 737 of file mrf24j40.c.

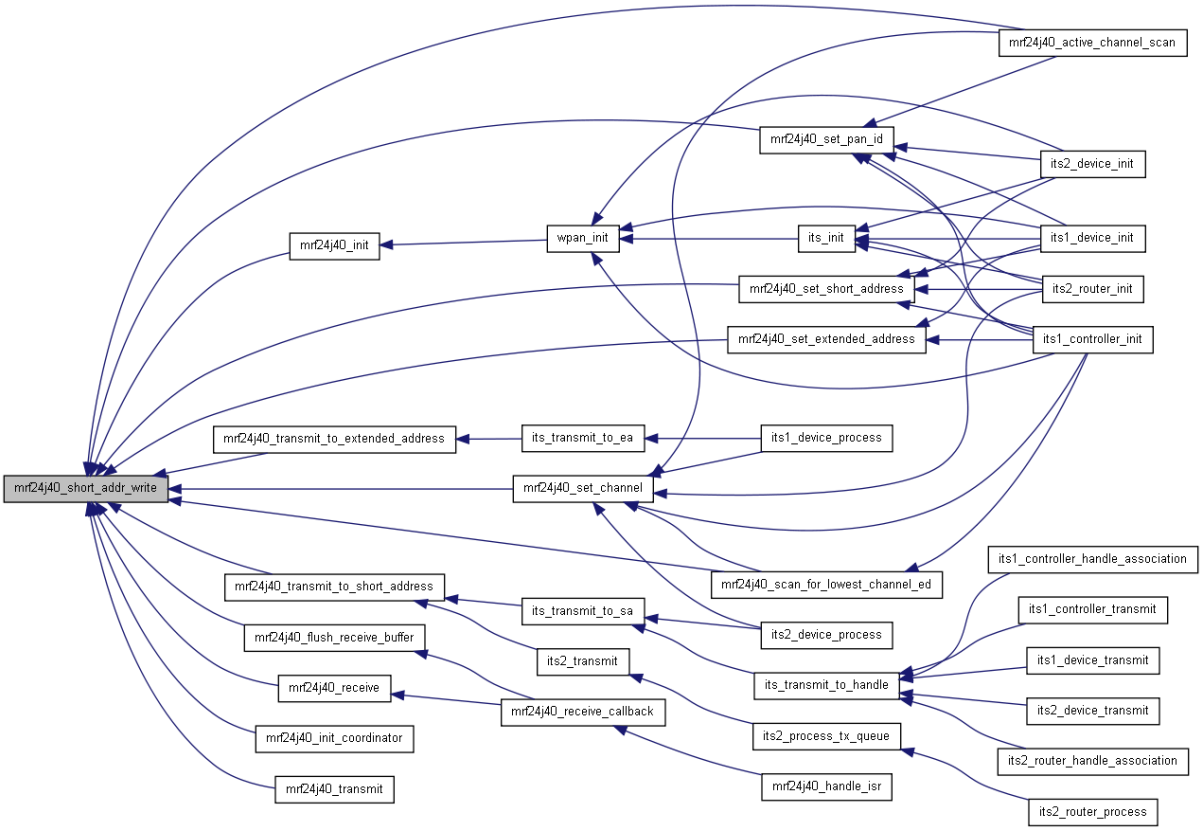
References clear_pin, set_pin, and spi_hw_transmit().

Referenced by mrf24j40_active_channel_scan(), mrf24j40_flush_receive_buffer(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_receive(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_channel(), mrf24j40_set_extended_address(), mrf24j40_set_pan_id(), mrf24j40_set_short_address(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

Here is the call graph for this function:




Here is the caller graph for this function:



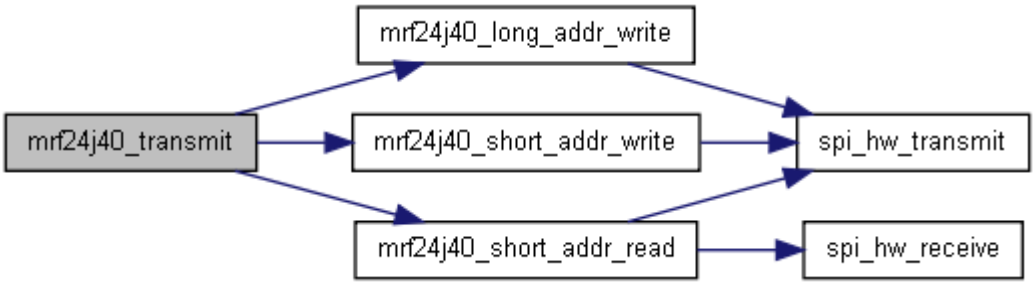

### void mrf24j40_transmit (uns8 * *data*, uns8 *bytes_to_transmit*)

Transmit a raw packet - this assumes you have already created an 802.15.4 compatible packet. Normally you would use transmit_to_extended_adddress or transmit_to_short_address instead.

Definition at line 470 of file mrf24j40.c.

References data_sequence_number, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), TXNCON, TXNCON_TXNTRIG, and uns8.

Here is the call graph for this function:




**void mrf24j40_transmit_callback (uns8 *status*, uns8 *retries*, uns8 *channel_busy*)**

Callback indicating the mrf24j40 has finished the transmission sequence.

**Parameters:**

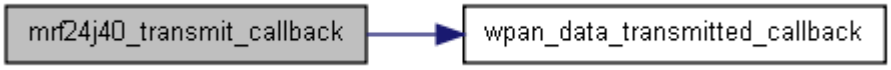
*status* Set to 0 for success or 1 for failure
*retries* Set to the number of retries
*channel_busy* Set to 1 if failure was due to the channel being busy.

Definition at line 75 of file wpan.c.

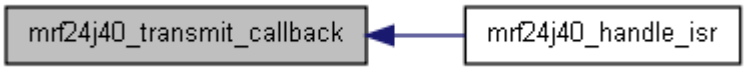
References debug_int, debug_str, and wpan_data_transmitted_callback().

Referenced by mrf24j40_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_transmit_to_extended_address (uns8 *frame_type*, uns16 *dest_pan_id*, uns8 * *dest_extended_address*, uns8 * *data*, uns8 *data_length*, uns8 *ack*)**

Requests that the mrf24j40 transmit the packet to the specified extended address

**Parameters:**

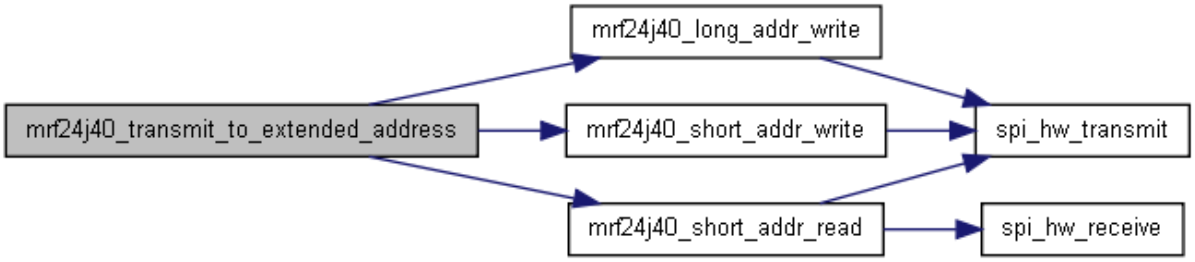
*frame_type* 802.15.4 frame type
*dest_pan_id* PAN id of destination
*dest_extended_address* Pointer to uns8 array indicating extended address of destination
*data* Pointer to uns8 array of bytes to transmit
*bytes_to_transmit*
*ack* Either MRF_ACK or MRF_NO_ACK depending if you want hardware acknowledgement

Definition at line 350 of file mrf24j40.c.

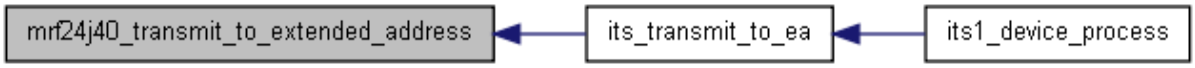
References data_sequence_number, extended_address, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), pan_id, TXNCON, TXNCON_TXNACKREQ, TXNCON_TXNTRIG, and uns8.

Referenced by its_transmit_to_ea().

Here is the call graph for this function:




Here is the caller graph for this function:




**void mrf24j40_transmit_to_short_address (uns8 *frame_type*, uns16 *dest_pan_id*, uns16 *dest_short_address*, uns8 * *data*, uns8 *bytes_to_transmit*, uns8 *ack*)**

Requests that the mrf24j40 transmit the packet to the specified short address

**Parameters:**

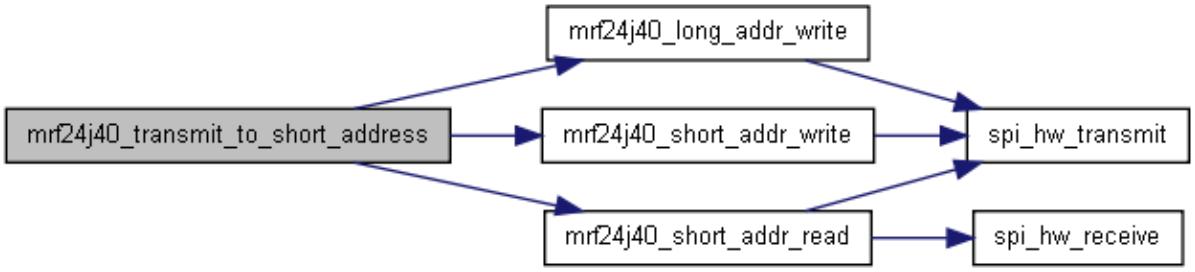
*frame_type* 802.15.4 frame type
*dest_pan_id* PAN id of destination
*dest_short_address* 16 bit short address of destination
*data* Pointer to uns8 array of bytes to transmit
*bytes_to_transmit*
*ack* Either MRF_ACK or MRF_NO_ACK depending if you want hardware acknowledgement

Definition at line 416 of file mrf24j40.c.

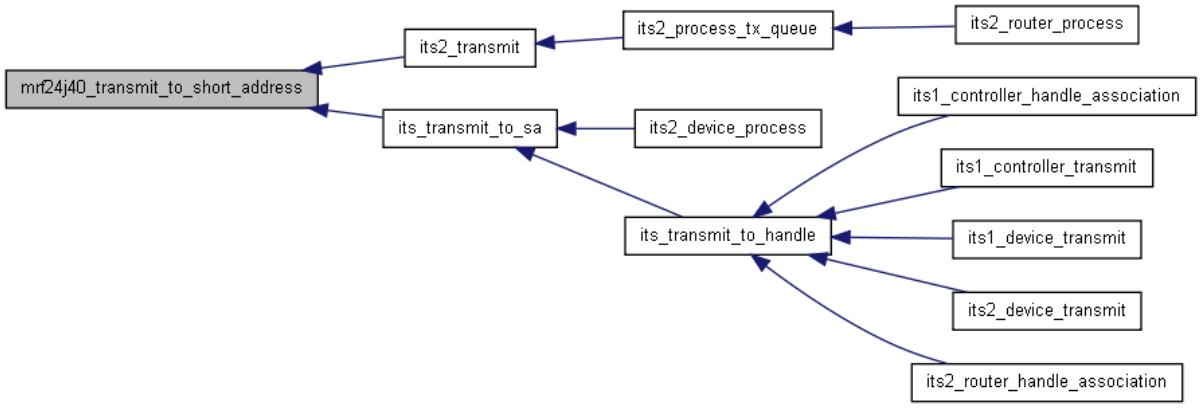
References data_sequence_number, mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), pan_id, short_address, TXNCON, TXNCON_TXNACKREQ, TXNCON_TXNTRIG, and uns8.

Referenced by its2_transmit(), and its_transmit_to_sa().

Here is the call graph for this function:



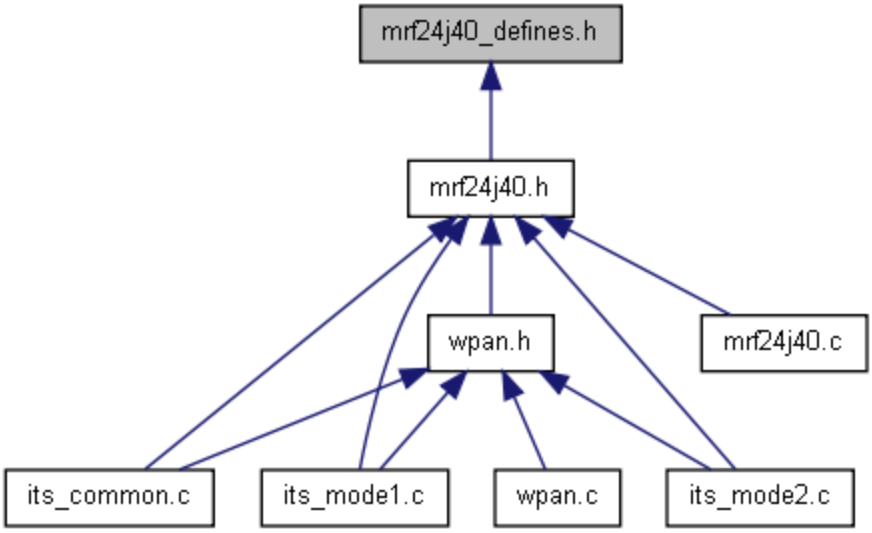

Here is the caller graph for this function:




---

# mrf24j40_defines.h File Reference

Defines for MRF24J40 chip - generated from the datasheet.
This graph shows which files directly or indirectly include this file:




## Defines

- #define ACKTMOUT 0x12
- #define ACKTMOUT_DRPACK 7
- #define ACKTMOUT_MAWD0 0
- #define ACKTMOUT_MAWD1 1
- #define ACKTMOUT_MAWD2 2
- #define ACKTMOUT_MAWD3 3
- #define ACKTMOUT_MAWD4 4
- #define ACKTMOUT_MAWD5 5
- #define ACKTMOUT_MAWD6 6

- #define ASSOEADR0 0x230
- #define ASSOEADR1 0x231
- #define ASSOEADR2 0x232
- #define ASSOEADR3 0x233
- #define ASSOEADR4 0x234
- #define ASSOEADR5 0x235
- #define ASSOEADR6 0x236
- #define ASSOEADR7 0x237
- #define ASSOSADR0 0x238
- #define ASSOSADR1 0x239
- #define BBREG0 0x38
- #define BBREG0_TURBO 0
- #define BBREG1 0x39
- #define BBREG1_RXDECINV 2
- #define BBREG2 0x3A
- #define BBREG2_CCACSTH0 2
- #define BBREG2_CCACSTH1 3
- #define BBREG2_CCACSTH2 4
- #define BBREG2_CCACSTH3 5
- #define BBREG2_CCAMODE0 6
- #define BBREG2_CCAMODE1 7
- #define BBREG3 0x3B
- #define BBREG3_PREDETTH0 1
- #define BBREG3_PREDETTH1 2
- #define BBREG3_PREDETTH2 3
- #define BBREG3_PREVALIDTH0 4
- #define BBREG3_PREVALIDTH1 5
- #define BBREG3_PREVALIDTH2 6
- #define BBREG3_PREVALIDTH3 7
- #define BBREG4 0x3C
- #define BBREG4_CSTH0 5
- #define BBREG4_CSTH1 6
- #define BBREG4_CSTH2 7
- #define BBREG4_PRECNT0 2
- #define BBREG4_PRECNT1 3
- #define BBREG4_PRECNT2 4
- #define BBREG6 0x3E
- #define BBREG6_RSSIMODE1 7
- #define BBREG6_RSSIMODE2 6
- #define BBREG6_RSSIRDY 0
- #define CCAEDTH 0x3F
- #define CCAEDTH_CCAEDTH0 0
- #define CCAEDTH_CCAEDTH1 1
- #define CCAEDTH_CCAEDTH2 2
- #define CCAEDTH_CCAEDTH3 3
- #define CCAEDTH_CCAEDTH4 4
- #define CCAEDTH_CCAEDTH5 5
- #define CCAEDTH_CCAEDTH6 6
- #define CCAEDTH_CCAEDTH7 7
- #define EADR0 0x05
- #define EADR1 0x06
- #define EADR2 0x07
- #define EADR3 0x08

- #define EADR4 0x09
- #define EADR5 0x0A
- #define EADR6 0x0B
- #define EADR7 0x0C
- #define ESLOTG1 0x13
- #define ESLOTG23 0x1E
- #define ESLOTG45 0x1F
- #define ESLOTG67 0x20
- #define FRMOFFSET 0x23
- #define FRMOFFSET_OFFSET0 0
- #define FRMOFFSET_OFFSET1 1
- #define FRMOFFSET_OFFSET2 2
- #define FRMOFFSET_OFFSET3 3
- #define FRMOFFSET_OFFSET4 4
- #define FRMOFFSET_OFFSET5 5
- #define FRMOFFSET_OFFSET6 6
- #define FRMOFFSET_OFFSET7 7
- #define GATECLK 0x26
- #define GATECLK_GTSON 3
- #define GPIO 0x33
- #define GPIO_GPIO0 0
- #define GPIO_GPIO1 1
- #define GPIO_GPIO2 2
- #define GPIO_GPIO3 3
- #define GPIO_GPIO4 4
- #define GPIO_GPIO5 5
- #define HSYMTMRH 0x29
- #define HSYMTMRH_HSYMTMR08 0
- #define HSYMTMRH_HSYMTMR09 1
- #define HSYMTMRH_HSYMTMR10 2
- #define HSYMTMRH_HSYMTMR11 3
- #define HSYMTMRH_HSYMTMR12 4
- #define HSYMTMRH_HSYMTMR13 5
- #define HSYMTMRH_HSYMTMR14 6
- #define HSYMTMRH_HSYMTMR15 7
- #define HSYMTMRL 0x28
- #define HSYMTMRL_HSYMTMR0 0
- #define HSYMTMRL_HSYMTMR1 1
- #define HSYMTMRL_HSYMTMR2 2
- #define HSYMTMRL_HSYMTMR3 3
- #define HSYMTMRL_HSYMTMR4 4
- #define HSYMTMRL_HSYMTMR5 5
- #define HSYMTMRL_HSYMTMR6 6
- #define HSYMTMRL_HSYMTMR7 7
- #define INTSTAT 0x31
- #define INTSTAT_HSYMTMRIF 5
- #define INTSTAT_RXIF 3
- #define INTSTAT_SECIF 4
- #define INTSTAT_SLPIF 7
- #define INTSTAT_TXG1IF 1
- #define INTSTAT_TXG2IF 2
- #define INTSTAT_TXNIF 0
- #define INTSTAT_WAKEIF 6

- #define MAINCNT0 0x226
- #define MAINCNT0_MAINCNT0 0
- #define MAINCNT0_MAINCNT1 1
- #define MAINCNT0_MAINCNT2 2
- #define MAINCNT0_MAINCNT3 3
- #define MAINCNT0_MAINCNT4 4
- #define MAINCNT0_MAINCNT5 5
- #define MAINCNT0_MAINCNT6 6
- #define MAINCNT0_MAINCNT7 7
- #define MAINCNT1 0x227
- #define MAINCNT1_MAINCNT10 2
- #define MAINCNT1_MAINCNT11 3
- #define MAINCNT1_MAINCNT12 4
- #define MAINCNT1_MAINCNT13 5
- #define MAINCNT1_MAINCNT14 6
- #define MAINCNT1_MAINCNT15 7
- #define MAINCNT1_MAINCNT8 0
- #define MAINCNT1_MAINCNT9 1
- #define MAINCNT2 0x228
- #define MAINCNT2_MAINCNT16 0
- #define MAINCNT2_MAINCNT17 1
- #define MAINCNT2_MAINCNT18 2
- #define MAINCNT2_MAINCNT19 3
- #define MAINCNT2_MAINCNT20 4
- #define MAINCNT2_MAINCNT21 5
- #define MAINCNT2_MAINCNT22 6
- #define MAINCNT2_MAINCNT23 7
- #define MAINCNT3 0x229
- #define MAINCNT3_MAINCNT24 0
- #define MAINCNT3_MAINCNT25 1
- #define MAINCNT3_STARTCNT 7
- #define MRF_INTCON 0x32
- #define MRF_INTCON_HSYMTMRIE 5
- #define MRF_INTCON_RXIE 3
- #define MRF_INTCON_SECIE 4
- #define MRF_INTCON_SLPIE 7
- #define MRF_INTCON_TXG1IE 1
- #define MRF_INTCON_TXG2IE 2
- #define MRF_INTCON_TXNIE 0
- #define MRF_INTCON_WAKEIE 6
- #define ORDER 0x10
- #define ORDER_BO0 4
- #define ORDER_BO1 5
- #define ORDER_BO2 6
- #define ORDER_BO3 7
- #define ORDER_SO0 0
- #define ORDER_SO1 1
- #define ORDER_SO2 2
- #define ORDER_SO3 3
- #define PACON0 0x16
- #define PACON0_PAONT0 0
- #define PACON0_PAONT1 1
- #define PACON0_PAONT2 2

- #define PACON0_PAONT3 3
- #define PACON0_PAONT4 4
- #define PACON0_PAONT5 5
- #define PACON0_PAONT6 6
- #define PACON0_PAONT7 7
- #define PACON1 0x17
- #define PACON1_PAONT8 0
- #define PACON1_PAONTS0 1
- #define PACON1_PAONTS1 2
- #define PACON1_PAONTS2 3
- #define PACON1_PAONTS3 4
- #define PACON2 0x18
- #define PACON2_FIFOEN 7
- #define PACON2_TXONT7 0
- #define PACON2_TXONT8 1
- #define PACON2_TXONTS0 2
- #define PACON2_TXONTS1 3
- #define PACON2_TXONTS2 4
- #define PACON2_TXONTS3 5
- #define PANIDH 0x02
- #define PANIDL 0x01
- #define REMCNTH 0x225
- #define REMCNTH_REMCNT10 2
- #define REMCNTH_REMCNT11 3
- #define REMCNTH_REMCNT12 4
- #define REMCNTH_REMCNT13 5
- #define REMCNTH_REMCNT14 6
- #define REMCNTH_REMCNT15 7
- #define REMCNTH_REMCNT8 0
- #define REMCNTH_REMCNT9 1
- #define REMCNTL 0x224
- #define REMCNTL_REMCNT0 0
- #define REMCNTL_REMCNT1 1
- #define REMCNTL_REMCNT2 2
- #define REMCNTL_REMCNT3 3
- #define REMCNTL_REMCNT4 4
- #define REMCNTL_REMCNT5 5
- #define REMCNTL_REMCNT6 6
- #define REMCNTL_REMCNT7 7
- #define RFCON0 0x200
- #define RFCON0_CHANNEL0 4
- #define RFCON0_CHANNEL1 5
- #define RFCON0_CHANNEL2 6
- #define RFCON0_CHANNEL3 7
- #define RFCON0_RFOPT0 0
- #define RFCON0_RFOPT1 1
- #define RFCON0_RFOPT2 2
- #define RFCON0_RFOPT3 3
- #define RFCON1 0x201
- #define RFCON1_VCOOPT0 0
- #define RFCON1_VCOOPT1 1
- #define RFCON1_VCOOPT2 2
- #define RFCON1_VCOOPT3 3

- #define RFCON1_VCOOPT4 4
- #define RFCON1_VCOOPT5 5
- #define RFCON1_VCOOPT6 6
- #define RFCON1_VCOOPT7 7
- #define RFCON2 0x202
- #define RFCON2_PLLEN 7
- #define RFCON3 0x203
- #define RFCON3_TXPWRL0 6
- #define RFCON3_TXPWRL1 7
- #define RFCON3_TXPWRS0 3
- #define RFCON3_TXPWRS1 4
- #define RFCON3_TXPWRS2 5
- #define RFCON5 0x205
- #define RFCON5_BATTH0 4
- #define RFCON5_BATTH1 5
- #define RFCON5_BATTH2 6
- #define RFCON5_BATTH3 7
- #define RFCON6 0x206
- #define RFCON6_20MRECVR 4
- #define RFCON6_BATEN 3
- #define RFCON6_TXFIL 7
- #define RFCON7 0x207
- #define RFCON7_CLKOUTMODE0 0
- #define RFCON7_CLKOUTMODE1 1
- #define RFCON7_SLPCLKSEL0 6
- #define RFCON7_SLPCLKSEL1 7
- #define RFCON8 0x208
- #define RFCON8_RFVCO 4
- #define RFCTL 0x36
- #define RFCTL_RFRST 2
- #define RFCTL_WAKECNT7 3
- #define RFCTL_WAKECNT8 4
- #define RFSTATE 0x20F
- #define RFSTATE_RFSTATE0 5
- #define RFSTATE_RFSTATE1 6
- #define RFSTATE_RFSTATE2 7
- #define RSSI 0x210
- #define RSSI_RSSI0 0
- #define RSSI_RSSI1 1
- #define RSSI_RSSI2 2
- #define RSSI_RSSI3 3
- #define RSSI_RSSI4 4
- #define RSSI_RSSI5 5
- #define RSSI_RSSI6 6
- #define RSSI_RSSI7 7
- #define RXFLUSH 0x0D
- #define RXFLUSH_BCNONLY 1
- #define RXFLUSH_CMDONLY 3
- #define RXFLUSH_DATAONLY 2
- #define RXFLUSH_RXFLUSH 0
- #define RXFLUSH_WAKEPAD 5
- #define RXFLUSH_WAKEPOL 6
- #define RXMCR 0x00

- #define RXMCR_COORD 2
- #define RXMCR_ERRPKT 1
- #define RXMCR_NOACKRSP 5
- #define RXMCR_PANCOORD 3
- #define RXMCR_PROMI 0
- #define RXSR 0x30
- #define RXSR_BATIND 5
- #define RXSR_UPSECERR 6
- #define SADRH 0x04
- #define SADRL 0x03
- #define SECCON0 0x2C
- #define SECCON0_RXCIPHER0 3
- #define SECCON0_RXCIPHER1 4
- #define SECCON0_RXCIPHER2 5
- #define SECCON0_SECIGNORE 7
- #define SECCON0_SECSTART 6
- #define SECCON0_TXNCIPHER0 0
- #define SECCON0_TXNCIPHER1 1
- #define SECCON0_TXNCIPHER2 2
- #define SECCON1 0x2D
- #define SECCON1_DISDEC 1
- #define SECCON1_DISENC 0
- #define SECCON1_TXBCIPHER0 4
- #define SECCON1_TXBCIPHER1 5
- #define SECCON1_TXBCIPHER2 6
- #define SECCR2 0x37
- #define SECCR2_TXG1CIPHER0 0
- #define SECCR2_TXG1CIPHER1 1
- #define SECCR2_TXG1CIPHER2 2
- #define SECCR2_TXG2CIPHER0 3
- #define SECCR2_TXG2CIPHER1 4
- #define SECCR2_TXG2CIPHER2 5
- #define SECCR2_UPDEC 7
- #define SECCR2_UPENC 6
- #define SLPACK 0x35
- #define SLPACK_SLPACK 7
- #define SLPACK_WAKECNT0 0
- #define SLPACK_WAKECNT1 1
- #define SLPACK_WAKECNT2 2
- #define SLPACK_WAKECNT3 3
- #define SLPACK_WAKECNT4 4
- #define SLPACK_WAKECNT5 5
- #define SLPACK_WAKECNT6 6
- #define SLPCAL0 0x209
- #define SLPCAL0_SLPCAL0 0
- #define SLPCAL0_SLPCAL1 1
- #define SLPCAL0_SLPCAL2 2
- #define SLPCAL0_SLPCAL3 3
- #define SLPCAL0_SLPCAL4 4
- #define SLPCAL0_SLPCAL5 5
- #define SLPCAL0_SLPCAL6 6
- #define SLPCAL0_SLPCAL7 7
- #define SLPCAL1 0x20A

- #define SLPCAL1_SLPCAL10 2
- #define SLPCAL1_SLPCAL11 3
- #define SLPCAL1_SLPCAL12 4
- #define SLPCAL1_SLPCAL13 5
- #define SLPCAL1_SLPCAL14 6
- #define SLPCAL1_SLPCAL15 7
- #define SLPCAL1_SLPCAL8 0
- #define SLPCAL1_SLPCAL9 1
- #define SLPCAL2 0x20B
- #define SLPCAL2_SLPCAL16 0
- #define SLPCAL2_SLPCAL17 1
- #define SLPCAL2_SLPCAL18 2
- #define SLPCAL2_SLPCAL19 3
- #define SLPCAL2_SLPCALEN 4
- #define SLPCAL2_SLPCALRDY 7
- #define SLPCON0 0x211
- #define SLPCON0_INTEDGE 1
- #define SLPCON0_SLPCLKEN 0
- #define SLPCON1 0x220
- #define SLPCON1_CLKOUTEN 5
- #define SLPCON1_SLPCLKDIV0 0
- #define SLPCON1_SLPCLKDIV1 1
- #define SLPCON1_SLPCLKDIV2 2
- #define SLPCON1_SLPCLKDIV3 3
- #define SLPCON1_SLPCLKDIV4 4
- #define SOFTRST 0x2A
- #define SOFTRST_RSTBB 1
- #define SOFTRST_RSTMAC 0
- #define SOFTRST_RSTPWR 2
- #define SYMTICKH 0x15
- #define SYMTICKH_TICKP8 0
- #define SYMTICKH_TXONT0 1
- #define SYMTICKH_TXONT1 2
- #define SYMTICKH_TXONT2 3
- #define SYMTICKH_TXONT3 4
- #define SYMTICKH_TXONT4 5
- #define SYMTICKH_TXONT5 6
- #define SYMTICKH_TXONT6 7
- #define SYMTICKL 0x14
- #define SYMTICKL_TICKP0 0
- #define SYMTICKL_TICKP1 1
- #define SYMTICKL_TICKP2 2
- #define SYMTICKL_TICKP3 3
- #define SYMTICKL_TICKP4 4
- #define SYMTICKL_TICKP5 5
- #define SYMTICKL_TICKP6 6
- #define SYMTICKL_TICKP7 7
- #define TESTMODE 0x22F
- #define TESTMODE_RSSIWAIT0 3
- #define TESTMODE_RSSIWAIT1 4
- #define TESTMODE_TESTMODE0 0
- #define TESTMODE_TESTMODE1 1
- #define TESTMODE_TESTMODE2 2

- #define TRISGPIO 0x34
- #define TRISGPIO_TRISGP0 0
- #define TRISGPIO_TRISGP1 1
- #define TRISGPIO_TRISGP2 2
- #define TRISGPIO_TRISGP3 3
- #define TRISGPIO_TRISGP4 4
- #define TRISGPIO_TRISGP5 5
- #define TXBCON0 0x1A
- #define TXBCON0_TXBSECEN 1
- #define TXBCON0_TXBTRIG 0
- #define TXBCON1 0x25
- #define TXG1CON 0x1C
- #define TXG1CON_TXG1ACKREQ 2
- #define TXG1CON_TXG1RETRY0 6
- #define TXG1CON_TXG1RETRY1 7
- #define TXG1CON_TXG1SECEN 1
- #define TXG1CON_TXG1SLOT0 3
- #define TXG1CON_TXG1SLOT1 4
- #define TXG1CON_TXG1SLOT2 5
- #define TXG1CON_TXG1TRIG 0
- #define TXG2CON 0x1D
- #define TXG2CON_TXG2ACKREQ 2
- #define TXG2CON_TXG2RETRY0 6
- #define TXG2CON_TXG2RETRY1 7
- #define TXG2CON_TXG2SECEN 1
- #define TXG2CON_TXG2SLOT0 3
- #define TXG2CON_TXG2SLOT1 4
- #define TXG2CON_TXG2SLOT2 5
- #define TXG2CON_TXG2TRIG 0
- #define TXMCR 0x11
- #define TXMCR_BATLIFEXT 6
- #define TXMCR_CSMABF0 0
- #define TXMCR_CSMABF1 1
- #define TXMCR_CSMABF2 2
- #define TXMCR_MACMINBE0 3
- #define TXMCR_MACMINBE1 4
- #define TXMCR_NOCSMA 7
- #define TXMCR_SLOTTED 5
- #define TXNCON 0x1B
- #define TXNCON_FPSTAT 4
- #define TXNCON_INDIRECT 3
- #define TXNCON_TXNACKREQ 2
- #define TXNCON_TXNSECEN 1
- #define TXNCON_TXNTRIG 0
- #define TXPEND 0x21
- #define TXPEND_FPACK 0
- #define TXPEND_GTSSWITCH 1
- #define TXPEND_MLIFS0 2
- #define TXPEND_MLIFS1 3
- #define TXPEND_MLIFS2 4
- #define TXPEND_MLIFS3 5
- #define TXPEND_MLIFS4 6
- #define TXPEND_MLIFS5 7

- #define TXSTAT 0x24
- #define TXSTAT_CCAFAIL 5
- #define TXSTAT_TXG1FNT 3
- #define TXSTAT_TXG1STAT 1
- #define TXSTAT_TXG2FNT 4
- #define TXSTAT_TXG2STAT 2
- #define TXSTAT_TXNRETRY0 6
- #define TXSTAT_TXNRETRY1 7
- #define TXSTAT_TXNSTAT 0
- #define TXSTBL 0x2E
- #define TXSTBL_MSIFS0 0
- #define TXSTBL_MSIFS1 1
- #define TXSTBL_MSIFS2 2
- #define TXSTBL_MSIFS3 3
- #define TXSTBL_RFSTBL0 4
- #define TXSTBL_RFSTBL1 5
- #define TXSTBL_RFSTBL2 6
- #define TXSTBL_RFSTBL3 7
- #define TXTIME 0x27
- #define TXTIME_TURNTIME0 4
- #define TXTIME_TURNTIME1 5
- #define TXTIME_TURNTIME2 6
- #define TXTIME_TURNTIME3 7
- #define UPNONCE0 0x240
- #define UPNONCE1 0x241
- #define UPNONCE10 0x24A
- #define UPNONCE11 0x24B
- #define UPNONCE12 0x24C
- #define UPNONCE2 0x242
- #define UPNONCE3 0x243
- #define UPNONCE4 0x244
- #define UPNONCE5 0x245
- #define UPNONCE6 0x246
- #define UPNONCE7 0x247
- #define UPNONCE8 0x248
- #define UPNONCE9 0x249
- #define WAKECON 0x22
- #define WAKECON_IMMWAKE 7
- #define WAKECON_REGWAKE 6
- #define WAKETIMEH 0x223
- #define WAKETIMEH_WAKETIME10 2
- #define WAKETIMEH_WAKETIME8 0
- #define WAKETIMEH_WAKETIME9 1
- #define WAKETIMEL 0x222
- #define WAKETIMEL_WAKETIME0 0
- #define WAKETIMEL_WAKETIME1 1
- #define WAKETIMEL_WAKETIME2 2
- #define WAKETIMEL_WAKETIME3 3
- #define WAKETIMEL_WAKETIME4 4
- #define WAKETIMEL_WAKETIME5 5
- #define WAKETIMEL_WAKETIME6 6
- #define WAKETIMEL_WAKETIME7 7

## Detailed Description

Definition in file mrf24j40_defines.h.

## Define Documentation

### #define ACKTMOUT 0x12

Definition at line 102 of file mrf24j40_defines.h.

### #define ACKTMOUT_DRPACK 7

Definition at line 105 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD0 0

Definition at line 112 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD1 1

Definition at line 111 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD2 2

Definition at line 110 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD3 3

Definition at line 109 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD4 4

Definition at line 108 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD5 5

Definition at line 107 of file mrf24j40_defines.h.

### #define ACKTMOUT_MAWD6 6

Definition at line 106 of file mrf24j40_defines.h.

**#define ASSOEADR0 0x230**

Definition at line 867 of file mrf24j40_defines.h.

**#define ASSOEADR1 0x231**

Definition at line 868 of file mrf24j40_defines.h.

**#define ASSOEADR2 0x232**

Definition at line 869 of file mrf24j40_defines.h.

**#define ASSOEADR3 0x233**

Definition at line 870 of file mrf24j40_defines.h.

**#define ASSOEADR4 0x234**

Definition at line 871 of file mrf24j40_defines.h.

**#define ASSOEADR5 0x235**

Definition at line 872 of file mrf24j40_defines.h.

**#define ASSOEADR6 0x236**

Definition at line 873 of file mrf24j40_defines.h.

**#define ASSOEADR7 0x237**

Definition at line 874 of file mrf24j40_defines.h.

**#define ASSOSADR0 0x238**

Definition at line 875 of file mrf24j40_defines.h.

**#define ASSOSADR1 0x239**

Definition at line 876 of file mrf24j40_defines.h.

**#define BBREG0 0x38**

Definition at line 470 of file mrf24j40_defines.h.

**#define BBREG0_TURBO 0**

Definition at line 480 of file mrf24j40_defines.h.

**#define BBREG1 0x39**

Definition at line 482 of file mrf24j40_defines.h.
Referenced by mrf24j40_receive().

**#define BBREG1_RXDECINV 2**

Definition at line 490 of file mrf24j40_defines.h.
Referenced by mrf24j40_receive().

**#define BBREG2 0x3A**

Definition at line 494 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define BBREG2_CCACSTH0 2**

Definition at line 502 of file mrf24j40_defines.h.

**#define BBREG2_CCACSTH1 3**

Definition at line 501 of file mrf24j40_defines.h.

**#define BBREG2_CCACSTH2 4**

Definition at line 500 of file mrf24j40_defines.h.

**#define BBREG2_CCACSTH3 5**

Definition at line 499 of file mrf24j40_defines.h.

**#define BBREG2_CCAMODE0 6**

Definition at line 498 of file mrf24j40_defines.h.

**#define BBREG2_CCAMODE1 7**

Definition at line 497 of file mrf24j40_defines.h.

**#define BBREG3 0x3B**

Definition at line 506 of file mrf24j40_defines.h.

**#define BBREG3_PREDETTH0 1**

Definition at line 515 of file mrf24j40_defines.h.

**#define BBREG3_PREDETTH1 2**

Definition at line 514 of file mrf24j40_defines.h.

**#define BBREG3_PREDETTH2 3**

Definition at line 513 of file mrf24j40_defines.h.

**#define BBREG3_PREVALIDTH0 4**

Definition at line 512 of file mrf24j40_defines.h.

**#define BBREG3_PREVALIDTH1 5**

Definition at line 511 of file mrf24j40_defines.h.

**#define BBREG3_PREVALIDTH2 6**

Definition at line 510 of file mrf24j40_defines.h.

**#define BBREG3_PREVALIDTH3 7**

Definition at line 509 of file mrf24j40_defines.h.

**#define BBREG4 0x3C**

Definition at line 518 of file mrf24j40_defines.h.

**#define BBREG4_CSTH0 5**

Definition at line 523 of file mrf24j40_defines.h.

**#define BBREG4_CSTH1 6**

Definition at line 522 of file mrf24j40_defines.h.

**#define BBREG4_CSTH2 7**

Definition at line 521 of file mrf24j40_defines.h.

**#define BBREG4_PRECNT0 2**

Definition at line 526 of file mrf24j40_defines.h.

**#define BBREG4_PRECNT1 3**

Definition at line 525 of file mrf24j40_defines.h.

**#define BBREG4_PRECNT2 4**

Definition at line 524 of file mrf24j40_defines.h.

**#define BBREG6 0x3E**

Definition at line 531 of file mrf24j40_defines.h.

Referenced by mrf24j40_active_channel_scan(), mrf24j40_init(), mrf24j40_init_coordinator(), and mrf24j40_scan_for_lowest_channel_ed().

**#define BBREG6_RSSIMODE1 7**

Definition at line 534 of file mrf24j40_defines.h.

Referenced by mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

**#define BBREG6_RSSIMODE2 6**

Definition at line 535 of file mrf24j40_defines.h.

Referenced by mrf24j40_init(), and mrf24j40_scan_for_lowest_channel_ed().

**#define BBREG6_RSSIRDY 0**

Definition at line 541 of file mrf24j40_defines.h.

Referenced by mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

**#define CCAEDTH 0x3F**

Definition at line 543 of file mrf24j40_defines.h.

Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define CCAEDTH_CCAEDTH0 0**

Definition at line 553 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH1 1**

Definition at line 552 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH2 2**

Definition at line 551 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH3 3**

Definition at line 550 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH4 4**

Definition at line 549 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH5 5**

Definition at line 548 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH6 6**

Definition at line 547 of file mrf24j40_defines.h.

**#define CCAEDTH_CCAEDTH7 7**

Definition at line 546 of file mrf24j40_defines.h.

**#define EADR0 0x05**

Definition at line 56 of file mrf24j40_defines.h.

**#define EADR1 0x06**

Definition at line 57 of file mrf24j40_defines.h.

**#define EADR2 0x07**

Definition at line 58 of file mrf24j40_defines.h.

**#define EADR3 0x08**

Definition at line 59 of file mrf24j40_defines.h.

**#define EADR4 0x09**

Definition at line 60 of file mrf24j40_defines.h.

**#define EADR5 0x0A**

Definition at line 61 of file mrf24j40_defines.h.

**#define EADR6 0x0B**

Definition at line 62 of file mrf24j40_defines.h.

**#define EADR7 0x0C**

Definition at line 63 of file mrf24j40_defines.h.
Referenced by mrf24j40_set_extended_address().

**#define ESLOTG1 0x13**

Definition at line 114 of file mrf24j40_defines.h.

**#define ESLOTG23 0x1E**

Definition at line 224 of file mrf24j40_defines.h.

**#define ESLOTG45 0x1F**

Definition at line 225 of file mrf24j40_defines.h.

**#define ESLOTG67 0x20**

Definition at line 226 of file mrf24j40_defines.h.

**#define FRMOFFSET 0x23**

Definition at line 251 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET0 0**

Definition at line 261 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET1 1**

Definition at line 260 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET2 2**

Definition at line 259 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET3 3**

Definition at line 258 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET4 4**

Definition at line 257 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET5 5**

Definition at line 256 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET6 6**

Definition at line 255 of file mrf24j40_defines.h.

**#define FRMOFFSET_OFFSET7 7**

Definition at line 254 of file mrf24j40_defines.h.

**#define GATECLK 0x26**

Definition at line 276 of file mrf24j40_defines.h.

**#define GATECLK_GTSON 3**

Definition at line 283 of file mrf24j40_defines.h.

**#define GPIO 0x33**

Definition at line 410 of file mrf24j40_defines.h.
Referenced by mrf24j40_scan_for_lowest_channel_ed().

**#define GPIO_GPIO0 0**

Definition at line 420 of file mrf24j40_defines.h.

**#define GPIO_GPIO1 1**

Definition at line 419 of file mrf24j40_defines.h.

**#define GPIO_GPIO2 2**

Definition at line 418 of file mrf24j40_defines.h.

**#define GPIO_GPIO3 3**

Definition at line 417 of file mrf24j40_defines.h.

**#define GPIO_GPIO4 4**

Definition at line 416 of file mrf24j40_defines.h.

**#define GPIO_GPIO5 5**

Definition at line 415 of file mrf24j40_defines.h.

**#define HSYMTMRH 0x29**

Definition at line 312 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR08 0**

Definition at line 322 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR09 1**

Definition at line 321 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR10 2**

Definition at line 320 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR11 3**

Definition at line 319 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR12 4**

Definition at line 318 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR13 5**

Definition at line 317 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR14 6**

Definition at line 316 of file mrf24j40_defines.h.

**#define HSYMTMRH_HSYMTMR15 7**

Definition at line 315 of file mrf24j40_defines.h.

**#define HSYMTMRL 0x28**

Definition at line 300 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR0 0**

Definition at line 310 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR1 1**

Definition at line 309 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR2 2**

Definition at line 308 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR3 3**

Definition at line 307 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR4 4**

Definition at line 306 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR5 5**

Definition at line 305 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR6 6**

Definition at line 304 of file mrf24j40_defines.h.

**#define HSYMTMRL_HSYMTMR7 7**

Definition at line 303 of file mrf24j40_defines.h.

**#define INTSTAT 0x31**

Definition at line 386 of file mrf24j40_defines.h.
Referenced by mrf24j40_handle_isr().

**#define INTSTAT_HSYMTMRIF 5**

Definition at line 391 of file mrf24j40_defines.h.

**#define INTSTAT_RXIF 3**

Definition at line 393 of file mrf24j40_defines.h.
Referenced by mrf24j40_handle_isr().

**#define INTSTAT_SECIF 4**

Definition at line 392 of file mrf24j40_defines.h.

**#define INTSTAT_SLPIF 7**

Definition at line 389 of file mrf24j40_defines.h.

**#define INTSTAT_TXG1IF 1**

Definition at line 395 of file mrf24j40_defines.h.

**#define INTSTAT_TXG2IF 2**

Definition at line 394 of file mrf24j40_defines.h.

**#define INTSTAT_TXNIF 0**

Definition at line 396 of file mrf24j40_defines.h.
Referenced by mrf24j40_handle_isr().

**#define INTSTAT_WAKEIF 6**

Definition at line 390 of file mrf24j40_defines.h.

**#define MAINCNT0 0x226**

Definition at line 802 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT0 0**

Definition at line 812 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT1 1**

Definition at line 811 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT2 2**

Definition at line 810 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT3 3**

Definition at line 809 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT4 4**

Definition at line 808 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT5 5**

Definition at line 807 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT6 6**

Definition at line 806 of file mrf24j40_defines.h.

**#define MAINCNT0_MAINCNT7 7**

Definition at line 805 of file mrf24j40_defines.h.

**#define MAINCNT1 0x227**

Definition at line 814 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT10 2**

Definition at line 822 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT11 3**

Definition at line 821 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT12 4**

Definition at line 820 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT13 5**

Definition at line 819 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT14 6**

Definition at line 818 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT15 7**

Definition at line 817 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT8 0**

Definition at line 824 of file mrf24j40_defines.h.

**#define MAINCNT1_MAINCNT9 1**

Definition at line 823 of file mrf24j40_defines.h.

**#define MAINCNT2 0x228**

Definition at line 826 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT16 0**

Definition at line 836 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT17 1**

Definition at line 835 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT18 2**

Definition at line 834 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT19 3**

Definition at line 833 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT20 4**

Definition at line 832 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT21 5**

Definition at line 831 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT22 6**

Definition at line 830 of file mrf24j40_defines.h.

**#define MAINCNT2_MAINCNT23 7**

Definition at line 829 of file mrf24j40_defines.h.

**#define MAINCNT3 0x229**

Definition at line 838 of file mrf24j40_defines.h.

**#define MAINCNT3_MAINCNT24 0**

Definition at line 848 of file mrf24j40_defines.h.

**#define MAINCNT3_MAINCNT25 1**

Definition at line 847 of file mrf24j40_defines.h.

**#define MAINCNT3_STARTCNT 7**

Definition at line 841 of file mrf24j40_defines.h.

**#define MRF_INTCON 0x32**

Definition at line 398 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define MRF_INTCON_HSYMTMRIE 5**

Definition at line 403 of file mrf24j40_defines.h.

**#define MRF_INTCON_RXIE 3**

Definition at line 405 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define MRF_INTCON_SECIE 4**

Definition at line 404 of file mrf24j40_defines.h.

**#define MRF_INTCON_SLPIE 7**

Definition at line 401 of file mrf24j40_defines.h.

**#define MRF_INTCON_TXG1IE 1**

Definition at line 407 of file mrf24j40_defines.h.

**#define MRF_INTCON_TXG2IE 2**

Definition at line 406 of file mrf24j40_defines.h.

**#define MRF_INTCON_TXNIE 0**

Definition at line 408 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define MRF_INTCON_WAKEIE 6**

Definition at line 402 of file mrf24j40_defines.h.

**#define ORDER 0x10**

Definition at line 78 of file mrf24j40_defines.h.

**#define ORDER_BO0 4**

Definition at line 84 of file mrf24j40_defines.h.

**#define ORDER_BO1 5**

Definition at line 83 of file mrf24j40_defines.h.

**#define ORDER_BO2 6**

Definition at line 82 of file mrf24j40_defines.h.

**#define ORDER_BO3 7**

Definition at line 81 of file mrf24j40_defines.h.

**#define ORDER_SO0 0**

Definition at line 88 of file mrf24j40_defines.h.

**#define ORDER_SO1 1**

Definition at line 87 of file mrf24j40_defines.h.

**#define ORDER_SO2 2**

Definition at line 86 of file mrf24j40_defines.h.

**#define ORDER_SO3 3**

Definition at line 85 of file mrf24j40_defines.h.

**#define PACON0 0x16**

Definition at line 139 of file mrf24j40_defines.h.

**#define PACON0_PAONT0 0**

Definition at line 149 of file mrf24j40_defines.h.

**#define PACON0_PAONT1 1**

Definition at line 148 of file mrf24j40_defines.h.

**#define PACON0_PAONT2 2**

Definition at line 147 of file mrf24j40_defines.h.

**#define PACON0_PAONT3 3**

Definition at line 146 of file mrf24j40_defines.h.

**#define PACON0_PAONT4 4**

Definition at line 145 of file mrf24j40_defines.h.

**#define PACON0_PAONT5 5**

Definition at line 144 of file mrf24j40_defines.h.

**#define PACON0_PAONT6 6**

Definition at line 143 of file mrf24j40_defines.h.

**#define PACON0_PAONT7 7**

Definition at line 142 of file mrf24j40_defines.h.

**#define PACON1 0x17**

Definition at line 151 of file mrf24j40_defines.h.

**#define PACON1_PAONT8 0**

Definition at line 161 of file mrf24j40_defines.h.

**#define PACON1_PAONTS0 1**

Definition at line 160 of file mrf24j40_defines.h.

**#define PACON1_PAONTS1 2**

Definition at line 159 of file mrf24j40_defines.h.

**#define PACON1_PAONTS2 3**

Definition at line 158 of file mrf24j40_defines.h.

**#define PACON1_PAONTS3 4**

Definition at line 157 of file mrf24j40_defines.h.

**#define PACON2 0x18**

Definition at line 163 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define PACON2_FIFOEN 7**

Definition at line 166 of file mrf24j40_defines.h.

**#define PACON2_TXONT7 0**

Definition at line 173 of file mrf24j40_defines.h.

**#define PACON2_TXONT8 1**

Definition at line 172 of file mrf24j40_defines.h.

**#define PACON2_TXONTS0 2**

Definition at line 171 of file mrf24j40_defines.h.

**#define PACON2_TXONTS1 3**

Definition at line 170 of file mrf24j40_defines.h.

**#define PACON2_TXONTS2 4**

Definition at line 169 of file mrf24j40_defines.h.

**#define PACON2_TXONTS3 5**

Definition at line 168 of file mrf24j40_defines.h.

**#define PANIDH 0x02**

Definition at line 53 of file mrf24j40_defines.h.
Referenced by mrf24j40_set_pan_id().

**#define PANIDL 0x01**

Definition at line 52 of file mrf24j40_defines.h.
Referenced by mrf24j40_set_pan_id().

**#define REMCNTH 0x225**

Definition at line 790 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT10 2**

Definition at line 798 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT11 3**

Definition at line 797 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT12 4**

Definition at line 796 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT13 5**

Definition at line 795 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT14 6**

Definition at line 794 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT15 7**

Definition at line 793 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT8 0**

Definition at line 800 of file mrf24j40_defines.h.

**#define REMCNTH_REMCNT9 1**

Definition at line 799 of file mrf24j40_defines.h.

**#define REMCNTL 0x224**

Definition at line 778 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT0 0**

Definition at line 788 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT1 1**

Definition at line 787 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT2 2**

Definition at line 786 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT3 3**

Definition at line 785 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT4 4**

Definition at line 784 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT5 5**

Definition at line 783 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT6 6**

Definition at line 782 of file mrf24j40_defines.h.

**#define REMCNTL_REMCNT7 7**

Definition at line 781 of file mrf24j40_defines.h.

**#define RFCON0 0x200**

Definition at line 555 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), mrf24j40_init_coordinator(), and mrf24j40_set_channel().

**#define RFCON0_CHANNEL0 4**

Definition at line 561 of file mrf24j40_defines.h.

**#define RFCON0_CHANNEL1 5**

Definition at line 560 of file mrf24j40_defines.h.

**#define RFCON0_CHANNEL2 6**

Definition at line 559 of file mrf24j40_defines.h.

**#define RFCON0_CHANNEL3 7**

Definition at line 558 of file mrf24j40_defines.h.

**#define RFCON0_RFOPT0 0**

Definition at line 565 of file mrf24j40_defines.h.

**#define RFCON0_RFOPT1 1**

Definition at line 564 of file mrf24j40_defines.h.

**#define RFCON0_RFOPT2 2**

Definition at line 563 of file mrf24j40_defines.h.

**#define RFCON0_RFOPT3 3**

Definition at line 562 of file mrf24j40_defines.h.

**#define RFCON1 0x201**

Definition at line 567 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define RFCON1_VCOOPT0 0**

Definition at line 577 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT1 1**

Definition at line 576 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT2 2**

Definition at line 575 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT3 3**

Definition at line 574 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT4 4**

Definition at line 573 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT5 5**

Definition at line 572 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT6 6**

Definition at line 571 of file mrf24j40_defines.h.

**#define RFCON1_VCOOPT7 7**

Definition at line 570 of file mrf24j40_defines.h.

**#define RFCON2 0x202**

Definition at line 579 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define RFCON2_PLLEN 7**

Definition at line 582 of file mrf24j40_defines.h.

**#define RFCON3 0x203**

Definition at line 591 of file mrf24j40_defines.h.
Referenced by mrf24j40_init().

**#define RFCON3_TXPWRL0 6**

Definition at line 595 of file mrf24j40_defines.h.

**#define RFCON3_TXPWRL1 7**

Definition at line 594 of file mrf24j40_defines.h.

**#define RFCON3_TXPWRS0 3**

Definition at line 598 of file mrf24j40_defines.h.

**#define RFCON3_TXPWRS1 4**

Definition at line 597 of file mrf24j40_defines.h.

**#define RFCON3_TXPWRS2 5**

Definition at line 596 of file mrf24j40_defines.h.

**#define RFCON5 0x205**

Definition at line 604 of file mrf24j40_defines.h.

**#define RFCON5_BATTH0 4**

Definition at line 610 of file mrf24j40_defines.h.

**#define RFCON5_BATTH1 5**

Definition at line 609 of file mrf24j40_defines.h.

**#define RFCON5_BATTH2 6**

Definition at line 608 of file mrf24j40_defines.h.

**#define RFCON5_BATTH3 7**

Definition at line 607 of file mrf24j40_defines.h.

**#define RFCON6 0x206**

Definition at line 616 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define RFCON6_20MRECVR 4**

Definition at line 622 of file mrf24j40_defines.h.

**#define RFCON6_BATEN 3**

Definition at line 623 of file mrf24j40_defines.h.

**#define RFCON6_TXFIL 7**

Definition at line 619 of file mrf24j40_defines.h.

**#define RFCON7 0x207**

Definition at line 628 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define RFCON7_CLKOUTMODE0 0**

Definition at line 638 of file mrf24j40_defines.h.

**#define RFCON7_CLKOUTMODE1 1**

Definition at line 637 of file mrf24j40_defines.h.

**#define RFCON7_SLPCLKSEL0 6**

Definition at line 632 of file mrf24j40_defines.h.

**#define RFCON7_SLPCLKSEL1 7**

Definition at line 631 of file mrf24j40_defines.h.

**#define RFCON8 0x208**

Definition at line 640 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define RFCON8_RFVCO 4**

Definition at line 646 of file mrf24j40_defines.h.

**#define RFCTL 0x36**

Definition at line 446 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), mrf24j40_init_coordinator(), and mrf24j40_set_channel().

**#define RFCTL_RFRST 2**

Definition at line 454 of file mrf24j40_defines.h.

**#define RFCTL_WAKECNT7 3**

Definition at line 453 of file mrf24j40_defines.h.

**#define RFCTL_WAKECNT8 4**

Definition at line 452 of file mrf24j40_defines.h.

**#define RFSTATE 0x20F**

Definition at line 691 of file mrf24j40_defines.h.
Referenced by mrf24j40_init().

**#define RFSTATE_RFSTATE0 5**

Definition at line 696 of file mrf24j40_defines.h.

**#define RFSTATE_RFSTATE1 6**

Definition at line 695 of file mrf24j40_defines.h.

**#define RFSTATE_RFSTATE2 7**

Definition at line 694 of file mrf24j40_defines.h.

**#define RSSI 0x210**

Definition at line 703 of file mrf24j40_defines.h.
Referenced by mrf24j40_active_channel_scan(), and mrf24j40_scan_for_lowest_channel_ed().

**#define RSSI_RSSI0 0**

Definition at line 713 of file mrf24j40_defines.h.

**#define RSSI_RSSI1 1**

Definition at line 712 of file mrf24j40_defines.h.

**#define RSSI_RSSI2 2**

Definition at line 711 of file mrf24j40_defines.h.

**#define RSSI_RSSI3 3**

Definition at line 710 of file mrf24j40_defines.h.

**#define RSSI_RSSI4 4**

Definition at line 709 of file mrf24j40_defines.h.

**#define RSSI_RSSI5 5**

Definition at line 708 of file mrf24j40_defines.h.

**#define RSSI_RSSI6 6**

Definition at line 707 of file mrf24j40_defines.h.

**#define RSSI_RSSI7 7**

Definition at line 706 of file mrf24j40_defines.h.

**#define RXFLUSH 0x0D**

Definition at line 64 of file mrf24j40_defines.h.
Referenced by mrf24j40_active_channel_scan(), and mrf24j40_flush_receive_buffer().

**#define RXFLUSH_BCNONLY 1**

Definition at line 73 of file mrf24j40_defines.h.
Referenced by mrf24j40_active_channel_scan().

**#define RXFLUSH_CMDONLY 3**

Definition at line 71 of file mrf24j40_defines.h.

**#define RXFLUSH_DATAONLY 2**

Definition at line 72 of file mrf24j40_defines.h.

**#define RXFLUSH_RXFLUSH 0**

Definition at line 74 of file mrf24j40_defines.h.
Referenced by mrf24j40_flush_receive_buffer().

**#define RXFLUSH_WAKEPAD 5**

Definition at line 69 of file mrf24j40_defines.h.

**#define RXFLUSH_WAKEPOL 6**

Definition at line 68 of file mrf24j40_defines.h.

**#define RXMCR 0x00**

Definition at line 40 of file mrf24j40_defines.h.
Referenced by mrf24j40_init_coordinator().

**#define RXMCR_COORD 2**

Definition at line 48 of file mrf24j40_defines.h.

**#define RXMCR_ERRPKT 1**

Definition at line 49 of file mrf24j40_defines.h.

**#define RXMCR_NOACKRSP 5**

Definition at line 45 of file mrf24j40_defines.h.

**#define RXMCR_PANCOORD 3**

Definition at line 47 of file mrf24j40_defines.h.
Referenced by mrf24j40_init_coordinator().

**#define RXMCR_PROMI 0**

Definition at line 50 of file mrf24j40_defines.h.

**#define RXSR 0x30**

Definition at line 374 of file mrf24j40_defines.h.

**#define RXSR_BATIND 5**

Definition at line 379 of file mrf24j40_defines.h.

**#define RXSR_UPSECERR 6**

Definition at line 378 of file mrf24j40_defines.h.

**#define SADRH 0x04**

Definition at line 55 of file mrf24j40_defines.h.
Referenced by mrf24j40_set_short_address().

**#define SADRL 0x03**

Definition at line 54 of file mrf24j40_defines.h.
Referenced by mrf24j40_set_short_address().

**#define SECCON0 0x2C**

Definition at line 337 of file mrf24j40_defines.h.

**#define SECCON0_RXCIPHER0 3**

Definition at line 344 of file mrf24j40_defines.h.

**#define SECCON0_RXCIPHER1 4**

Definition at line 343 of file mrf24j40_defines.h.

**#define SECCON0_RXCIPHER2 5**

Definition at line 342 of file mrf24j40_defines.h.

**#define SECCON0_SECIGNORE 7**

Definition at line 340 of file mrf24j40_defines.h.

**#define SECCON0_SECSTART 6**

Definition at line 341 of file mrf24j40_defines.h.

**#define SECCON0_TXNCIPHER0 0**

Definition at line 347 of file mrf24j40_defines.h.

**#define SECCON0_TXNCIPHER1 1**

Definition at line 346 of file mrf24j40_defines.h.

**#define SECCON0_TXNCIPHER2 2**

Definition at line 345 of file mrf24j40_defines.h.

**#define SECCON1 0x2D**

Definition at line 349 of file mrf24j40_defines.h.

**#define SECCON1_DISDEC 1**

Definition at line 358 of file mrf24j40_defines.h.

**#define SECCON1_DISENC 0**

Definition at line 359 of file mrf24j40_defines.h.

**#define SECCON1_TXBCIPHER0 4**

Definition at line 355 of file mrf24j40_defines.h.

**#define SECCON1_TXBCIPHER1 5**

Definition at line 354 of file mrf24j40_defines.h.

**#define SECCON1_TXBCIPHER2 6**

Definition at line 353 of file mrf24j40_defines.h.

**#define SECCR2 0x37**

Definition at line 458 of file mrf24j40_defines.h.

**#define SECCR2_TXG1CIPHER0 0**

Definition at line 468 of file mrf24j40_defines.h.

**#define SECCR2_TXG1CIPHER1 1**

Definition at line 467 of file mrf24j40_defines.h.

**#define SECCR2_TXG1CIPHER2 2**

Definition at line 466 of file mrf24j40_defines.h.

**#define SECCR2_TXG2CIPHER0 3**

Definition at line 465 of file mrf24j40_defines.h.

**#define SECCR2_TXG2CIPHER1 4**

Definition at line 464 of file mrf24j40_defines.h.

**#define SECCR2_TXG2CIPHER2 5**

Definition at line 463 of file mrf24j40_defines.h.

**#define SECCR2_UPDEC 7**

Definition at line 461 of file mrf24j40_defines.h.

**#define SECCR2_UPENC 6**

Definition at line 462 of file mrf24j40_defines.h.

**#define SLPACK 0x35**

Definition at line 434 of file mrf24j40_defines.h.

**#define SLPACK_SLPACK 7**

Definition at line 437 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT0 0**

Definition at line 444 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT1 1**

Definition at line 443 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT2 2**

Definition at line 442 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT3 3**

Definition at line 441 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT4 4**

Definition at line 440 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT5 5**

Definition at line 439 of file mrf24j40_defines.h.

**#define SLPACK_WAKECNT6 6**

Definition at line 438 of file mrf24j40_defines.h.

**#define SLPCAL0 0x209**

Definition at line 652 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL0 0**

Definition at line 662 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL1 1**

Definition at line 661 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL2 2**

Definition at line 660 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL3 3**

Definition at line 659 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL4 4**

Definition at line 658 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL5 5**

Definition at line 657 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL6 6**

Definition at line 656 of file mrf24j40_defines.h.

**#define SLPCAL0_SLPCAL7 7**

Definition at line 655 of file mrf24j40_defines.h.

**#define SLPCAL1 0x20A**

Definition at line 664 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL10 2**

Definition at line 672 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL11 3**

Definition at line 671 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL12 4**

Definition at line 670 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL13 5**

Definition at line 669 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL14 6**

Definition at line 668 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL15 7**

Definition at line 667 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL8 0**

Definition at line 674 of file mrf24j40_defines.h.

**#define SLPCAL1_SLPCAL9 1**

Definition at line 673 of file mrf24j40_defines.h.

**#define SLPCAL2 0x20B**

Definition at line 676 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCAL16 0**

Definition at line 686 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCAL17 1**

Definition at line 685 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCAL18 2**

Definition at line 684 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCAL19 3**

Definition at line 683 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCALEN 4**

Definition at line 682 of file mrf24j40_defines.h.

**#define SLPCAL2_SLPCALRDY 7**

Definition at line 679 of file mrf24j40_defines.h.

**#define SLPCON0 0x211**

Definition at line 715 of file mrf24j40_defines.h.

**#define SLPCON0_INTEDGE 1**

Definition at line 724 of file mrf24j40_defines.h.

**#define SLPCON0_SLPCLKEN 0**

Definition at line 725 of file mrf24j40_defines.h.

**#define SLPCON1 0x220**

Definition at line 741 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define SLPCON1_CLKOUTEN 5**

Definition at line 746 of file mrf24j40_defines.h.

**#define SLPCON1_SLPCLKDIV0 0**

Definition at line 751 of file mrf24j40_defines.h.

**#define SLPCON1_SLPCLKDIV1 1**

Definition at line 750 of file mrf24j40_defines.h.

**#define SLPCON1_SLPCLKDIV2 2**

Definition at line 749 of file mrf24j40_defines.h.

**#define SLPCON1_SLPCLKDIV3 3**

Definition at line 748 of file mrf24j40_defines.h.

**#define SLPCON1_SLPCLKDIV4 4**

Definition at line 747 of file mrf24j40_defines.h.

**#define SOFTRST 0x2A**

Definition at line 324 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define SOFTRST_RSTBB 1**

Definition at line 333 of file mrf24j40_defines.h.

**#define SOFTRST_RSTMAC 0**

Definition at line 334 of file mrf24j40_defines.h.

**#define SOFTRST_RSTPWR 2**

Definition at line 332 of file mrf24j40_defines.h.

**#define SYMTICKH 0x15**

Definition at line 127 of file mrf24j40_defines.h.

**#define SYMTICKH_TICKP8 0**

Definition at line 137 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT0 1**

Definition at line 136 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT1 2**

Definition at line 135 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT2 3**

Definition at line 134 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT3 4**

Definition at line 133 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT4 5**

Definition at line 132 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT5 6**

Definition at line 131 of file mrf24j40_defines.h.

**#define SYMTICKH_TXONT6 7**

Definition at line 130 of file mrf24j40_defines.h.

**#define SYMTICKL 0x14**

Definition at line 115 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP0 0**

Definition at line 125 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP1 1**

Definition at line 124 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP2 2**

Definition at line 123 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP3 3**

Definition at line 122 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP4 4**

Definition at line 121 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP5 5**

Definition at line 120 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP6 6**

Definition at line 119 of file mrf24j40_defines.h.

**#define SYMTICKL_TICKP7 7**

Definition at line 118 of file mrf24j40_defines.h.

**#define TESTMODE 0x22F**

Definition at line 855 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_scan_for_lowest_channel_ed().

**#define TESTMODE_RSSIWAIT0 3**

Definition at line 862 of file mrf24j40_defines.h.

**#define TESTMODE_RSSIWAIT1 4**

Definition at line 861 of file mrf24j40_defines.h.

**#define TESTMODE_TESTMODE0 0**

Definition at line 865 of file mrf24j40_defines.h.

**#define TESTMODE_TESTMODE1 1**

Definition at line 864 of file mrf24j40_defines.h.

**#define TESTMODE_TESTMODE2 2**

Definition at line 863 of file mrf24j40_defines.h.

**#define TRISGPIO 0x34**

Definition at line 422 of file mrf24j40_defines.h.
Referenced by mrf24j40_scan_for_lowest_channel_ed().

**#define TRISGPIO_TRISGP0 0**

Definition at line 432 of file mrf24j40_defines.h.

**#define TRISGPIO_TRISGP1 1**

Definition at line 431 of file mrf24j40_defines.h.

**#define TRISGPIO_TRISGP2 2**

Definition at line 430 of file mrf24j40_defines.h.

**#define TRISGPIO_TRISGP3 3**

Definition at line 429 of file mrf24j40_defines.h.

**#define TRISGPIO_TRISGP4 4**

Definition at line 428 of file mrf24j40_defines.h.

**#define TRISGPIO_TRISGP5 5**

Definition at line 427 of file mrf24j40_defines.h.

**#define TXBCON0 0x1A**

Definition at line 176 of file mrf24j40_defines.h.

**#define TXBCON0_TXBSECEN 1**

Definition at line 185 of file mrf24j40_defines.h.

**#define TXBCON0_TXBTRIG 0**

Definition at line 186 of file mrf24j40_defines.h.

**#define TXBCON1 0x25**

Definition at line 275 of file mrf24j40_defines.h.

**#define TXG1CON 0x1C**

Definition at line 200 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1ACKREQ 2**

Definition at line 208 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1RETRY0 6**

Definition at line 204 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1RETRY1 7**

Definition at line 203 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1SECEN 1**

Definition at line 209 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1SLOT0 3**

Definition at line 207 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1SLOT1 4**

Definition at line 206 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1SLOT2 5**

Definition at line 205 of file mrf24j40_defines.h.

**#define TXG1CON_TXG1TRIG 0**

Definition at line 210 of file mrf24j40_defines.h.

**#define TXG2CON 0x1D**

Definition at line 212 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2ACKREQ 2**

Definition at line 220 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2RETRY0 6**

Definition at line 216 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2RETRY1 7**

Definition at line 215 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2SECEN 1**

Definition at line 221 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2SLOT0 3**

Definition at line 219 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2SLOT1 4**

Definition at line 218 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2SLOT2 5**

Definition at line 217 of file mrf24j40_defines.h.

**#define TXG2CON_TXG2TRIG 0**

Definition at line 222 of file mrf24j40_defines.h.

**#define TXMCR 0x11**

Definition at line 90 of file mrf24j40_defines.h.
Referenced by mrf24j40_init_coordinator().

**#define TXMCR_BATLIFEXT 6**

Definition at line 94 of file mrf24j40_defines.h.

**#define TXMCR_CSMABF0 0**

Definition at line 100 of file mrf24j40_defines.h.

**#define TXMCR_CSMABF1 1**

Definition at line 99 of file mrf24j40_defines.h.

**#define TXMCR_CSMABF2 2**

Definition at line 98 of file mrf24j40_defines.h.

**#define TXMCR_MACMINBE0 3**

Definition at line 97 of file mrf24j40_defines.h.

**#define TXMCR_MACMINBE1 4**

Definition at line 96 of file mrf24j40_defines.h.

**#define TXMCR_NOCSMA 7**

Definition at line 93 of file mrf24j40_defines.h.

**#define TXMCR_SLOTTED 5**

Definition at line 95 of file mrf24j40_defines.h.

Referenced by mrf24j40_init_coordinator().

**#define TXNCON 0x1B**

Definition at line 188 of file mrf24j40_defines.h.

Referenced by mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

**#define TXNCON_FPSTAT 4**

Definition at line 194 of file mrf24j40_defines.h.

**#define TXNCON_INDIRECT 3**

Definition at line 195 of file mrf24j40_defines.h.

**#define TXNCON_TXNACKREQ 2**

Definition at line 196 of file mrf24j40_defines.h.

Referenced by mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

**#define TXNCON_TXNSECEN 1**

Definition at line 197 of file mrf24j40_defines.h.

**#define TXNCON_TXNTRIG 0**

Definition at line 198 of file mrf24j40_defines.h.

Referenced by mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), and mrf24j40_transmit_to_short_address().

**#define TXPEND 0x21**

Definition at line 227 of file mrf24j40_defines.h.

**#define TXPEND_FPACK 0**

Definition at line 237 of file mrf24j40_defines.h.

**#define TXPEND_GTSSWITCH 1**

Definition at line 236 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS0 2**

Definition at line 235 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS1 3**

Definition at line 234 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS2 4**

Definition at line 233 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS3 5**

Definition at line 232 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS4 6**

Definition at line 231 of file mrf24j40_defines.h.

**#define TXPEND_MLIFS5 7**

Definition at line 230 of file mrf24j40_defines.h.

**#define TXSTAT 0x24**

Definition at line 263 of file mrf24j40_defines.h.
Referenced by mrf24j40_handle_isr().

**#define TXSTAT_CCAFAIL 5**

Definition at line 268 of file mrf24j40_defines.h.
Referenced by mrf24j40_handle_isr().

**#define TXSTAT_TXG1FNT 3**

Definition at line 270 of file mrf24j40_defines.h.

**#define TXSTAT_TXG1STAT 1**

Definition at line 272 of file mrf24j40_defines.h.

**#define TXSTAT_TXG2FNT 4**

Definition at line 269 of file mrf24j40_defines.h.

**#define TXSTAT_TXG2STAT 2**

Definition at line 271 of file mrf24j40_defines.h.

**#define TXSTAT_TXNRETRY0 6**

Definition at line 267 of file mrf24j40_defines.h.

**#define TXSTAT_TXNRETRY1 7**

Definition at line 266 of file mrf24j40_defines.h.

**#define TXSTAT_TXNSTAT 0**

Definition at line 273 of file mrf24j40_defines.h.

**#define TXSTBL 0x2E**

Definition at line 361 of file mrf24j40_defines.h.
Referenced by mrf24j40_init(), and mrf24j40_init_coordinator().

**#define TXSTBL_MSIFS0 0**

Definition at line 371 of file mrf24j40_defines.h.

**#define TXSTBL_MSIFS1 1**

Definition at line 370 of file mrf24j40_defines.h.

**#define TXSTBL_MSIFS2 2**

Definition at line 369 of file mrf24j40_defines.h.

**#define TXSTBL_MSIFS3 3**

Definition at line 368 of file mrf24j40_defines.h.

**#define TXSTBL_RFSTBL0 4**

Definition at line 367 of file mrf24j40_defines.h.

**#define TXSTBL_RFSTBL1 5**

Definition at line 366 of file mrf24j40_defines.h.

**#define TXSTBL_RFSTBL2 6**

Definition at line 365 of file mrf24j40_defines.h.

**#define TXSTBL_RFSTBL3 7**

Definition at line 364 of file mrf24j40_defines.h.

**#define TXTIME 0x27**

Definition at line 288 of file mrf24j40_defines.h.

**#define TXTIME_TURNTIME0 4**

Definition at line 294 of file mrf24j40_defines.h.

**#define TXTIME_TURNTIME1 5**

Definition at line 293 of file mrf24j40_defines.h.

**#define TXTIME_TURNTIME2 6**

Definition at line 292 of file mrf24j40_defines.h.

**#define TXTIME_TURNTIME3 7**

Definition at line 291 of file mrf24j40_defines.h.

**#define UPNONCE0 0x240**

Definition at line 883 of file mrf24j40_defines.h.

**#define UPNONCE1 0x241**

Definition at line 884 of file mrf24j40_defines.h.

**#define UPNONCE10 0x24A**

Definition at line 893 of file mrf24j40_defines.h.

**#define UPNONCE11 0x24B**

Definition at line 894 of file mrf24j40_defines.h.

**#define UPNONCE12 0x24C**

Definition at line 895 of file mrf24j40_defines.h.

**#define UPNONCE2 0x242**

Definition at line 885 of file mrf24j40_defines.h.

**#define UPNONCE3 0x243**

Definition at line 886 of file mrf24j40_defines.h.

**#define UPNONCE4 0x244**

Definition at line 887 of file mrf24j40_defines.h.

**#define UPNONCE5 0x245**

Definition at line 888 of file mrf24j40_defines.h.

**#define UPNONCE6 0x246**

Definition at line 889 of file mrf24j40_defines.h.

**#define UPNONCE7 0x247**

Definition at line 890 of file mrf24j40_defines.h.

**#define UPNONCE8 0x248**

Definition at line 891 of file mrf24j40_defines.h.

**#define UPNONCE9 0x249**

Definition at line 892 of file mrf24j40_defines.h.

**#define WAKECON 0x22**

Definition at line 239 of file mrf24j40_defines.h.

**#define WAKECON_IMMWAKE 7**

Definition at line 242 of file mrf24j40_defines.h.

**#define WAKECON_REGWAKE 6**

Definition at line 243 of file mrf24j40_defines.h.

**#define WAKETIMEH 0x223**

Definition at line 766 of file mrf24j40_defines.h.

**#define WAKETIMEH_WAKETIME10 2**

Definition at line 774 of file mrf24j40_defines.h.

**#define WAKETIMEH_WAKETIME8 0**

Definition at line 776 of file mrf24j40_defines.h.

**#define WAKETIMEH_WAKETIME9 1**

Definition at line 775 of file mrf24j40_defines.h.

**#define WAKETIMEL 0x222**

Definition at line 754 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME0 0**

Definition at line 764 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME1 1**

Definition at line 763 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME2 2**

Definition at line 762 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME3 3**

Definition at line 761 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME4 4**

Definition at line 760 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME5 5**

Definition at line 759 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME6 6**

Definition at line 758 of file mrf24j40_defines.h.

**#define WAKETIMEL_WAKETIME7 7**

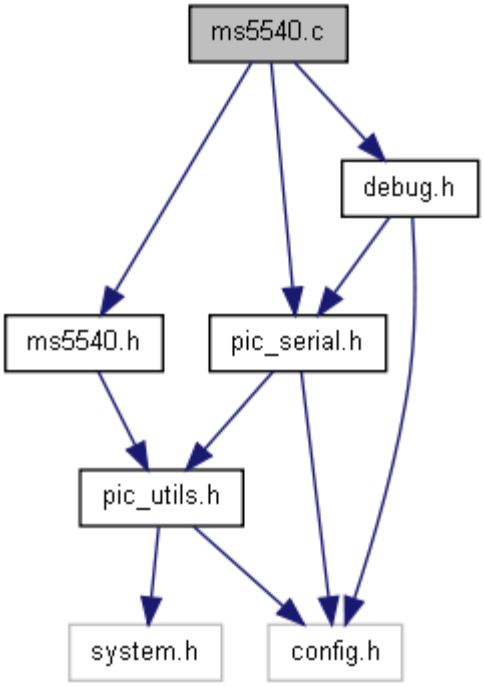
Definition at line 757 of file mrf24j40_defines.h.

---

## ms5540.c File Reference

```
#include "ms5540.h"
#include "pic_serial.h"
#include "debug.h"
```

Include dependency graph for ms5540.c:




## Defines

- #define MS5540_DELAY_AMOUNT 2

## Functions

- void ms5540_calc_temp_and_pressure ()
- uns16 ms5540_get_config (uns8 config_word)
- uns16 ms5540_get_raw_pressure ()
- uns16 ms5540_get_raw_temp ()
- void ms5540_init ()
- void ms5540_pulse_sclk ()
- void ms5540_reset ()
- void ms5540_send_start ()
- void ms5540_send_stop ()
- void ms5540_setup_io (void)

## Variables

- int16 c1
- int16 c2
- int16 c3
- int16 c4
- int16 c5
- int16 c6

---

## Define Documentation

### #define MS5540_DELAY_AMOUNT 2

Definition at line 42 of file ms5540.c.

Referenced by ms5540_pulse_sclk().

---

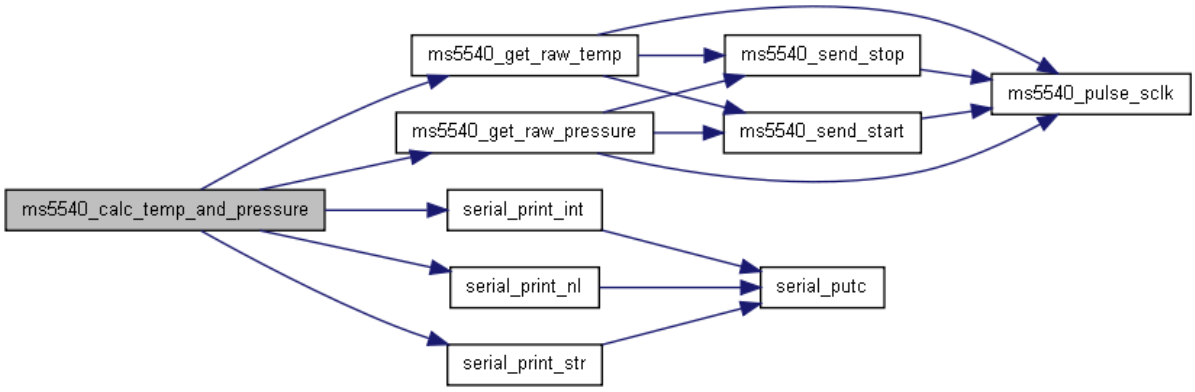
## Function Documentation

### void ms5540_calc_temp_and_pressure ()

Definition at line 322 of file ms5540.c.

References c1, c2, c3, c4, c5, c6, int16, int32, ms5540_get_raw_pressure(), ms5540_get_raw_temp(), serial_print_int(), serial_print_nl(), and serial_print_str().

Here is the call graph for this function:



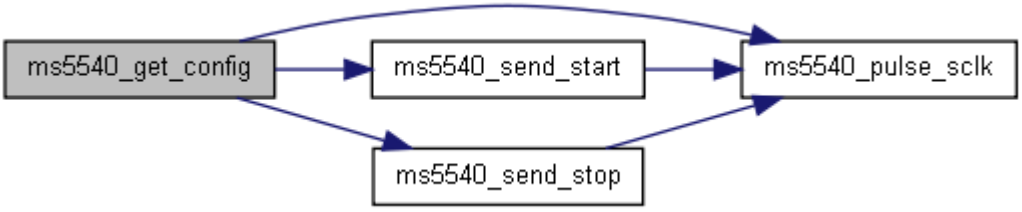

### uns16 ms5540_get_config (uns8 *config_word*)

Definition at line 102 of file ms5540.c.

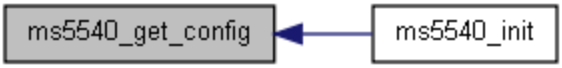
References change_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), test_pin, uns16, and uns8.

Referenced by ms5540_init().

Here is the call graph for this function:




Here is the caller graph for this function:



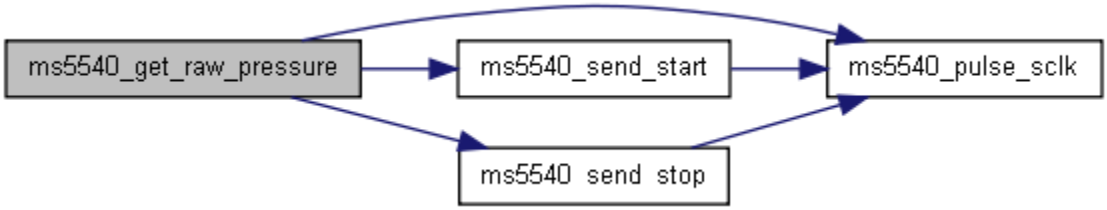

### uns16 ms5540_get_raw_pressure ()

Definition at line 154 of file ms5540.c.

References clear_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), set_pin, test_pin, uns16, and uns8.

Referenced by ms5540_calc_temp_and_pressure().

Here is the call graph for this function:




Here is the caller graph for this function:

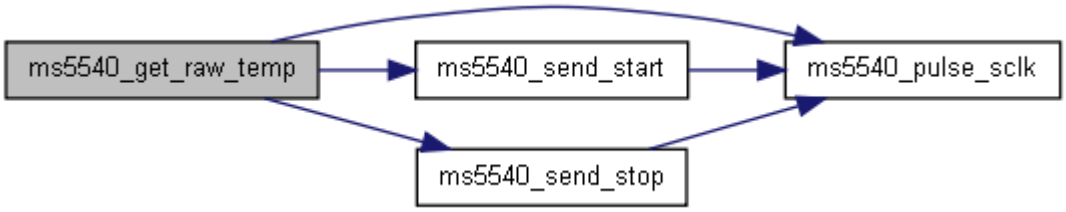

### uns16 ms5540_get_raw_temp ()

Definition at line 200 of file ms5540.c.

References clear_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), set_pin, test_pin, uns16, and uns8.

Referenced by ms5540_calc_temp_and_pressure().

Here is the call graph for this function:



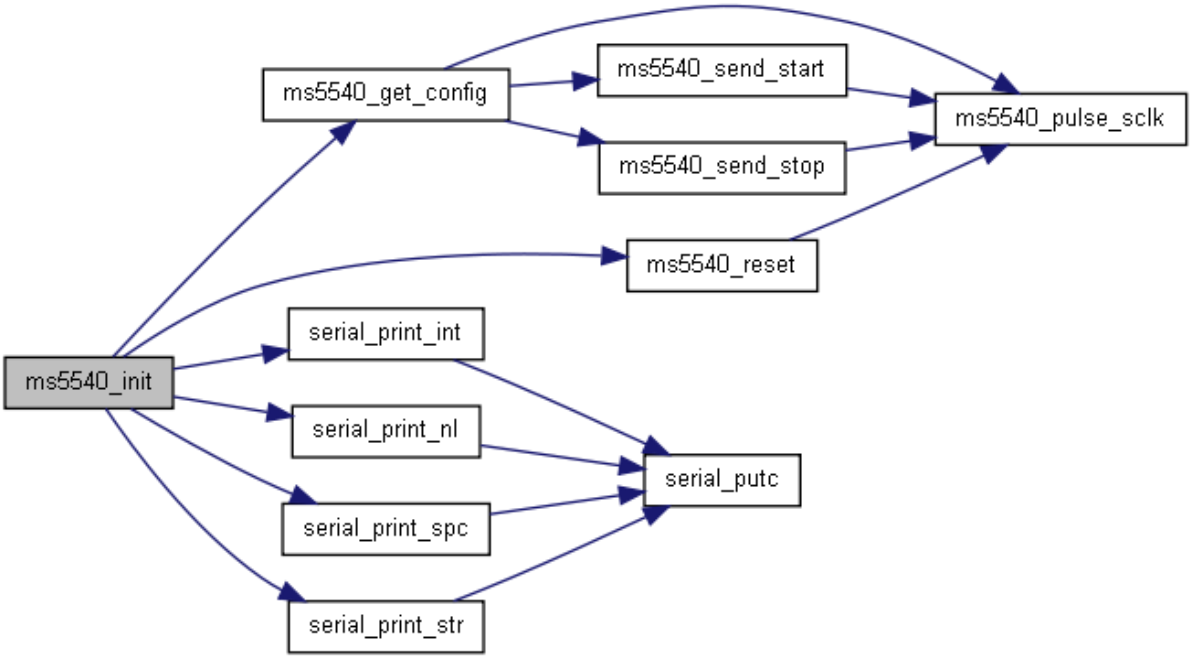

Here is the caller graph for this function:


### void ms5540_init ()

Definition at line 247 of file ms5540.c.

References c1, c2, c3, c4, c5, c6, ms5540_get_config(), ms5540_reset(), serial_print_int(), serial_print_nl(), serial_print_spc(), serial_print_str(), and uns16.

Here is the call graph for this function:



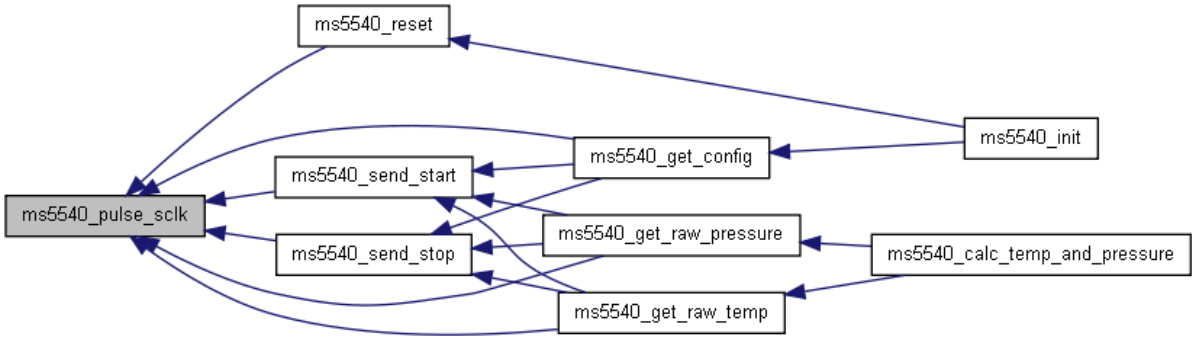

### void ms5540_pulse_sclk ()

Definition at line 54 of file ms5540.c.

References clear_pin, MS5540_DELAY_AMOUNT, and set_pin.

Referenced by ms5540_get_config(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), ms5540_reset(), ms5540_send_start(), and ms5540_send_stop().

Here is the caller graph for this function:



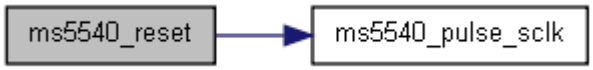

### void ms5540_reset ()

Definition at line 80 of file ms5540.c.

References change_pin, ms5540_pulse_sclk(), uns16, and uns8.

Referenced by ms5540_init().

Here is the call graph for this function:




Here is the caller graph for this function:


### void ms5540_send_start ()

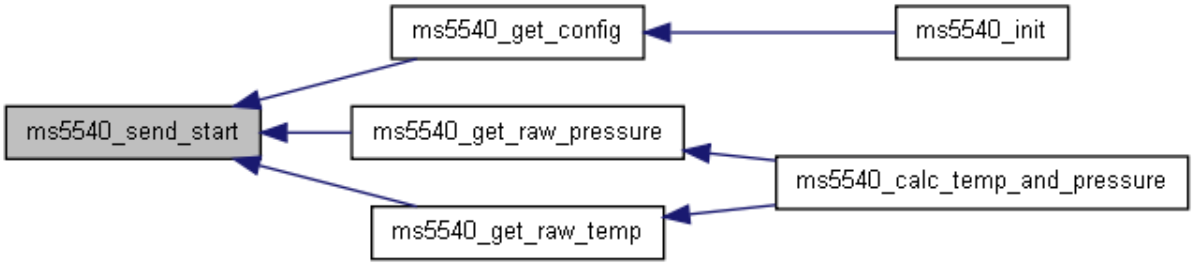
Definition at line 63 of file ms5540.c.

References ms5540_pulse_sclk(), and set_pin.

Referenced by ms5540_get_config(), ms5540_get_raw_pressure(), and ms5540_get_raw_temp().

Here is the call graph for this function:


Here is the caller graph for this function:




### void ms5540_send_stop ()

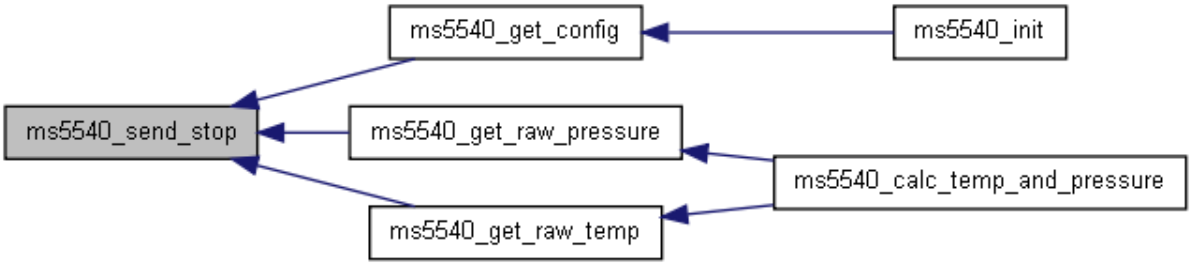
Definition at line 71 of file ms5540.c.

References clear_pin, and ms5540_pulse_sclk().

Referenced by ms5540_get_config(), ms5540_get_raw_pressure(), and ms5540_get_raw_temp().

Here is the call graph for this function:


Here is the caller graph for this function:




**void ms5540_setup_io (void)**

Definition at line 46 of file ms5540.c.

References clear_pin, make_input, and make_output.

---

## Variable Documentation

**int16 c1**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), ms5540_init(), and sht15_fix_humidity_r().

**int16 c2**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), ms5540_init(), and sht15_fix_humidity_r().

**int16 c3**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), ms5540_init(), and sht15_fix_humidity_r().

**int16 c4**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), and ms5540_init().

**int16 c5**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), and ms5540_init().

**int16 c6**

Definition at line 44 of file ms5540.c.

Referenced by ms5540_calc_temp_and_pressure(), and ms5540_init().

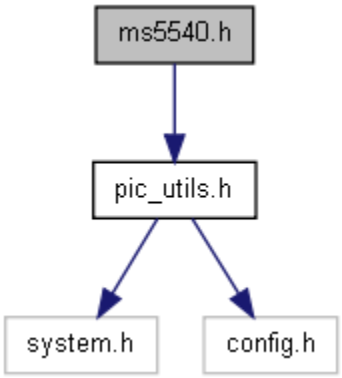
---

# ms5540.h File Reference

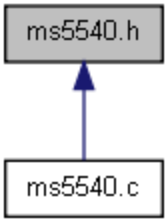
MS5540 temperature and pressure sensor routines.

```
#include "pic_utils.h"
```

Include dependency graph for ms5540.h:




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




## Defines

- #define ms5540_setup() ms5540_setup_io()

## *Setup ms5540 ports and pins.* Functions

- void ms5540_calc_temp_and_pressure ()
- uns16 ms5540_get_config (uns8 config_word)
- uns16 ms5540_get_raw_pressure ()
- uns16 ms5540_get_raw_temp ()
- void ms5540_init ()
- void ms5540_reset ()
- void ms5540_setup_io (void)

---

## Detailed Description

A library to communicate with the ms5540 sensor

Definition in file ms5540.h.

---

## Define Documentation

### #define ms5540_setup() ms5540_setup_io()

Definition at line 72 of file ms5540.h.

---

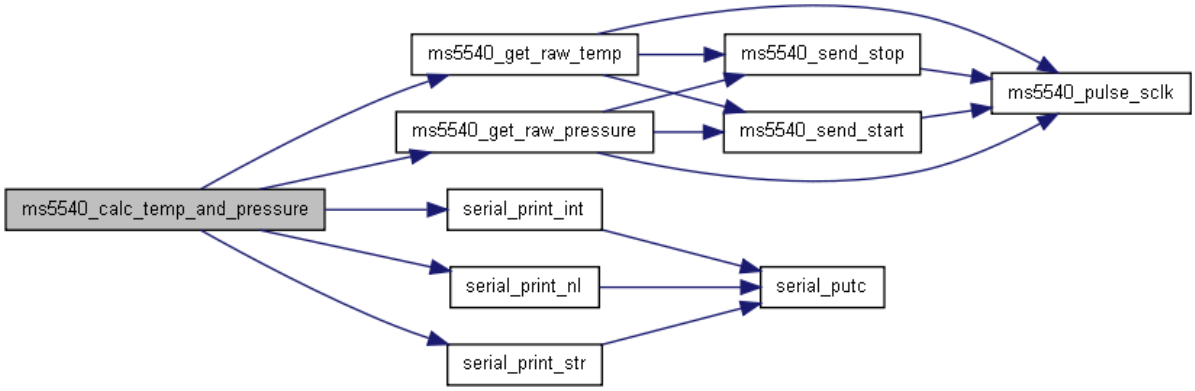
## Function Documentation

### void ms5540_calc_temp_and_pressure ()

Definition at line 322 of file ms5540.c.

References c1, c2, c3, c4, c5, c6, int16, int32, ms5540_get_raw_pressure(), ms5540_get_raw_temp(), serial_print_int(), serial_print_nl(), and serial_print_str().

Here is the call graph for this function:



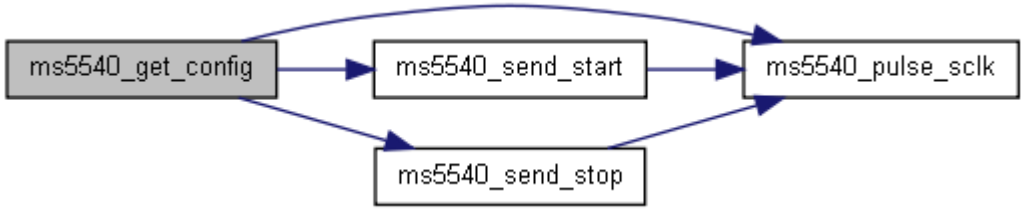

### uns16 ms5540_get_config (uns8 *config_word*)

Definition at line 102 of file ms5540.c.

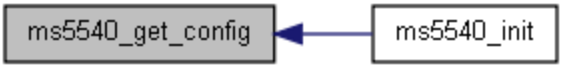
References change_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), test_pin, uns16, and uns8.

Referenced by ms5540_init().

Here is the call graph for this function:




Here is the caller graph for this function:



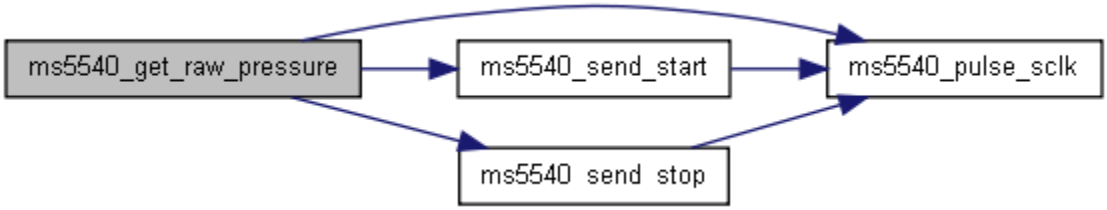

### uns16 ms5540_get_raw_pressure ()

Definition at line 154 of file ms5540.c.

References clear_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), set_pin, test_pin, uns16, and uns8.

Referenced by ms5540_calc_temp_and_pressure().

Here is the call graph for this function:




Here is the caller graph for this function:

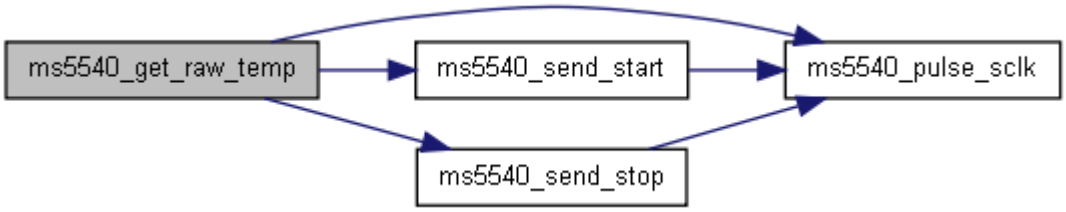

### uns16 ms5540_get_raw_temp ()

Definition at line 200 of file ms5540.c.

References clear_pin, ms5540_pulse_sclk(), ms5540_send_start(), ms5540_send_stop(), set_pin, test_pin, uns16, and uns8.

Referenced by ms5540_calc_temp_and_pressure().

Here is the call graph for this function:



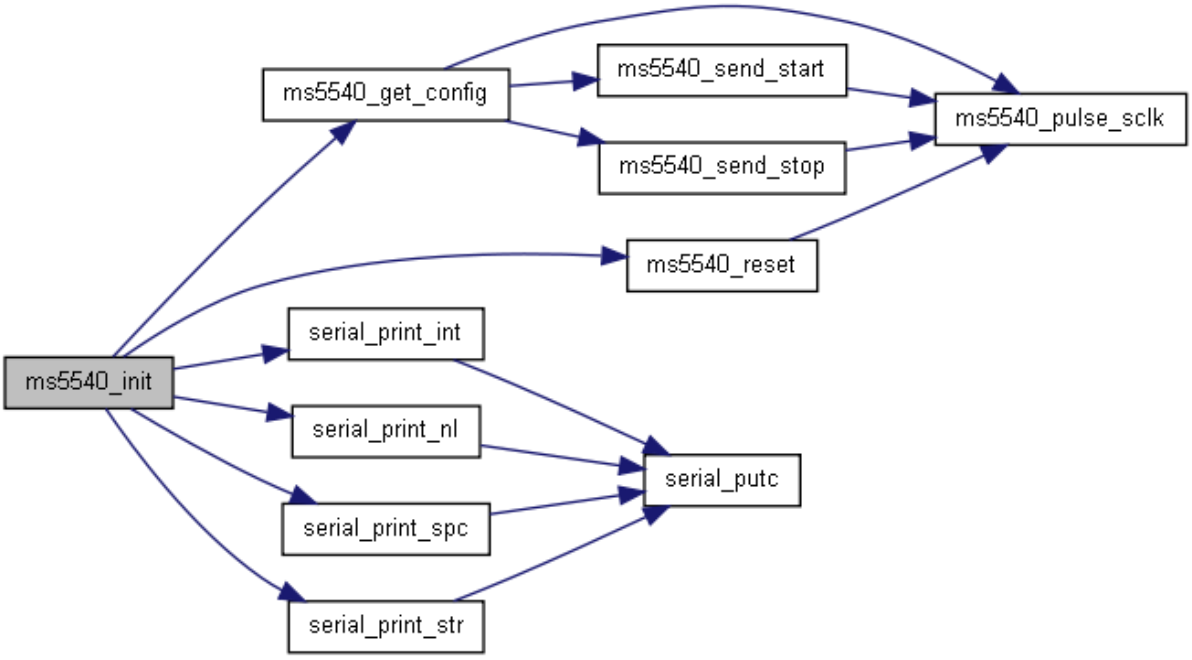

Here is the caller graph for this function:


### void ms5540_init ()

Definition at line 247 of file ms5540.c.

References c1, c2, c3, c4, c5, c6, ms5540_get_config(), ms5540_reset(), serial_print_int(), serial_print_nl(), serial_print_spc(), serial_print_str(), and uns16.

Here is the call graph for this function:




### void ms5540_reset ()

Definition at line 80 of file ms5540.c.

References change_pin, ms5540_pulse_sclk(), uns16, and uns8.

Referenced by ms5540_init().

Here is the call graph for this function:


Here is the caller graph for this function:


### void ms5540_setup_io (void)

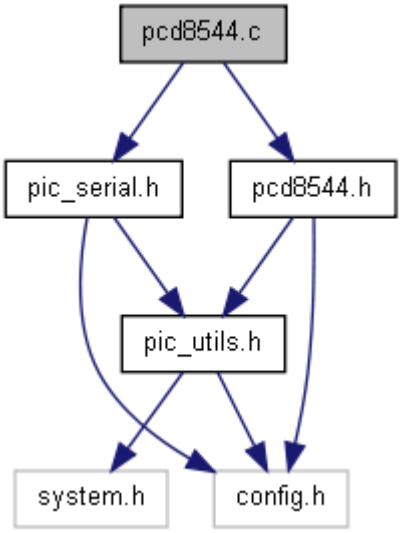
Definition at line 46 of file ms5540.c.

References clear_pin, make_input, and make_output.

# pcd8544.c File Reference

```
#include "pcd8544.h"
#include "pic_serial.h"
```

Include dependency graph for pcd8544.c:




## Functions

- void pcd8544_init ()
- void pcd8544_send_byte (uns8 b)
- void pcd8544_send_command (uns8 cmd)
- void pcd8544_send_data (uns8 data)
- void pcd8544_setup_io ()

---

## Function Documentation

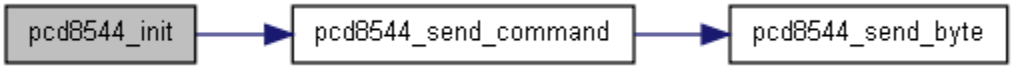
### void pcd8544_init ()

Definition at line 55 of file pcd8544.c.

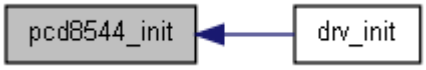
References clear_pin, pcd8544_send_command(), and set_pin.

Referenced by drv_init().

Here is the call graph for this function:




Here is the caller graph for this function:



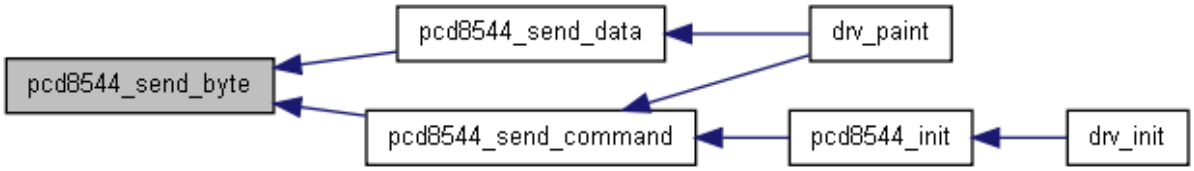
### void pcd8544_send_byte (uns8 *b*)

Definition at line 107 of file pcd8544.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by pcd8544_send_command(), and pcd8544_send_data().

Here is the caller graph for this function:




### void pcd8544_send_command (uns8 *cmd*)

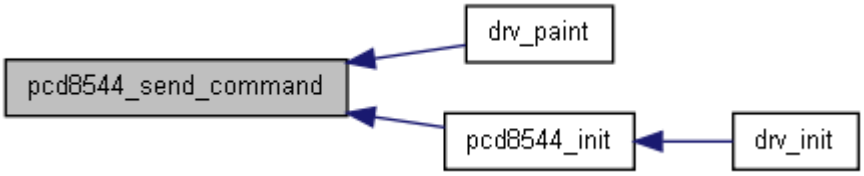
Definition at line 91 of file pcd8544.c.

References clear_pin, pcd8544_send_byte(), and set_pin.

Referenced by drv_paint(), and pcd8544_init().

Here is the call graph for this function:


Here is the caller graph for this function:




### void pcd8544_send_data (uns8 *data*)

Definition at line 99 of file pcd8544.c.

References clear_pin, pcd8544_send_byte(), and set_pin.

Referenced by drv_paint().

Here is the call graph for this function:


Here is the caller graph for this function:

### void pcd8544_setup_io ()

Definition at line 40 of file pcd8544.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:

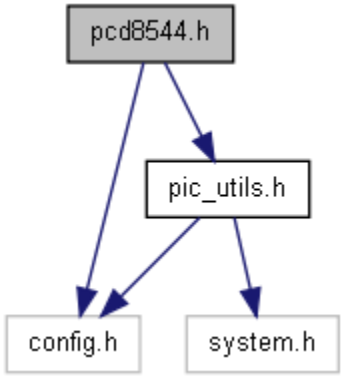

---

## pcd8544.h File Reference

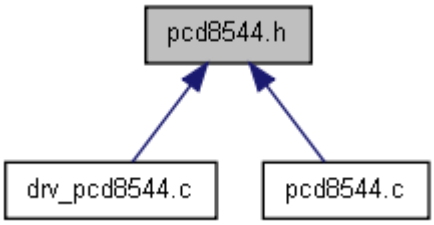
PCD8544 Interface routines (used in Nokia 3310 LCD).

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

Include dependency graph for pcd8544.h:




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




### Defines

- #define pcd8544_setup() pcd8544_setup_io()

### Functions

- void pcd8544_clear ()
- void pcd8544_init ()

- void pcd8544_send_byte (uns8 b)
- void pcd8544_send_command (uns8 command)
- void pcd8544_send_data (uns8 data)
- void pcd8544_set_pixel (uns8 x, uns8 y, uns8 colour)
- void pcd8544_setup_io ()
- void pcd8544_write (uns8 mem_addr, uns8 data)

---

## Detailed Description

Routines to communicate with Nokia 3310 LCD display via the PCD8544 chip

Definition in file pcd8544.h.

---

## Define Documentation

### #define pcd8544_setup() pcd8544_setup_io()

Definition at line 76 of file pcd8544.h.

---

## Function Documentation

### void pcd8544_clear ()

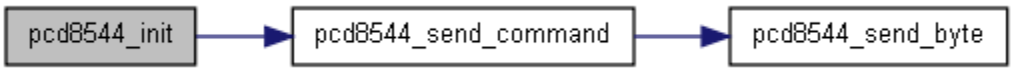
### void pcd8544_init ()

Definition at line 55 of file pcd8544.c.

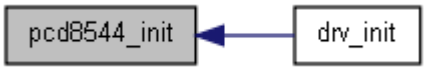
References clear_pin, pcd8544_send_command(), and set_pin.

Referenced by drv_init().

Here is the call graph for this function:




Here is the caller graph for this function:



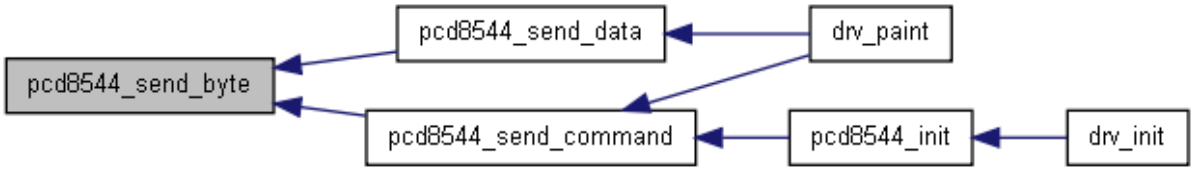

### void pcd8544_send_byte (uns8 *b*)

Definition at line 107 of file pcd8544.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by pcd8544_send_command(), and pcd8544_send_data().

Here is the caller graph for this function:




**void pcd8544_send_command (uns8 *command*)**

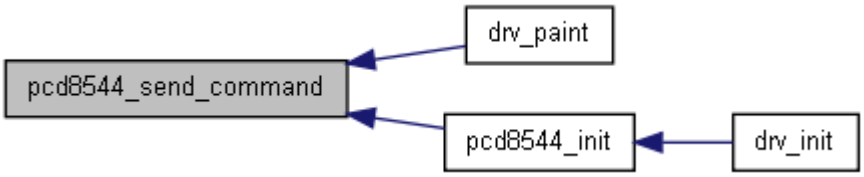
Definition at line 91 of file pcd8544.c.

References clear_pin, pcd8544_send_byte(), and set_pin.

Referenced by drv_paint(), and pcd8544_init().

Here is the call graph for this function:


Here is the caller graph for this function:




**void pcd8544_send_data (uns8 *data*)**

Definition at line 99 of file pcd8544.c.

References clear_pin, pcd8544_send_byte(), and set_pin.

Referenced by drv_paint().

Here is the call graph for this function:


Here is the caller graph for this function:


**void pcd8544_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

**void pcd8544_setup_io ()**

Definition at line 40 of file pcd8544.c.

References clear_pin, make_output, and set_pin.

Referenced by drv_setup_io().

Here is the caller graph for this function:

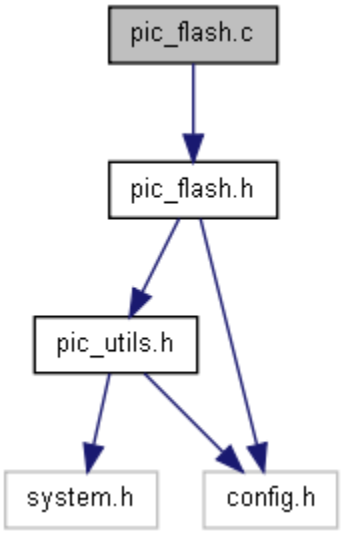

**void pcd8544_write (uns8 *mem_addr*, uns8 *data*)**

---

## pic_flash.c File Reference

```
#include "pic_flash.h"
```

Include dependency graph for pic_flash.c:



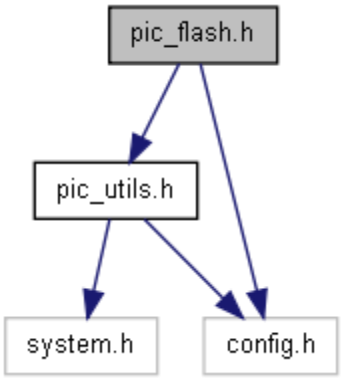

---

## pic_flash.h File Reference

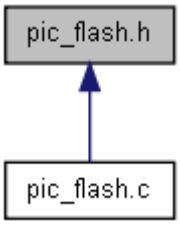
Flash write / erase routines.

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

Include dependency graph for pic_flash.h:




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




### Functions

- void flash_erase (FLASH_MEM_ADDR_TYPE address)
- void flash_write (FLASH_MEM_ADDR_TYPE address, uns8 count, uns8 *data)

---

### Detailed Description

Definition in file pic_flash.h.

---

### Function Documentation

**void flash_erase (FLASH_MEM_ADDR_TYPE *address*)**

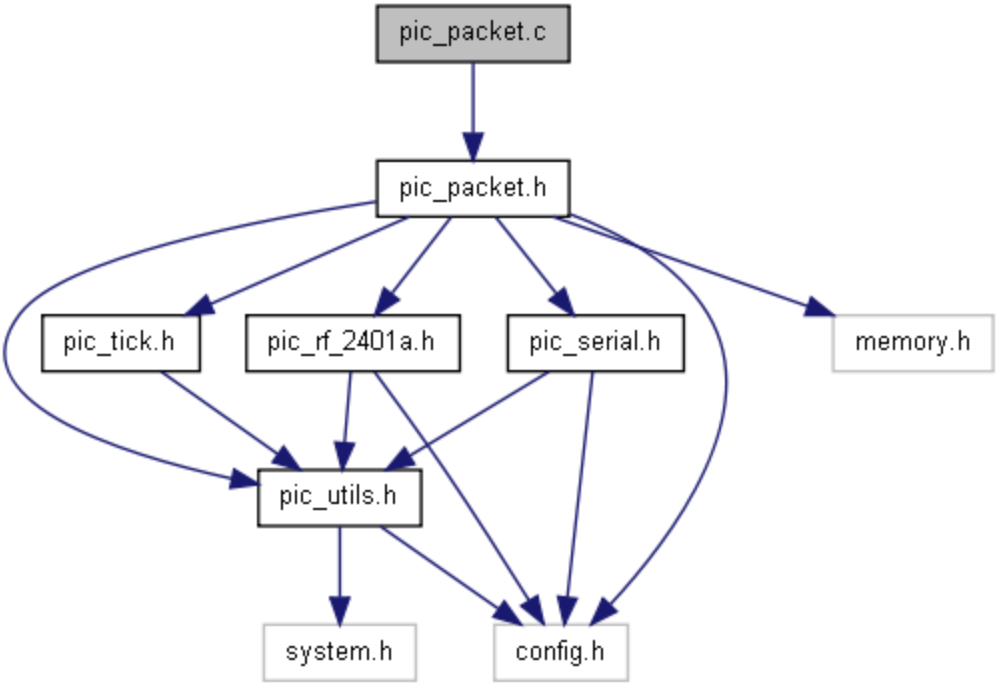
**void flash_write (FLASH_MEM_ADDR_TYPE *address*, uns8 *count*, uns8 * *data*)**

---

## pic_packet.c File Reference

```
#include "pic_packet.h"
```

Include dependency graph for pic_packet.c:




## Data Structures

- union rf_packet
- struct seen_packet
- struct sending_item

## Defines

- #define PKT_FLAG_DELETED 0xff

## Functions

- void pkt_calc_check_byte (rf_packet *packet)
- uns8 pkt_check_check_byte (rf_packet *packet)
- void pkt_init (uns16 my_addr, uns16 last_sent_id)
- *Initialise packet delivery system.* uns8 pkt_print_packet (rf_packet *my_packet)
- uns8 pkt_process_rf_data (uns8 *data_in)
- *Process received RF data.* void pkt_process_tx_queue ()
- *Handle queued items.* uns8 pkt_queue_packet (rf_packet *packet, uns8 resend)
- uns8 pkt_seen (uns16 pkt_id, uns16 source_addr)
- void pkt_send_packet (rf_packet *packet)
- uns8 pkt_send_payload (uns16 dest_addr, uns8 *payload, uns8 resend)

## *Send a payload via the packet delivery system.* Variables

- uns16 pkt_my_addr = 0x66
- uns16 pkt_my_next_pkt_id = 0
- static seen_packet pkt_seen_list [PKT_SEEN_LIST_SIZE]
- uns8 pkt_seen_list_last = 0
- static sending_item pkt_tx_queue [PKT_TX_QUEUE_SIZE]

## Define Documentation

### #define PKT_FLAG_DELETED 0xff

Definition at line 78 of file pic_packet.c.

Referenced by pkt_init(), pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_queue_packet().

---

## Function Documentation

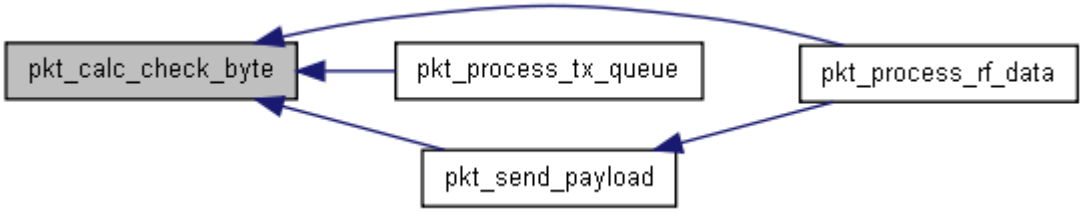
### void pkt_calc_check_byte (rf_packet * *packet*)

Definition at line 93 of file pic_packet.c.

References rf_packet::a, rf_packet_det::check_byte, rf_packet::d, PKT_PACKET_SIZE, and uns8.

Referenced by pkt_process_rf_data(), pkt_process_tx_queue(), and pkt_send_payload().

Here is the caller graph for this function:




### uns8 pkt_check_check_byte (rf_packet * *packet*)

Definition at line 104 of file pic_packet.c.

References rf_packet::a, rf_packet_det::check_byte, rf_packet::d, PKT_PACKET_SIZE, and uns8.

Referenced by pkt_process_rf_data().

Here is the caller graph for this function:


### void pkt_init (uns16 *my_addr*, uns16 *last_sent_pkt_id*)

Initialise the packet delivery system ready for use. This routine clears the transmit queue and seen queue. If you don't store the last_sent_pkt_id (eg, in EEPROM) then set this to 0.

**Parameters:**

*my_addr* The address of this system
*last_sent_pkt_id* The last pkt_id used by this system.

Definition at line 423 of file pic_packet.c.

References sending_item::flag, PKT_FLAG_DELETED, pkt_my_addr, pkt_my_next_pkt_id, seen_packet::source_addr, uns16, and uns8.

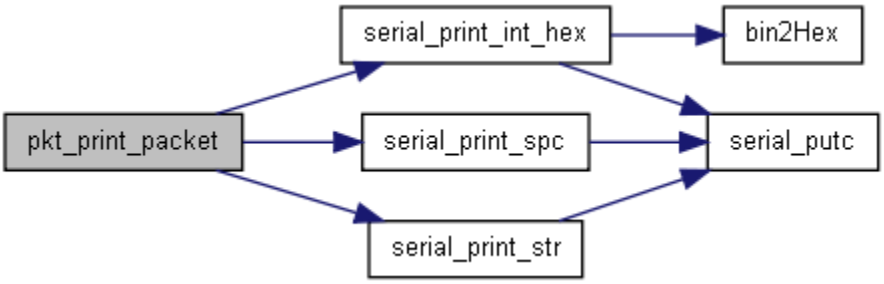
### uns8 pkt_print_packet (rf_packet * *my_packet*)

Definition at line 120 of file pic_packet.c.

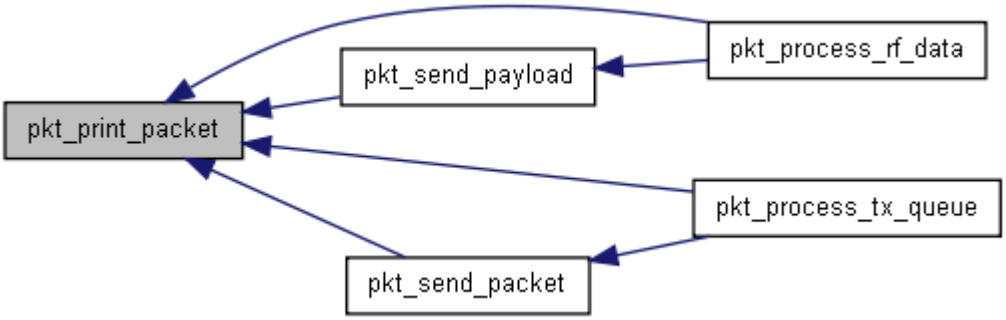
References rf_packet::d, rf_packet_det::dest_addr, rf_packet_det::payload, rf_packet_det::pkt_id, rf_packet_det::r1_addr, rf_packet_det::r2_addr, rf_packet_det::r3_addr, serial_print_int_hex(), serial_print_spc(), serial_print_str(), rf_packet_det::source_addr, and uns8.

Referenced by pkt_process_rf_data(), pkt_process_tx_queue(), pkt_send_packet(), and pkt_send_payload().

Here is the call graph for this function:




Here is the caller graph for this function:




### uns8 pkt_process_rf_data (uns8 * *data_in*)

Call this routine when your RF device has received a chunk of data from somewhere. The packet delivery system will handle everything including acknowledgements and ignoring packets that it has already seen.

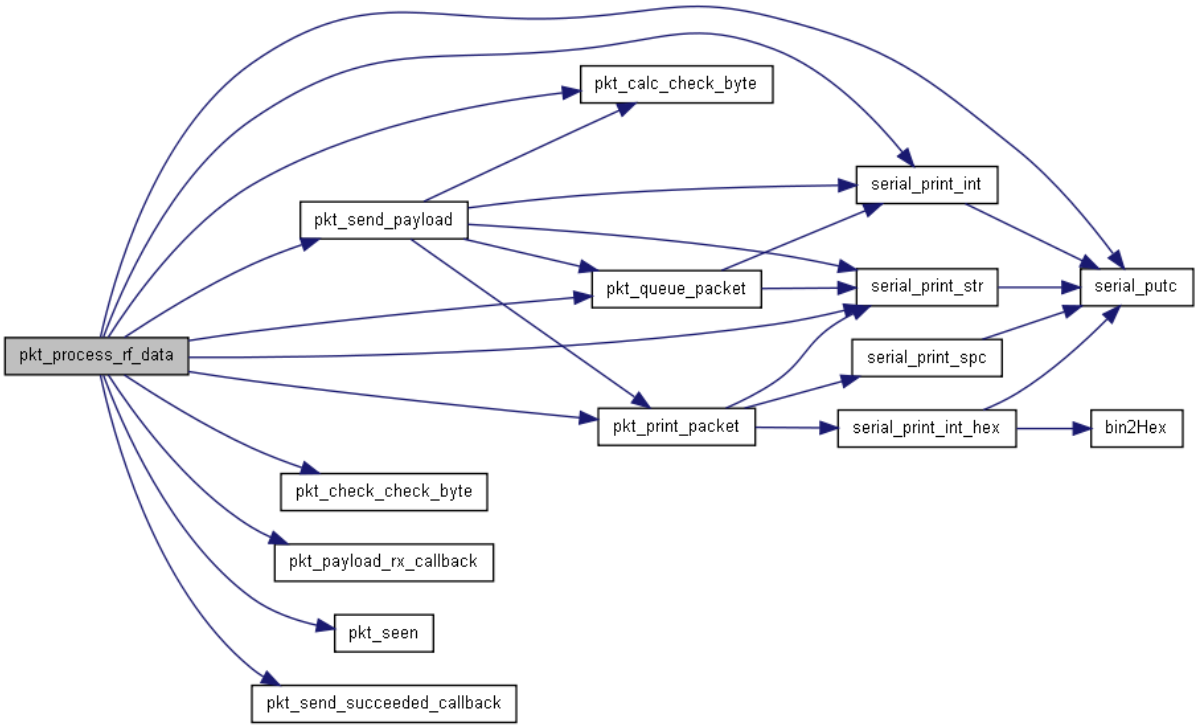
**Parameters:**

*pkt_in* A pointer to the received data. It is assumed that this will point to PKT_PACKET_SIZE bytes of data.

Definition at line 174 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, end_crit_sec, sending_item::flag, sending_item::packet, PKT_BROADCAST_ADDR, pkt_calc_check_byte(), pkt_check_check_byte(), PKT_DIRECT_SEND_ADDR, PKT_FLAG_DELETED, PKT_FLAG_NO_RESEND, seen_packet::pkt_id, rf_packet_det::pkt_id, pkt_my_addr, PKT_PACKET_SIZE, pkt_payload_rx_callback(), pkt_print_packet(), pkt_queue_packet(), pkt_seen(), pkt_seen_list_last, pkt_send_payload(), pkt_send_succeeded_callback(), PKT_STATUS_CHECK_FAIL, PKT_STATUS_DIRECT_SEND, PKT_STATUS_I_AM_SENDER, PKT_STATUS_NEED_TO_REBROADCAST, PKT_STATUS_PKT_FOR_ME_BUT_SEEN, PKT_STATUS_PKT_IS_ACK_FOR_ME, PKT_STATUS_PKT_IS_FACK_FOR_ME, PKT_STATUS_PKT_IS_FOR_ME, PKT_STATUS_PREVIOUS_ROUTED_VIA_ME, PKT_STATUS_ROUTING_FULL, PKT_STATUS_SEEN_BEFORE, serial_print_int(), serial_print_str(), serial_putc(), seen_packet::source_addr, start_crit_sec, uns16, and uns8.

Here is the call graph for this function:



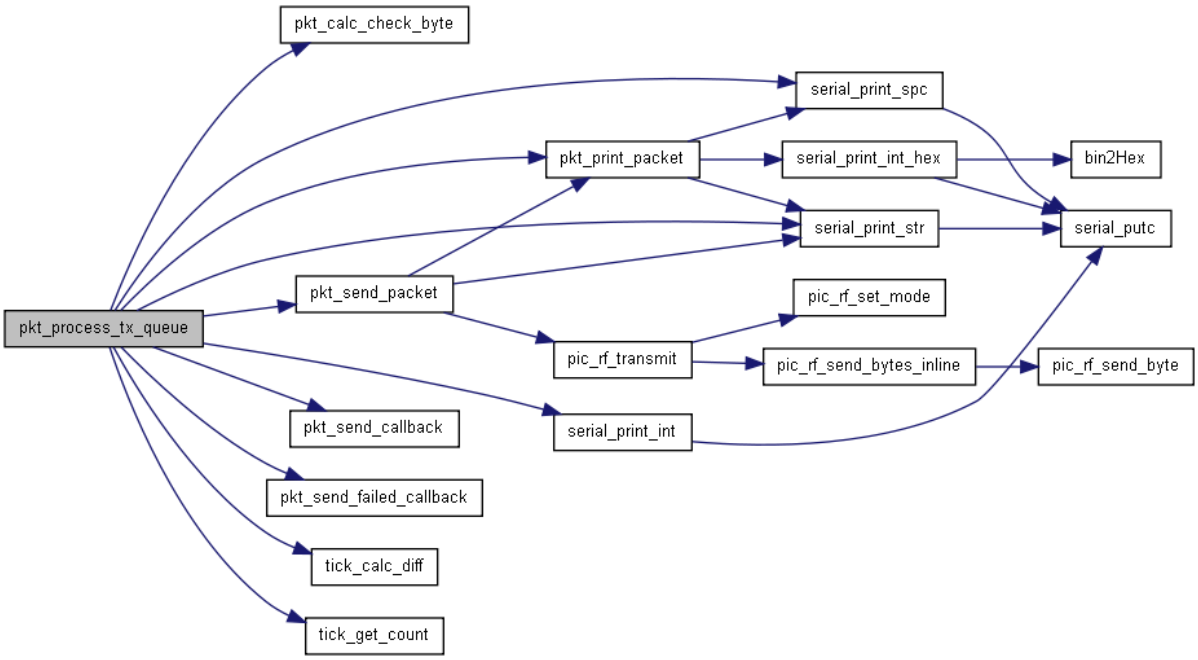

### void pkt_process_tx_queue ()

Call this routine regularly (as often as possible) in your main loop in order for the packet delivery system to send any queued packets when it is appropriate to do so. It will also remove any packets from the tx queue that have been there too long.

Definition at line 354 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, sending_item::flag, sending_item::packet, rf_packet_det::payload, pkt_calc_check_byte(), PKT_FLAG_DELETED, PKT_FLAG_RESEND, rf_packet_det::pkt_id, pkt_print_packet(), pkt_send_callback(), pkt_send_failed_callback(), pkt_send_packet(), rf_packet_det::r1_addr, sending_item::sent_count, serial_print_int(), serial_print_spc(), serial_print_str(), tick_calc_diff(), tick_get_count(), sending_item::tick_sent, uns16, and uns8.

Here is the call graph for this function:



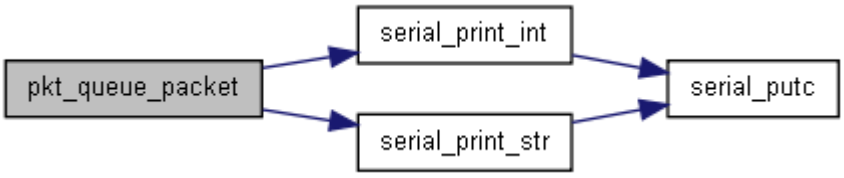

**uns8 pkt_queue_packet (rf_packet * *packet*, uns8 *resend*)**

Definition at line 144 of file pic_packet.c.

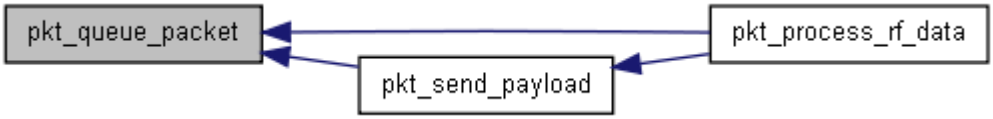
References sending_item::flag, PKT_FLAG_DELETED, PKT_PACKET_SIZE, PKT_STATUS_QUEUED, PKT_STATUS_TX_QUEUE_FULL, sending_item::sent_count, serial_print_int(), serial_print_str(), and uns8.

Referenced by pkt_process_rf_data(), and pkt_send_payload().

Here is the call graph for this function:




Here is the caller graph for this function:




**uns8 pkt_seen (uns16 *pkt_id*, uns16 *source_addr*)**

Definition at line 80 of file pic_packet.c.

References uns8.

Referenced by pkt_process_rf_data().

Here is the caller graph for this function:

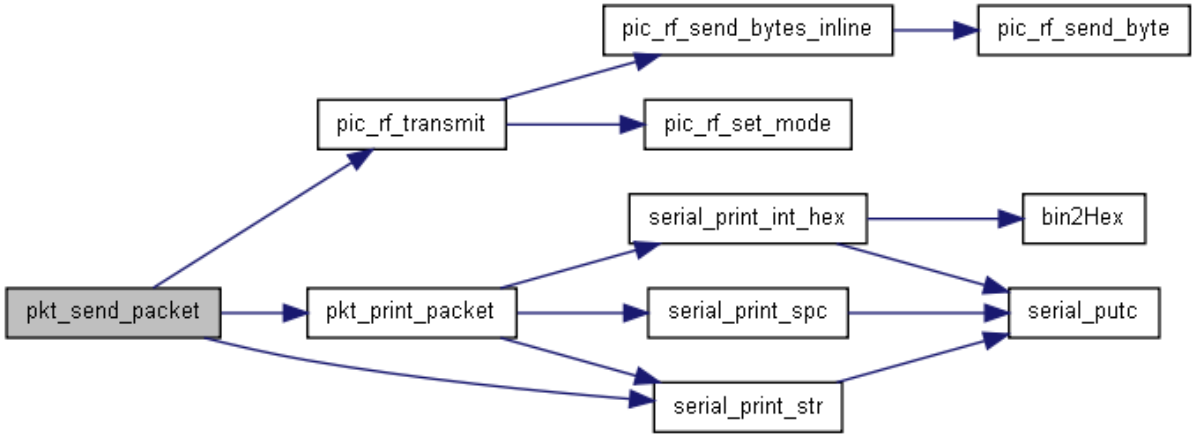

**void pkt_send_packet (rf_packet * *packet*) `[inline]`**

Definition at line 328 of file pic_packet.c.

References rf_packet::a, pic_rf_transmit(), PKT_PACKET_SIZE, pkt_print_packet(), serial_print_str(), tx_buffer, and uns8.

Referenced by pkt_process_tx_queue().

Here is the call graph for this function:




Here is the caller graph for this function:


**uns8 pkt_send_payload (uns16 *dest_addr*, uns8 * *payload*, uns8 *resend*)**

Use this routine to send a payload of data to the destination address. The payload must point to PKT_PAYLOAD_SIZE bytes of data (as defined in your config.h). The packet will be constructed, setting destination address, sender address (set previously in pkt_init) payload, initial routing directions and the check byte calculated. It will then be placed into the tx queue and will actually be sent next time pkt_process_tx_queue is called.

**Parameters:**

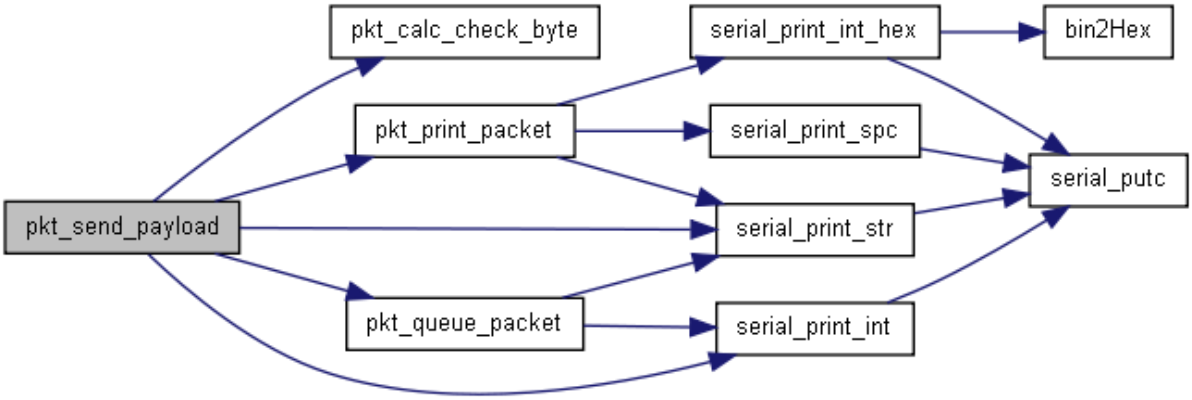
*dest_addr* Send payload to this address. Use address of PKT_BROADCAST_ADDR to broadcast to all listening local addresses
*pay_load* Pointer to PKT_PAYLOAD_SIZE array of bytes
*resend* Set to PKT_FLAG_RESEND or PKT_FLAG_NO_RESEND

Definition at line 440 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, rf_packet_det::payload, PKT_BROADCAST_ADDR, pkt_calc_check_byte(), PKT_DIRECT_SEND_ADDR, rf_packet_det::pkt_id, pkt_my_addr, pkt_my_next_pkt_id, pkt_print_packet(), pkt_queue_packet(), rf_packet_det::r1_addr, rf_packet_det::r2_addr, rf_packet_det::r3_addr, serial_print_int(), serial_print_str(), rf_packet_det::source_addr, and uns8.

Referenced by pkt_process_rf_data().

Here is the call graph for this function:




Here is the caller graph for this function:


---

## Variable Documentation

**uns16 pkt_my_addr = 0x66**

Definition at line 74 of file pic_packet.c.
Referenced by pkt_init(), pkt_process_rf_data(), and pkt_send_payload().

**uns16 pkt_my_next_pkt_id = 0**

Definition at line 75 of file pic_packet.c.
Referenced by pkt_init(), and pkt_send_payload().

**seen_packet pkt_seen_list[PKT_SEEN_LIST_SIZE] `[static]`**

Definition at line 70 of file pic_packet.c.

**uns8 pkt_seen_list_last = 0**

Definition at line 73 of file pic_packet.c.
Referenced by pkt_process_rf_data().

**sending_item pkt_tx_queue[PKT_TX_QUEUE_SIZE] `[static]`**

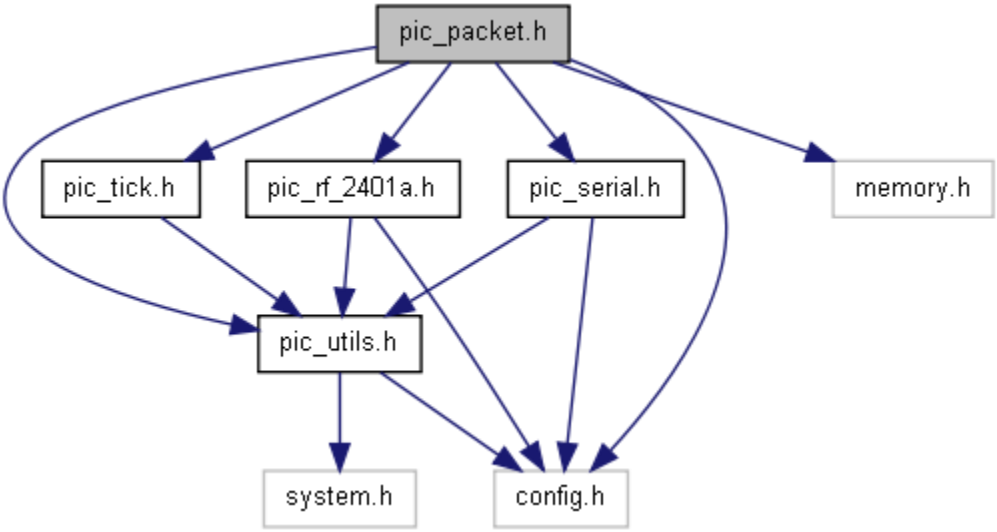
Definition at line 71 of file pic_packet.c.

# pic_packet.h File Reference

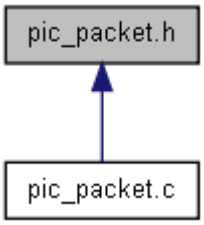
Pic meshed packed network library.

```
#include "pic_utils.h"
#include "pic_serial.h"
#include "pic_tick.h"
#include "config.h"
#include <memory.h>
#include "pic_rf_2401a.h"
```

Include dependency graph for pic_packet.h:




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




## Data Structures

- struct rf_packet_det

## Defines

- #define PKT_BROADCAST_ADDR 0xfffe
- #define PKT_CONFIG_ADDR 0xfffd
- #define PKT_DIRECT_SEND_ADDR 0xffff
- #define PKT_FLAG_BROADCAST 2
- #define PKT_FLAG_NO_RESEND 0
- #define PKT_FLAG_RESEND 1

- #define PKT_PACKET_SIZE sizeof(rf_packet_det)
- #define PKT_STATUS_CHECK_FAIL 12
- #define PKT_STATUS_DIRECT_SEND 6
- #define PKT_STATUS_I_AM_SENDER 2
- #define PKT_STATUS_NEED_TO_REBROADCAST 9
- #define PKT_STATUS_PKT_FOR_ME_BUT_SEEN 13
- #define PKT_STATUS_PKT_IS_ACK_FOR_ME 4
- #define PKT_STATUS_PKT_IS_FACK_FOR_ME 5
- #define PKT_STATUS_PKT_IS_FOR_ME 3
- #define PKT_STATUS_PREVIOUS_ROUTED_VIA_ME 7
- #define PKT_STATUS_QUEUED 10
- #define PKT_STATUS_ROUTING_FULL 8
- #define PKT_STATUS_SEEN_BEFORE 1
- #define PKT_STATUS_TX_QUEUE_FULL 11
- #define RF_RX_BUFFER_SIZE PKT_PACKET_SIZE

## Functions

- void pkt_init (uns16 my_addr, uns16 last_sent_pkt_id)
- *Initialise packet delivery system.* void pkt_payload_rx_callback (uns16 source_addr, uns16 pkt_id, uns8 *payload)
- *Called when a packet has been received.* uns8 pkt_process_rf_data (uns8 *data_in)
- *Process received RF data.* void pkt_process_tx_queue ()
- *Handle queued items.* void pkt_send_callback (uns16 dest_addr, uns16 pkt_id)
- void pkt_send_failed_callback (uns16 dest_addr, uns16 pkt_id)
- uns8 pkt_send_payload (uns16 dest_addr, uns8 *payload, uns8 resend)
- *Send a payload via the packet delivery system.* void pkt_send_succeeded_callback (uns16 dest_addr, uns16 pkt_id)

---

## Detailed Description

Assumes either nrf2401a or nrf24l01 library

Definition in file pic_packet.h.

---

## Define Documentation

### #define PKT_BROADCAST_ADDR 0xfffe

Send to anyone that will listen

Definition at line 172 of file pic_packet.h.

Referenced by pkt_process_rf_data(), and pkt_send_payload().

### #define PKT_CONFIG_ADDR 0xfffd

Magic config packet address

Definition at line 169 of file pic_packet.h.

### #define PKT_DIRECT_SEND_ADDR 0xffff

Router address for direct send only (no routing)

Definition at line 175 of file pic_packet.h.

Referenced by pkt_process_rf_data(), and pkt_send_payload().

**#define PKT_FLAG_BROADCAST 2**

Packet should be broadcast to anyone on the network

Definition at line 165 of file pic_packet.h.

**#define PKT_FLAG_NO_RESEND 0**

Packet should not be resent if it fails to reach destination

Definition at line 159 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_FLAG_RESEND 1**

Packet should be resent if it fails to reach destination

Definition at line 162 of file pic_packet.h.

Referenced by pkt_process_tx_queue().

**#define PKT_PACKET_SIZE sizeof(rf_packet_det)**

Definition at line 202 of file pic_packet.h.

Referenced by pkt_calc_check_byte(), pkt_check_check_byte(), pkt_process_rf_data(), pkt_queue_packet(), and pkt_send_packet().

**#define PKT_STATUS_CHECK_FAIL 12**

Packet is corrupt, ignoring

Definition at line 151 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_DIRECT_SEND 6**

Packet is direct send, but not for me, ignoring

Definition at line 133 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_I_AM_SENDER 2**

I have sent this packet, so ignoring

Definition at line 121 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_NEED_TO_REBROADCAST 9**

Packet is not for me, but protocol states I need to rebroadcast it

Definition at line 142 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_PKT_FOR_ME_BUT_SEEN 13**

Packet is for me but seen previously

Definition at line 154 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_PKT_IS_ACK_FOR_ME 4**

Packet is ACK for me, but didn't find in my transmit queue

Definition at line 127 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_PKT_IS_FACK_FOR_ME 5**

Packet is ACK for me, found in and removed from transmit queue (successful send)

Definition at line 130 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_PKT_IS_FOR_ME 3**

Packet is for me

Definition at line 124 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_PREVIOUS_ROUTED_VIA_ME 7**

Packet has my address in the router list, ignoring

Definition at line 136 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_QUEUED 10**

Packet queued for transmition

Definition at line 145 of file pic_packet.h.

Referenced by pkt_queue_packet().

**#define PKT_STATUS_ROUTING_FULL 8**

Packet not retransmitted since its routing list is full

Definition at line 139 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_SEEN_BEFORE 1**

Packet is already in seen list, ignoring

Definition at line 118 of file pic_packet.h.

Referenced by pkt_process_rf_data().

**#define PKT_STATUS_TX_QUEUE_FULL 11**

Packet has not been queued for transmition since my transmit queue is full

Definition at line 148 of file pic_packet.h.

Referenced by pkt_queue_packet().

#### #define RF_RX_BUFFER_SIZE PKT_PACKET_SIZE

Definition at line 204 of file pic_packet.h.

---

## Function Documentation

#### void pkt_init (uns16 *my_addr*, uns16 *last_sent_pkt_id*)

Initialise the packet delivery system ready for use. This routine clears the transmit queue and seen queue. If you don't store the last_sent_pkt_id (eg, in EEPROM) then set this to 0.

**Parameters:**

*my_addr* The address of this system
*last_sent_pkt_id* The last pkt_id used by this system.

Definition at line 423 of file pic_packet.c.

References sending_item::flag, PKT_FLAG_DELETED, pkt_my_addr, pkt_my_next_pkt_id, seen_packet::source_addr, uns16, and uns8.

#### void pkt_payload_rx_callback (uns16 *source_addr*, uns16 *pkt_id*, uns8 * *payload*)

Once a packet has been received, if it has not been seen before and it is destined to this address (set using pkt_init), then this routine will be called. You must have this routine defined in your own code.

**Parameters:**

*source_addr* The address of the system that sent the packet
*pkt_id* The ID of the packet (the source_addr and pkt_id are used to uniquely identify the packet
*payload* A pointer to PKT_PACKET_SIZE bytes received

Referenced by pkt_process_rf_data().

Here is the caller graph for this function:


#### uns8 pkt_process_rf_data (uns8 * *data_in*)

Call this routine when your RF device has received a chunk of data from somewhere. The packet delivery system will handle everything including acknowledgements and ignoring packets that it has already seen.

**Parameters:**

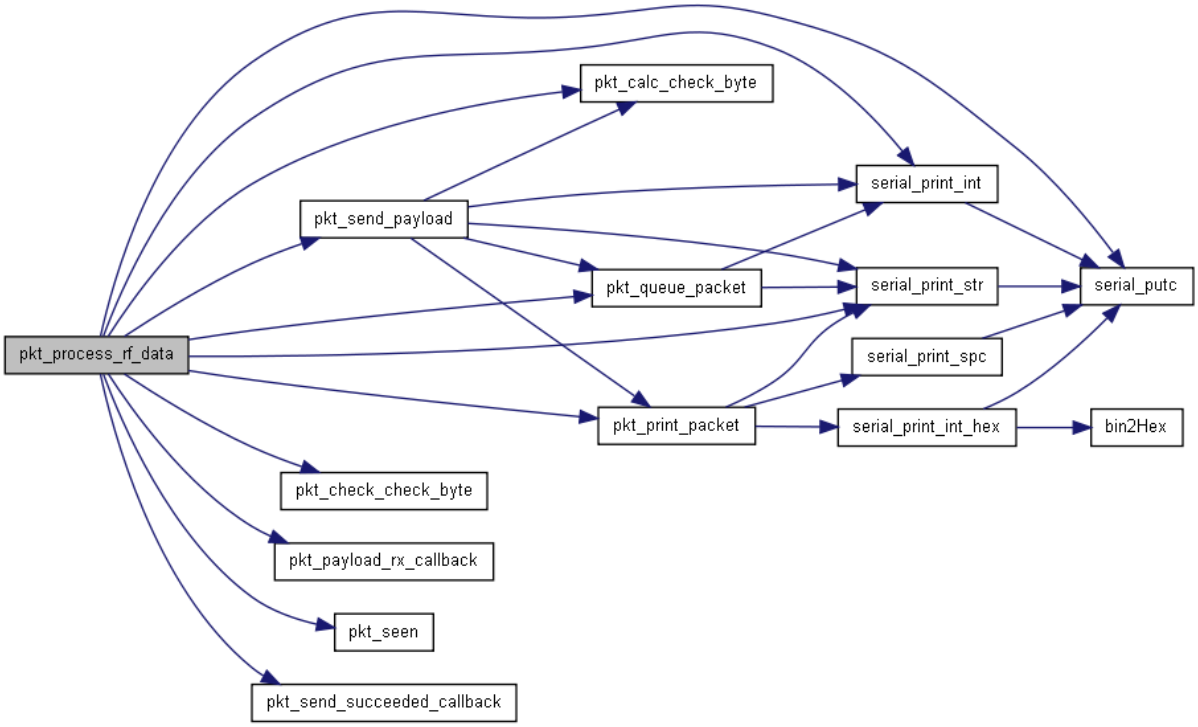
*pkt_in* A pointer to the received data. It is assumed that this will point to PKT_PACKET_SIZE bytes of data.

Definition at line 174 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, end_crit_sec, sending_item::flag, sending_item::packet, PKT_BROADCAST_ADDR, pkt_calc_check_byte(), pkt_check_check_byte(), PKT_DIRECT_SEND_ADDR, PKT_FLAG_DELETED, PKT_FLAG_NO_RESEND, seen_packet::pkt_id, rf_packet_det::pkt_id, pkt_my_addr, PKT_PACKET_SIZE, pkt_payload_rx_callback(), pkt_print_packet(), pkt_queue_packet(), pkt_seen(), pkt_seen_list_last, pkt_send_payload(), pkt_send_succeeded_callback(),

PKT_STATUS_CHECK_FAIL, PKT_STATUS_DIRECT_SEND, PKT_STATUS_I_AM_SENDER, PKT_STATUS_NEED_TO_REBROADCAST, PKT_STATUS_PKT_FOR_ME_BUT_SEEN, PKT_STATUS_PKT_IS_ACK_FOR_ME, PKT_STATUS_PKT_IS_FACK_FOR_ME, PKT_STATUS_PKT_IS_FOR_ME, PKT_STATUS_PREVIOUS_ROUTED_VIA_ME, PKT_STATUS_ROUTING_FULL, PKT_STATUS_SEEN_BEFORE, serial_print_int(), serial_print_str(), serial_putc(), seen_packet::source_addr, start_crit_sec, uns16, and uns8.

Here is the call graph for this function:



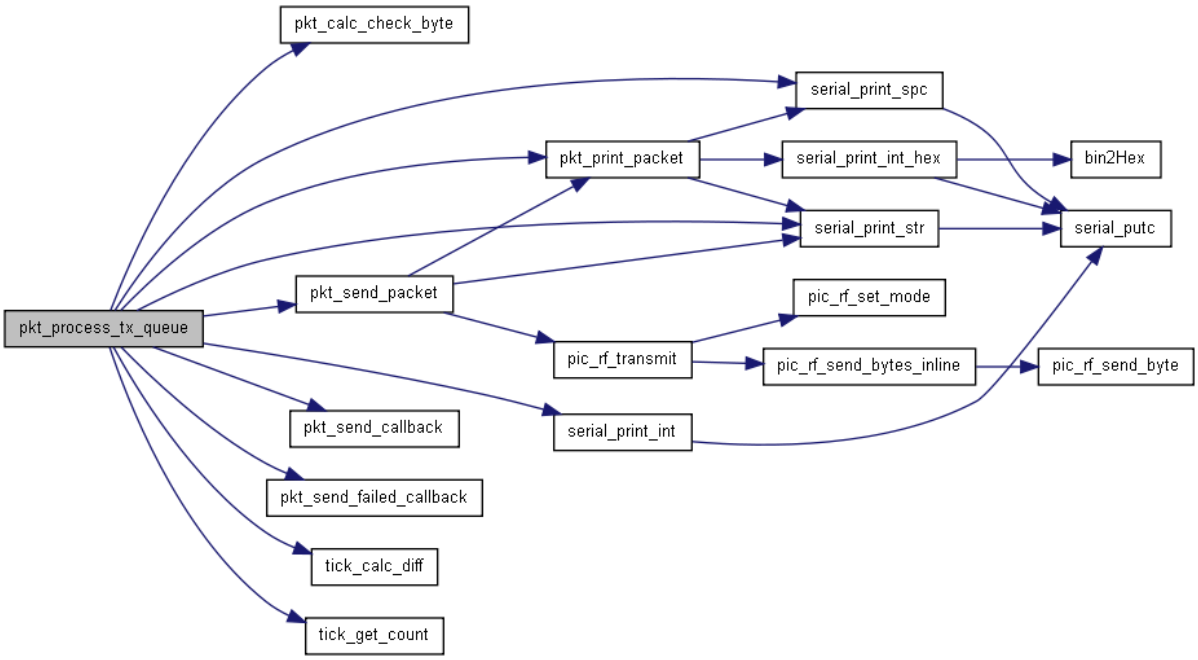

### void pkt_process_tx_queue ()

Call this routine regularly (as often as possible) in your main loop in order for the packet delivery system to send any queued packets when it is appropriate to do so. It will also remove any packets from the tx queue that have been there too long.

Definition at line 354 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, sending_item::flag, sending_item::packet, rf_packet_det::payload, pkt_calc_check_byte(), PKT_FLAG_DELETED, PKT_FLAG_RESEND, rf_packet_det::pkt_id, pkt_print_packet(), pkt_send_callback(), pkt_send_failed_callback(), pkt_send_packet(), rf_packet_det::r1_addr, sending_item::sent_count, serial_print_int(), serial_print_spc(), serial_print_str(), tick_calc_diff(), tick_get_count(), sending_item::tick_sent, uns16, and uns8.

Here is the call graph for this function:




**void pkt_send_callback (uns16 *dest_addr*, uns16 *pkt_id*)**

Referenced by pkt_process_tx_queue().

Here is the caller graph for this function:


**void pkt_send_failed_callback (uns16 *dest_addr*, uns16 *pkt_id*)**

Referenced by pkt_process_tx_queue().

Here is the caller graph for this function:


**uns8 pkt_send_payload (uns16 *dest_addr*, uns8 * *payload*, uns8 *resend*)**

Use this routine to send a payload of data to the destination address. The payload must point to PKT_PAYLOAD_SIZE bytes of data (as defined in your config.h). The packet will be constructed, setting destination address, sender address (set previously in pkt_init) payload, initial routing directions and the check byte calculated. It will then be placed into the tx queue and will actually be sent next time pkt_process_tx_queue is called.

**Parameters:**

*dest_addr* Send payload to this address. Use address of PKT_BROADCAST_ADDR to broadcast to all listening local addresses

*pay_load* Pointer to PKT_PAYLOAD_SIZE array of bytes

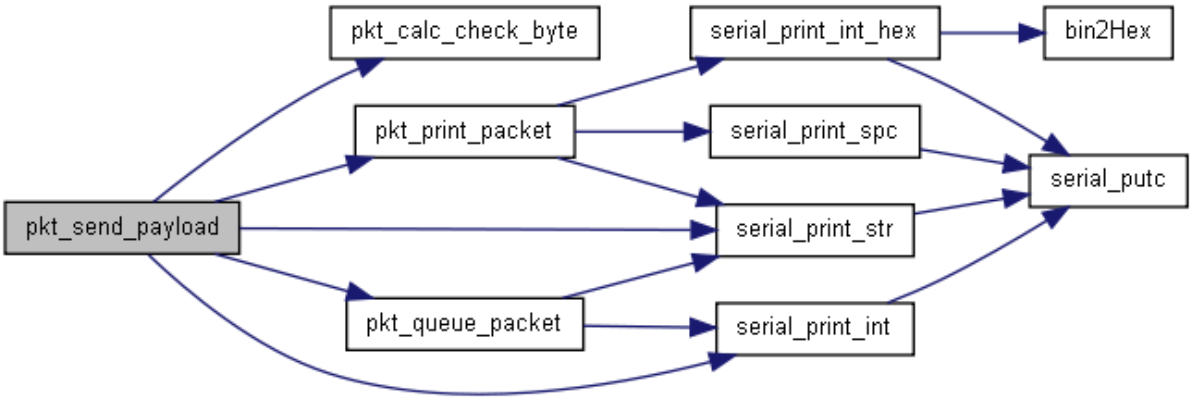
*resend* Set to PKT_FLAG_RESEND or PKT_FLAG_NO_RESEND

Definition at line 440 of file pic_packet.c.

References rf_packet::d, rf_packet_det::dest_addr, rf_packet_det::payload, PKT_BROADCAST_ADDR, pkt_calc_check_byte(), PKT_DIRECT_SEND_ADDR, rf_packet_det::pkt_id, pkt_my_addr, pkt_my_next_pkt_id, pkt_print_packet(), pkt_queue_packet(), rf_packet_det::r1_addr, rf_packet_det::r2_addr, rf_packet_det::r3_addr, serial_print_int(), serial_print_str(), rf_packet_det::source_addr, and uns8.

Referenced by pkt_process_rf_data().

Here is the call graph for this function:

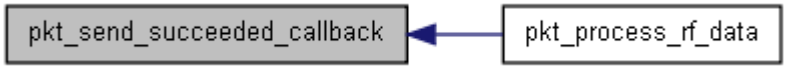



Here is the caller graph for this function:


**void pkt_send_succeeded_callback (uns16 *dest_addr*, uns16 *pkt_id*)**

Referenced by pkt_process_rf_data().

Here is the caller graph for this function:

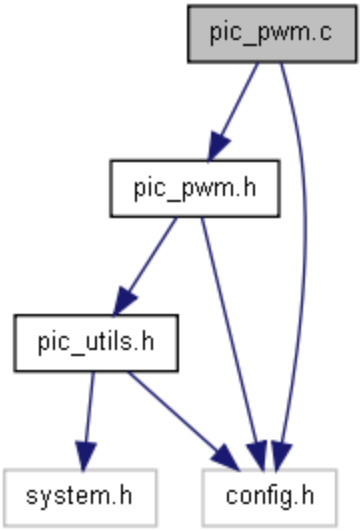



---

## pic_pwm.c File Reference

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

Include dependency graph for pic_pwm.c:




## Functions

- uns8 pwm_get_level (uns8 pwm_item)
- void pwm_handle ()
- void pwm_set_level (uns8 pwm_item, uns8 level)
- void pwm_set_transition (uns8 pwm_item, uns8 to_level, uns16 steps)
- void pwm_setup_io ()

## Variables

- uns8 pwm_count
- uns8 pwm_level [PWM_NUM_PINS]

---

## Function Documentation

### uns8 pwm_get_level (uns8 *pwm_item*)

Definition at line 50 of file pic_pwm.c.

References pwm_level.

### void pwm_handle ()

Definition at line 70 of file pic_pwm.c.

References clear_pin, pwm_count, pwm_level, set_pin, and uns8.

### void pwm_set_level (uns8 *pwm_item*, uns8 *level*)

Definition at line 45 of file pic_pwm.c.

References pwm_level.

**void pwm_set_transition (uns8 *pwm_item*, uns8 *to_level*, uns16 *steps*)**

Definition at line 54 of file pic_pwm.c.

**void pwm_setup_io ()**

Definition at line 58 of file pic_pwm.c.
References make_output.

---

## Variable Documentation

**uns8 pwm_count**

Definition at line 41 of file pic_pwm.c.
Referenced by pwm_handle().

**uns8 pwm_level[PWM_NUM_PINS]**

Definition at line 42 of file pic_pwm.c.
Referenced by pwm_get_level(), pwm_handle(), and pwm_set_level().

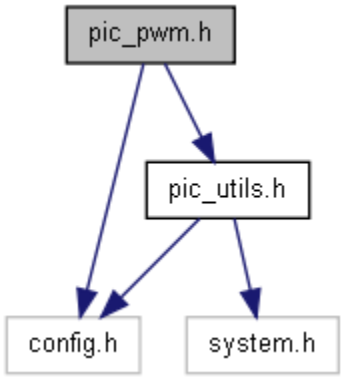
---

# pic_pwm.h File Reference

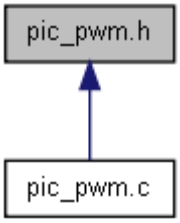
Software (timer-based) PWM.

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

Include dependency graph for pic_pwm.h:




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




## Functions

- uns8 pwm_get_level (uns8 pwm_item)
- void pwm_handle ()
- void pwm_set_level (uns8 pwm_item, uns8 level)
- void pwm_set_transition (uns8 to_level, uns16 steps)
- void pwm_setup_io ()

## Variables

- uns8 pwm_count
- uns8 pwm_level [PWM_NUM_PINS]

---

## Detailed Description

Definition in file pic_pwm.h.

---

## Function Documentation

### uns8 pwm_get_level (uns8 *pwm_item*)

Definition at line 50 of file pic_pwm.c.

References pwm_level.

### void pwm_handle ()

Definition at line 70 of file pic_pwm.c.

References clear_pin, pwm_count, pwm_level, set_pin, and uns8.

### void pwm_set_level (uns8 *pwm_item*, uns8 *level*)

Definition at line 45 of file pic_pwm.c.

References pwm_level.

**void pwm_set_transition (uns8 *to_level*, uns16 *steps*)**

**void pwm_setup_io ()**

Definition at line 58 of file pic_pwm.c.

References make_output.

---

## Variable Documentation

**uns8 pwm_count**

Definition at line 41 of file pic_pwm.c.

Referenced by pwm_handle().

**uns8 pwm_level[PWM_NUM_PINS]**

Definition at line 42 of file pic_pwm.c.

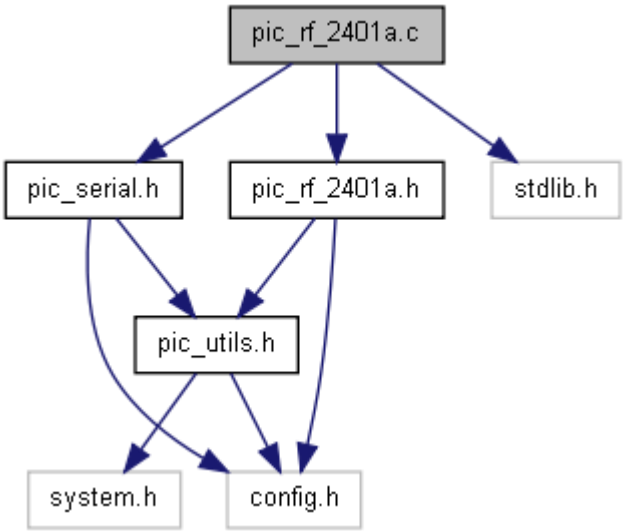
Referenced by pwm_get_level(), pwm_handle(), and pwm_set_level().

---

# pic_rf_2401a.c File Reference

```
#include "pic_rf_2401a.h"
#include "pic_serial.h"
#include <stdlib.h>
```

Include dependency graph for pic_rf_2401a.c:



## Functions

- void pic_rf_init (rf_config *my_config)
- *Initialise nrf2401a chip with config.* void pic_rf_quick_init (char *my_config, uns8 my_channel, bit my_receive_on)
- *Initialise nrf2401a chip with quick config.* void pic_rf_receive (uns8 *data, uns8 bytes_to_receive)
- *Receive data from nrf2401a.* void pic_rf_send_byte (uns8 b)
- *Internal routine to send a byte to nrf2401a.* void pic_rf_send_bytes (char *bytes, uns8 num_bytes)
- *Internal routine to send bytes to nrf2401a.* void pic_rf_set_channel (uns8 channel)
- *Change channel on the nrf2401a.* void pic_rf_set_mode (uns8 mode)
- *Set rf mode to transmit or receive.* void pic_rf_setup ()
- *Setup ports and pins for communication with nrf2401a.* void pic_rf_transmit (char *data, uns8 bytes_to_transmit)

*Transmit data from nrf2401a.*

---

## Function Documentation

### void pic_rf_init (rf_config * *my_config*)

Initialise nRF24L01 chip with config.

Sends the configuration to the Nordic nrf2401a chip ready to begin communication. This routine assumes you have already set my_config to the correct values.

Definition at line 83 of file pic_rf_2401a.c.

### void pic_rf_quick_init (char * *my_config*, uns8 *my_channel*, bit *my_receive_on*)

While the usual pic_rf_init() routine is excellent when you want to programatically change the 2401a config, if you're only doing this once (at the start) then it's likely you're burning a lot of instructions (154 words on a PIC16 device) just to send some bytes of config out to the 2401a. If you know your config in advance, then you can just send the byte-stream config using this routine. Use the nrf2401a_config.pl script in the tools directory to generate this string.

Definition at line 62 of file pic_rf_2401a.c.

### void pic_rf_receive (uns8 * *data*, uns8 *bytes_to_receive*)

Receive data from nRF24L01.

Having been notified that there is data available, call this routine to clock the data in from the nrf2401a.

!pic_rf_chip_enable(0); // save power

pic_rf_chip_enable(1); // turn chip back on

Definition at line 130 of file pic_rf_2401a.c.

### void pic_rf_send_byte (uns8 *b*)

Clock a byte into the nRF24L01.

Internal routine to send a byte to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

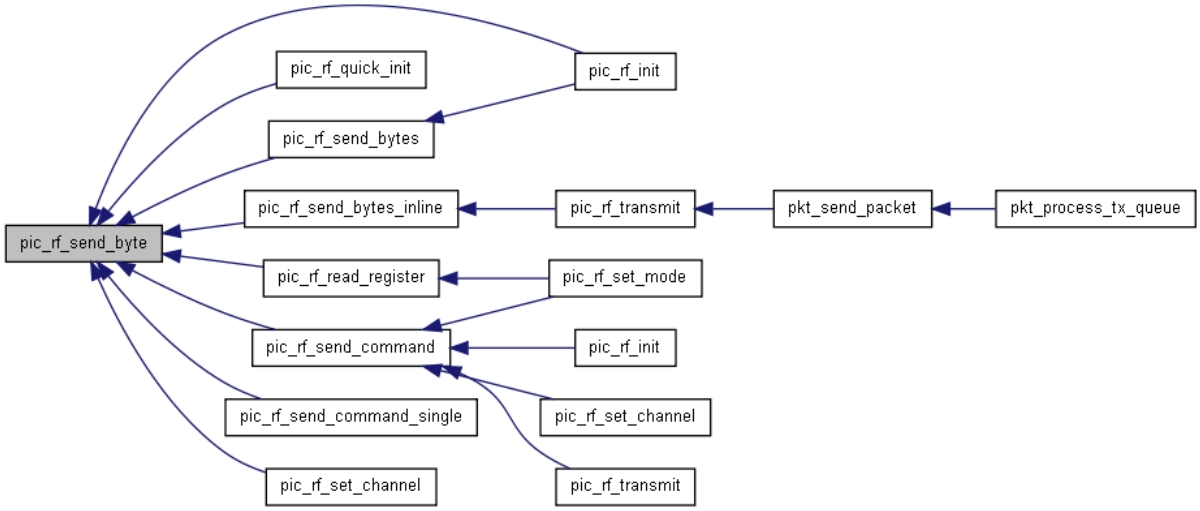
**See also:**

pic_rf_transmit

Definition at line 41 of file pic_rf_2401a.c.

Referenced by pic_rf_init(), pic_rf_quick_init(), pic_rf_read_register(), pic_rf_send_bytes(), pic_rf_send_bytes_inline(), pic_rf_send_command(), pic_rf_send_command_single(), and pic_rf_set_channel().

Here is the caller graph for this function:




### void pic_rf_send_bytes (char * *bytes*, uns8 *num_bytes*)

Internal routine to send bytes to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

**See also:**

pic_rf_transmit

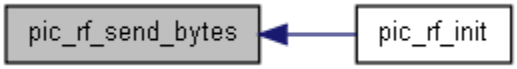
Definition at line 54 of file pic_rf_2401a.c.

References pic_rf_send_byte(), and uns8.

Referenced by pic_rf_init().

Here is the call graph for this function:


Here is the caller graph for this function:




### void pic_rf_set_channel (uns8 *channel*)

Receive data from nRF24L01 (inline).

Reclocks the essential config to change the current channel used by the nrf2401a.

Definition at line 199 of file pic_rf_2401a.c.

**void pic_rf_set_mode (uns8 *mode*)**

Pass RECEIVE_MODE or TRANSMIT_MODE to change current mode. Generally, you shouldn't need to call this routine. The library assumes you want to receive until you transmit, in which case it switches automatically to transmit mode and back to receive afterwards.

Definition at line 178 of file pic_rf_2401a.c.

Referenced by pic_rf_transmit().

Here is the caller graph for this function:


**void pic_rf_setup ()**

Setup ports and pins for communication with nRF24L01.

Set up ports and pins to correct input/output for communication with Nordif nrf2401a

Definition at line 221 of file pic_rf_2401a.c.

**void pic_rf_transmit (char * *data*, uns8 *bytes_to_transmit*)**

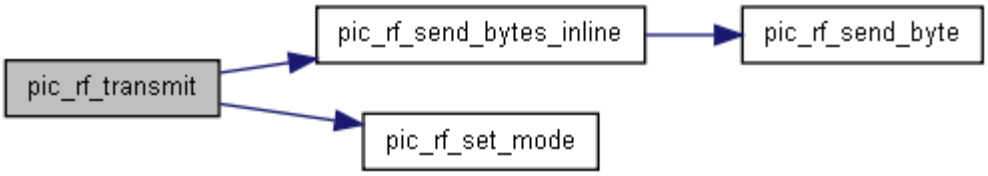
Changes to transmit mode, clocks data into the nRF2401A and hits the shockburst button. Returns to receive mode when finished.

Definition at line 154 of file pic_rf_2401a.c.

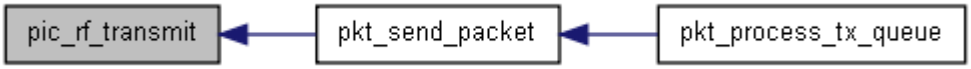
References kill_interrupts, make_output, pic_rf_chip_enable, pic_rf_send_bytes_inline(), pic_rf_set_mode(), RECEIVE_MODE, rf_current_mode_receive, TRANSMIT_MODE, and uns8.

Referenced by pkt_send_packet().

Here is the call graph for this function:




Here is the caller graph for this function:



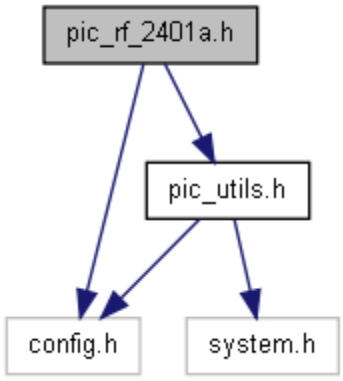

---

## pic_rf_2401a.h File Reference

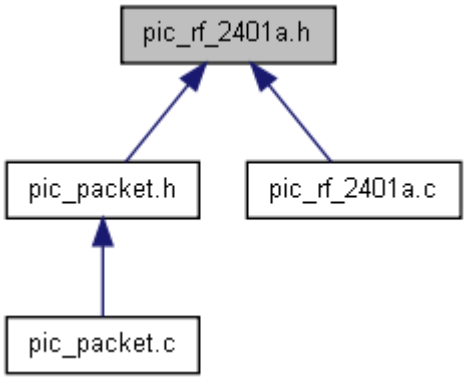
Pic Nordic nrf2401a routines.

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

Include dependency graph for pic_rf_2401a.h:




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




## Data Structures

- struct rf_config

## Defines

- #define OP_ENABLE_1_MBPS 5
- #define OP_ENABLE_CH2 7
- #define OP_ENABLE_CRC 2
- #define OP_ENABLE_RECEIVE 0
- #define OP_ENABLE_SHOCKBURST 6
- #define OP_LONG_CRC 1
- #define pic_rf_chip_enable(value) change_pin(rf_ce_port, rf_ce_pin, value);
- #define pic_rf_chip_select(value) change_pin(rf_cs_port, rf_cs_pin, value);
- #define pic_rf_init_inline(my_config)
- *Inline version of the pic_rf_init routine.* #define pic_rf_receive_mode() pic_rf_set_mode(RECEIVE_MODE)
- #define pic_rf_transmit_mode() pic_rf_set_mode(TRANSMIT_MODE)
- #define RECEIVE_MODE 1
- #define TRANSMIT_MODE 0

## Functions

- void pic_rf_init (rf_config *my_config)
- *Initialise nrf2401a chip with config.* void pic_rf_quick_init (char *my_config, uns8 my_channel, bit my_receive_on)
- *Initialise nrf2401a chip with quick config.* void pic_rf_receive (uns8 *data, uns8 bytes_to_receive)

- *Receive data from nrf2401a.* void pic_rf_send_byte (uns8 b)
- *Internal routine to send a byte to nrf2401a.* void pic_rf_send_bytes (char *bytes, uns8 num_bytes)
- *Internal routine to send bytes to nrf2401a.* void pic_rf_send_bytes_inline (char *bytes, uns8 num_bytes)
- *Inline version of send_bytes.* void pic_rf_set_channel (uns8 channel)
- *Change channel on the nrf2401a.* void pic_rf_set_mode (uns8 mode)
- *Set rf mode to transmit or receive.* void pic_rf_setup ()
- *Setup ports and pins for communication with nrf2401a.* void pic_rf_transmit (char *data, uns8 bytes_to_transmit)

### *Transmit data from nrf2401a.* Variables

- static uns8 rf_current_channel = 2
- static bit rf_current_mode_receive = 0

---

## Detailed Description

Nordic nrf2401a routines

Definition in file pic_rf_2401a.h.

---

## Define Documentation

#### #define OP_ENABLE_1_MBPS 5

rf_config options - enable 1 MPS transmition (otherwise 250Kbps)

Definition at line 101 of file pic_rf_2401a.h.

#### #define OP_ENABLE_CH2 7

rf_config options - enable channel 2 reception

Definition at line 105 of file pic_rf_2401a.h.

#### #define OP_ENABLE_CRC 2

rf_config options - enable CRC (recommended!)

Definition at line 99 of file pic_rf_2401a.h.

#### #define OP_ENABLE_RECEIVE 0

rf_config options - enable receive

Definition at line 95 of file pic_rf_2401a.h.

#### #define OP_ENABLE_SHOCKBURST 6

rf_config options - enable shockburst transmition

Definition at line 103 of file pic_rf_2401a.h.

#### #define OP_LONG_CRC 1

rf_config options - enable 16 bit CRC (otherwise 8 bit CRC if 0)

Definition at line 97 of file pic_rf_2401a.h.

**#define pic_rf_chip_enable(value)  change_pin(rf_ce_port, rf_ce_pin, value);**

Set chip enable line (CE) to the given value

Definition at line 183 of file pic_rf_2401a.h.

Referenced by pic_rf_init(), pic_rf_quick_init(), pic_rf_set_channel(), pic_rf_set_mode(), and pic_rf_transmit().

**#define pic_rf_chip_select(value)  change_pin(rf_cs_port, rf_cs_pin, value);**

Set chip select line (CS) to the given value

Definition at line 180 of file pic_rf_2401a.h.

Referenced by pic_rf_init(), pic_rf_quick_init(), pic_rf_set_channel(), and pic_rf_set_mode().

**#define pic_rf_init_inline(my_config)**

Depending upon your chip, if you're using the programmatic method of configuring the nrf2401a you may find that this routine helps you use less flash and/or stack space

Definition at line 192 of file pic_rf_2401a.h.

**#define pic_rf_receive_mode()  pic_rf_set_mode(RECEIVE_MODE)**

Change to receive mode

Definition at line 174 of file pic_rf_2401a.h.

**#define pic_rf_transmit_mode()  pic_rf_set_mode(TRANSMIT_MODE)**

Change to transmit mode

Definition at line 177 of file pic_rf_2401a.h.

**#define RECEIVE_MODE  1**

Mode selection - receive

Definition at line 109 of file pic_rf_2401a.h.

Referenced by pic_rf_set_mode(), and pic_rf_transmit().

**#define TRANSMIT_MODE  0**

Mode selection - transmit

Definition at line 111 of file pic_rf_2401a.h.

Referenced by pic_rf_set_mode(), and pic_rf_transmit().

---

## Function Documentation

**void pic_rf_init (rf_config * *my_config*)**

Sends the configuration to the Nordic nrf2401a chip ready to begin communication. This routine assumes you have already set my_config to the correct values.

Definition at line 83 of file pic_rf_2401a.c.

**void pic_rf_quick_init (char * *my_config*, uns8 *my_channel*, bit *my_receive_on*)**

While the usual pic_rf_init() routine is excellent when you want to programatically change the 2401a config, if you're only doing this once (at the start) then it's likely you're burning a lot of instructions (154 words on a PIC16 device) just to send some bytes of config out to the 2401a. If you know your config in advance, then you can just send the byte-stream config using this routine. Use the nrf2401a_config.pl script in the tools directory to generate this string.

Definition at line 62 of file pic_rf_2401a.c.

**void pic_rf_receive (uns8 * *data*, uns8 *bytes_to_receive*)**

Having been notified that there is data available, call this routine to clock the data in from the nrf2401a.

!pic_rf_chip_enable(0); // save power

pic_rf_chip_enable(1); // turn chip back on

Definition at line 130 of file pic_rf_2401a.c.

**void pic_rf_send_byte (uns8 *b*)**

Internal routine to send a byte to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

**See also:**

pic_rf_transmit

Definition at line 41 of file pic_rf_2401a.c.

**void pic_rf_send_bytes (char * *bytes*, uns8 *num_bytes*)**

Internal routine to send bytes to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

**See also:**

pic_rf_transmit

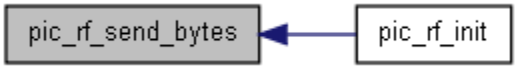
Definition at line 54 of file pic_rf_2401a.c.

References pic_rf_send_byte(), and uns8.

Referenced by pic_rf_init().

Here is the call graph for this function:


Here is the caller graph for this function:




**void pic_rf_send_bytes_inline (char * *bytes*, uns8 *num_bytes*) `[inline]`**

Inline version of pc_rf_send_bytes. This is an internal routine, and generally you shouldn't need to call it. Use pic_rf_transmit instead.

**See also:**

pic_rf_transmit

Definition at line 273 of file pic_rf_2401a.h.

References pic_rf_send_byte(), and uns8.

Referenced by pic_rf_transmit().

Here is the call graph for this function:


Here is the caller graph for this function:


### void pic_rf_set_channel (uns8 *channel*)

Reclocks the essential config to change the current channel used by the nrf2401a.

Definition at line 199 of file pic_rf_2401a.c.

### void pic_rf_set_mode (uns8 *mode*)

Pass RECEIVE_MODE or TRANSMIT_MODE to change current mode. Generally, you shouldn't need to call this routine. The library assumes you want to receive until you transmit, in which case it switches automatically to transmit mode and back to receive afterwards.

Definition at line 178 of file pic_rf_2401a.c.

### void pic_rf_setup ()

Set up ports and pins to correct input/output for communication with Nordif nrf2401a

Definition at line 221 of file pic_rf_2401a.c.

### void pic_rf_transmit (char * *data*, uns8 *bytes_to_transmit*)

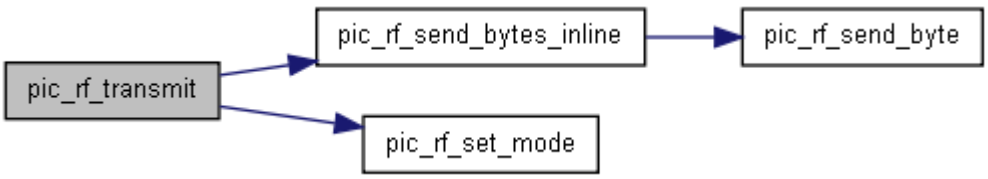
Changes to transmit mode, clocks data into the nRF2401A and hits the shockburst button. Returns to receive mode when finished.

Definition at line 154 of file pic_rf_2401a.c.

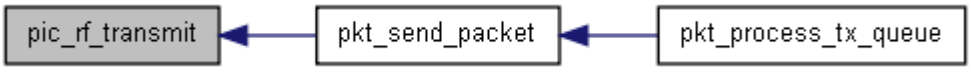
References kill_interrupts, make_output, pic_rf_chip_enable, pic_rf_send_bytes_inline(), pic_rf_set_mode(), RECEIVE_MODE, rf_current_mode_receive, TRANSMIT_MODE, and uns8.

Referenced by pkt_send_packet().

Here is the call graph for this function:




Here is the caller graph for this function:



## Variable Documentation

### uns8 rf_current_channel = 2 [static]

Maintain state of current channel

Definition at line 116 of file pic_rf_2401a.h.

Referenced by pic_rf_init(), pic_rf_quick_init(), and pic_rf_set_channel().

### bit rf_current_mode_receive = 0 [static]

Maintain state of current mode (1 = receive mode)

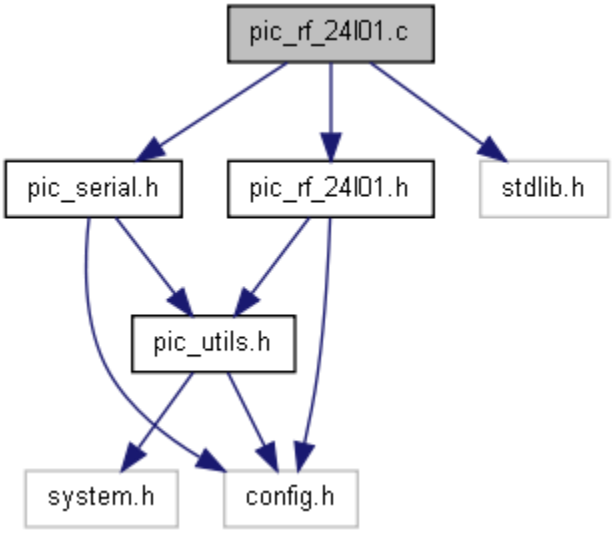
Definition at line 114 of file pic_rf_2401a.h.

Referenced by pic_rf_init(), pic_rf_quick_init(), pic_rf_set_channel(), pic_rf_set_mode(), and pic_rf_transmit().

# pic_rf_24l01.c File Reference

```
#include "pic_rf_24l01.h"
#include "pic_serial.h"
#include <stdlib.h>
```

Include dependency graph for pic_rf_24l01.c:




## Functions

- void pic_rf_init (rf_config *my_config)
- *Initialise nrf2401a chip with config.* void pic_rf_quick_init (char *my_config, uns8 my_channel, bit my_receive_on)
- *Initialise nrf2401a chip with quick config.* uns8 pic_rf_read_register (uns8 cmd, uns8 *data, uns8 data_len)
- *Read nRF24L01 register.* uns8 pic_rf_read_register_int (uns8 cmd, uns8 *data, uns8 data_len)

- uns8 pic_rf_receive (uns8 *data, uns8 bytes_to_receive)
- *Receive data from nrf2401a.* void pic_rf_receive2 (uns8 *data, uns8 bytes_to_receive)
- void pic_rf_receive_inline (uns8 *data, uns8 bytes_to_receive)
- *Receive data from nRF24L01.* uns8 pic_rf_send_byte (uns8 b)
- *Internal routine to send a byte to nrf2401a.* uns8 pic_rf_send_byte_int (uns8 b)
- *Clock a byte into the nRF24L01.* uns8 pic_rf_send_command (uns8 cmd, uns8 *data, uns8 data_len)
- *Send command to the nrf24l01.* uns8 pic_rf_send_command_single (uns8 cmd, uns8 data)
- *Send a single data byte command to the nrf24l01.* void pic_rf_set_channel (uns8 channel)
- *Change channel on the nrf2401a.* void pic_rf_set_mode (uns8 requested_mode)
- *Set rf mode to transmit or receive.* void pic_rf_setup ()
- *Setup ports and pins for communication with nrf2401a.* void pic_rf_transmit (uns8 *data, uns8 bytes_to_transmit)

*Transmit data from nRF24L01.*

## Function Documentation

### void pic_rf_init (rf_config * *my_config*)

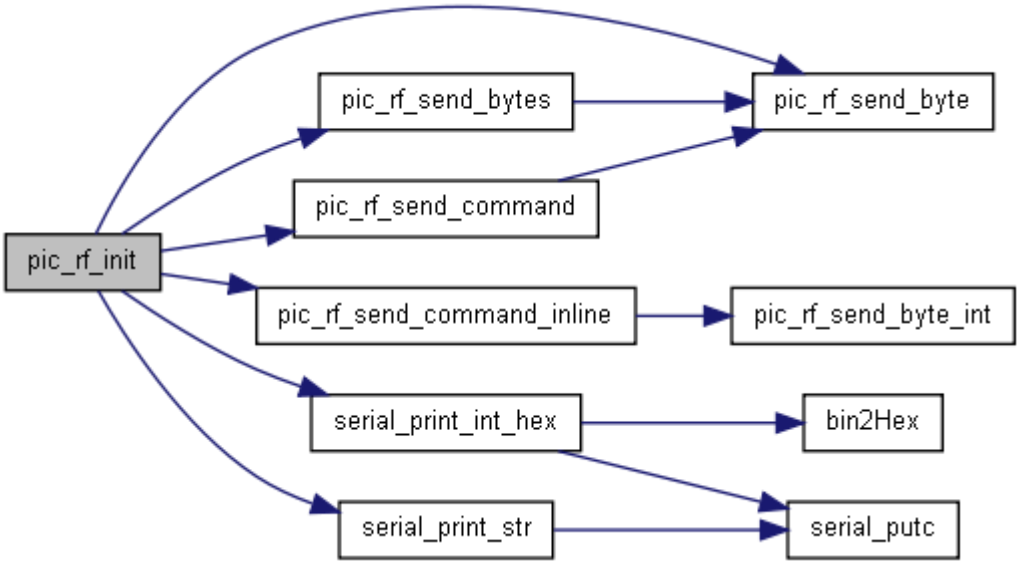
Initialise nRF24L01 chip with config.

Sends the configuration to the Nordic nrf2401a chip ready to begin communication. This routine assumes you have already set my_config to the correct values.

Definition at line 253 of file pic_rf_24l01.c.

References rf_config::address_ch1, rf_config::address_ch2, rf_config::address_width, rf_config::channel, clear_pin, rf_config::crystal, end_crit_sec, make_output, rf_config::options, rf_config::output_power, rf_config::payload_width_ch1, rf_config::payload_width_ch2, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), pic_rf_send_bytes(), pic_rf_send_command(), pic_rf_send_command_inline(), rf_current_channel, rf_current_mode_receive, RF_FLUSH_RX, RF_FLUSH_TX, RF_WR_REG_CONFIG_REG, RF_WR_REG_EN_AA, RF_WR_REG_RF_CH, RF_WR_REG_RF_SETUP, RF_WR_REG_RX_ADDR_P0, RF_WR_REG_RX_PW_P0, RF_WR_REG_SETUP_AW, RF_WR_REG_SETUP_RETR, RF_WR_REG_STATUS, RF_WR_REG_TX_ADDR, serial_print_int_hex(), serial_print_str(), set_pin, start_crit_sec, and uns8.

Here is the call graph for this function:



**void pic_rf_quick_init (char * *my_config*, uns8 *my_channel*, bit *my_receive_on*)**

While the usual pic_rf_init() routine is excellent when you want to programatically change the 2401a config, if you're only doing this once (at the start) then it's likely you're burning a lot of instructions (154 words on a PIC16 device) just to send some bytes of config out to the 2401a. If you know your config in advance, then you can just send the byte-stream config using this routine. Use the nrf2401a_config.pl script in the tools directory to generate this string.

Definition at line 157 of file pic_rf_24l01.c.

References clear_pin, make_output, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), rf_current_channel, rf_current_mode_receive, and uns8.

Here is the call graph for this function:


**uns8 pic_rf_read_register (uns8 *cmd*, uns8 * *data*, uns8 *data_len*)**

Internal routine to read a particular nRF24L01 register. Clocks out data_len bytes from the chip. Internal routine.

**Parameters:**

*cmd* Read register command, eg RF_RD_REG_STATUS
*data* Pointer to array of bytes where data will be put
*data_len* Number of bytes to clock out

Definition at line 92 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Referenced by pic_rf_set_mode().

Here is the call graph for this function:


Here is the caller graph for this function:


**uns8 pic_rf_read_register_int (uns8 *cmd*, uns8 * *data*, uns8 *data_len*)**

Definition at line 106 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte_int(), set_pin, and uns8.

Here is the call graph for this function:


**uns8 pic_rf_receive (uns8 * *data*, uns8 *bytes_to_receive*)**

Receive data from nRF24L01.

Having been notified that there is data available, call this routine to clock the data in from the nrf2401a.

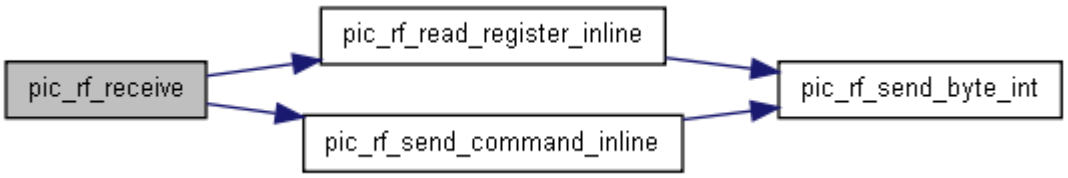
!pic_rf_chip_enable(0); // save power

pic_rf_chip_enable(1); // turn chip back on

Definition at line 41 of file pic_rf_24l01.c.

References clear_pin, kill_interrupts, make_input, pic_rf_read_register_inline(), pic_rf_send_command_inline(), RF_R_RX_PAYLOAD, RF_RD_REG_FIFO_STATUS, RF_WR_REG_STATUS, set_pin, test_pin, and uns8.

Here is the call graph for this function:




**void pic_rf_receive2 (uns8 * *data*, uns8 *bytes_to_receive*)**

Definition at line 352 of file pic_rf_24l01.c.

References pic_rf_receive_inline().

Here is the call graph for this function:


**void pic_rf_receive_inline (uns8 * *data*, uns8 *bytes_to_receive*) `[inline]`**

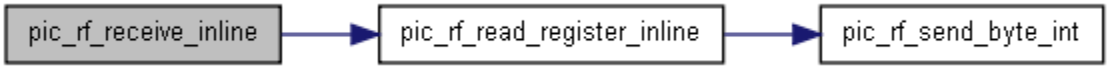
Having been notified that there is data available, call this routine to clock the data in from the nRF24L01.

Definition at line 348 of file pic_rf_24l01.c.

References pic_rf_read_register_inline(), and RF_R_RX_PAYLOAD.

Referenced by pic_rf_receive2().

Here is the call graph for this function:




Here is the caller graph for this function:


**uns8 pic_rf_send_byte (uns8 *b*)**

Clock a byte into the nRF24L01.

Internal routine to send a byte to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

**See also:**

pic_rf_transmit

Definition at line 123 of file pic_rf_24l01.c.

References change_pin, clear_pin, set_pin, test_pin, and uns8.

### uns8 pic_rf_send_byte_int (uns8 *b*)

Clock one byte into the nRF24L01. Internal routine.

**Parameters:**

*b* The byte to send

**Returns:**

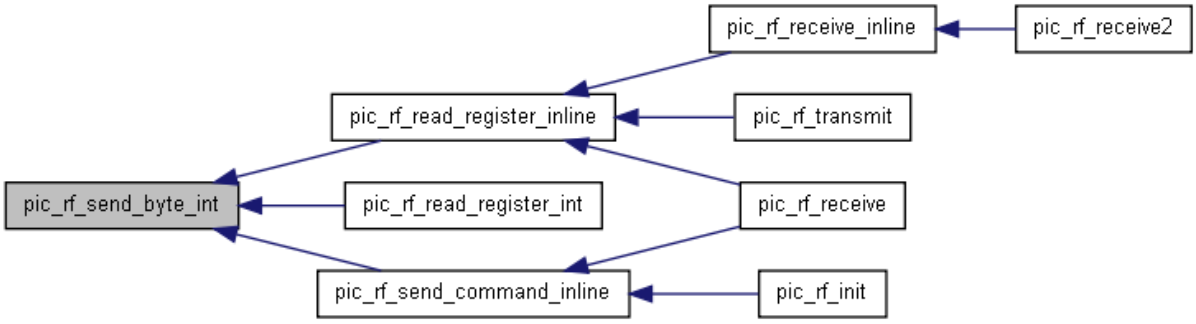
nRF24L01 status

Definition at line 140 of file pic_rf_24l01.c.

References change_pin, clear_pin, set_pin, test_pin, and uns8.

Referenced by pic_rf_read_register_inline(), pic_rf_read_register_int(), and pic_rf_send_command_inline().

Here is the caller graph for this function:




### uns8 pic_rf_send_command (uns8 *cmd*, uns8 * *data*, uns8 *data_len*)

Send a command and associated data to the nRF24L01

**Parameters:**

*cmd* Command to send, eg, RF_WR_REG_SETUP_RETR
*data* Pointer to an array of bytes to send as data for the command
*data_len* Number of bytes in the array

**Returns:**

nRF24L01 status

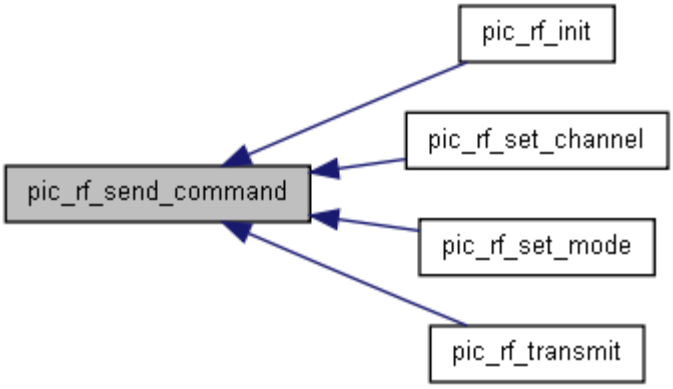
Definition at line 66 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Referenced by pic_rf_init(), pic_rf_set_channel(), pic_rf_set_mode(), and pic_rf_transmit().

Here is the call graph for this function:


Here is the caller graph for this function:




**uns8 pic_rf_send_command_single (uns8 *cmd*, uns8 *data*)**

Send a command and 1 byte of data to the nRF24L01

**Parameters:**

*cmd* Command to send, eg, RF_WR_REG_SETUP_RETR
*data* One byte of data for the command

**Returns:**

nRF24L01 status

Definition at line 80 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Here is the call graph for this function:


**void pic_rf_set_channel (uns8 *channel*)**

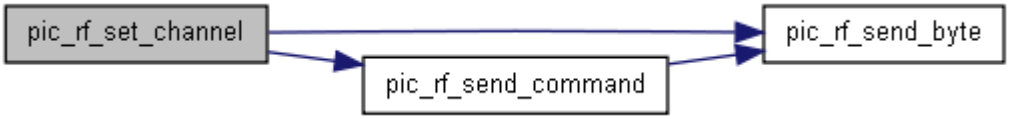
Receive data from nRF24L01 (inline).

Reclocks the essential config to change the current channel used by the nrf2401a.

Definition at line 385 of file pic_rf_24l01.c.

References clear_pin, end_crit_sec, kill_interrupts, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), pic_rf_send_command(), rf_current_channel, rf_current_mode_receive, RF_WR_REG_RF_CH, set_pin, and start_crit_sec.

Here is the call graph for this function:



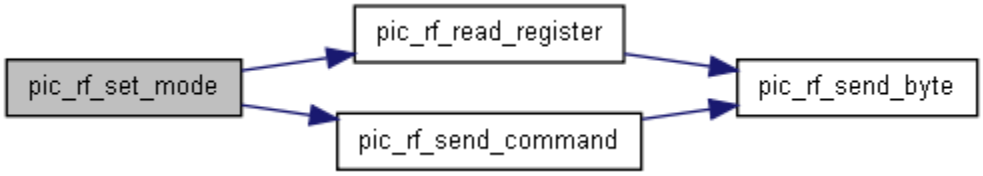

**void pic_rf_set_mode (uns8 *mode*)**

Pass RECEIVE_MODE or TRANSMIT_MODE to change current mode. Generally, you shouldn't need to call this routine. The library assumes you want to receive until you transmit, in which case it switches automatically to transmit mode and back to receive afterwards.

Definition at line 358 of file pic_rf_24l01.c.

References change_pin, clear_pin, CONFIG_PRIM_RX, end_crit_sec, kill_interrupts, make_output, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_read_register(), pic_rf_send_command(), RECEIVE_MODE, rf_current_mode_receive, RF_RD_REG_CONFIG_REG, RF_WR_REG_CONFIG_REG, set_pin, start_crit_sec, TRANSMIT_MODE, and uns8.

Here is the call graph for this function:




### void pic_rf_setup ()

Setup ports and pins for communication with nRF24L01.

Set up ports and pins to correct input/output for communication with Nordif nrf2401a

Definition at line 400 of file pic_rf_24l01.c.

References clear_pin, make_input, make_output, and set_pin.

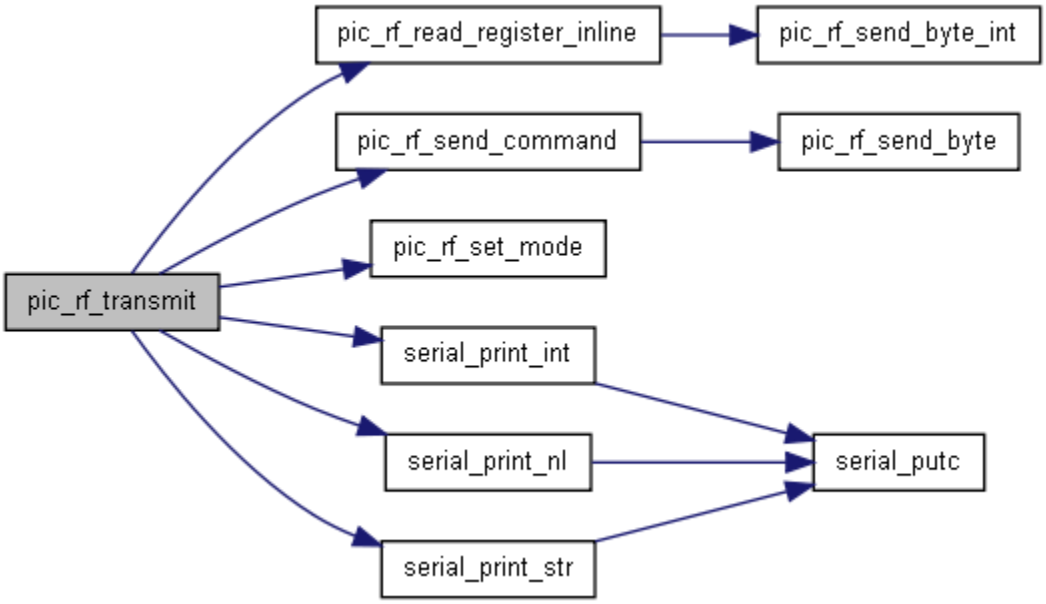
### void pic_rf_transmit (uns8 * *data*, uns8 *bytes_to_transmit*)

Changes to transmit mode, clocks data into the nrf24L01 and hits the shockburst button. Returns to receive mode when finished.

Definition at line 326 of file pic_rf_24l01.c.

References clear_pin, end_crit_sec, pic_rf_read_register_inline(), pic_rf_send_command(), pic_rf_set_mode(), RECEIVE_MODE, RF_RD_REG_CD, RF_W_TX_PAYLOAD, serial_print_int(), serial_print_nl(), serial_print_str(), set_pin, start_crit_sec, TRANSMIT_MODE, and uns8.

Here is the call graph for this function:

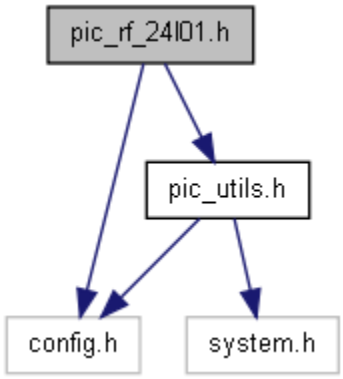


# pic_rf_24l01.h File Reference

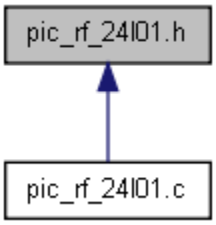
RF routines for the Nordic nRF24L01 chip.

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

Include dependency graph for pic_rf_24l01.h:




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




## Data Structures

- struct rf_config

## Defines

- #define CONFIG_CRCO 2
- #define CONFIG_EN_CRC 3
- #define CONFIG_MASK_MAX_RT 4
- #define CONFIG_MASK_RX_DR 6
- #define CONFIG_MASK_TX_DS 5
- #define CONFIG_PRIM_RX 0
- #define CONFIG_PWR_UP 1
- #define FIFO_STATUS_RX_EMPTY 0
- #define FIFO_STATUS_RX_FULL 1
- #define FIFO_STATUS_TX_EMPTY 4
- #define FIFO_STATUS_TX_FULL 5
- #define FIFO_STATUS_TX_REUSE 6
- #define OP_ENABLE_1_MBPS 5
- #define OP_ENABLE_CH2 7
- #define OP_ENABLE_CRC 2

- #define OP_ENABLE_RECEIVE 0
- #define OP_ENABLE_SHOCKBURST 6
- #define OP_LONG_CRC 1
- #define pic_rf_get_status() pic_rf_read_register(RF_NOP, 0, 0)
- #define pic_rf_receive_mode() pic_rf_set_mode(RECEIVE_MODE)
- #define pic_rf_set_status(status) pic_rf_send_command(RF_WR_REG_STATUS, status, 1)
- #define pic_rf_transmit_mode() pic_rf_set_mode(TRANSMIT_MODE)
- #define RECEIVE_MODE 1
- #define RF_FLUSH_RX 0b11100010
- #define RF_FLUSH_TX 0b11100001
- #define RF_NOP 0b11111111
- #define RF_R_RX_PAYLOAD 0b01100001
- #define RF_RD_REG_CD 0b00001001
- #define RF_RD_REG_CONFIG_REG 0b00000000
- #define RF_RD_REG_FIFO_STATUS 0b00010111
- #define RF_RD_REG_RX_PW_P0 0b00010001
- #define RF_RD_REG_STATUS 0b00000111
- #define RF_W_TX_PAYLOAD 0b10100000
- #define RF_WR_REG_CONFIG_REG 0b00100000
- #define RF_WR_REG_EN_AA 0b00100001
- #define RF_WR_REG_RF_CH 0b00100101
- #define RF_WR_REG_RF_SETUP 0b00100110
- #define RF_WR_REG_RX_ADDR_P0 0b00101010
- #define RF_WR_REG_RX_PW_P0 0b00110001
- #define RF_WR_REG_SETUP_AW 0b00100011
- #define RF_WR_REG_SETUP_RETR 0b00100100
- #define RF_WR_REG_STATUS 0b00100111
- #define RF_WR_REG_TX_ADDR 0b00110000
- #define STATUS_MAX_RT 4
- #define STATUS_RX_DR 6
- #define STATUS_TX_DS 5
- #define STATUS_TX_FULL 0
- #define TRANSMIT_MODE 0

## Functions

- void pic_rf_init (rf_config *my_config)
- *Initialise nRF24L01 chip with config.* void pic_rf_quick_init (char *my_config, uns8 my_channel, bit my_receive_on)
- *Initialise nrf2401a chip with quick config.* uns8 pic_rf_read_register (uns8 cmd, uns8 *data, uns8 data_len)
- *Read nRF24L01 register.* uns8 pic_rf_read_register_inline (uns8 cmd, uns8 *data, uns8 data_len)
- *Read nRF24L01 register (inline).* uns8 pic_rf_receive (uns8 *data, uns8 bytes_to_receive)
- *Receive data from nRF24L01.* void pic_rf_receive_inline (uns8 *data, uns8 bytes_to_receive)
- *Receive data from nRF24L01.* uns8 pic_rf_send_byte (uns8 b)
- *Clock a byte into the nRF24L01.* uns8 pic_rf_send_byte_int (uns8 b)
- *Clock a byte into the nRF24L01.* uns8 pic_rf_send_command (uns8 cmd, uns8 *data, uns8 data_len)
- *Send command to the nrf24l01.* uns8 pic_rf_send_command_inline (uns8 cmd, uns8 *data, uns8 data_len)
- *Send command to the nrf24l01 (inline).* uns8 pic_rf_send_command_single (uns8 cmd, uns8 data)
- *Send a single data byte command to the nrf24l01.* void pic_rf_set_channel (uns8 channel)
- *Receive data from nRF24L01 (inline).* void pic_rf_set_mode (uns8 mode)
- *Set rf mode to transmit or receive.* void pic_rf_setup ()
- *Setup ports and pins for communication with nRF24L01.* void pic_rf_transmit (uns8 *data, uns8 bytes_to_transmit)

## *Transmit data from nRF24L01.* Variables

- static uns8 rf_current_channel = 2
- static bit rf_current_mode_receive = 0

---

## Detailed Description

RF routines for the Nordic nRF24L01 chip

Definition in file pic_rf_24l01.h.

---

## Define Documentation

### #define CONFIG_CRCO 2

Definition at line 159 of file pic_rf_24l01.h.

### #define CONFIG_EN_CRC 3

Definition at line 158 of file pic_rf_24l01.h.

### #define CONFIG_MASK_MAX_RT 4

Definition at line 157 of file pic_rf_24l01.h.

### #define CONFIG_MASK_RX_DR 6

Definition at line 155 of file pic_rf_24l01.h.

### #define CONFIG_MASK_TX_DS 5

Definition at line 156 of file pic_rf_24l01.h.

### #define CONFIG_PRIM_RX 0

Definition at line 161 of file pic_rf_24l01.h.

Referenced by pic_rf_set_mode().

### #define CONFIG_PWR_UP 1

Definition at line 160 of file pic_rf_24l01.h.

### #define FIFO_STATUS_RX_EMPTY 0

Definition at line 175 of file pic_rf_24l01.h.

**#define FIFO_STATUS_RX_FULL 1**

Definition at line 174 of file pic_rf_24l01.h.

**#define FIFO_STATUS_TX_EMPTY 4**

Definition at line 173 of file pic_rf_24l01.h.

**#define FIFO_STATUS_TX_FULL 5**

Definition at line 172 of file pic_rf_24l01.h.

**#define FIFO_STATUS_TX_REUSE 6**

Definition at line 171 of file pic_rf_24l01.h.

**#define OP_ENABLE_1_MBPS 5**

rf_config options - enable 1 MPS transmition (otherwise 250Kbps)

Definition at line 107 of file pic_rf_24l01.h.

**#define OP_ENABLE_CH2 7**

rf_config options - enable channel 2 reception

Definition at line 111 of file pic_rf_24l01.h.

**#define OP_ENABLE_CRC 2**

rf_config options - enable CRC (recommended!)

Definition at line 105 of file pic_rf_24l01.h.

**#define OP_ENABLE_RECEIVE 0**

rf_config options - enable receive

Definition at line 101 of file pic_rf_24l01.h.

**#define OP_ENABLE_SHOCKBURST 6**

rf_config options - enable shockburst transmition

Definition at line 109 of file pic_rf_24l01.h.

**#define OP_LONG_CRC 1**

rf_config options - enable 16 bit CRC (otherwise 8 bit CRC if 0)

Definition at line 103 of file pic_rf_24l01.h.

**#define pic_rf_get_status() pic_rf_read_register(RF_NOP, 0, 0)**

Definition at line 369 of file pic_rf_24l01.h.

**#define pic_rf_receive_mode()  pic_rf_set_mode(RECEIVE_MODE)**

Definition at line 375 of file pic_rf_24l01.h.

**#define pic_rf_set_status(status)  pic_rf_send_command(RF_WR_REG_STATUS, status, 1)**

Definition at line 372 of file pic_rf_24l01.h.

**#define pic_rf_transmit_mode()  pic_rf_set_mode(TRANSMIT_MODE)**

Definition at line 376 of file pic_rf_24l01.h.

**#define RECEIVE_MODE  1**

Mode selection - receive

Definition at line 115 of file pic_rf_24l01.h.

**#define RF_FLUSH_RX  0b11100010**

Definition at line 145 of file pic_rf_24l01.h.

Referenced by pic_rf_init().

**#define RF_FLUSH_TX  0b11100001**

Definition at line 144 of file pic_rf_24l01.h.

Referenced by pic_rf_init().

**#define RF_NOP  0b11111111**

Definition at line 152 of file pic_rf_24l01.h.

**#define RF_R_RX_PAYLOAD  0b01100001**

Definition at line 149 of file pic_rf_24l01.h.

Referenced by pic_rf_receive(), and pic_rf_receive_inline().

**#define RF_RD_REG_CD  0b00001001**

Definition at line 143 of file pic_rf_24l01.h.

Referenced by pic_rf_transmit().

**#define RF_RD_REG_CONFIG_REG  0b00000000**

Definition at line 129 of file pic_rf_24l01.h.

Referenced by pic_rf_set_mode().

**#define RF_RD_REG_FIFO_STATUS 0b00010111**

Definition at line 142 of file pic_rf_24l01.h.
Referenced by pic_rf_receive().

**#define RF_RD_REG_RX_PW_P0 0b00010001**

Definition at line 140 of file pic_rf_24l01.h.

**#define RF_RD_REG_STATUS 0b00000111**

Definition at line 141 of file pic_rf_24l01.h.

**#define RF_W_TX_PAYLOAD 0b10100000**

Definition at line 148 of file pic_rf_24l01.h.
Referenced by pic_rf_transmit().

**#define RF_WR_REG_CONFIG_REG 0b00100000**

Definition at line 128 of file pic_rf_24l01.h.
Referenced by pic_rf_init(), and pic_rf_set_mode().

**#define RF_WR_REG_EN_AA 0b00100001**

Definition at line 135 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define RF_WR_REG_RF_CH 0b00100101**

Definition at line 137 of file pic_rf_24l01.h.
Referenced by pic_rf_init(), and pic_rf_set_channel().

**#define RF_WR_REG_RF_SETUP 0b00100110**

Definition at line 132 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define RF_WR_REG_RX_ADDR_P0 0b00101010**

Definition at line 134 of file pic_rf_24l01.h.

Referenced by pic_rf_init().

**#define RF_WR_REG_RX_PW_P0 0b00110001**

Definition at line 136 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define RF_WR_REG_SETUP_AW 0b00100011**

Definition at line 131 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define RF_WR_REG_SETUP_RETR 0b00100100**

Definition at line 130 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define RF_WR_REG_STATUS 0b00100111**

Definition at line 138 of file pic_rf_24l01.h.
Referenced by pic_rf_init(), and pic_rf_receive().

**#define RF_WR_REG_TX_ADDR 0b00110000**

Definition at line 133 of file pic_rf_24l01.h.
Referenced by pic_rf_init().

**#define STATUS_MAX_RT 4**

Definition at line 166 of file pic_rf_24l01.h.

**#define STATUS_RX_DR 6**

Definition at line 164 of file pic_rf_24l01.h.

**#define STATUS_TX_DS 5**

Definition at line 165 of file pic_rf_24l01.h.

**#define STATUS_TX_FULL 0**

Definition at line 168 of file pic_rf_24l01.h.

**#define TRANSMIT_MODE 0**

Mode selection - transmit

Definition at line 117 of file pic_rf_24l01.h.

---

## Function Documentation

**void pic_rf_init (rf_config * *my_config*)**

Sends the configuration to the Nordic nRF24L01 chip ready to begin communication. This routine assumes you have already set my_config to the correct values.

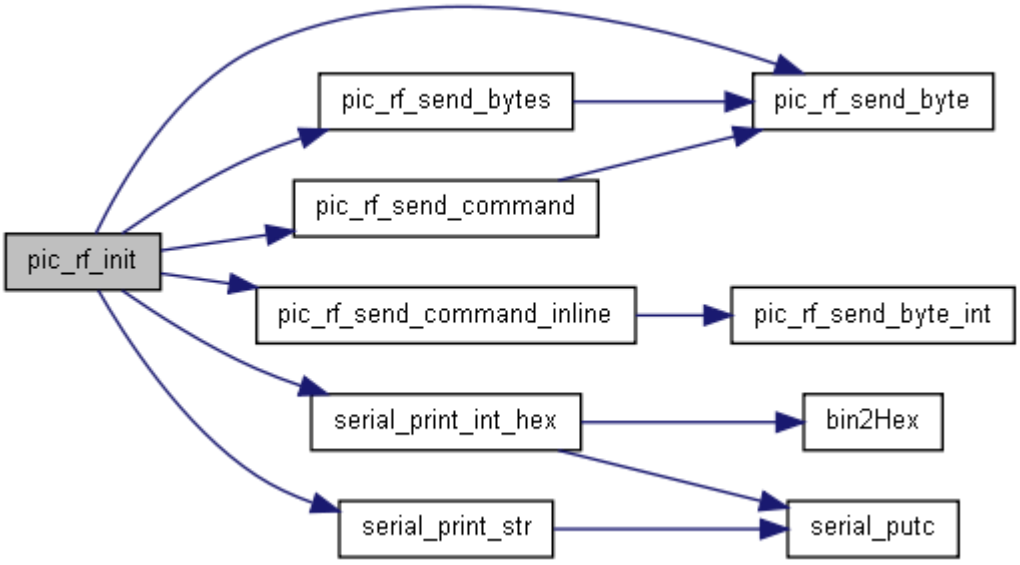
Initialise nRF24L01 chip with config.

Sends the configuration to the Nordic nrf2401a chip ready to begin communication. This routine assumes you have already set my_config to the correct values.

Definition at line 83 of file pic_rf_2401a.c.

References rf_config::address_ch1, rf_config::address_ch2, rf_config::address_width, rf_config::channel, clear_pin, rf_config::crystal, end_crit_sec, make_output, rf_config::options, rf_config::output_power, rf_config::payload_width_ch1, rf_config::payload_width_ch2, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), pic_rf_send_bytes(), pic_rf_send_command(), pic_rf_send_command_inline(), rf_current_channel, rf_current_mode_receive, RF_FLUSH_RX, RF_FLUSH_TX, RF_WR_REG_CONFIG, RF_WR_REG_EN_AA, RF_WR_REG_RF_CH, RF_WR_REG_RF_SETUP, RF_WR_REG_RX_ADDR_P0, RF_WR_REG_RX_PW_P0, RF_WR_REG_SETUP_AW, RF_WR_REG_SETUP_RETR, RF_WR_REG_STATUS, RF_WR_REG_TX_ADDR, serial_print_int_hex(), serial_print_str(), set_pin, start_crit_sec, and uns8.

Here is the call graph for this function:




**void pic_rf_quick_init (char * *my_config*, uns8 *my_channel*, bit *my_receive_on*)**

While the usual pic_rf_init() routine is excellent when you want to programatically change the 2401a config, if you're only doing this once (at the start) then it's likely you're burning a lot of instructions (154 words on a PIC16 device) just to send some bytes of config out to the 2401a. If you know your

config in advance, then you can just send the byte-stream config using this routine. Use the nrf2401a_config.pl script in the tools directory to generate this string.

Definition at line 62 of file pic_rf_2401a.c.

References clear_pin, make_output, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), rf_current_channel, rf_current_mode_receive, and uns8.

Here is the call graph for this function:


### uns8 pic_rf_read_register (uns8 *cmd*, uns8 * *data*, uns8 *data_len*)

Internal routine to read a particular nRF24L01 register. Clocks out data_len bytes from the chip. Internal routine.

**Parameters:**

*cmd* Read register command, eg RF_RD_REG_STATUS
*data* Pointer to array of bytes where data will be put
*data_len* Number of bytes to clock out

Definition at line 92 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Referenced by pic_rf_set_mode().

Here is the call graph for this function:


Here is the caller graph for this function:


### uns8 pic_rf_read_register_inline (uns8 *cmd*, uns8 * *data*, uns8 *data_len*) `[inline]`

Internal routine to read a particular nRF24L01 register. Clocks out data_len bytes from the chip. Internal routine. Inline version.

**Parameters:**

*cmd* Read register command, eg RF_RD_REG_STATUS
*data* Pointer to array of bytes where data will be put
*data_len* Number of bytes to clock out

**Returns:**

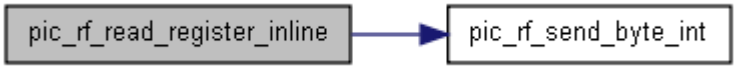
nRF24L01 status

Definition at line 326 of file pic_rf_24l01.h.

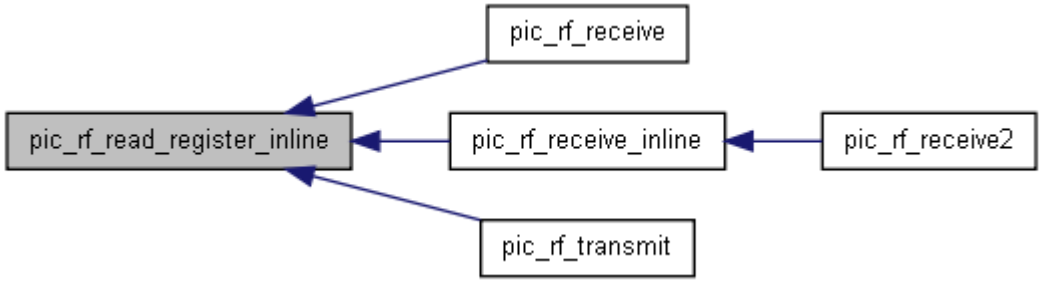
References clear_pin, pic_rf_send_byte_int(), set_pin, and uns8.

Referenced by pic_rf_receive(), pic_rf_receive_inline(), and pic_rf_transmit().

Here is the call graph for this function:



Here is the caller graph for this function:




**uns8 pic_rf_receive (uns8 * *data*, uns8 *bytes_to_receive*)**

Having been notified that there is data available, call this routine to clock the data in from the nRF24L01.

Receive data from nRF24L01.

Having been notified that there is data available, call this routine to clock the data in from the nrf2401a.

!pic_rf_chip_enable(0); // save power

pic_rf_chip_enable(1); // turn chip back on

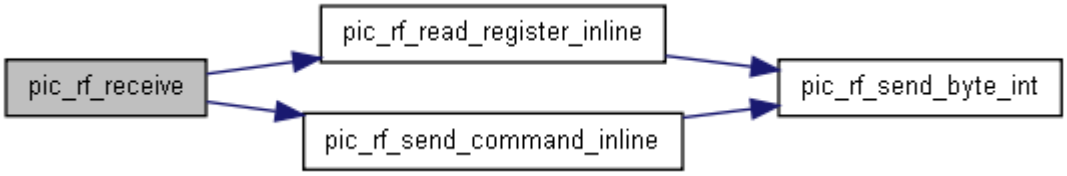
!pic_rf_chip_enable(0); // save power

pic_rf_chip_enable(1); // turn chip back on

Definition at line 130 of file pic_rf_2401a.c.

References clear_pin, kill_interrupts, make_input, pic_rf_read_register_inline(), pic_rf_send_command_inline(), RF_R_RX_PAYLOAD, RF_RD_REG_FIFO_STATUS, RF_WR_REG_STATUS, set_pin, test_pin, and uns8.

Here is the call graph for this function:




**void pic_rf_receive_inline (uns8 * *data*, uns8 *bytes_to_receive*) `[inline]`**

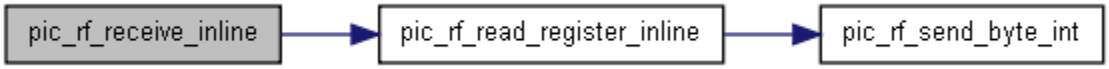
Having been notified that there is data available, call this routine to clock the data in from the nRF24L01.

Definition at line 348 of file pic_rf_24l01.c.

References pic_rf_read_register_inline(), and RF_R_RX_PAYLOAD.

Referenced by pic_rf_receive2().

Here is the call graph for this function:



Here is the caller graph for this function:


### uns8 pic_rf_send_byte (uns8 *b*)

Clock one byte into the nRF24L01. Internal routine.

**Parameters:**

*b* The byte to send

**Returns:**

nRF24L01 status

Clock a byte into the nRF24L01.

Internal routine to send a byte to the nrf2401a. Generally you shouldn't need to use this, see pic_rf_transmit instead

**See also:**

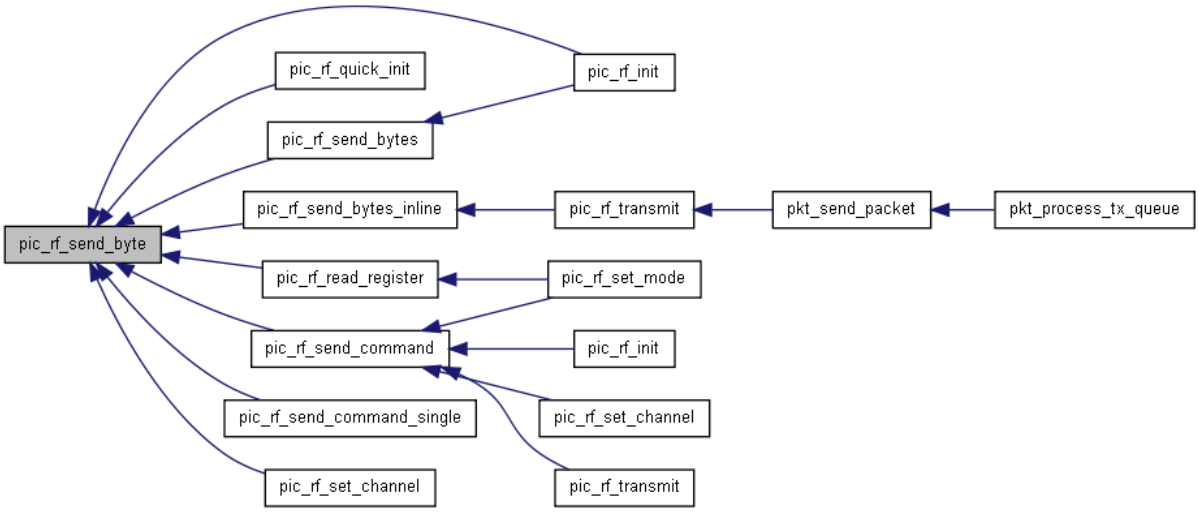
pic_rf_transmit

Definition at line 41 of file pic_rf_2401a.c.

References change_pin, clear_pin, set_pin, test_pin, and uns8.

Referenced by pic_rf_init(), pic_rf_quick_init(), pic_rf_read_register(), pic_rf_send_bytes(), pic_rf_send_bytes_inline(), pic_rf_send_command(), pic_rf_send_command_single(), and pic_rf_set_channel().

Here is the caller graph for this function:




### uns8 pic_rf_send_byte_int (uns8 *b*)

Clock one byte into the nRF24L01. Internal routine.

**Parameters:**

*b* The byte to send

**Returns:**

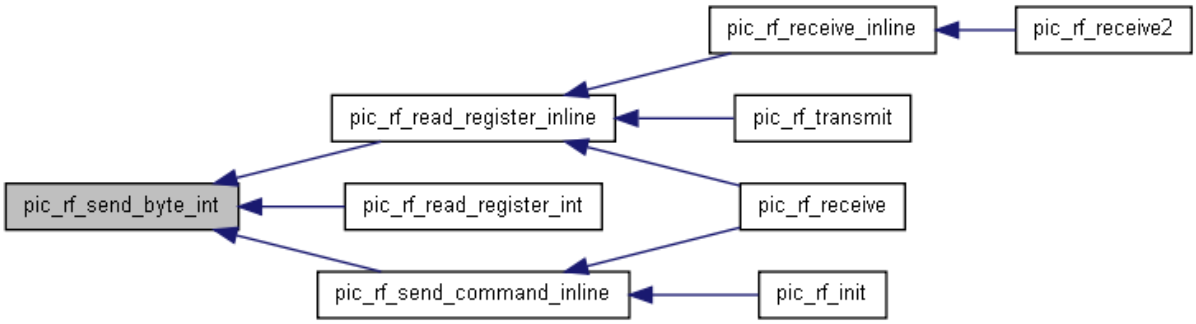
nRF24L01 status

Definition at line 140 of file pic_rf_24l01.c.

References change_pin, clear_pin, set_pin, test_pin, and uns8.

Referenced by pic_rf_read_register_inline(), pic_rf_read_register_int(), and pic_rf_send_command_inline().

Here is the caller graph for this function:




**uns8 pic_rf_send_command (uns8 *cmd*, uns8 * *data*, uns8 *data_len*)**

Send a command and associated data to the nRF24L01

**Parameters:**

*cmd* Command to send, eg, RF_WR_REG_SETUP_RETR
*data* Pointer to an array of bytes to send as data for the command
*data_len* Number of bytes in the array

**Returns:**

nRF24L01 status

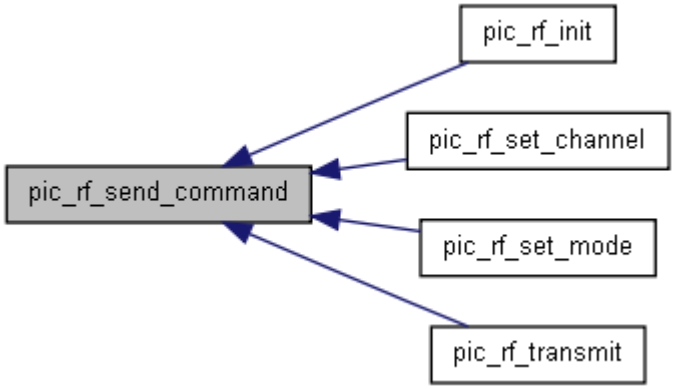
Definition at line 66 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Referenced by pic_rf_init(), pic_rf_set_channel(), pic_rf_set_mode(), and pic_rf_transmit().

Here is the call graph for this function:


Here is the caller graph for this function:




**uns8 pic_rf_send_command_inline (uns8 *cmd*, uns8 * *data*, uns8 *data_len*) `[inline]`**

Send a command and associated data to the nRF24L01. Internal routine. Inline version.

**Parameters:**

*cmd* Command to send, eg, RF_WR_REG_SETUP_RETR
*data* Pointer to an array of bytes to send as data for the command
*data_len* Number of bytes in the array

**Returns:**

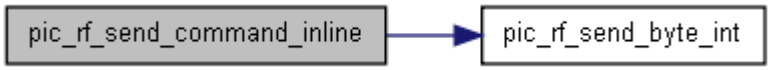
nRF24L01 status

Definition at line 354 of file pic_rf_24l01.h.

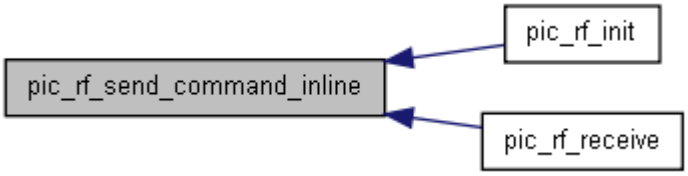
References clear_pin, pic_rf_send_byte_int(), set_pin, and uns8.

Referenced by pic_rf_init(), and pic_rf_receive().

Here is the call graph for this function:




Here is the caller graph for this function:




### uns8 pic_rf_send_command_single (uns8 *cmd*, uns8 *data*)

Send a command and 1 byte of data to the nRF24L01

**Parameters:**

*cmd* Command to send, eg, RF_WR_REG_SETUP_RETR
*data* One byte of data for the command

**Returns:**

nRF24L01 status

Definition at line 80 of file pic_rf_24l01.c.

References clear_pin, pic_rf_send_byte(), set_pin, and uns8.

Here is the call graph for this function:


### void pic_rf_set_channel (uns8 *channel*)

Having been notified that there is data available, call this routine to clock the data in from the nRF24L01. Change channel on the nRF24L01 Changes the current channel used by the nRF24L01.

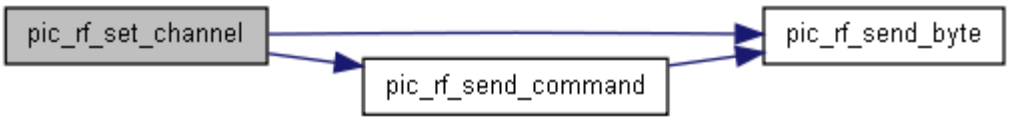
Receive data from nRF24L01 (inline).

Reclocks the essential config to change the current channel used by the nrf2401a.

Definition at line 199 of file pic_rf_2401a.c.

References clear_pin, end_crit_sec, kill_interrupts, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_send_byte(), pic_rf_send_command(), rf_current_channel, rf_current_mode_receive, RF_WR_REG_RF_CH, set_pin, and start_crit_sec.

Here is the call graph for this function:




**void pic_rf_set_mode (uns8 *mode*)**

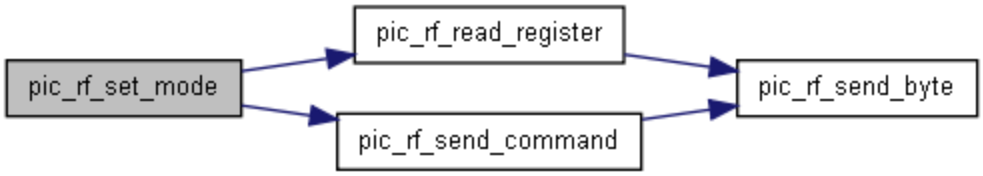
Pass RECEIVE_MODE or TRANSMIT_MODE to change current mode. Generally, you shouldn't need to call this routine. The library assumes you want to receive until you transmit, in which case it switches automatically to transmit mode and back to receive afterwards.

Definition at line 178 of file pic_rf_2401a.c.

References change_pin, clear_pin, CONFIG_PRIM_RX, end_crit_sec, kill_interrupts, make_output, pic_rf_chip_enable, pic_rf_chip_select, pic_rf_read_register(), pic_rf_send_command(), RECEIVE_MODE, rf_current_mode_receive, RF_RD_REG_CONFIG_REG, RF_WR_REG_CONFIG_REG, set_pin, start_crit_sec, TRANSMIT_MODE, and uns8.

Referenced by pic_rf_transmit().

Here is the call graph for this function:




Here is the caller graph for this function:


**void pic_rf_setup ()**

Set up ports and pins to correct input/output for communication with Nordic nRF24L01

Setup ports and pins for communication with nRF24L01.

Set up ports and pins to correct input/output for communication with Nordif nrf2401a

Definition at line 221 of file pic_rf_2401a.c.

References clear_pin, make_input, make_output, and set_pin.

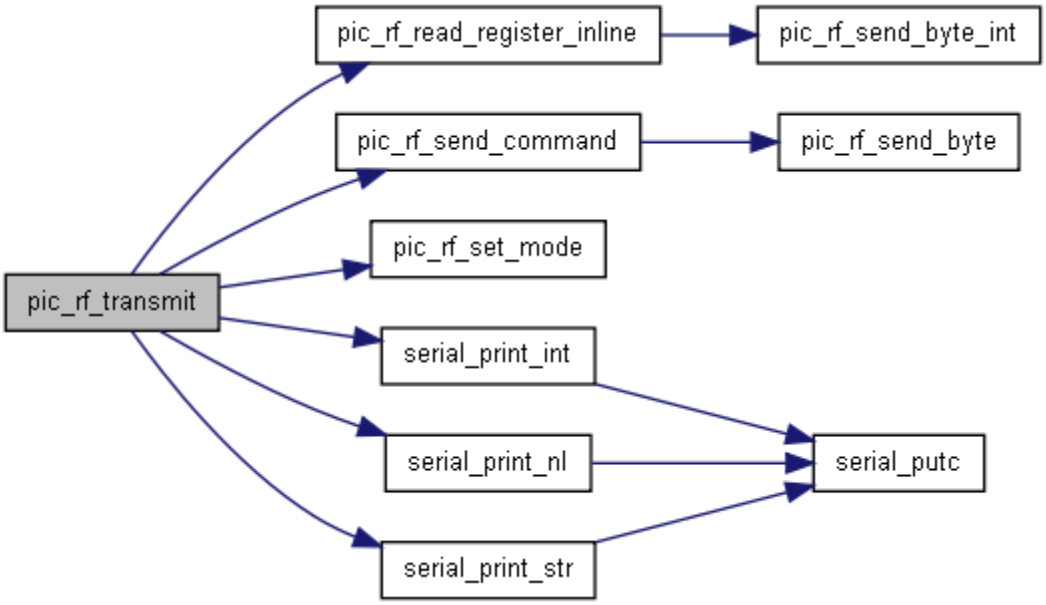
**void pic_rf_transmit (uns8 * *data*, uns8 *bytes_to_transmit*)**

Changes to transmit mode, clocks data into the nrf24L01 and hits the shockburst button. Returns to receive mode when finished.

Definition at line 326 of file pic_rf_24l01.c.

References clear_pin, end_crit_sec, pic_rf_read_register_inline(), pic_rf_send_command(), pic_rf_set_mode(), RECEIVE_MODE, RF_RD_REG_CD, RF_W_TX_PAYLOAD, serial_print_int(), serial_print_nl(), serial_print_str(), set_pin, start_crit_sec, TRANSMIT_MODE, and uns8.

Here is the call graph for this function:




### Variable Documentation

**uns8 rf_current_channel = 2 [static]**

Maintain state of current channel

Definition at line 122 of file pic_rf_24l01.h.

**bit rf_current_mode_receive = 0 [static]**

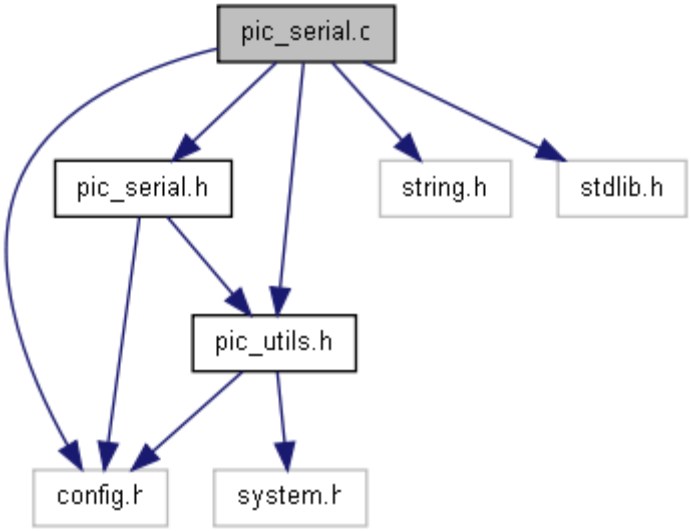
Maintain state of current mode (1 = receive mode)

Definition at line 120 of file pic_rf_24l01.h.

## pic_serial.c File Reference

```
#include "config.h"
#include "pic_utils.h"
#include "pic_serial.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for pic_serial.c:




## Functions

- uns8 bin2Hex (uns8 x)
- uns8 serial_getc (void)
- *Retrieve a character from the serial port.* void serial_print_int (uns16 i)
- *Print a 16 bit number to the serial port.* void serial_print_int_hex (uns8 i)
- *Print an 8 bit number in hex to the serial port.* void serial_print_int_hex_16bit (uns16 i)
- *Print a 16 bit number in hex to the serial port.* void serial_print_nl ()
- *Print a newline.* void serial_print_spc ()
- *Print a space.* void serial_print_str (rom char *str)
- *Print a rom string out to the serial port.* void serial_print_str (char *str)
- *Print a string out to the serial port.* void serial_print_var (char *str, uns16 i)
- void serial_putc (uns8 c)
- *Transmit a single character.* uns8 serial_rx_avail ()
- *Tests if the serial rx fifo has a character available.* void serial_rx_isr ()
- *Serial receive interrupt service routine.* void serial_setup (uns8 req_spbrg)
- *Configure the pic for serial communicaitons.* uns8 serial_tx_empty ()
- *Tests if the serial tx fifo is empty.* void serial_tx_isr ()

## *Serial transmit interrupt service routine.* Variables

- uns8 rx_buffer [SERIAL_RX_BUFFER_SIZE]
- uns8 rx_end = 0
- uns8 rx_start = 0
- uns8 tx_buffer [SERIAL_TX_BUFFER_SIZE]
- uns8 tx_end = 0
- uns8 tx_start = 0

---

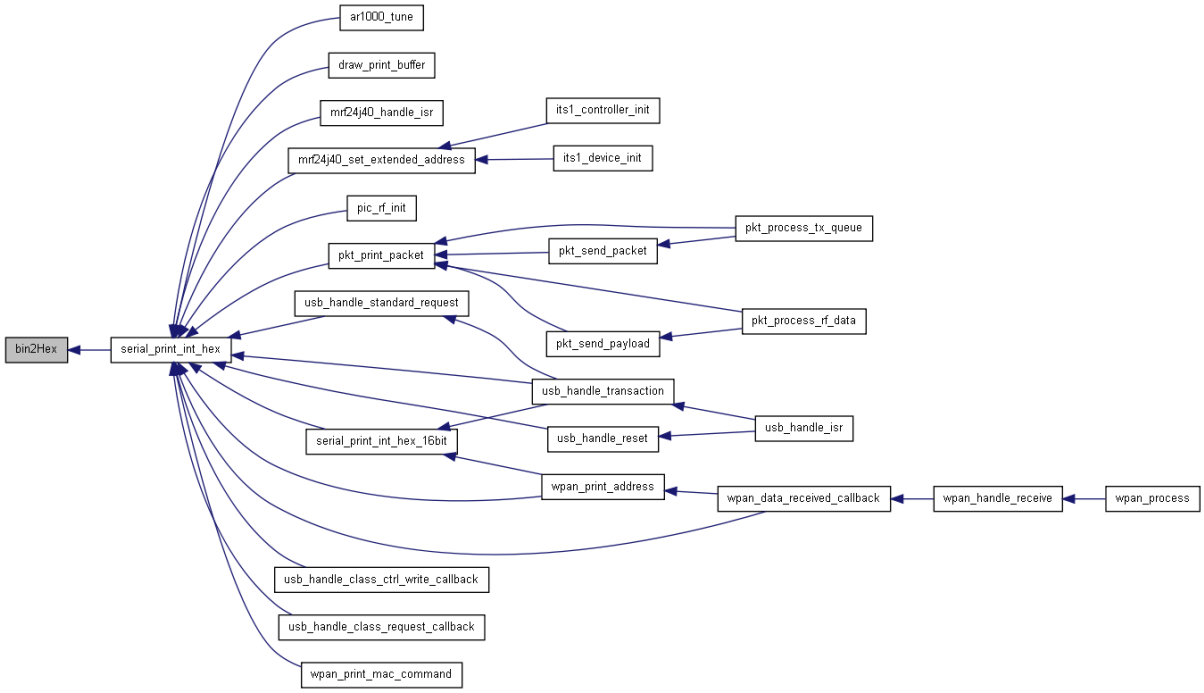
## Function Documentation

### uns8 bin2Hex (uns8 *x*) `[inline]`

Definition at line 65 of file pic_serial.c.

Referenced by serial_print_int_hex().

Here is the caller graph for this function:




### uns8 serial_getc (void)

Retrieve character from the serial port. Note that if there is nothing in the fifo, this function will wait until a character is received - and this will never happen if interrupts are turned off when this is called! So, be careful not to call getc during a critical section or during an ISR unless* you're sure there's something in the fifo. You can do this by calling the serial_rx_avail() routine. In any other situation, you can call getc() and happily wait for a character to arrive.

Definition at line 276 of file pic_serial.c.

References end_crit_sec, rx_buffer, rx_end, rx_start, start_crit_sec, and uns8.

Referenced by term_process().

Here is the caller graph for this function:


### void serial_print_int (uns16 *i*)

Print a 16 bit unsigned number in decimal to the serial port

**Parameters:**

*i* the 16 bit number to be printed

Definition at line 322 of file pic_serial.c.

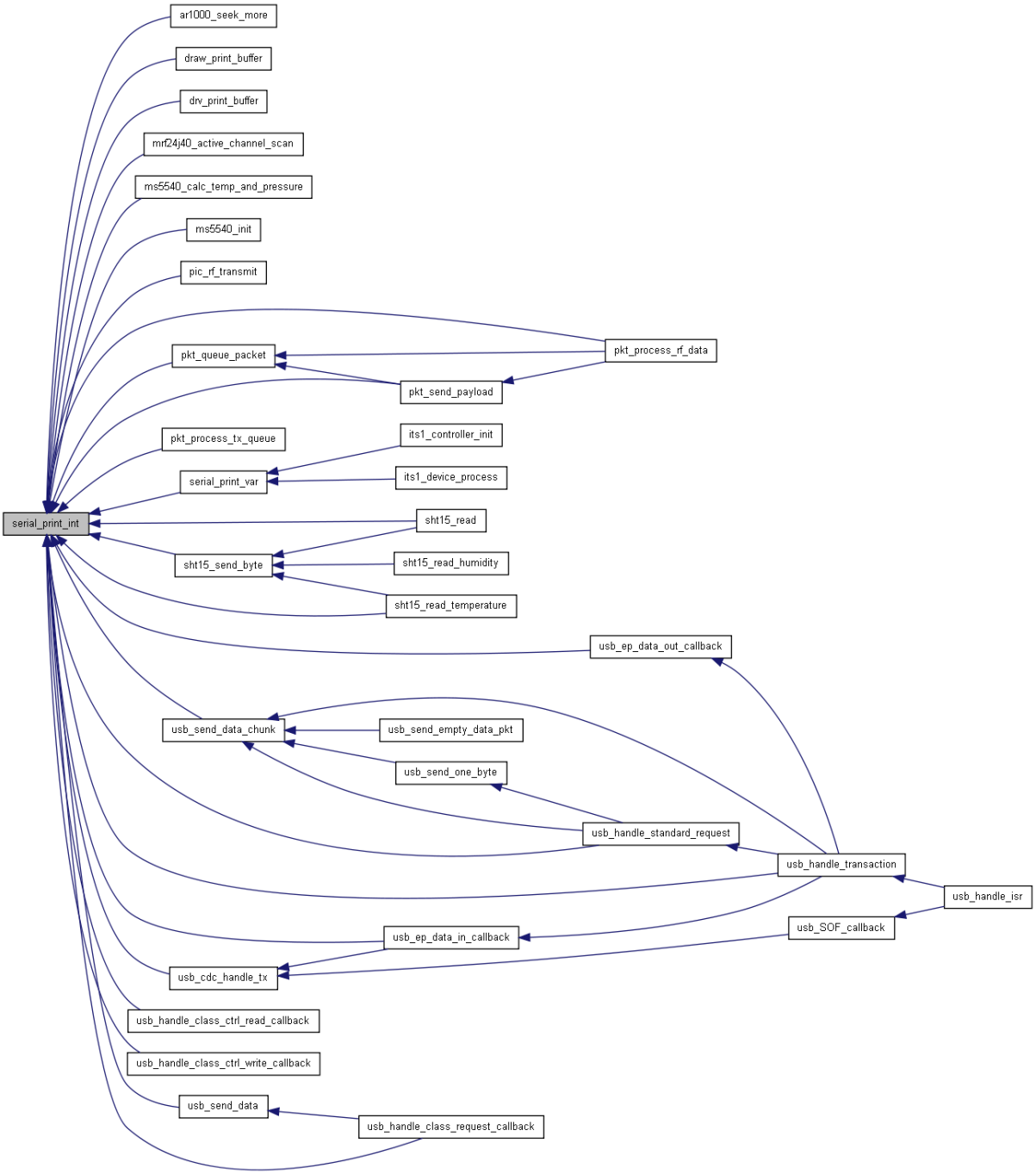
References serial_putc(), and uns8.

Referenced by ar1000_seek_more(), draw_print_buffer(), drv_print_buffer(), mrf24j40_active_channel_scan(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_transmit(), pkt_process_rf_data(), pkt_process_tx_queue(), pkt_queue_packet(), pkt_send_payload(), serial_print_var(), sht15_read(),

sht15_read_temperature(), sht15_send_byte(), usb_cdc_handle_tx(), usb_ep_data_in_callback(), usb_ep_data_out_callback(), usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data(), and usb_send_data_chunk().

Here is the call graph for this function:


Here is the caller graph for this function:



**void serial_print_int_hex (uns8 *i*)**

Print a 8 bit unsigned number in hex to the serial port

**Parameters:**

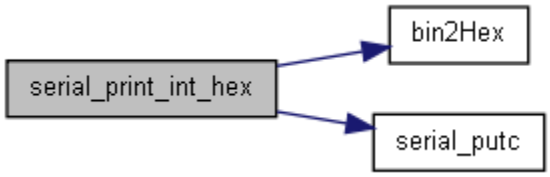
*i* 8 bit number to be printed

Definition at line 343 of file pic_serial.c.

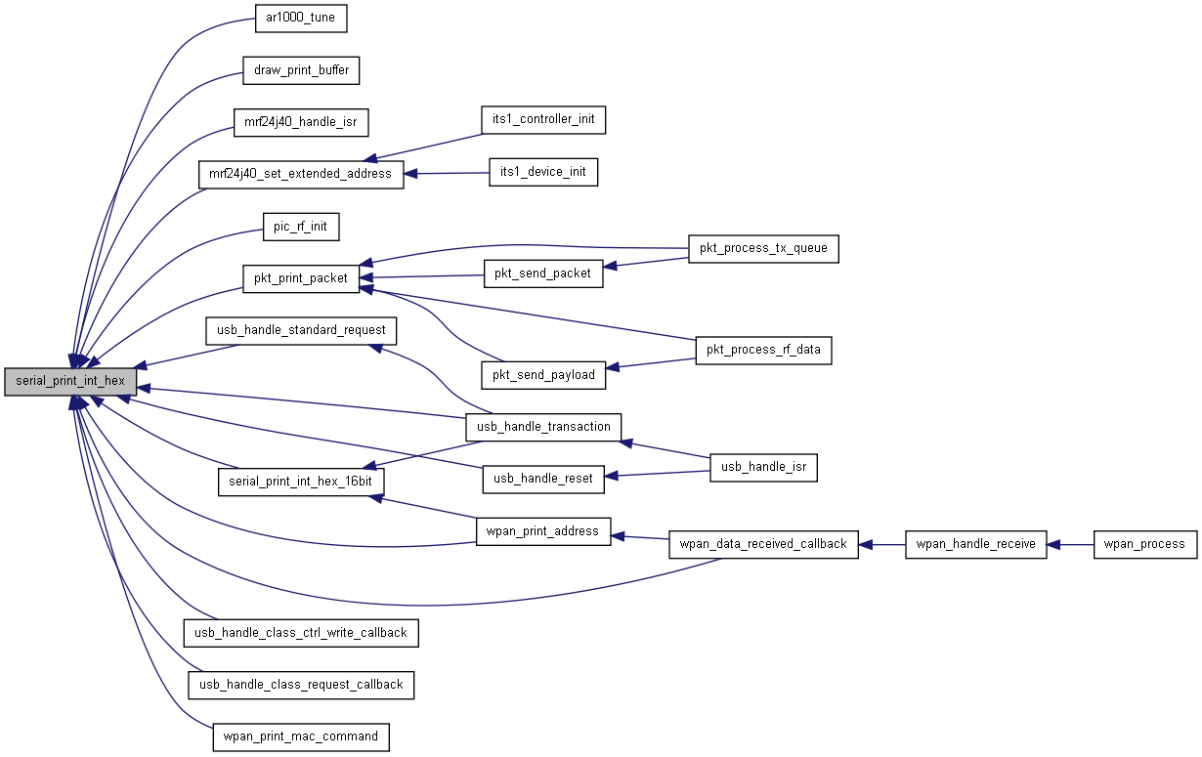
References bin2Hex(), and serial_putc().

Referenced by ar1000_tune(), draw_print_buffer(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), pic_rf_init(), pkt_print_packet(), serial_print_int_hex_16bit(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_standard_request(), usb_handle_transaction(), wpan_data_received_callback(), wpan_print_address(), and wpan_print_mac_command().

Here is the call graph for this function:




Here is the caller graph for this function:




**void serial_print_int_hex_16bit (uns16 *i*)**

Print a 16 bit unsigned number in hex to the serial port

**Parameters:**

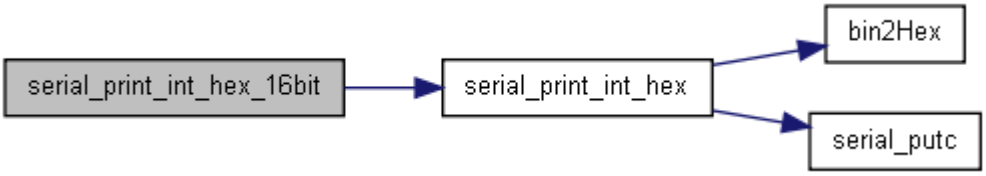
*i* 16 bit number to be printed

Definition at line 350 of file pic_serial.c.

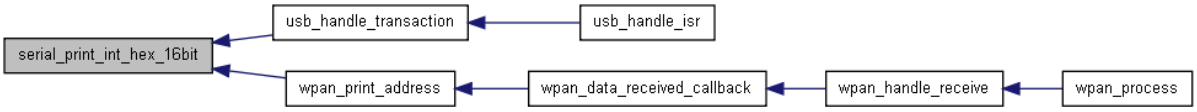
References serial_print_int_hex().

Referenced by usb_handle_transaction(), and wpan_print_address().

Here is the call graph for this function:




Here is the caller graph for this function:




### void serial_print_nl ()

Print a new line out the serial port - if you do this often, this routine can be used to save a couple of instructions. Always helps!

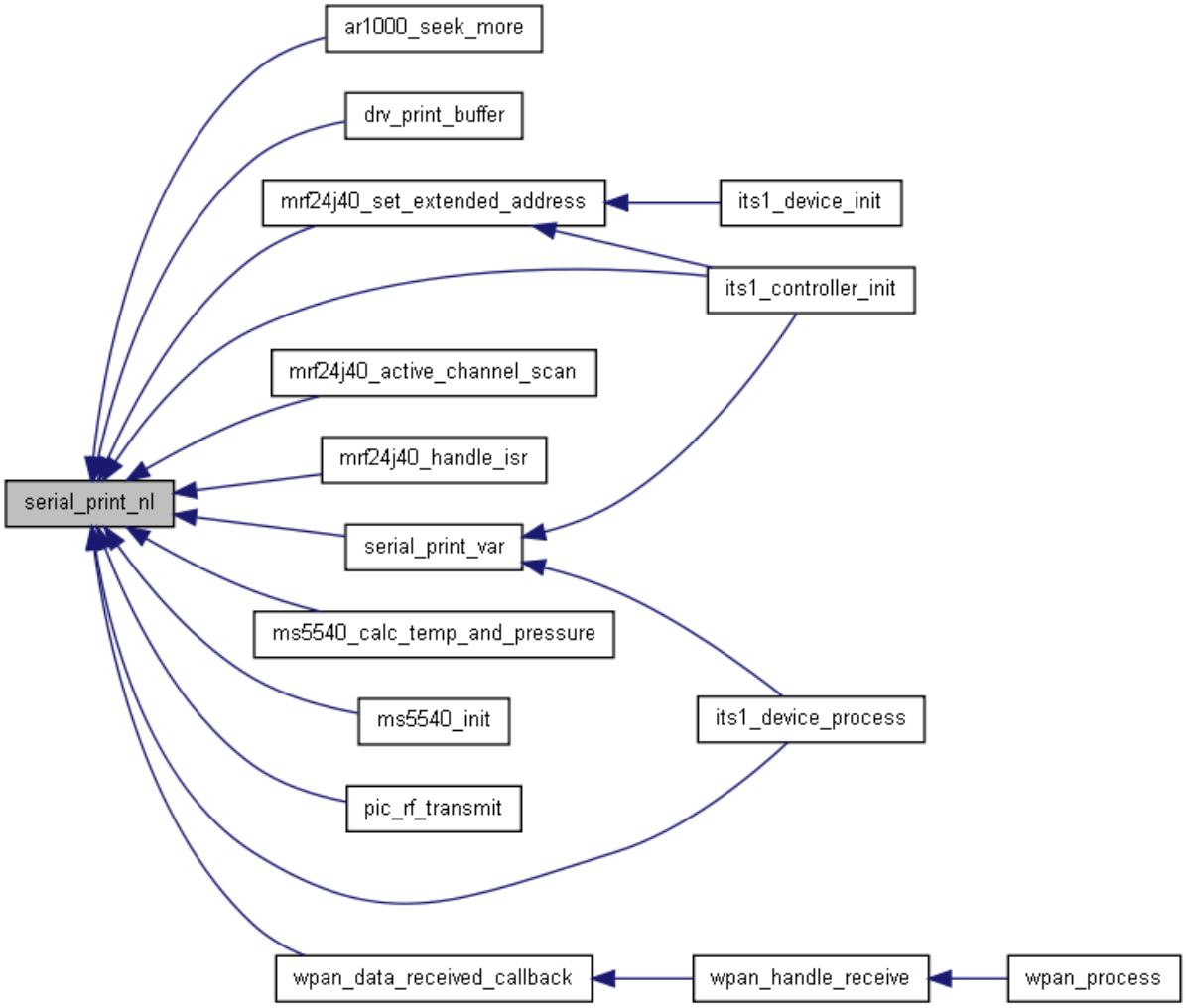
Definition at line 361 of file pic_serial.c.

References serial_putc().

Referenced by ar1000_seek_more(), drv_print_buffer(), its1_controller_init(), its1_device_process(), mrf24j40_active_channel_scan(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_transmit(), serial_print_var(), and wpan_data_received_callback().

Here is the call graph for this function:


Here is the caller graph for this function:




**void serial_print_spc ()**

Print a space out the serial port - if you do this often, this routine can be used to save a couple of instructions. Always helps!

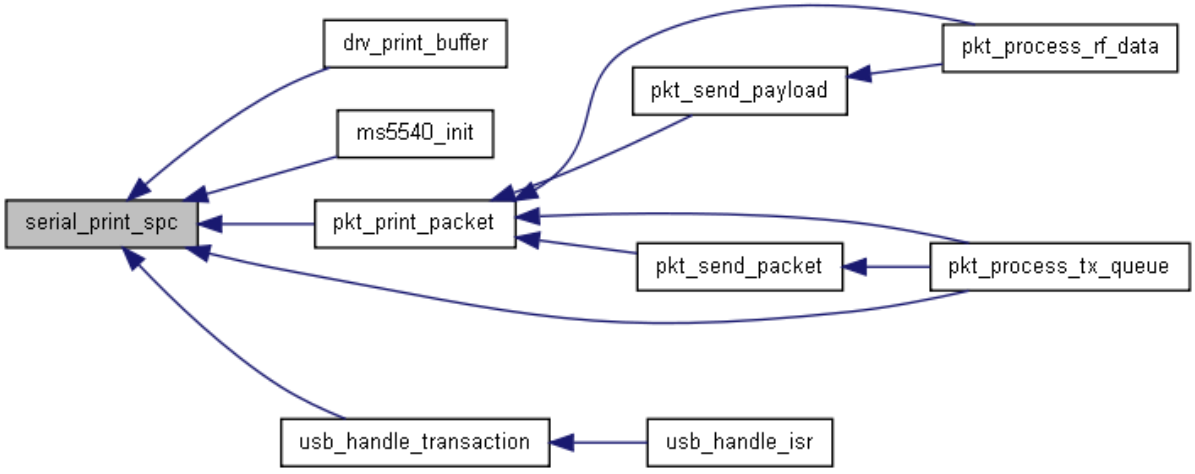
Definition at line 356 of file pic_serial.c.

References serial_putc().

Referenced by drv_print_buffer(), ms5540_init(), pkt_print_packet(), pkt_process_tx_queue(), and usb_handle_transaction().

Here is the call graph for this function:


Here is the caller graph for this function:




**void serial_print_str (rom char * *str*)**

Send a null terminated rom string out the serial port

**Parameters:**

*str* the rom string to be sent

Definition at line 310 of file pic_serial.c.

References serial_putc(), and uns8.

Here is the call graph for this function:


**void serial_print_str (char * *str*)**

Send a null terminated string out the serial port

**Parameters:**

*str* the string to be sent

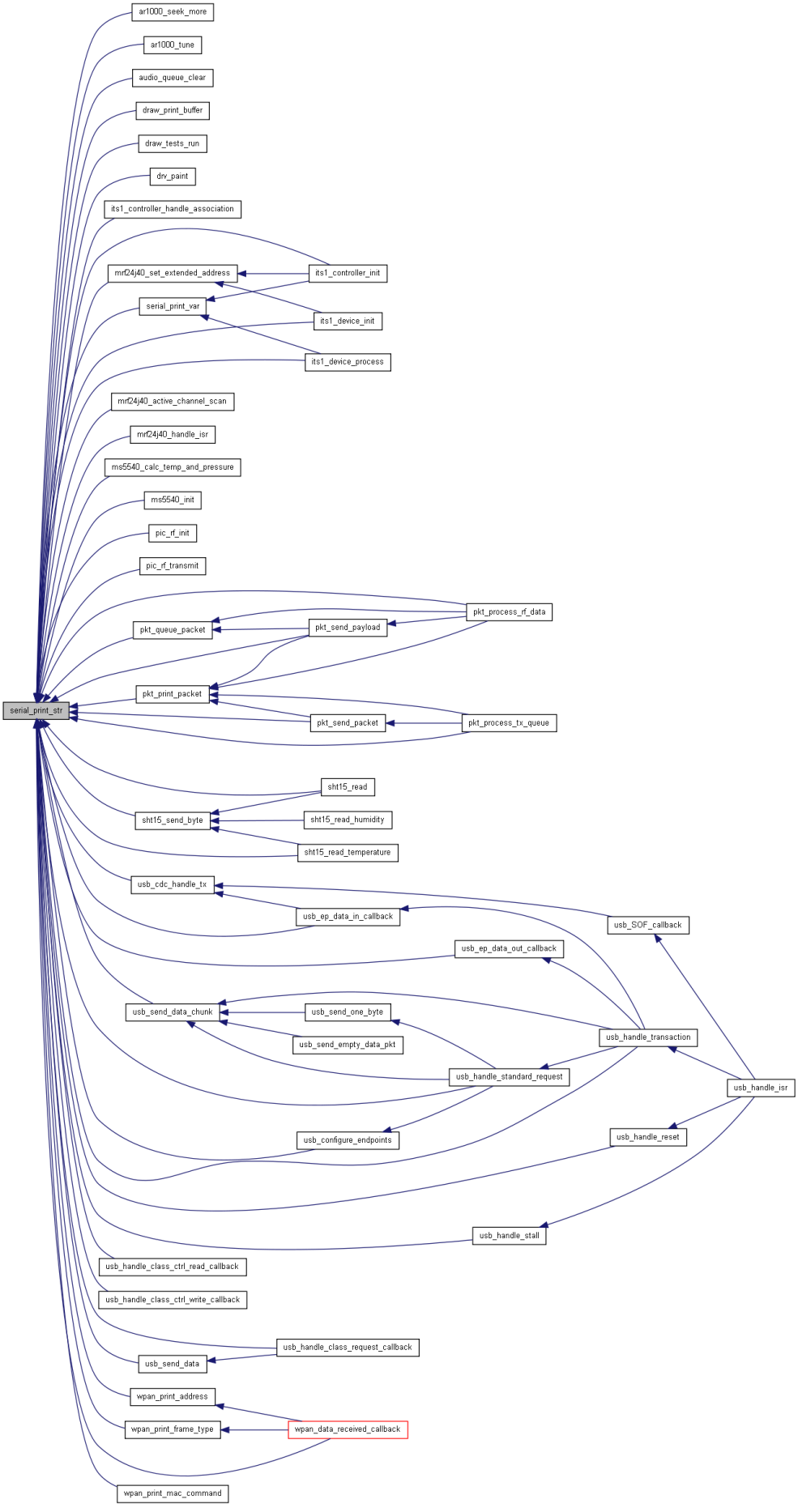
Definition at line 298 of file pic_serial.c.

References serial_putc(), and uns8.

Referenced by ar1000_seek_more(), ar1000_tune(), audio_queue_clear(), draw_print_buffer(), draw_tests_run(), drv_paint(), its1_controller_handle_association(), its1_controller_init(), its1_device_init(), its1_device_process(), mrf24j40_active_channel_scan(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_init(), pic_rf_transmit(), pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), pkt_queue_packet(), pkt_send_packet(), pkt_send_payload(), serial_print_var(), sht15_read(), sht15_read_temperature(), sht15_send_byte(), usb_cdc_handle_tx(), usb_configure_endpoints(), usb_ep_data_in_callback(), usb_ep_data_out_callback(), usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_stall(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data(), usb_send_data_chunk(), wpan_data_received_callback(), wpan_print_address(), wpan_print_frame_type(), and wpan_print_mac_command().

Here is the call graph for this function:

Here is the caller graph for this function:


ar1000_seek_more
ar1000_tune
audio_queue_clear
draw_print_buffer
draw_tests_run
drv_paint
its1_controller_handle_association
mrf24j40_set_extended_address
its1_controller_init
serial_print_var
its1_device_init
its1_device_process
mrf24j40_active_channel_scan
mrf24j40_handle_isr
ms5540_calc_temp_and_pressure
ms5540_init
pic_rf_init
pic_rf_transmit
pkt_process_rf_data
pkt_queue_packet
pkt_send_payload
pkt_print_packet
serial_print_str
pkt_send_packet
pkt_process_tx_queue
sht15_read
sht15_send_byte
sht15_read_humidity
sht15_read_temperature
usb_cdc_handle_tx
usb_ep_data_in_callback
usb_SOF_callback
usb_ep_data_out_callback
usb_send_data_chunk
usb_send_one_byte
usb_handle_transaction
usb_send_empty_data_pkt
usb_handle_standard_request
usb_handle_isr
usb_configure_endpoints
usb_handle_reset
usb_handle_stall
usb_handle_class_ctrl_read_callback
usb_handle_class_ctrl_write_callback
usb_handle_class_request_callback
usb_send_data
wpan_print_address
wpan_print_frame_type
wpan_data_received_callback
wpan_print_mac_command

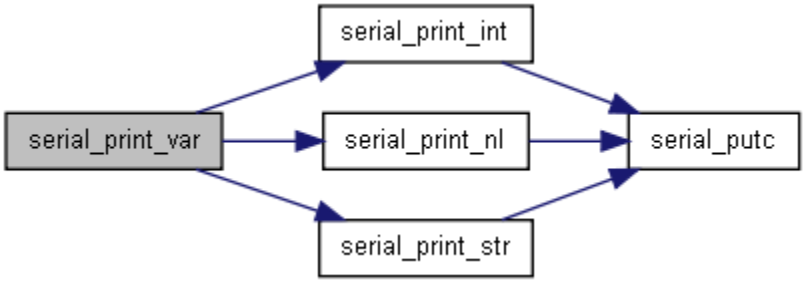
### void serial_print_var (char * *str*, uns16 *i*)

Definition at line 365 of file pic_serial.c.

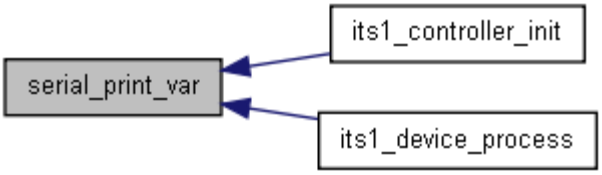
References serial_print_int(), serial_print_nl(), and serial_print_str().

Referenced by its1_controller_init(), and its1_device_process().

Here is the call graph for this function:




Here is the caller graph for this function:




### void serial_putc (uns8 *c*)

Sends a single character out the serial connection. It is sent straight out if possible, otherwise put into the fifo. Note that if you fill the fifo while interrupts are off (eg, in an interrupt routine or a critical section) then this routine will hang the pic, since it's waiting for an interrupt to clear the fifo, which never comes... The moral is to keep your fifo big enough or don't send too much while interrupts are off (eg, in an interrupt response routine). Of course, you *can* send things in an ISR - just don't fill the fifo up.

**Parameters:**

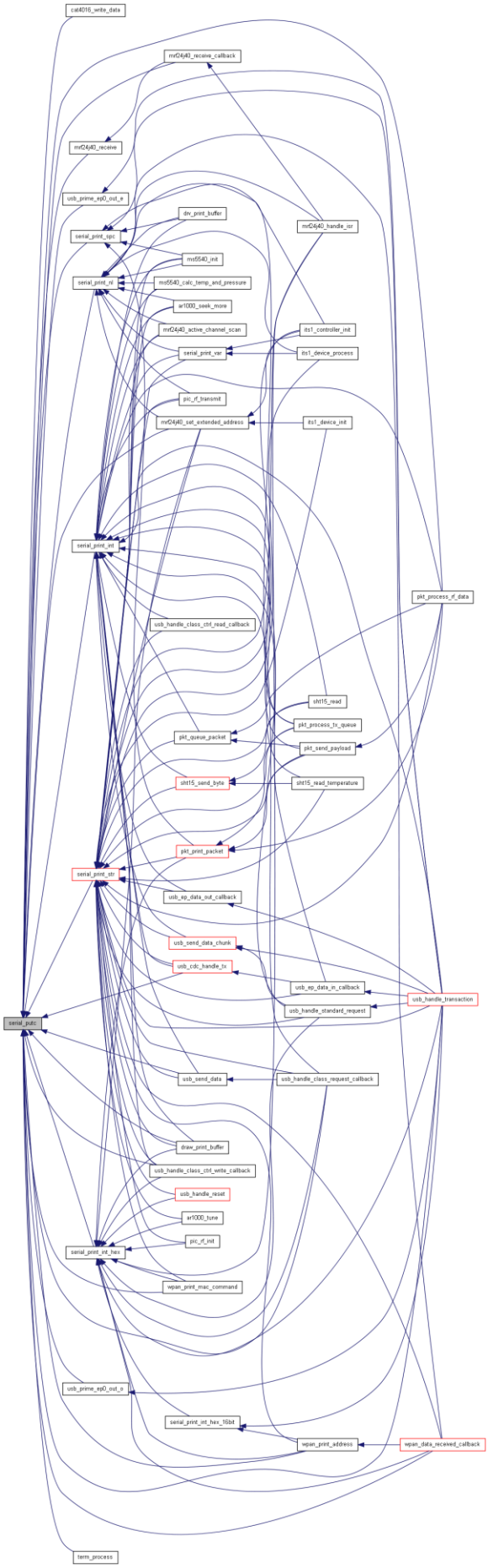
*c* the character to transmit

Definition at line 172 of file pic_serial.c.

References kill_interrupts, serial_handle_tx_isr, tx_buffer, tx_end, tx_start, and uns8.

Referenced by cat4016_write_data(), draw_print_buffer(), mrf24j40_receive(), mrf24j40_receive_callback(), mrf24j40_set_extended_address(), pkt_process_rf_data(), serial_print_int(), serial_print_int_hex(), serial_print_nl(), serial_print_spc(), serial_print_str(), term_process(), usb_cdc_handle_tx(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), wpan_data_received_callback(), wpan_print_address(), and wpan_print_mac_command().

Here is the caller graph for this function:


cat4016_write_data
mrf24j40_receive_callback
mrf24j40_receive
usb_prime_ep0_out_s
dtv_print_buffer
serial_print_spc
mrf24j40_handle_isr
ms5540_init
serial_print_nl
ms5540_calc_temp_and_pressure
ar1000_seek_more
mrf24j40_active_channel_scan
its1_controller_init
serial_print_var
its1_device_process
pic_rf_transmit
mrf24j40_set_extended_address
its1_device_init
serial_print_int
pkt_process_rf_data
usb_handle_class_ctrl_read_callback
sht15_read
pkt_process_tx_queue
pkt_queue_packet
pkt_send_payload
sht15_send_byte
sht15_read_temperature
pkt_print_packet
serial_print_str
usb_ep_data_out_callback
usb_send_data_chunk
usb_cdc_handle_tx
usb_ep_data_in_callback
usb_handle_transaction
usb_handle_standard_request
serial_putc
usb_send_data
usb_handle_class_request_callback
draw_print_buffer
usb_handle_class_ctrl_write_callback
usb_handle_reset
ar1000_tune
pic_rf_init
serial_print_int_hex
wpan_print_mac_command
usb_prime_ep0_out_o
serial_print_int_hex_16bit
wpan_print_address
wpan_data_received_callback
term_process

### uns8 serial_rx_avail ()

Tests to see if the serial receive fifo has a character available. Useful to call before getc() if interrupts are not enabled in that section of code.

**Returns:**

true (non zero) if there are one or more characters waiting in the fifo queue, false (zero) otherwise

Definition at line 371 of file pic_serial.c.

References rx_end, and rx_start.

Referenced by term_process().

Here is the caller graph for this function:


### void serial_rx_isr ()

This routine needs to be called from your interrupt() routine when the receive hardware interrupt occurs in order to put received bytes into the fifo buffer.

Definition at line 236 of file pic_serial.c.

References rx_buffer, rx_end, rx_start, and uns8.

### void serial_setup (uns8 *req_spbrg*)

Configures the pic and gets ready for interrupt-driven serial communications. Includes setting the tris bits appropriately, and getting the baud rate generator set up. After calling this you can immediately start sending and receiving bytes.

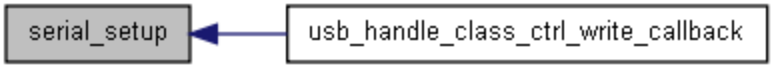
**Parameters:**

*brgh* See sprg defines earlier in pic_serial.h

Definition at line 77 of file pic_serial.c.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:




### uns8 serial_tx_empty ()

Tests to see if the serial transmit fifo is empty.

**Returns:**

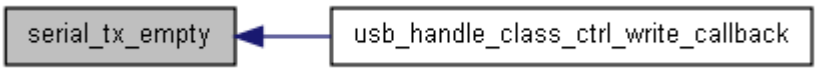
true (non zero) if tx fifo is empty, false (zero) otherwise

Definition at line 372 of file pic_serial.c.

References tx_end, and tx_start.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:



**void serial_tx_isr ()**

serial_load_tx

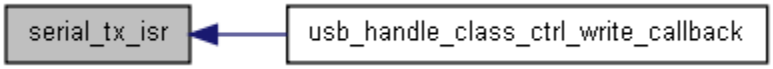
This routine needs to be called from your interrupt() routine when the transmit hardware interrupt occurs in order to send bytes that are waiting in the fifo buffer.

Definition at line 216 of file pic_serial.c.

References tx_buffer, tx_end, tx_start, and uns8.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:




---

## Variable Documentation

**uns8 rx_buffer[SERIAL_RX_BUFFER_SIZE]**

Receive fifo

Definition at line 51 of file pic_serial.c.

Referenced by serial_getc(), and serial_rx_isr().

**uns8 rx_end = 0**

Receive fifo end point

Definition at line 55 of file pic_serial.c.

Referenced by serial_getc(), serial_rx_avail(), and serial_rx_isr().

**uns8 rx_start = 0**

Receive fifo start point

Definition at line 53 of file pic_serial.c.

Referenced by serial_getc(), serial_rx_avail(), and serial_rx_isr().

**uns8 tx_buffer[SERIAL_TX_BUFFER_SIZE]**

Transmit fifo

Definition at line 44 of file pic_serial.c.

Referenced by pkt_send_packet(), serial_putc(), and serial_tx_isr().

**uns8 tx_end = 0**

Transmit fifo end point

Definition at line 48 of file pic_serial.c.

Referenced by serial_putc(), serial_tx_empty(), and serial_tx_isr().

**uns8 tx_start = 0**

Transmit fifo start point

Definition at line 46 of file pic_serial.c.

Referenced by serial_putc(), serial_tx_empty(), and serial_tx_isr().

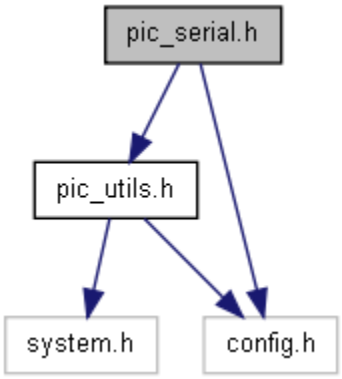
---

# pic_serial.h File Reference

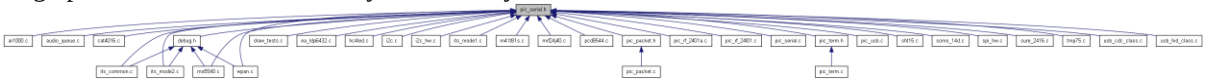
Interrupt driven fifo serial support.

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

Include dependency graph for pic_serial.h:




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



## Defines

- #define BRGH_HIGH_SPEED 1
- #define BRGH_LOW_SPEED 0
- #define serial_handle_rx_isr() if (pir1.RCIF) { serial_rx_isr(); clear_bit( pir1, RCIF ); }
- #define serial_handle_tx_isr() if (pir1.TXIF) { serial_tx_isr(); }
- #define serial_print_debug(string, variable) serial_print_str(string); serial_print_int(variable); serial_print_nl();

## Functions

- uns8 serial_getc (void)
- *Retrieve a character from the serial port.* void serial_print_int (uns16 i)
- *Print a 16 bit number to the serial port.* void serial_print_int_hex (uns8 i)
- *Print an 8 bit number in hex to the serial port.* void serial_print_int_hex_16bit (uns16 i)
- *Print a 16 bit number in hex to the serial port.* void serial_print_nl ()
- *Print a newline.* void serial_print_spc ()
- *Print a space.* void serial_print_str (rom char *str)
- *Print a rom string out to the serial port.* void serial_print_str (char *str)

- *Print a string out to the serial port.* void serial_print_var (char *str, uns16 i)
- void serial_putc (uns8 c)
- *Transmit a single character.* uns8 serial_rx_avail ()
- *Tests if the serial rx fifo has a character available.* void serial_rx_isr ()
- *Serial receive interrupt service routine.* void serial_setup (uns8 req_spbrg)
- *Configure the pic for serial communicaitons.* uns8 serial_tx_empty ()
- *Tests if the serial tx fifo is empty.* void serial_tx_full ()
- *Tests if the serial tx fifo is full.* void serial_tx_isr ()

*Serial transmit interrupt service routine.*

---

## Detailed Description

Put the following into your config.h

- - - - - - - - - - - - - - - - - - - pic_serial defines
- - - - - - - - - - - - - - - - - - -

define SERIAL_TX_BUFFER_SIZE 20 define SERIAL_RX_BUFFER_SIZE 4

Use this define if you want fine-grained control of what happens in the serial port define SERIAL_DEBUG_ON

Use this define if you are debugging in the IDE simulator and don't want it to hang waiting for serial interrupts that will never come... define SERIAL_IDE_DEBUG

Use thie define if you want to drop a character if the TX buffer is full, rather than the default behaviour, which is to wait until the TX buffer has a spare spot. define SERIAL_DISCARD_ON_TX_FULL_DURING_INT

- - - - - - - - - - - - - - - - - - -

Put the following in your ISR

serial_handle_tx_isr(); serial_handle_rx_isr();

Put the following in your system setup routine

serial_setup(uns8/16 req_spbrg);

Definition in file pic_serial.h.

---

## Define Documentation

### #define BRGH_HIGH_SPEED 1

Definition at line 338 of file pic_serial.h.

### #define BRGH_LOW_SPEED 0

Definition at line 339 of file pic_serial.h.

### #define serial_handle_rx_isr() if (pir1.RCIF) { serial_rx_isr(); clear_bit( pir1, RCIF ); }

include in your ISR

Definition at line 90 of file pic_serial.h.

**#define serial_handle_tx_isr()  if (pir1.TXIF) { serial_tx_isr(); }**

include in your ISR

Definition at line 87 of file pic_serial.h.

Referenced by serial_putc().

**#define serial_print_debug(string,  variable)  serial_print_str(string); serial_print_int(variable); serial_print_nl();**

Definition at line 93 of file pic_serial.h.

---

## Function Documentation

### uns8 serial_getc (void)

Retrieve character from the serial port. Note that if there is nothing in the fifo, this function will wait until a character is received - and this will never happen if interrupts are turned off when this is called! So, be careful not to call getc during a critical section or during an ISR unless* you're sure there's something in the fifo. You can do this by calling the serial_rx_avail() routine. In any other situation, you can call getc() and happily wait for a character to arrive.

Definition at line 276 of file pic_serial.c.

References end_crit_sec, rx_buffer, rx_end, rx_start, start_crit_sec, and uns8.

Referenced by term_process().

Here is the caller graph for this function:


### void serial_print_int (uns16 *i*)

Print a 16 bit unsigned number in decimal to the serial port

**Parameters:**

*i* the 16 bit number to be printed

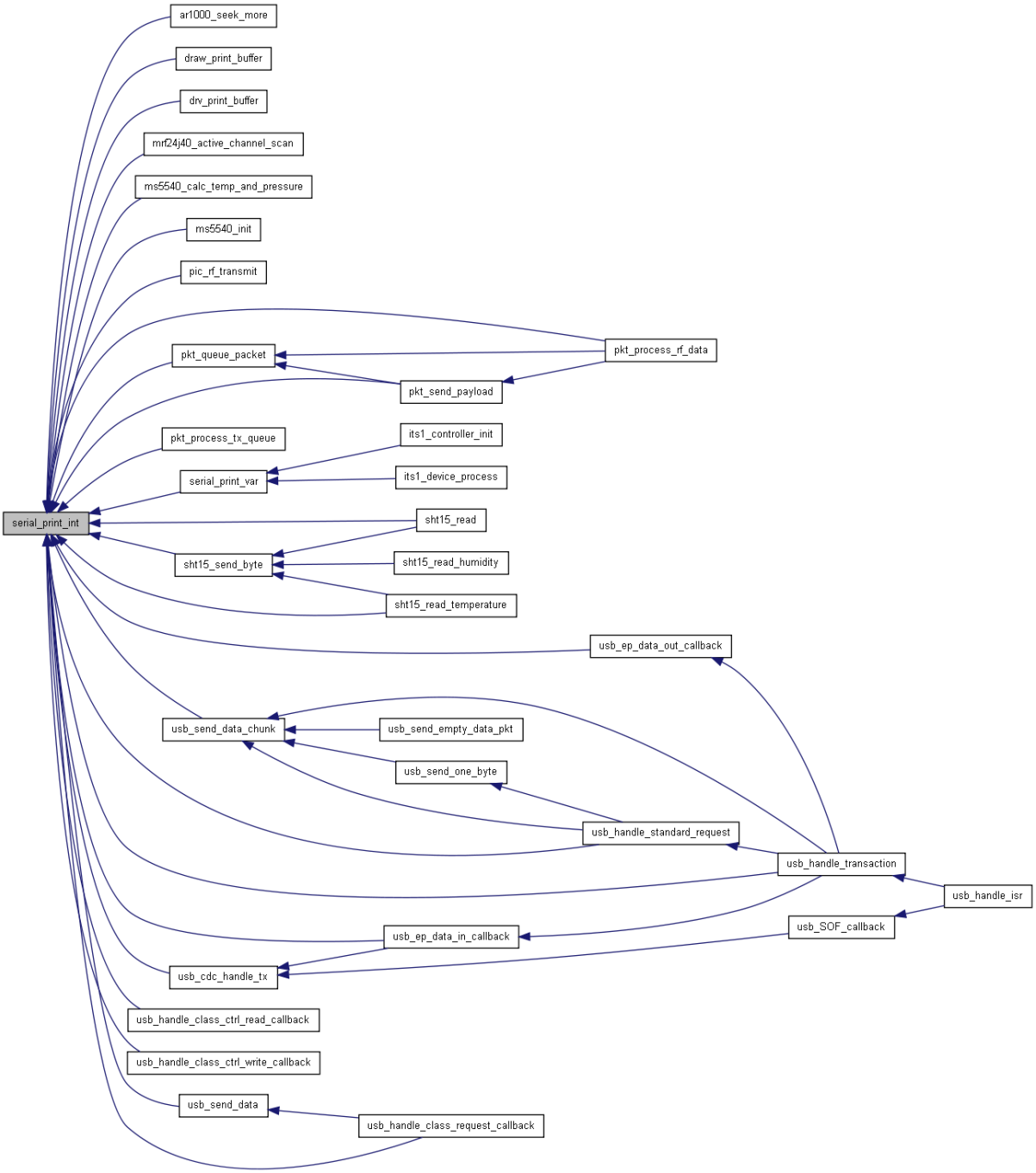
Definition at line 322 of file pic_serial.c.

References serial_putc(), and uns8.

Referenced by ar1000_seek_more(), draw_print_buffer(), drv_print_buffer(), mrf24j40_active_channel_scan(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_transmit(), pkt_process_rf_data(), pkt_process_tx_queue(), pkt_queue_packet(), pkt_send_payload(), serial_print_var(), sht15_read(), sht15_read_temperature(), sht15_send_byte(), usb_cdc_handle_tx(), usb_ep_data_in_callback(), usb_ep_data_out_callback(), usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data(), and usb_send_data_chunk().

Here is the call graph for this function:

Here is the caller graph for this function:




### void serial_print_int_hex (uns8 *i*)

Print a 8 bit unsigned number in hex to the serial port

**Parameters:**

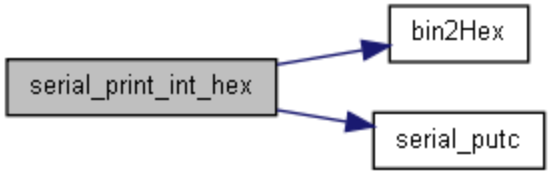
*i* 8 bit number to be printed

Definition at line 343 of file pic_serial.c.

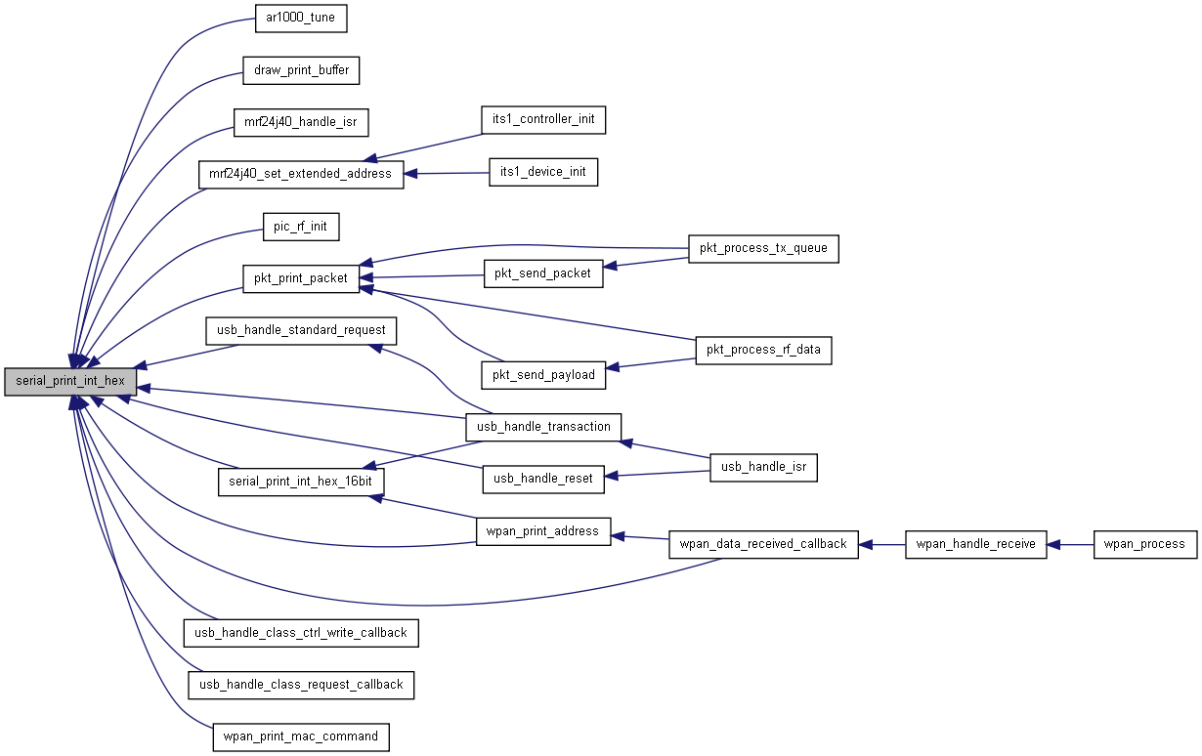
References bin2Hex(), and serial_putc().

Referenced by ar1000_tune(), draw_print_buffer(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), pic_rf_init(), pkt_print_packet(), serial_print_int_hex_16bit(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_standard_request(), usb_handle_transaction(), wpan_data_received_callback(), wpan_print_address(), and wpan_print_mac_command().

Here is the call graph for this function:




Here is the caller graph for this function:




### void serial_print_int_hex_16bit (uns16 *i*)

Print a 16 bit unsigned number in hex to the serial port

**Parameters:**

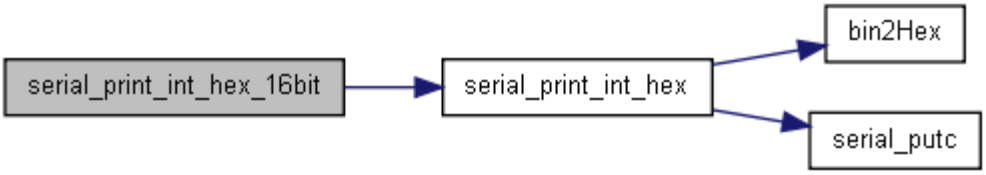
*i* 16 bit number to be printed

Definition at line 350 of file pic_serial.c.

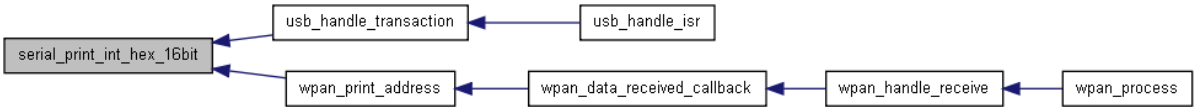
References serial_print_int_hex().

Referenced by usb_handle_transaction(), and wpan_print_address().

Here is the call graph for this function:




Here is the caller graph for this function:




**void serial_print_nl ()**

Print a new line out the serial port - if you do this often, this routine can be used to save a couple of instructions. Always helps!

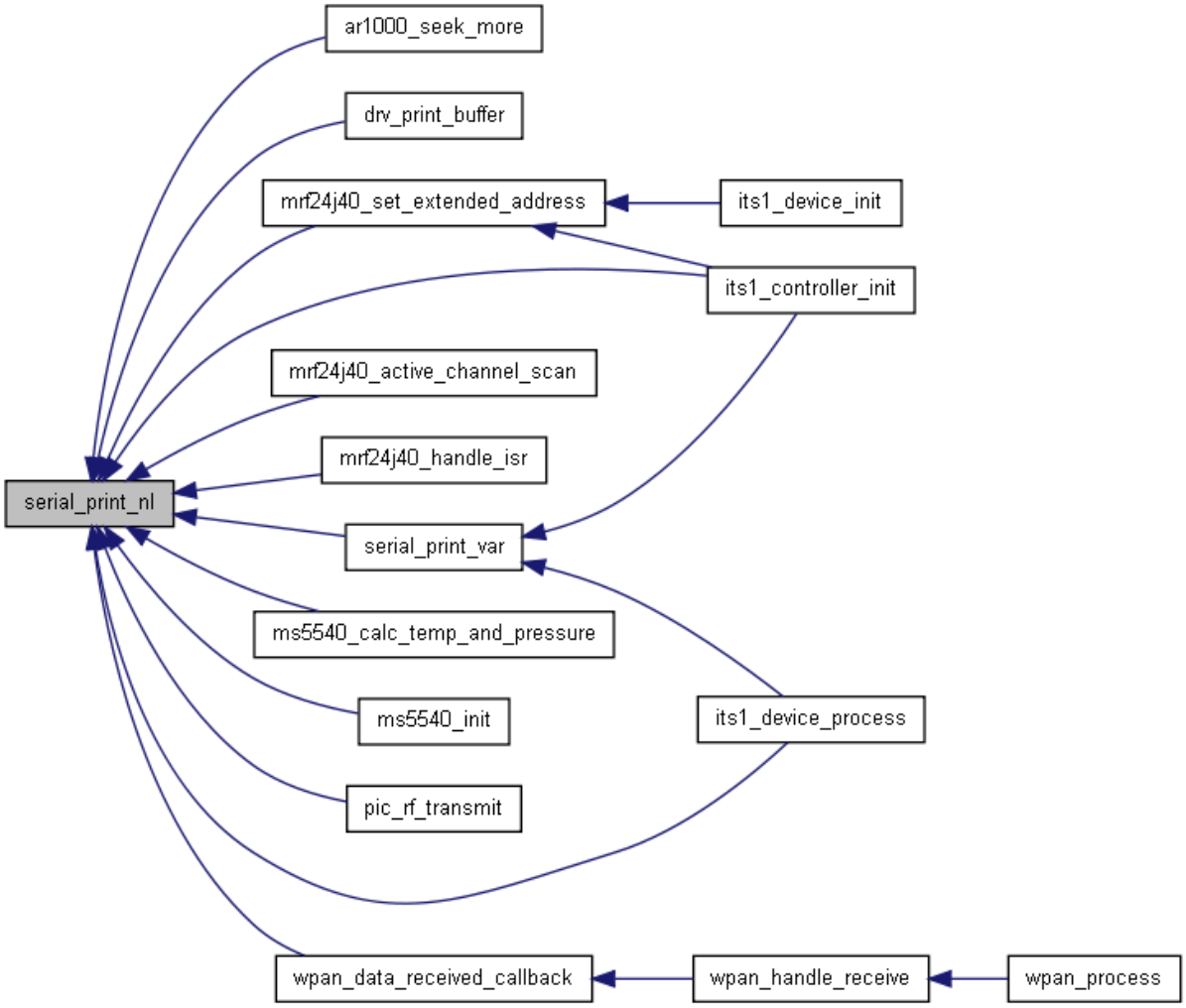
Definition at line 361 of file pic_serial.c.

References serial_putc().

Referenced by ar1000_seek_more(), drv_print_buffer(), its1_controller_init(), its1_device_process(), mrf24j40_active_channel_scan(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_transmit(), serial_print_var(), and wpan_data_received_callback().

Here is the call graph for this function:


Here is the caller graph for this function:




**void serial_print_spc ()**

Print a space out the serial port - if you do this often, this routine can be used to save a couple of instructions. Always helps!

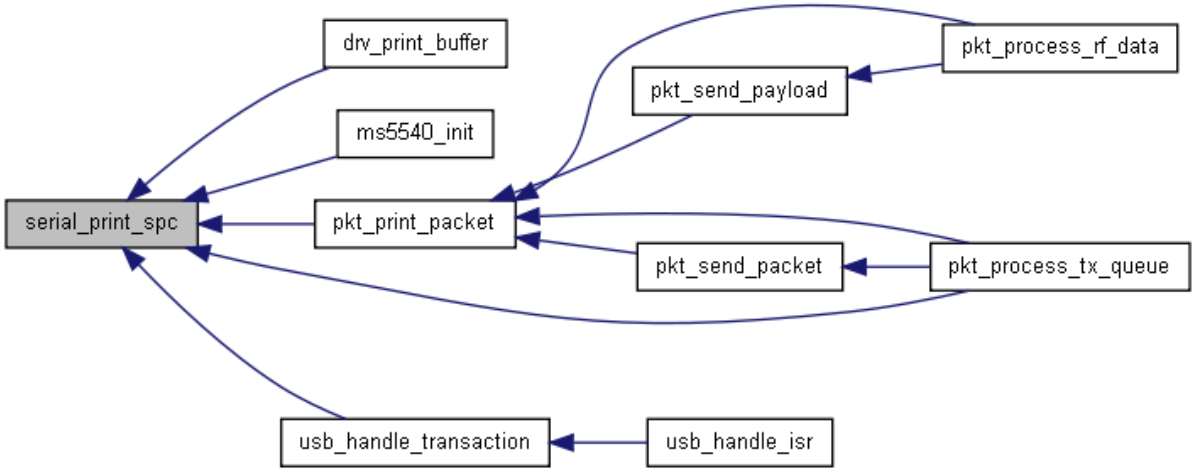
Definition at line 356 of file pic_serial.c.

References serial_putc().

Referenced by drv_print_buffer(), ms5540_init(), pkt_print_packet(), pkt_process_tx_queue(), and usb_handle_transaction().

Here is the call graph for this function:


Here is the caller graph for this function:




**void serial_print_str (rom char * *str*)**

Send a null terminated rom string out the serial port

**Parameters:**

*str* the rom string to be sent

Definition at line 310 of file pic_serial.c.

References serial_putc(), and uns8.

Here is the call graph for this function:


**void serial_print_str (char * *str*)**

Send a null terminated string out the serial port

**Parameters:**

*str* the string to be sent

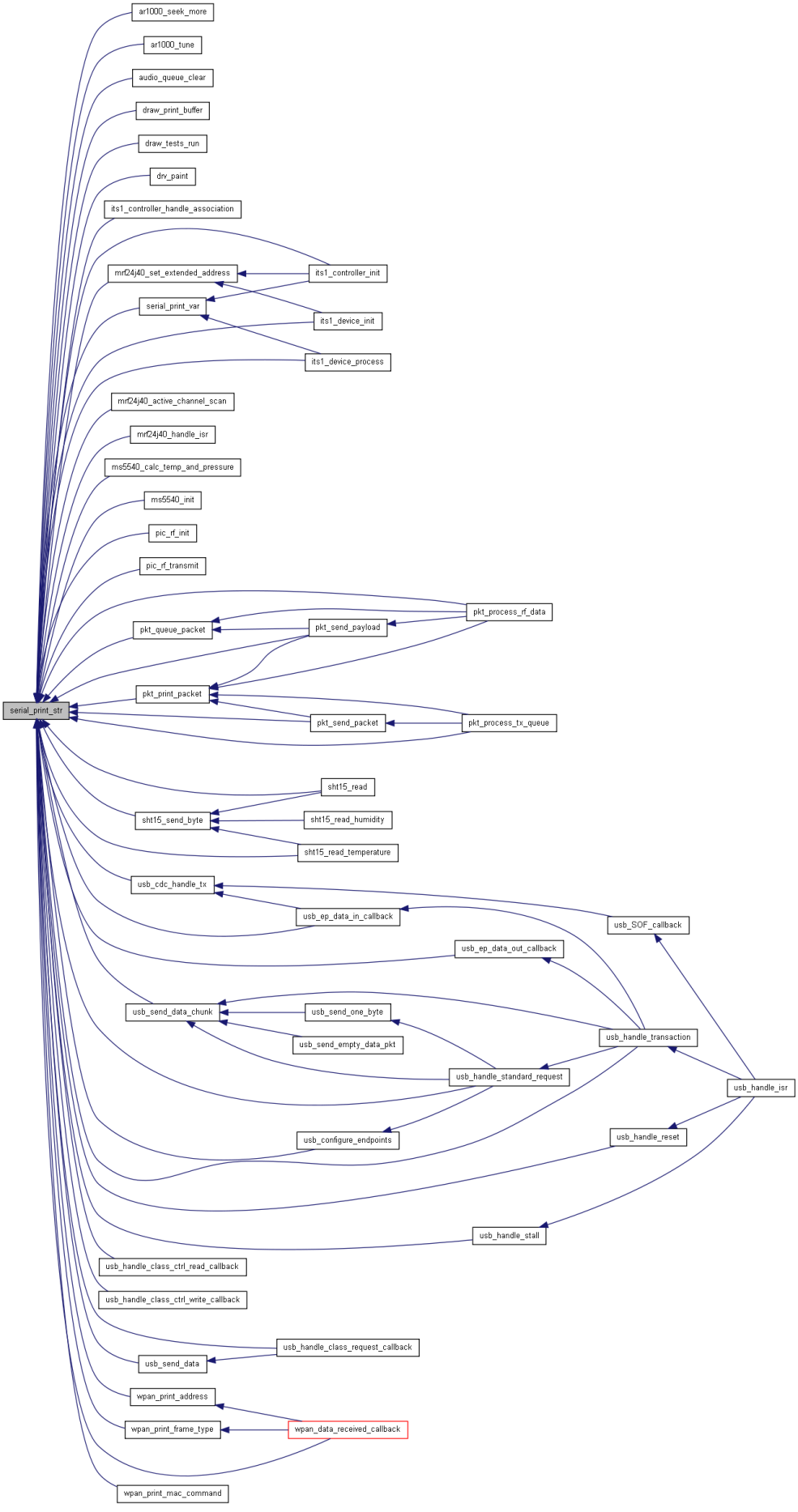
Definition at line 298 of file pic_serial.c.

References serial_putc(), and uns8.

Referenced by ar1000_seek_more(), ar1000_tune(), audio_queue_clear(), draw_print_buffer(), draw_tests_run(), drv_paint(), its1_controller_handle_association(), its1_controller_init(), its1_device_init(), its1_device_process(), mrf24j40_active_channel_scan(), mrf24j40_handle_isr(), mrf24j40_set_extended_address(), ms5540_calc_temp_and_pressure(), ms5540_init(), pic_rf_init(), pic_rf_transmit(), pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), pkt_queue_packet(), pkt_send_packet(), pkt_send_payload(), serial_print_var(), sht15_read(), sht15_read_temperature(), sht15_send_byte(), usb_cdc_handle_tx(), usb_configure_endpoints(), usb_ep_data_in_callback(), usb_ep_data_out_callback(), usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_stall(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data(), usb_send_data_chunk(), wpan_data_received_callback(), wpan_print_address(), wpan_print_frame_type(), and wpan_print_mac_command().

Here is the call graph for this function:

Here is the caller graph for this function:


ar1000_seek_more
ar1000_tune
audio_queue_clear
draw_print_buffer
draw_tests_run
drv_paint
its1_controller_handle_association
mrf24j40_set_extended_address
its1_controller_init
serial_print_var
its1_device_init
its1_device_process
mrf24j40_active_channel_scan
mrf24j40_handle_isr
ms5540_calc_temp_and_pressure
ms5540_init
pic_rf_init
pic_rf_transmit
pkt_process_rf_data
pkt_queue_packet
pkt_send_payload
pkt_print_packet
serial_print_str
pkt_send_packet
pkt_process_tx_queue
sht15_read
sht15_send_byte
sht15_read_humidity
sht15_read_temperature
usb_cdc_handle_tx
usb_ep_data_in_callback
usb_SOF_callback
usb_ep_data_out_callback
usb_send_data_chunk
usb_send_one_byte
usb_handle_transaction
usb_send_empty_data_pkt
usb_handle_standard_request
usb_handle_isr
usb_configure_endpoints
usb_handle_reset
usb_handle_stall
usb_handle_class_ctrl_read_callback
usb_handle_class_ctrl_write_callback
usb_handle_class_request_callback
usb_send_data
wpan_print_address
wpan_print_frame_type
wpan_data_received_callback
wpan_print_mac_command

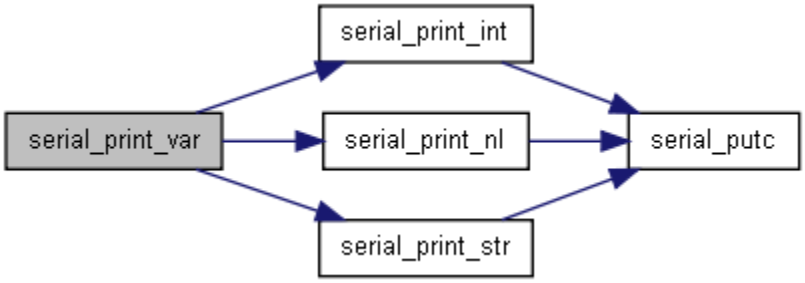
### void serial_print_var (char * *str*, uns16 *i*)

Definition at line 365 of file pic_serial.c.

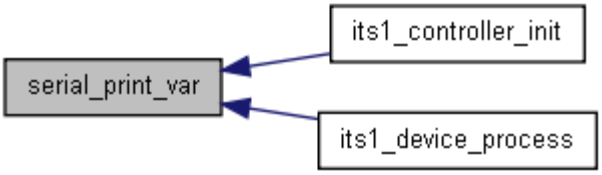
References serial_print_int(), serial_print_nl(), and serial_print_str().

Referenced by its1_controller_init(), and its1_device_process().

Here is the call graph for this function:




Here is the caller graph for this function:




### void serial_putc (uns8 *c*)

Sends a single character out the serial connection. It is sent straight out if possible, otherwise put into the fifo. Note that if you fill the fifo while interrupts are off (eg, in an interrupt routine or a critical section) then this routine will hang the pic, since it's waiting for an interrupt to clear the fifo, which never comes... The moral is to keep your fifo big enough or don't send too much while interrupts are off (eg, in an interrupt response routine). Of course, you *can* send things in an ISR - just don't fill the fifo up.

**Parameters:**

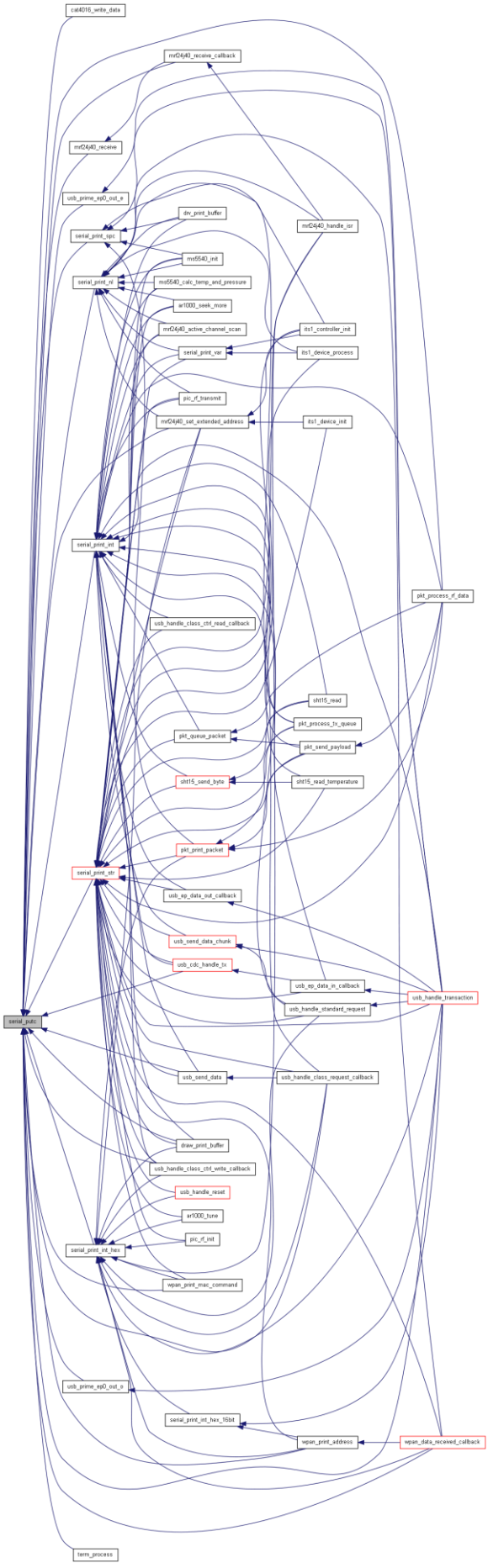
*c* the character to transmit

Definition at line 172 of file pic_serial.c.

References kill_interrupts, serial_handle_tx_isr, tx_buffer, tx_end, tx_start, and uns8.

Referenced by cat4016_write_data(), draw_print_buffer(), mrf24j40_receive(), mrf24j40_receive_callback(), mrf24j40_set_extended_address(), pkt_process_rf_data(), serial_print_int(), serial_print_int_hex(), serial_print_nl(), serial_print_spc(), serial_print_str(), term_process(), usb_cdc_handle_tx(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), wpan_data_received_callback(), wpan_print_address(), and wpan_print_mac_command().

Here is the caller graph for this function:


cat4016_write_data
mrf24j40_receive_callback
mrf24j40_receive
usb_prime_ep0_out_s
dtv_print_buffer
serial_print_spc
mrf24j40_handle_isr
ms5540_init
serial_print_nl
ms5540_calc_temp_and_pressure
ar1000_seek_more
mrf24j40_active_channel_scan
its1_controller_init
serial_print_var
its1_device_process
pic_rf_transmit
mrf24j40_set_extended_address
its1_device_init
serial_print_int
pkt_process_rf_data
usb_handle_class_ctrl_read_callback
sht15_read
pkt_process_tx_queue
pkt_queue_packet
pkt_send_payload
sht15_send_byte
sht15_read_temperature
pkt_print_packet
serial_print_str
usb_ep_data_out_callback
usb_send_data_chunk
usb_cdc_handle_tx
usb_ep_data_in_callback
usb_handle_transaction
usb_handle_standard_request
serial_putc
usb_send_data
usb_handle_class_request_callback
draw_print_buffer
usb_handle_class_ctrl_write_callback
usb_handle_reset
ar1000_tune
pic_rf_init
serial_print_int_hex
wpan_print_mac_command
usb_prime_ep0_out_o
serial_print_int_hex_16bit
wpan_print_address
wpan_data_received_callback
term_process

### uns8 serial_rx_avail ()

Tests to see if the serial receive fifo has a character available. Useful to call before getc() if interrupts are not enabled in that section of code.

**Returns:**

true (non zero) if there are one or more characters waiting in the fifo queue, false (zero) otherwise

Definition at line 371 of file pic_serial.c.

References rx_end, and rx_start.

Referenced by term_process().

Here is the caller graph for this function:


### void serial_rx_isr ()

This routine needs to be called from your interrupt() routine when the receive hardware interrupt occurs in order to put received bytes into the fifo buffer.

Definition at line 236 of file pic_serial.c.

References rx_buffer, rx_end, rx_start, and uns8.

### void serial_setup (uns8 *req_spbrg*)

Configures the pic and gets ready for interrupt-driven serial communications. Includes setting the tris bits appropriately, and getting the baud rate generator set up. After calling this you can immediately start sending and receiving bytes.

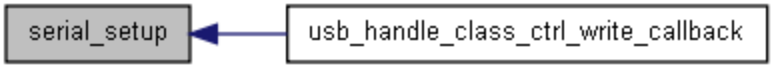
**Parameters:**

*brgh* See sprg defines earlier in pic_serial.h

Definition at line 77 of file pic_serial.c.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:




### uns8 serial_tx_empty ()

Tests to see if the serial transmit fifo is empty.

**Returns:**

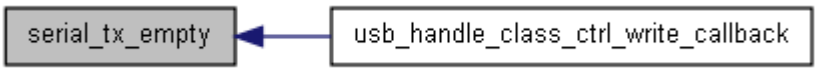
true (non zero) if tx fifo is empty, false (zero) otherwise

Definition at line 372 of file pic_serial.c.

References tx_end, and tx_start.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:



### void serial_tx_full ()

Tests to see if the serial transmit fifo is full.

**Returns:**

true (non zero) if tx fifo is full, false (zero) otherwise

### void serial_tx_isr ()

serial_load_tx

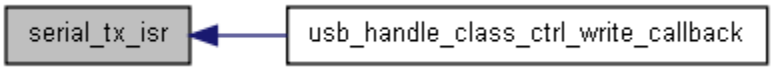
This routine needs to be called from your interrupt() routine when the transmit hardware interrupt occurs in order to send bytes that are waiting in the fifo buffer.

Definition at line 216 of file pic_serial.c.

References tx_buffer, tx_end, tx_start, and uns8.

Referenced by usb_handle_class_ctrl_write_callback().

Here is the caller graph for this function:

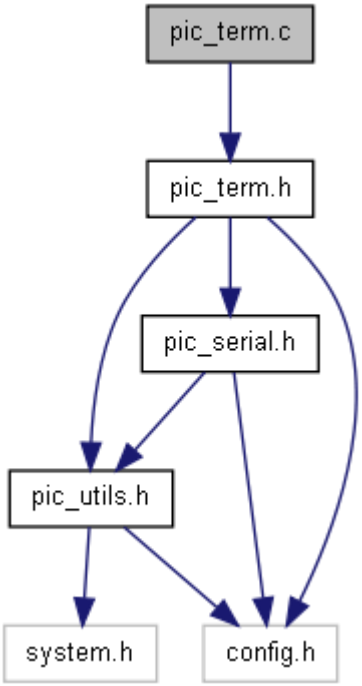



---

## pic_term.c File Reference

```
#include "pic_term.h"
```

Include dependency graph for pic_term.c:



## Functions

- void term_init ()
- void term_process ()

## Variables

- uns8 buffer_len
- uns8 term_buffer [TERM_BUFFER_SIZE]

---

## Function Documentation

### void term_init ()

Definition at line 82 of file pic_term.c.

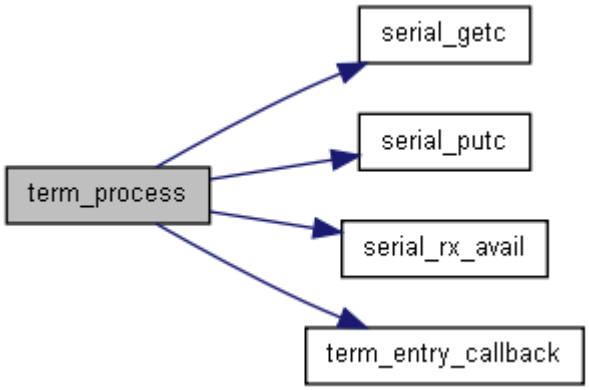
References buffer_len, and term_buffer.

### void term_process ()

Definition at line 43 of file pic_term.c.

References buffer_len, MAGIC_BOOSTBLOADER_REQUEST, serial_getc(), serial_putc(), serial_rx_avail(), term_buffer, term_entry_callback(), and uns8.

Here is the call graph for this function:




---

## Variable Documentation

### uns8 buffer_len

Definition at line 41 of file pic_term.c.

Referenced by term_init(), and term_process().

### uns8 term_buffer[TERM_BUFFER_SIZE]

Definition at line 40 of file pic_term.c.

Referenced by term_init(), and term_process().

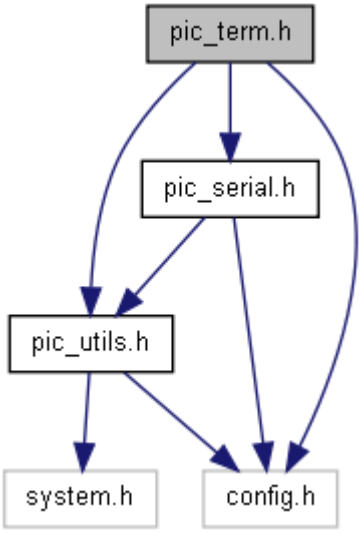
---

## pic_term.h File Reference

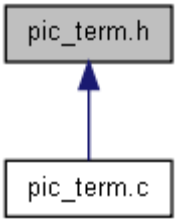
Pic terminal routines.

```
#include "pic_utils.h"
#include "config.h"
#include "pic_serial.h"
```

Include dependency graph for pic_term.h:




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




### Functions

- void term_entry_callback (uns8 *term_buffer)
- void term_init ()
- void term_process ()

---

### Detailed Description

Allows the use of a serial terminal for interaction with the PIC

Put the following into your config.h

```
// - - - - - - - - - - - - - - - - - - - -
// pic term defines
// - - - - - - - - - - - - - - - - - - - -

#define TERM_BUFFER_SIZE    10

// Reset (go to bootloader) if magic character received
#define TERM ALLOW BOOSTBLOADER
// Echo typing so user can see what they're doing
#define TERM ECHO INPUT

// - - - - - - - - - - - - - - - - - - - -

// Create term entry callback in your own code
// void term_entry_callback(uns8 *term_buffer);
```

Definition in file pic_term.h.

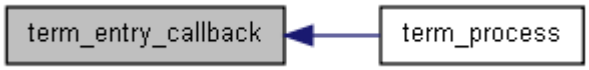
---

## Function Documentation

### void term_entry_callback (uns8 * *term_buffer*)

Referenced by term_process().

Here is the caller graph for this function:




### void term_init ()

Definition at line 82 of file pic_term.c.

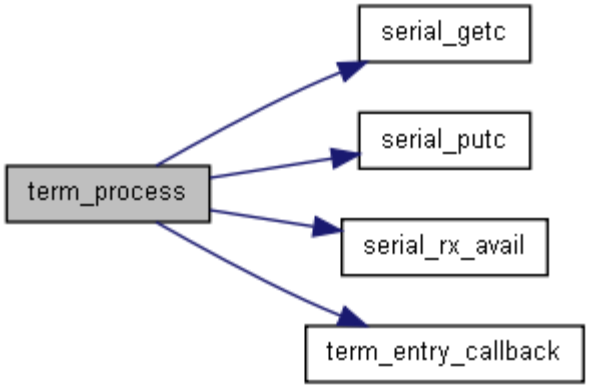
References buffer_len, and term_buffer.

### void term_process ()

Definition at line 43 of file pic_term.c.

References buffer_len, MAGIC_BOOSTBLOADER_REQUEST, serial_getc(), serial_putc(), serial_rx_avail(), term_buffer, term_entry_callback(), and uns8.

Here is the call graph for this function:

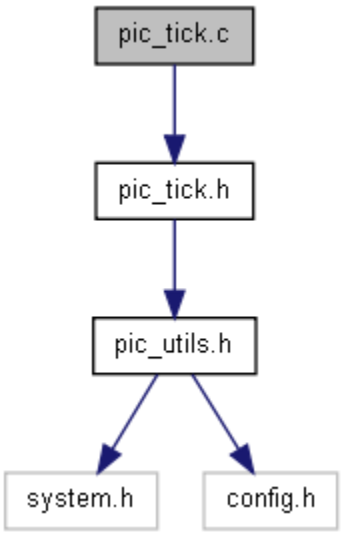



---

## pic_tick.c File Reference

`#include "pic_tick.h"`

Include dependency graph for pic_tick.c:




### Functions

- void handle_tick ()
- *Call this routine to increment tick count.* uns16 tick_calc_diff (uns16 a, uns16 b)
- *Calculate the tick time difference between two values.* uns16 tick_get_count ()
- *Return current tick count.* void timer_0_callback ()

*Timer 0 callback function.*

---

## Function Documentation

### void handle_tick ()

Typically called during the interrupt routine of a timer to increment the tick count. Note this routine assumes that interupts are off - which is always the case in an interrupt sub routine.

Definition at line 66 of file pic_tick.c.

References tick.

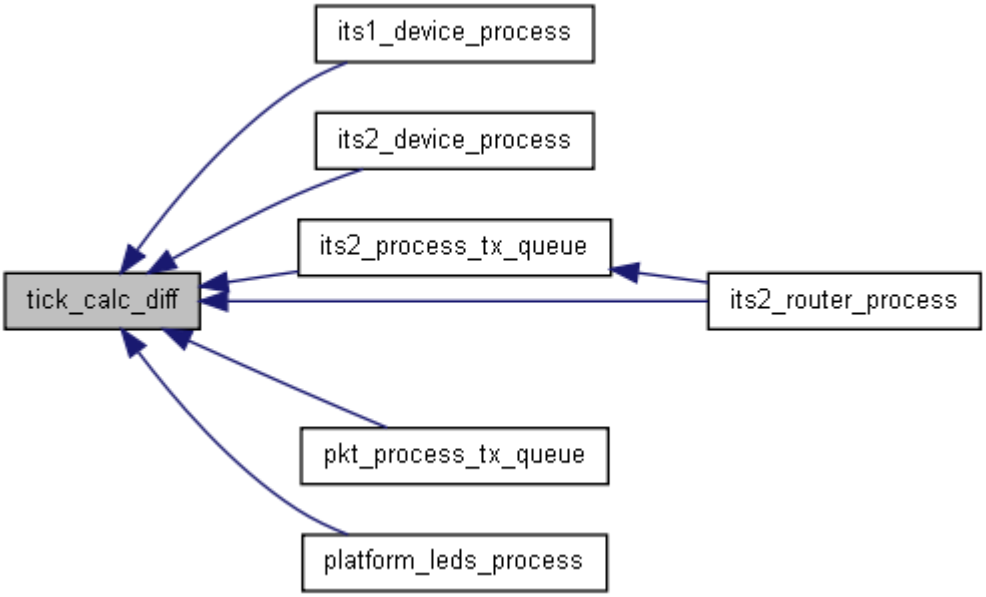
### uns16 tick_calc_diff (uns16 *a*, uns16 *b*)

Calculates how many ticks have elapsed between two tick values. Covers cases where the tick count wraps beyond its 16 bit value.

Definition at line 58 of file pic_tick.c.

Referenced by its1_device_process(), its2_device_process(), its2_process_tx_queue(), its2_router_process(), pkt_process_tx_queue(), and platform_leds_process().

Here is the caller graph for this function:




### uns16 tick_get_count ()

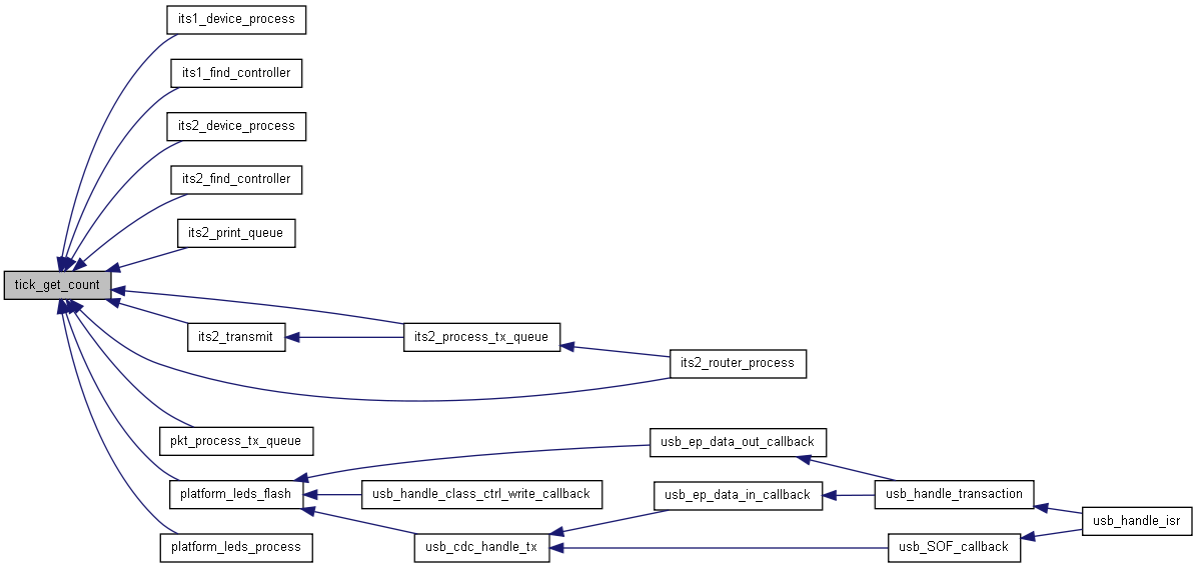
Returns the current tick count. Thread and interrupt safe.

Definition at line 47 of file pic_tick.c.

References end_crit_sec, start_crit_sec, tick, and uns16.

Referenced by its1_device_process(), its1_find_controller(), its2_device_process(), its2_find_controller(), its2_print_queue(), its2_process_tx_queue(), its2_router_process(), its2_transmit(), pkt_process_tx_queue(), platform_leds_flash(), and platform_leds_process().

Here is the caller graph for this function:




### void timer_0_callback ()

When a timer 0 interrupt occurs, after handling the interupt and timing issues, this callback function is executed. You will need to define this subroutine in your code, otherwise linking will fail.

Definition at line 43 of file pic_tick.c.

References handle_tick_inline().

Here is the call graph for this function:

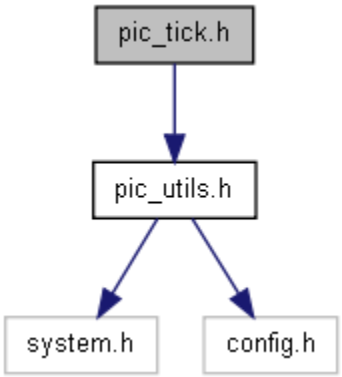

---

## pic_tick.h File Reference

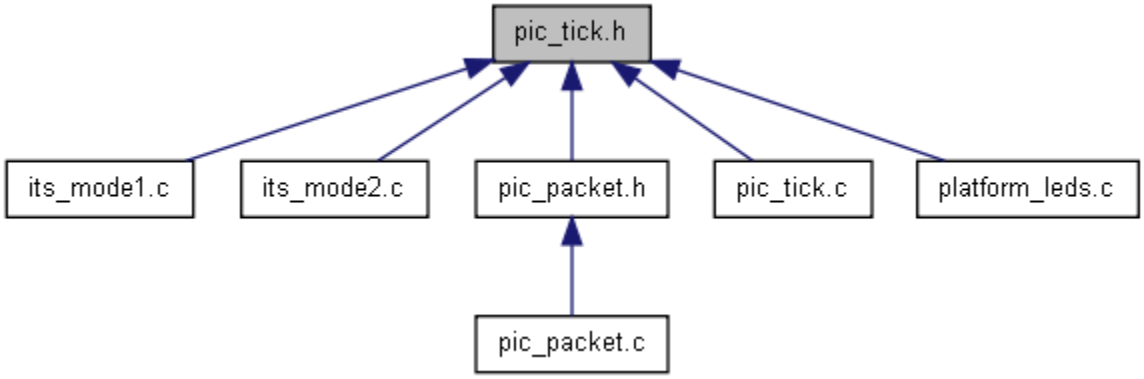
Timer helper routines.

```
#include "pic_utils.h"
```

Include dependency graph for pic_tick.h:



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




## Functions

- void handle_tick ()
- *Call this routine to increment tick count.* void handle_tick_inline ()
- *Call this routine to increment tick count - inline version.* uns16 tick_calc_diff (uns16 a, uns16 b)
- *Calculate the tick time difference between two values.* uns16 tick_get_count ()

## *Return current tick count.* Variables

- static uns16 tick = 0

---

## Detailed Description

Definition in file pic_tick.h.

---

## Function Documentation

### void handle_tick ()

Typically called during the interrupt routine of a timer to increment the tick count. Note this routine assumes that interupts are off - which is always the case in an interrupt sub routine.

Definition at line 66 of file pic_tick.c.

References tick.

### void handle_tick_inline () `[inline]`

Typically called during the interrupt routine of a timer to increment the tick count. Note this routine assumes that interupts are off - which is always the case in an interrupt sub routine. Inline version so you don't use up one stack level

Definition at line 84 of file pic_tick.h.

References tick.

Referenced by timer_0_callback().

Here is the caller graph for this function:

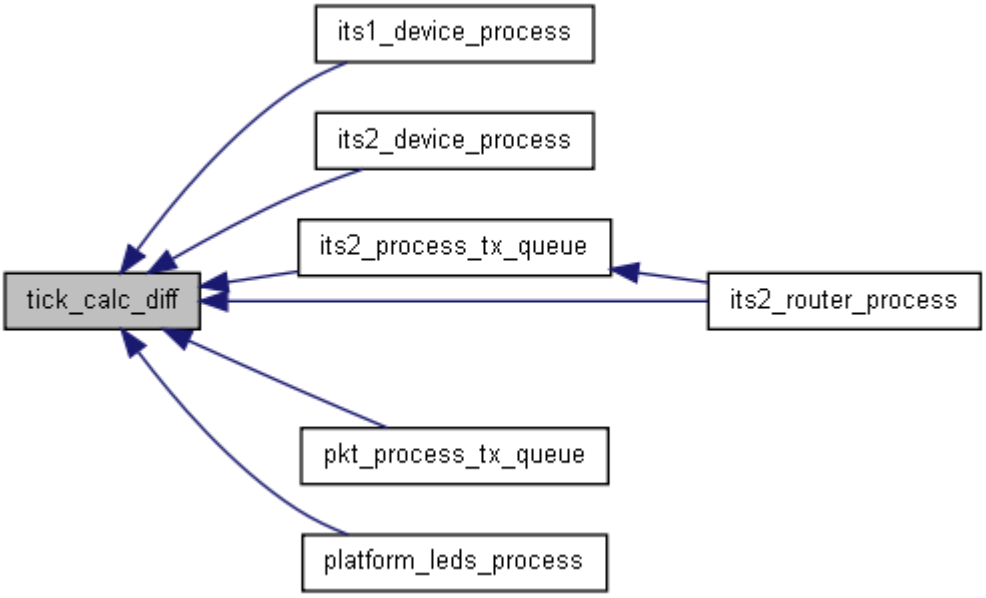

### uns16 tick_calc_diff (uns16 *a*, uns16 *b*)

Calculates how many ticks have elapsed between two tick values. Covers cases where the tick count wraps beyond its 16 bit value.

Definition at line 58 of file pic_tick.c.

Referenced by its1_device_process(), its2_device_process(), its2_process_tx_queue(), its2_router_process(), pkt_process_tx_queue(), and platform_leds_process().

Here is the caller graph for this function:




### uns16 tick_get_count ()

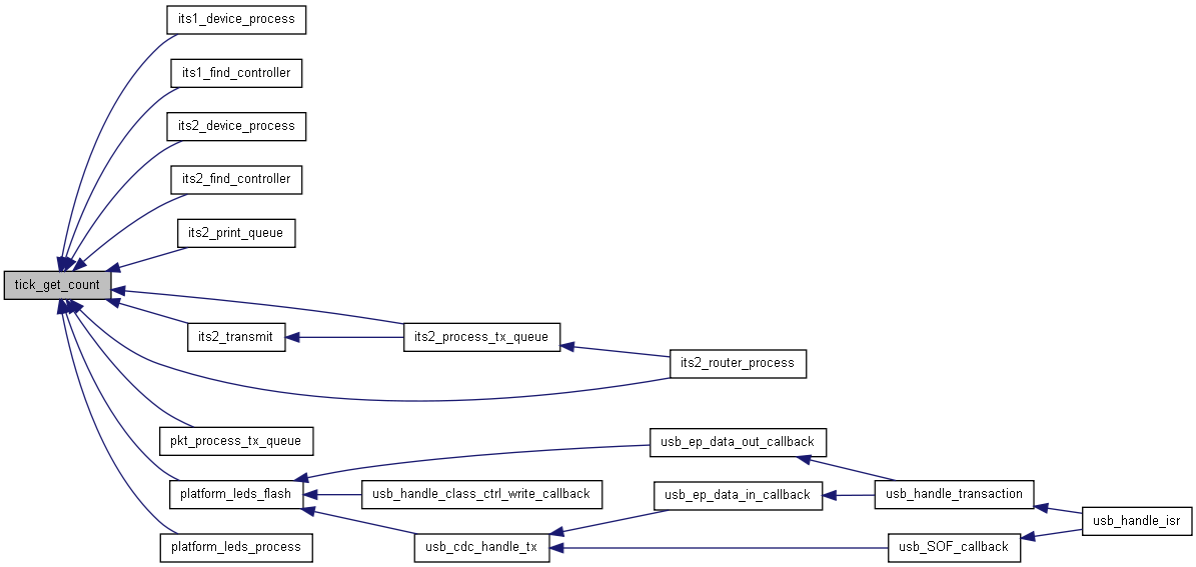
Returns the current tick count. Thread and interrupt safe.

Definition at line 47 of file pic_tick.c.

References end_crit_sec, start_crit_sec, tick, and uns16.

Referenced by its1_device_process(), its1_find_controller(), its2_device_process(), its2_find_controller(), its2_print_queue(), its2_process_tx_queue(), its2_router_process(), its2_transmit(), pkt_process_tx_queue(), platform_leds_flash(), and platform_leds_process().

Here is the caller graph for this function:




---

## Variable Documentation

### uns16 tick = 0 [static]

Global tick counter

Definition at line 51 of file pic_tick.h.

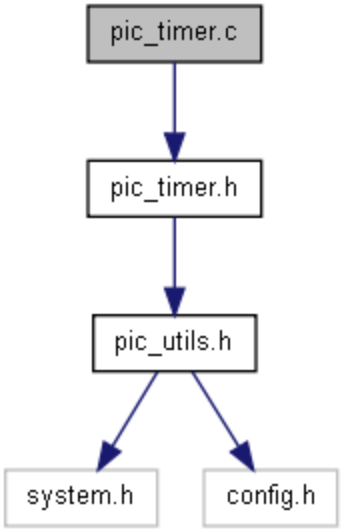
Referenced by handle_tick(), handle_tick_inline(), and tick_get_count().

---

# pic_timer.c File Reference

```
#include "pic_timer.h"
```

Include dependency graph for pic_timer.c:



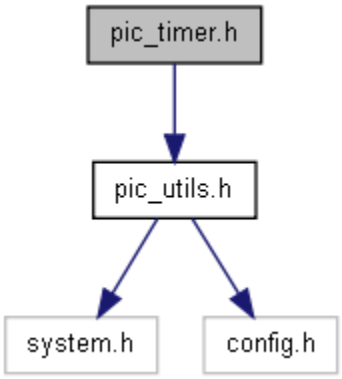

---

## pic_timer.h File Reference

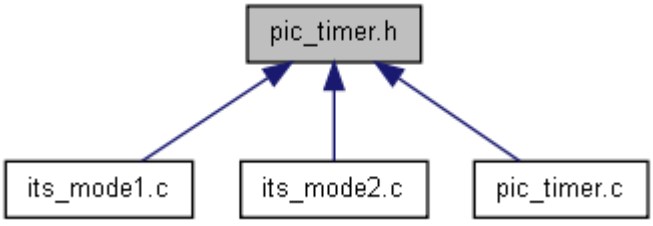
Access to timer 0.

```
#include "pic_utils.h"
```

Include dependency graph for pic_timer.h:




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



## Defines

- #define TIMER_16BIT_MODE 0
- #define TIMER_8BIT_MODE 1
- #define TIMER_PRESCALER_1_TO_128 0x06
- #define TIMER_PRESCALER_1_TO_16 0x03
- #define TIMER_PRESCALER_1_TO_2 0x00
- #define TIMER_PRESCALER_1_TO_256 0x07
- #define TIMER_PRESCALER_1_TO_32 0x04
- #define TIMER_PRESCALER_1_TO_4 0x01
- #define TIMER_PRESCALER_1_TO_64 0x05
- #define TIMER_PRESCALER_1_TO_8 0x02
- #define TIMER_PRESCALER_OFF 0xff

## Functions

- void timer_0_callback ()
- *Timer 0 callback function.* void timer_setup_0 (bit mode_16_bit, uns8 prescaler_setting, uns16 timer_start_value)
- *Setup timer zero with starting values.* void timer_start_0 ()
- *Start timer 0.* void timer_stop_0 ()

*Stop timer 0.*

---

## Detailed Description

Definition in file pic_timer.h.

---

## Define Documentation

### #define TIMER_16BIT_MODE 0

Timer mode for devices where this is applicable (16bit timer)

Definition at line 45 of file pic_timer.h.

### #define TIMER_8BIT_MODE 1

Timer mode for devices where this is applicable (8bit timer)

Definition at line 47 of file pic_timer.h.

### #define TIMER_PRESCALER_1_TO_128 0x06

Definition at line 57 of file pic_timer.h.

### #define TIMER_PRESCALER_1_TO_16 0x03

Definition at line 54 of file pic_timer.h.

### #define TIMER_PRESCALER_1_TO_2 0x00

Definition at line 51 of file pic_timer.h.

**#define TIMER_PRESCALER_1_TO_256 0x07**

Definition at line 58 of file pic_timer.h.

**#define TIMER_PRESCALER_1_TO_32 0x04**

Definition at line 55 of file pic_timer.h.

**#define TIMER_PRESCALER_1_TO_4 0x01**

Definition at line 52 of file pic_timer.h.

**#define TIMER_PRESCALER_1_TO_64 0x05**

Definition at line 56 of file pic_timer.h.

**#define TIMER_PRESCALER_1_TO_8 0x02**

Definition at line 53 of file pic_timer.h.

**#define TIMER_PRESCALER_OFF 0xff**

Definition at line 50 of file pic_timer.h.

---

## Function Documentation

**void timer_0_callback ()**

When a timer 0 interrupt occurs, after handling the interupt and timing issues, this callback function is executed. You will need to define this subroutine in your code, otherwise linking will fail.

Definition at line 43 of file pic_tick.c.

References handle_tick_inline().

Here is the call graph for this function:


**void timer_setup_0 (bit *mode_16_bit*, uns8 *prescaler_setting*, uns16 *timer_start_value*)**

Turns off timer zero, congures 16/8bit mode (only for 18f devices), prescaler setting and start value (which will be loaded on each reset.

**void timer_start_0 ()**

Kicks off timer 0. In pic18 devices this will turn the timer on, on pic16 devices this will turn on timer0 interrupts.

**void timer_stop_0 ()**

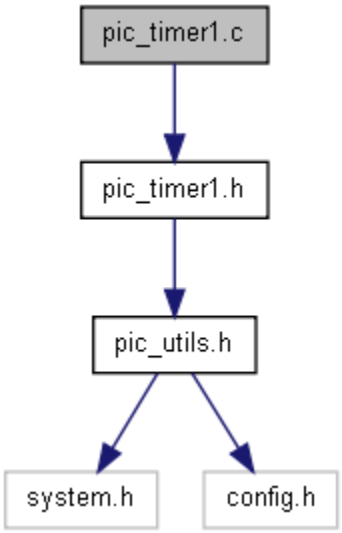
Stops timer 0. In pic18 devices, this will switch the timer off. On pic16 devices this will merely turn off the interrupt and the timer will continue running.

---

# pic_timer1.c File Reference

```
#include "pic_timer1.h"
```

Include dependency graph for pic_timer1.c:




## Functions

- void timer_setup_1 (uns8 prescaler_setting, uns16 timer_start_value)
- *Setup timer 1 with starting values.* void timer_start_1 ()
- *Start timer 1.* void timer_stop_1 ()

## *Stop timer 1.* Variables

- uns16 timer_1_start_value = 0

---

## Function Documentation

**void timer_setup_1 (uns8 *prescaler_setting*, uns16 *timer_start_value*)**

Turns off timer 1, sets prescaler setting and start value (which will be loaded on each reset).

Definition at line 43 of file pic_timer1.c.

References timer_1_start_value.

**void timer_start_1 ()**

Kicks off timer 1.

Definition at line 56 of file pic_timer1.c.

References timer_1_start_value.

**void timer_stop_1 ()**

Stops timer 1.

Definition at line 63 of file pic_timer1.c.

---

## Variable Documentation

**uns16 timer_1_start_value = 0**

Definition at line 40 of file pic_timer1.c.

Referenced by timer_handle_1_isr(), timer_setup_1(), and timer_start_1().

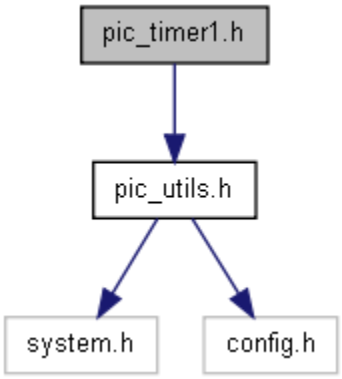
---

# pic_timer1.h File Reference

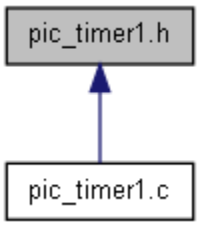
Timer 1 support.

```
#include "pic_utils.h"
```

Include dependency graph for pic_timer1.h:




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



## Defines

- #define TIMER1_PRESCALER_1_TO_2 0b00010000
- #define TIMER1_PRESCALER_1_TO_4 0b00100000
- #define TIMER1_PRESCALER_1_TO_8 0b00110000
- #define TIMER1_PRESCALER_OFF 0b00000000

## Functions

- void timer_1_callback ()
- *Timer 1 callback function.* void timer_handle_1_isr ()
- *handle timer 1 in interrupt service routine* void timer_setup_1 (uns8 prescaler_setting, uns16 timer_start_value)
- *Setup timer 1 with starting values.* void timer_start_1 ()
- *Start timer 1.* void timer_stop_1 ()

## *Stop timer 1.* Variables

- uns16 timer_1_start_value

---

## Detailed Description

Definition in file pic_timer1.h.

---

## Define Documentation

### #define TIMER1_PRESCALER_1_TO_2 0b00010000

Definition at line 44 of file pic_timer1.h.

### #define TIMER1_PRESCALER_1_TO_4 0b00100000

Definition at line 45 of file pic_timer1.h.

### #define TIMER1_PRESCALER_1_TO_8 0b00110000

Definition at line 46 of file pic_timer1.h.

### #define TIMER1_PRESCALER_OFF 0b00000000

Definition at line 43 of file pic_timer1.h.

---

## Function Documentation

### void timer_1_callback ()

When a timer 1 interrupt occurs, after handling the interupt and timing issues, this callback function is executed. You will need to define this subroutine in your code, otherwise linking will fail.

Referenced by timer_handle_1_isr().

Here is the caller graph for this function:


**void timer_handle_1_isr () `[inline]`**

Call this routine in your interrupt subroutine to automatically service timer 1 interrupts if they have occursed.

Definition at line 94 of file pic_timer1.h.

References timer_1_callback(), timer_1_start_value, and uns16.

Here is the call graph for this function:


**void timer_setup_1 (uns8 *prescaler_setting*, uns16 *timer_start_value*)**

Turns off timer 1, sets prescaler setting and start value (which will be loaded on each reset).

Definition at line 43 of file pic_timer1.c.

References timer_1_start_value.

**void timer_start_1 ()**

Kicks off timer 1.

Definition at line 56 of file pic_timer1.c.

References timer_1_start_value.

**void timer_stop_1 ()**

Stops timer 1.

Definition at line 63 of file pic_timer1.c.

---

## Variable Documentation

**uns16 timer_1_start_value**

Definition at line 40 of file pic_timer1.c.

Referenced by timer_handle_1_isr(), timer_setup_1(), and timer_start_1().

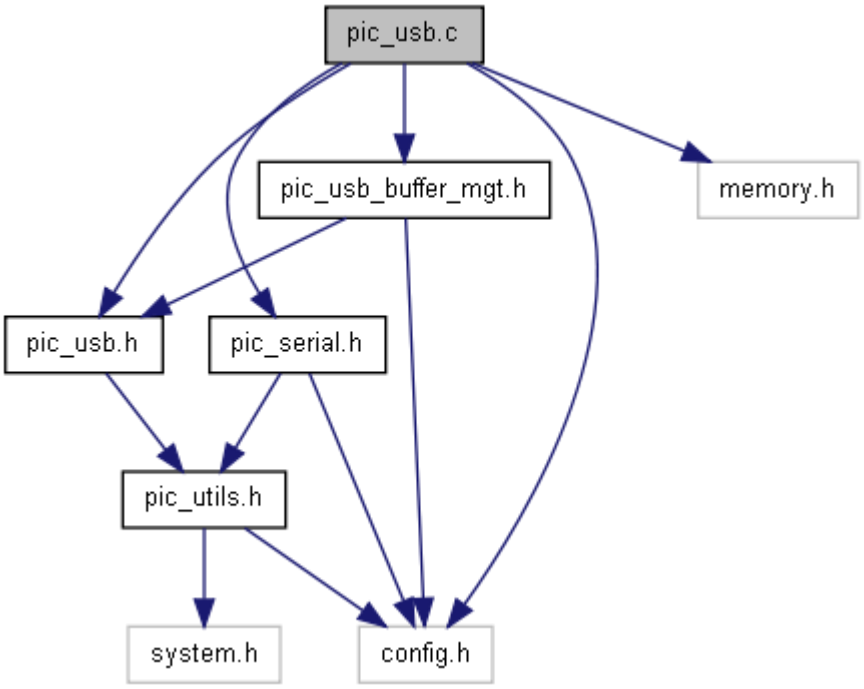
---

# pic_usb.c File Reference

```
#include "pic_usb.h"
#include "memory.h"
```

```
#include "config.h"
#include "pic_usb_buffer_mgt.h"
#include "pic_serial.h"
```

Include dependency graph for pic_usb.c:




## Functions

- void turn_usb_ints_on ()
- *Turn on USB interrupts.* void usb_configure_endpoints ()
- void usb_enable_module ()
- *Enables the USB hardware and starts USB negotiations.* usb_state_type usb_get_state ()
- *Query the current state of the USB connection.* void usb_handle_isr ()
- *Handle USB interrupts.* void usb_handle_reset ()
- void usb_handle_stall ()
- void usb_handle_standard_request (setup_data_packet sdp)
- void usb_handle_transaction (uns8 stat)
- void usb_prime_ep0_out_e ()
- void usb_prime_ep0_out_o ()
- void usb_send_data (uns8 ep, uns8 *data, uns8 send_count, bit first)
- *Send data over an endpoint pipe.* void usb_send_data_chunk ()
- void usb_send_empty_data_pkt ()
- *Send an empty data packet.* void usb_send_one_byte (uns8 data)
- void usb_setup ()
- *Setup USB hardware ready for use.* void usb_stall_ep0 ()
- *Send a stall on control transfer endpoint.* void usb_stall_on_in ()

## Variables

- uns8 buffer_byte
- control_mode_type control_mode
- buffer_descriptor * delivery_bd

- uns8 * delivery_buffer
- uns8 delivery_buffer_size
- uns16 delivery_bytes_max_send
- uns16 delivery_bytes_sent
- uns16 delivery_bytes_to_send
- uns8 * delivery_ptr
- uns8 usb_address
- setup_data_packet usb_sdp
- usb_state_type usb_state = st_POWERED
- usb_status_type usb_status

---

## Function Documentation

### void turn_usb_ints_on ()

If you are using interrupt-driven code (generally the best way of doing things) you can turn on USB interrupts using turn_usb_ints_on(). Don't forget that you will also need to call turn_global_ints_on() as well. Typically this is called in your system setup routine.

Definition at line 962 of file pic_usb.c.

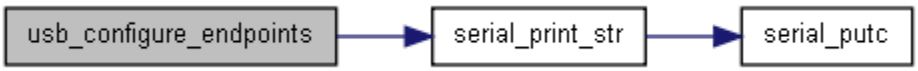
### void usb_configure_endpoints ()

Definition at line 66 of file pic_usb.c.

References buffer_descriptor::addr, BC8, BC9, bd1in, bd1out, bd2in, bd2out, bd3in, bd3out, BSTALL, buffer_descriptor::count, DTS, DTSEN, INCDIS, KEN, serial_print_str(), buffer_descriptor::stat, and UOWN.

Referenced by usb_handle_standard_request().

Here is the call graph for this function:




Here is the caller graph for this function:


### void usb_enable_module ()

After you've called usb_setup(), you can call usb_enable_module() whenever you're ready for USB negotiations to occur. Normally, this would need to occur relatively quickly after power-up if your PIC is powered by USB and it's purpose is to talk over USB. This is normally called from your main() routine once all other configuration is done.

Once the USB module has successfully negotiated a connection with the host, usb_device_configured_callback() will be called if you have requested this in your config.h file. This will indicate a successful connection. Because of the way USB works, there is no way to tell that it *hasn't* worked, except via a timer - if you haven't had a good connection in several seconds, you can assume it has failed (although this may just mean the user is hunting for a driver disk etc).

Definition at line 1033 of file pic_usb.c.

References st_DEFAULT, and usb_state.

**usb_state_type usb_get_state ()**

Returns the USB state, either powered, default, address or connect. This is updated as the negotiation progresses. It may be useful to query this if, after a suitable time-out, the connection has not been made.

Definition at line 1039 of file pic_usb.c.

References usb_state.

**void usb_handle_isr ()**

usb_handle_isr() should be inserted in your interrupt service routine. Alternatively, if you have reason not to want to do interrupt-driven USB, for example, a bootloader, you can poll this routine.

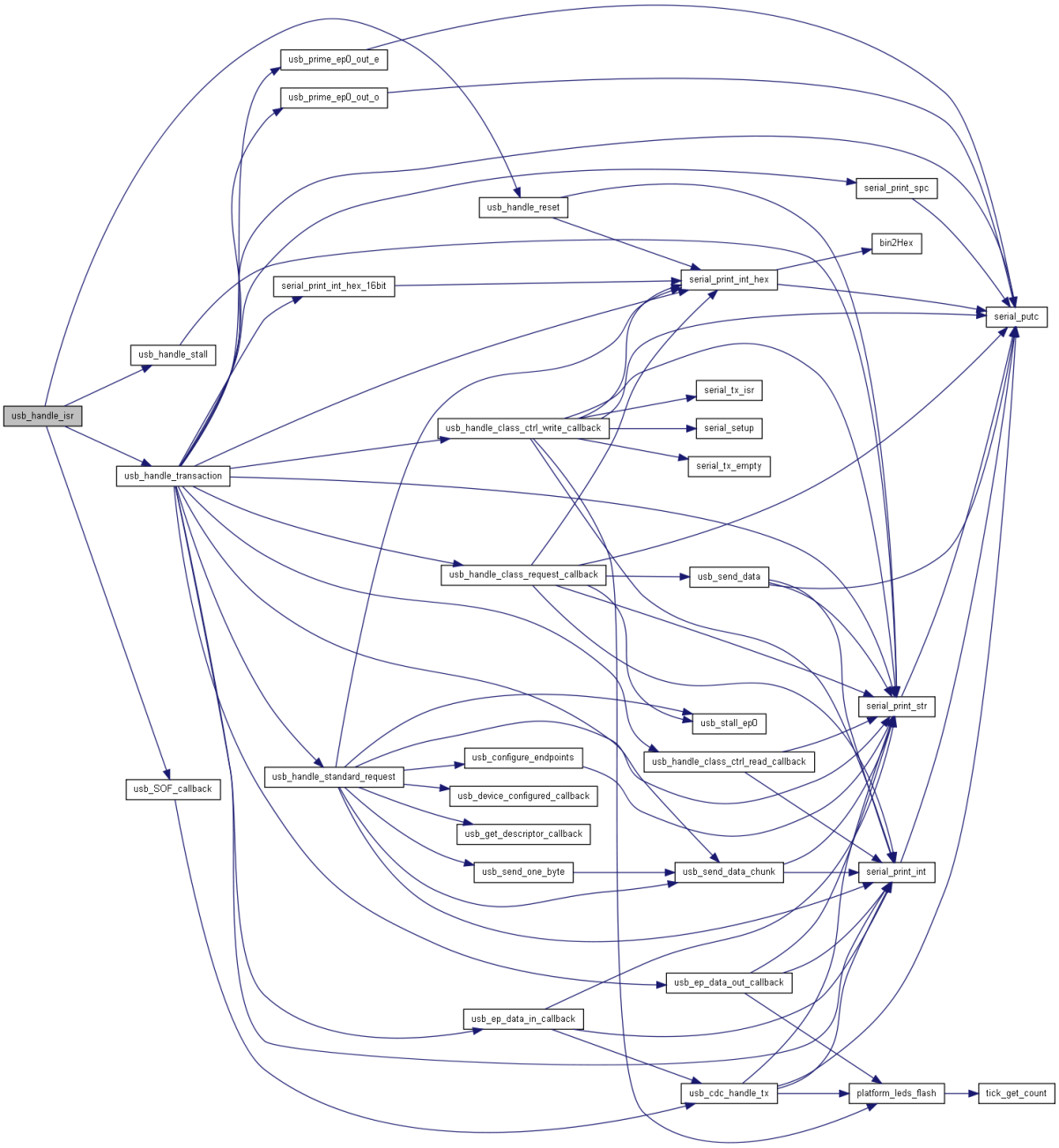
Make sure you call turn_usb_ints() and turn_global_ints_on() to ensure interrupts occur.

It will check for any of the USB interrupt flags and handle: USB transactions, USB reset, USB stall, USB Start Of Frame (including calling usb_SOF_callback() if configured in your config.h and most importantly USB transaction, which is where all the hard work is done.

Definition at line 928 of file pic_usb.c.

References uns8, usb_handle_reset(), usb_handle_stall(), usb_handle_transaction(), and usb_SOF_callback().

Here is the call graph for this function:



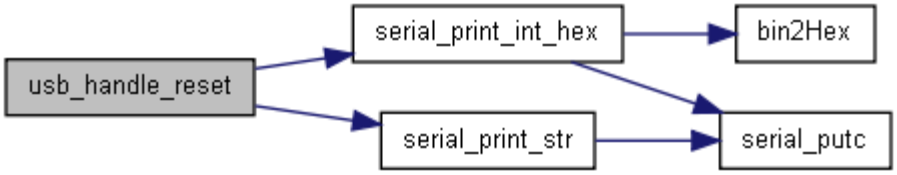

**void usb_handle_reset ()**

Definition at line 853 of file pic_usb.c.

References buffer_descriptor::addr, BC8, BC9, bd0in, bd0out_e, bd0out_o, BSTALL, cm_IDLE, control_mode, buffer_descriptor::count, DTS, DTSEN, INCDIS, KEN, serial_print_int_hex(), serial_print_str(), buffer_descriptor::stat, UOWN, us_IDLE, usb_address, USB_EP0_IN_ADDR, USB_EP0_OUT_E_ADDR, USB_EP0_OUT_O_ADDR, and usb_status.

Referenced by usb_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:

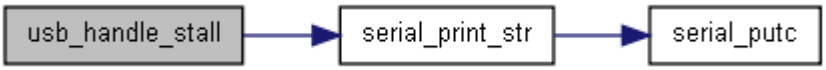

### void usb_handle_stall ()

Definition at line 918 of file pic_usb.c.

References bd0in, BSTALL, serial_print_str(), buffer_descriptor::stat, and UOWN.

Referenced by usb_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:

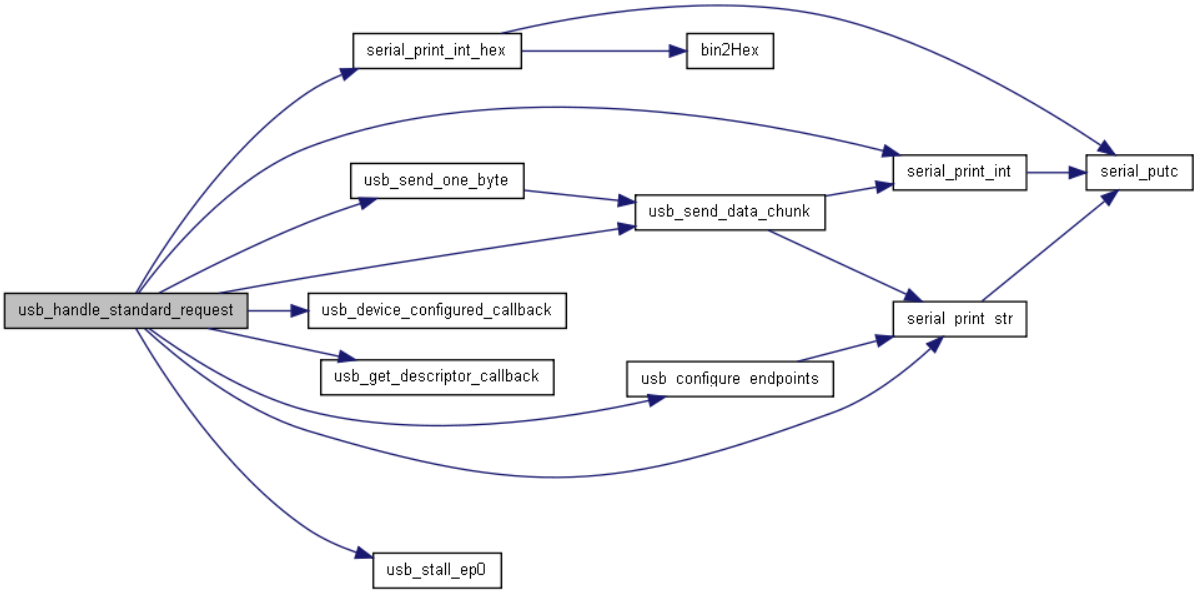

### void usb_handle_standard_request (setup_data_packet *sdp*)

Definition at line 428 of file pic_usb.c.

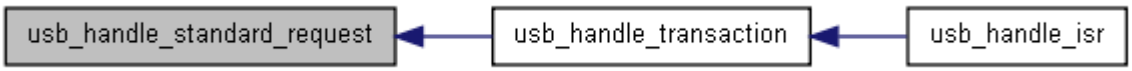
References bd0in, setup_data_packet::bRequest, cm_CTRL_READ_DATA_STAGE, cm_CTRL_WRITE_SENDING_STATUS, control_mode, delivery_buffer, delivery_buffer_size, delivery_bytes_max_send, delivery_bytes_sent, delivery_bytes_to_send, delivery_ptr, DTS, req_Get_Descriptor, req_Get_Interface, req_Get_Status, req_Set_Address, req_Set_Configuration, serial_print_int(), serial_print_int_hex(), serial_print_str(), st_CONFIGURED, buffer_descriptor::stat, uns8, us_SET_ADDRESS, usb_address, usb_configure_endpoints(), usb_device_configured_callback(), USB_EP0_IN_ADDR, usb_get_descriptor_callback(), usb_send_data_chunk(), usb_send_one_byte(), usb_send_status_ack, usb_stall_ep0(), usb_state, usb_status, setup_data_packet::wLength, and setup_data_packet::wValue.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




**void usb_handle_transaction (uns8 *stat*)**

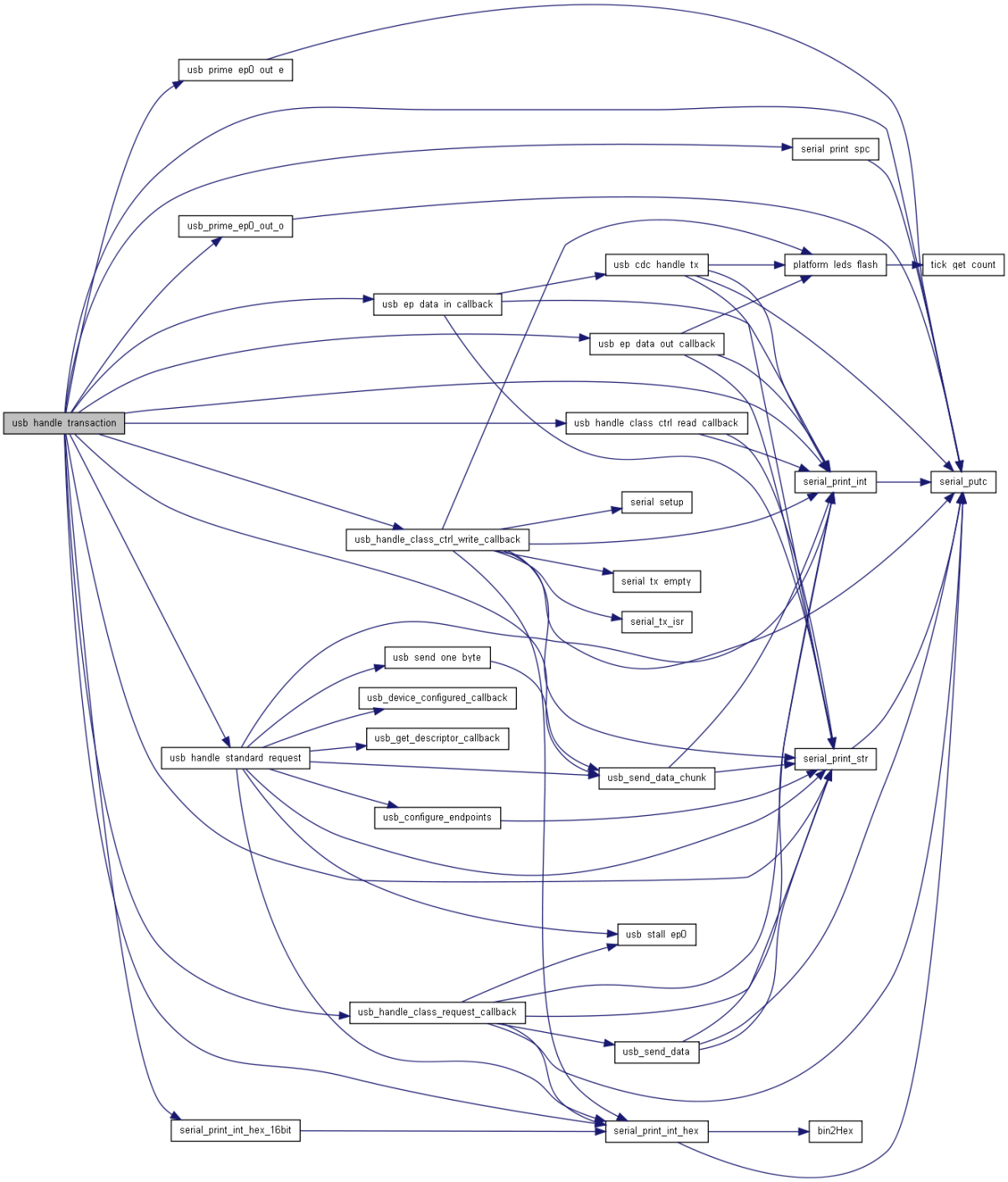
!usb_stall_on_in();

Definition at line 524 of file pic_usb.c.

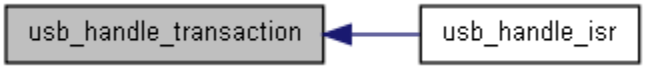
References buffer_descriptor::addr, BC8, BC9, bd0in, bd0out_e, bd0out_o, setup_data_packet::bmRequestType, setup_data_packet::bRequest, BSTALL, buffer_0_in, buffer_0_out_e, buffer_0_out_o, cm_CTRL_READ_AWAITING_STATUS, cm_CTRL_READ_DATA_STAGE, cm_CTRL_READ_DATA_STAGE_CLASS, cm_CTRL_WRITE_DATA_STAGE_CLASS, cm_CTRL_WRITE_SENDING_STATUS, cm_IDLE, control_mode, buffer_descriptor::count, DATA_STAGE_DIR, DTS, DTSEN, ep_in_bd_location, ep_out_bd_location, ep_out_buffer_location, ep_out_buffer_size, INCDIS, KEN, pid_ACK, pid_IN, pid_OUT, pid_SETUP, REQUEST_TYPE0, REQUEST_TYPE1, serial_print_int(), serial_print_int_hex(), serial_print_int_hex_16bit(), serial_print_spc(), serial_print_str(), serial_putc(), st_ADDRESS, buffer_descriptor::stat, uns16, uns8, UOWN, us_IDLE, us_SET_ADDRESS, usb_address, usb_ep_data_in_callback(), usb_ep_data_out_callback(), usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_standard_request(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data_chunk(), usb_state, usb_status, setup_data_packet::wIndex, setup_data_packet::wLength, and setup_data_packet::wValue.

Referenced by usb_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:




**void usb_prime_ep0_out_e ()**

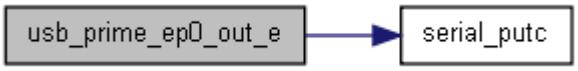
! clear

Definition at line 384 of file pic_usb.c.

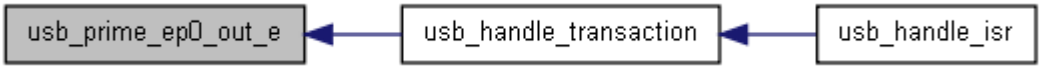
References buffer_descriptor::addr, BC8, BC9, bd0out_e, BSTALL, buffer_descriptor::count, DTS, DTSEN, INCDIS, KEN, serial_putc(), buffer_descriptor::stat, UOWN, and USB_EP0_OUT_E_ADDR.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




### void usb_prime_ep0_out_o ()

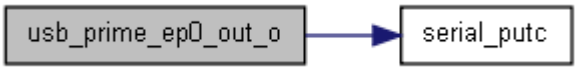
! clear

Definition at line 406 of file pic_usb.c.

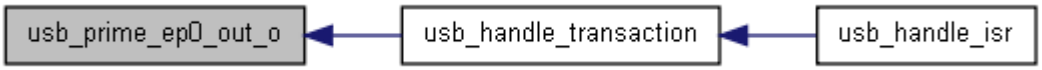
References buffer_descriptor::addr, BC8, BC9, bd0out_o, BSTALL, buffer_descriptor::count, DTS, DTSEN, INCDIS, KEN, serial_putc(), buffer_descriptor::stat, UOWN, and USB_EP0_OUT_O_ADDR.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




### void usb_send_data (uns8 *ep*, uns8 * *data*, uns8 *send_count*, bit *first*)

Use this routine to send data across the USB pipe.

**Parameters:**

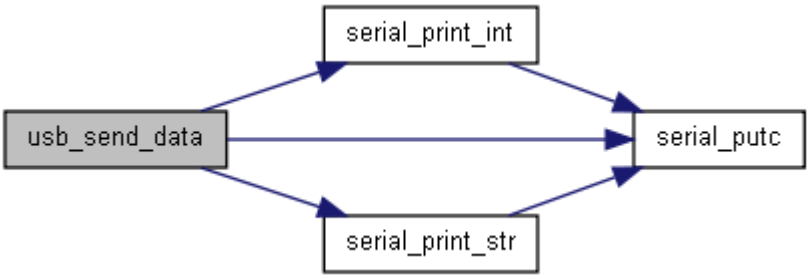
*ep* Endpoint that the data should be sent from
*data* pointer to the data
*send_count* the number of bytes to send
*first* True if this is the first in a series of sends. Generally, this can be set to False, since it will automatically be set to the right value on endpoint creation. However, in the case of control transfers, the data stage needs to have the first parameter set to True to ensure the DTS bit is set correctly.

Definition at line 211 of file pic_usb.c.

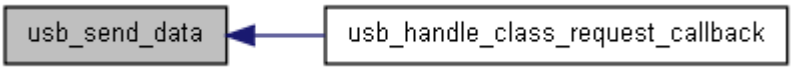
References buffer_descriptor::addr, BC8, BC9, BSTALL, buffer_descriptor::count, DTS, DTSEN, ep_in_bd_location, ep_in_buffer_location, INCDIS, KEN, serial_print_int(), serial_print_str(), serial_putc(), buffer_descriptor::stat, uns16, uns8, and UOWN.

Referenced by usb_handle_class_request_callback().

Here is the call graph for this function:




Here is the caller graph for this function:



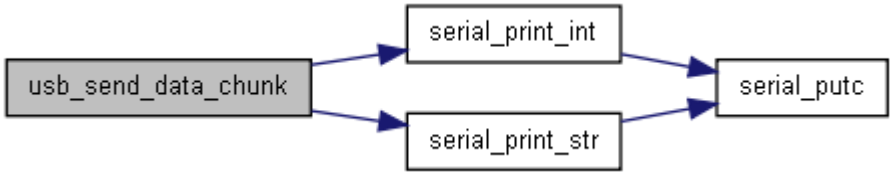

### void usb_send_data_chunk ()

Definition at line 260 of file pic_usb.c.

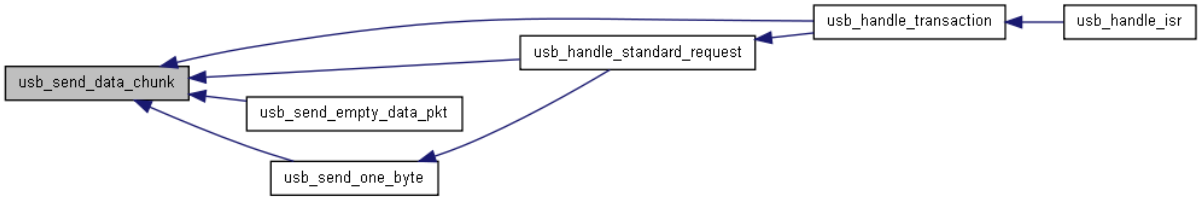
References buffer_descriptor::addr, BC8, BC9, bd0in, BSTALL, buffer_0_in, cm_CTRL_READ_AWAITING_STATUS, cm_CTRL_WRITE_SENDING_STATUS, control_mode, buffer_descriptor::count, delivery_buffer, delivery_buffer_size, delivery_bytes_max_send, delivery_bytes_sent, delivery_bytes_to_send, delivery_ptr, DTS, DTSEN, INCDIS, KEN, serial_print_int(), serial_print_str(), buffer_descriptor::stat, uns16, uns8, and UOWN.

Referenced by usb_handle_standard_request(), usb_handle_transaction(), usb_send_empty_data_pkt(), and usb_send_one_byte().

Here is the call graph for this function:




Here is the caller graph for this function:



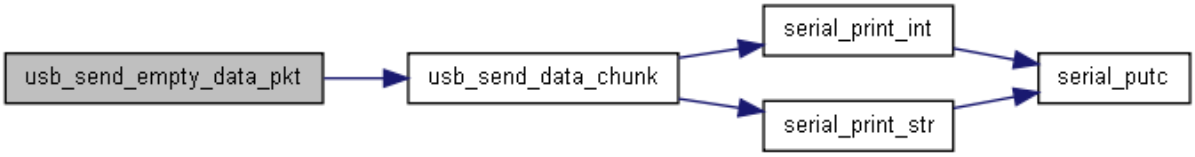

### void usb_send_empty_data_pkt ()

Use this routine to send an data across the USB pipe on endpoint 0. This is the equivalent of sending a status acknowledge.

Definition at line 341 of file pic_usb.c.

References bd0in, buffer_0_in, delivery_buffer, delivery_buffer_size, delivery_bytes_max_send, delivery_bytes_sent, delivery_bytes_to_send, delivery_ptr, DTS, buffer_descriptor::stat, uns8, and usb_send_data_chunk().

Here is the call graph for this function:



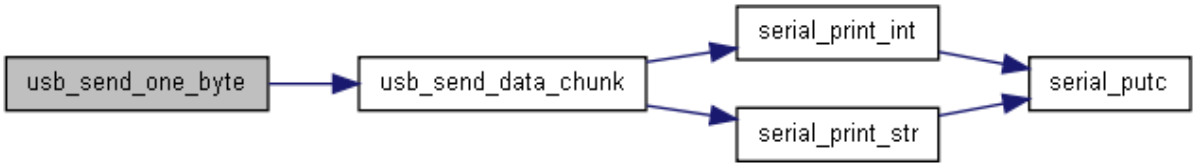

### void usb_send_one_byte (uns8 *data*)

Definition at line 370 of file pic_usb.c.

References bd0in, buffer_0_in, buffer_byte, delivery_buffer, delivery_buffer_size, delivery_bytes_max_send, delivery_bytes_sent, delivery_bytes_to_send, delivery_ptr, DTS, buffer_descriptor::stat, uns8, and usb_send_data_chunk().

Referenced by usb_handle_standard_request().

Here is the call graph for this function:




Here is the caller graph for this function:


### void usb_setup ()

usb_setup() configures the PIC USB hardware ready for use and prepares the internal data structures used to keep track of where the endpoint buffers are.

After calling usb_setup(), you are ready to call usb_enable_module() to actually start USB negotiations. Ensure that you have usb_handle_isr() in your interrupt service routine.

Definition at line 973 of file pic_usb.c.

References bd0in, bd0out_e, bd1in, bd1out, bd2in, bd2out, bd3in, bd3out, bd4in, bd4out, ep_in_bd_location, ep_out_bd_location, st_POWERED, and usb_state.

### void usb_stall_ep0 ()

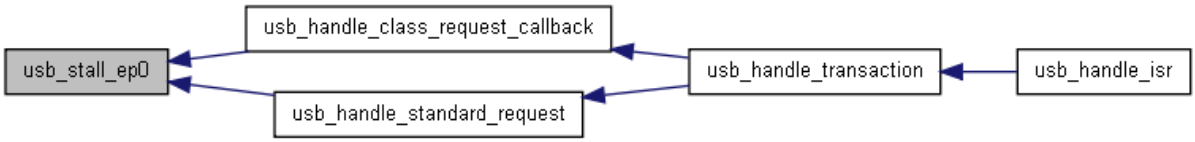
Use this routine to send a stall on the control transfer endpoint - usually used to indicate that the requested function is not available.

Definition at line 203 of file pic_usb.c.

References bd0in, BSTALL, buffer_descriptor::stat, and UOWN.

Referenced by usb_handle_class_request_callback(), and usb_handle_standard_request().

Here is the caller graph for this function:




**void usb_stall_on_in ()**

Definition at line 353 of file pic_usb.c.

References BC8, BC9, bd0in, BSTALL, DTS, DTSEN, INCDIS, KEN, buffer_descriptor::stat, and UOWN.

---

## Variable Documentation

**uns8 buffer_byte**

Definition at line 368 of file pic_usb.c.

Referenced by usb_send_one_byte().

**control_mode_type control_mode**

Store the control mode state

Definition at line 55 of file pic_usb.c.

Referenced by usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_standard_request(), usb_handle_transaction(), and usb_send_data_chunk().

**buffer_descriptor* delivery_bd**

Definition at line 62 of file pic_usb.c.

**uns8* delivery_buffer**

Definition at line 61 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns8 delivery_buffer_size**

Definition at line 60 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns16 delivery_bytes_max_send**

Definition at line 56 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns16 delivery_bytes_sent**

Definition at line 56 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns16 delivery_bytes_to_send**

Definition at line 56 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns8* delivery_ptr**

Definition at line 59 of file pic_usb.c.

Referenced by usb_handle_standard_request(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns8 usb_address**

Store the usb address

Definition at line 54 of file pic_usb.c.

Referenced by usb_handle_reset(), usb_handle_standard_request(), and usb_handle_transaction().

**setup_data_packet usb_sdp**

Store the last setup data packet

Definition at line 52 of file pic_usb.c.

Referenced by usb_handle_class_ctrl_read_callback(), and usb_handle_class_ctrl_write_callback().

**usb_state_type usb_state = st_POWERED**

Store the current USB device state

Definition at line 50 of file pic_usb.c.

Referenced by usb_enable_module(), usb_get_state(), usb_handle_standard_request(), usb_handle_transaction(), and usb_setup().

**usb_status_type usb_status**

Definition at line 64 of file pic_usb.c.

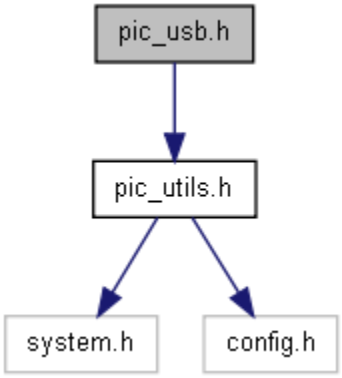
Referenced by usb_handle_reset(), usb_handle_standard_request(), and usb_handle_transaction().

# pic_usb.h File Reference

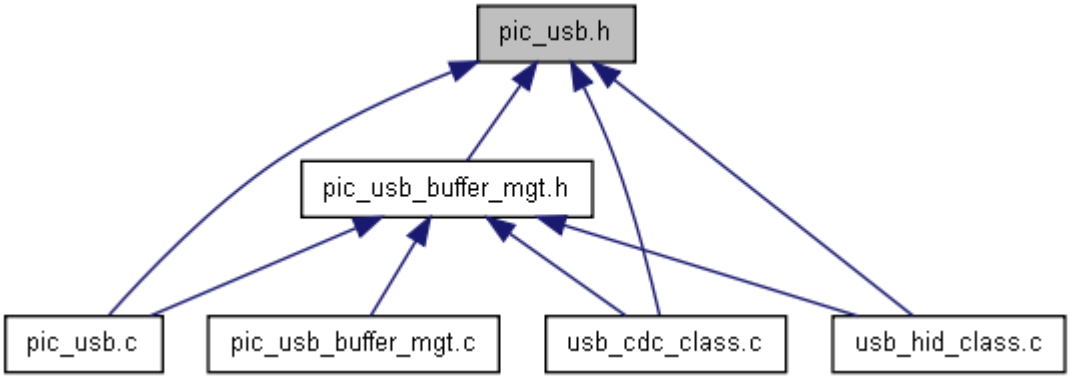
Pic USB routines.

```
#include "pic_utils.h"
```

Include dependency graph for pic_usb.h:




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




## Data Structures

- struct buffer_descriptor
- struct CDC_ACM_functional_descriptor
- struct CDC_call_mgt_functional_descriptor
- struct CDC_header_functional_descriptor
- struct CDC_union_functional_descriptor
- struct configuration_descriptor
- struct device_descriptor
- struct endpoint_descriptor
- struct hid_descriptor
- struct interface_descriptor
- struct setup_data_packet

## Defines

- #define BC8 0
- #define BC9 1
- #define BSTALL 2
- #define DATA_STAGE_DIR 7

- #define dt_CONFIGURATION 0x02
- #define dt_CS_INTERFACE 0x24
- #define dt_DEBUG 0x0a
- #define dt_DEVICE 0x01
- #define dt_DEVICE_QUALIFIER 0x06
- #define dt_ENDPOINT 0x05
- #define dt_HID 0x21
- #define dt_HID_REPORT 0x22
- #define dt_INTERFACE 0x04
- #define dt_INTERFACE_ASSOC 0x0b
- #define dt_INTERFACE_POWER 0x08
- #define dt_OTG 0x09
- #define dt_OTHER_SPEED_CONFIG 0x07
- #define dt_STRING 0x03
- #define DTS 6
- #define DTSEN 3
- #define INCDIS 4
- #define KEN 5
- #define PID0 2
- #define PID1 3
- #define PID2 4
- #define PID3 5
- #define pid_ACK 0b00000010
- #define pid_DATA0 0b00000011
- #define pid_DATA1 0b00001011
- #define pid_DATA2 0b00000111
- #define pid_IN 0b00001001
- #define pid_MDATA 0b00001111
- #define pid_NAK 0b00001010
- #define pid_NYET 0b00000110
- #define pid_OUT 0b00000001
- #define pid_SETUP 0b00001101
- #define pid_SOF 0b00000101
- #define pid_STALL 0b00001110
- #define req_Clear_Feature 0x01
- #define req_Get_Configuration 0x08
- #define req_Get_Descriptor 0x06
- #define req_Get_Interface 0x0a
- #define req_Get_Status 0x00
- #define req_Set_Address 0x05
- #define req_Set_Configuration 0x09
- #define req_Set_Descriptor 0x07
- #define req_Set_Feature 0x03
- #define req_Set_Interface 0x0b
- #define req_Synch_Frame 0x0c
- #define REQUEST_TYPE0 5
- #define REQUEST_TYPE1 6
- #define UOWN 7
- #define usb_send_status_ack() usb_send_empty_data_pkt()

## Enumerations

- enum control_mode_type { cm_IDLE, cm_CTRL_WRITE_DATA_STAGE, cm_CTRL_WRITE_DATA_STAGE_CLASS, cm_CTRL_READ_DATA_STAGE, cm_CTRL_READ_DATA_STAGE_CLASS, cm_CTRL_READ_AWAITING_STATUS, cm_CTRL_WRITE_SENDING_STATUS }
- enum usb_state_type { st_POWERED, st_DEFAULT, st_ADDRESS, st_CONFIGURED }
- enum usb_status_type { us_IDLE, us_SET_ADDRESS }

## Functions

- void turn_usb_ints_on ()
- *Turn on USB interrupts.* void usb_device_configured_callback ()
- *Callback routine triggered when a successful intial USB negotiation has completed.* void usb_enable_module ()
- *Enables the USB hardware and starts USB negotiations.* void usb_ep_data_in_callback (uns8 end_point, uns16 byte_count)
- *Callback routine triggered when data has been sent to the host.* void usb_ep_data_out_callback (uns8 end_point, uns8 *buffer_location, uns16 byte_count)
- *Callback routine triggered when data has been sent to the device.* void usb_get_descriptor_callback (uns8 descriptor_type, uns8 descriptor_num, uns8 **rtn_descriptor_ptr, uns16 *rtn_descriptor_size)
- *Callback routine triggered when the descriptor is requested by the host.* usb_state_type usb_get_state ()
- *Query the current state of the USB connection.* void usb_handle_class_ctrl_read_callback ()
- *Callback routine for a class control read.* void usb_handle_class_ctrl_write_callback (uns8 *data, uns16 count)
- *Callback routine for a class control write.* void usb_handle_class_request_callback (setup_data_packet sdp)
- *Callback routine for a control transfer request that is placed on the class.* void usb_handle_isr ()
- *Handle USB interrupts.* void usb_send_data (uns8 ep, uns8 *data, uns8 send_count, bit first)
- *Send data over an endpoint pipe.* void usb_send_empty_data_pkt ()
- *Send an empty data packet.* void usb_setup ()
- *Setup USB hardware ready for use.* void usb_SOF_callback (uns16 frame)
- *Callback routine triggered each time a start of frame (SOF) has been received.* void usb_stall_ep0 ()

## *Send a stall on control transfer endpoint.* Variables

- control_mode_type control_mode
- uns8 usb_address
- setup_data_packet usb_sdp
- usb_state_type usb_state

---

## Detailed Description

Put the following in your config.h

- - - - - - - - - - - - - - - - - - - pic_usb defines
- - - - - - - - - - - - - - - - - - -

Use this define if you would like to get USB negotiation and data transfer information out the serial (UART) port. You'll also need to include pic_serial.c in your project. define USB_DEBUG

Use this define if you would like a high level (ie, lots) of USB debug information printed out the serial (UART) port. define USB_DEBUG_HIGH

Define the highest numbered endpoint you will use (in this case, we choose 3). define USB_HIGHEST_EP 3

Define either USB_SELF_POWERED or USB_BUS_POWERED define USB_SELF_POWERED define USB_BUS_POWERED

Define your endpoint buffers. These start at 0x500 for a 18f4550. You'll always need endpoint 0, which is the control transfer endpoint, and others as well depending on your use. You don't have to declare the endpoints you don't use, even if they're not sequential.

define USB_EP0_OUT_SIZE 8 define USB_EP0_OUT_ADDR 0x0200

define USB_EP0_IN_SIZE 8 define USB_EP0_IN_ADDR 0x0208

EP1 not used

define USB_EP2_IN_SIZE 8 define USB_EP2_IN_ADDR 0x0210

define USB_EP3_OUT_SIZE 8 define USB_EP3_OUT_ADDR 0x0218 define USB_EP3_IN_SIZE 8 define USB_EP3_IN_ADDR 0x0220

Use this define if you want to get a callback each SOF (Start Of Frame), generally every 1ms define USB_CALLBACK_ON_SOF if you define it, you'll need to include this routine in your code: void usb_SOF_callback(uns16 frame) { }

Use this define if you would like to know when your device has been configured and is ready for use define USB_CALLBACK_ON_DEVICE_CONFIGURED if you define it, you'll need to include this routine in your code: void usb_device_configured_callback() { }

Use this define if your device uses class control transfers, eg, CDC (virtual serial port) is one that does define USB_CALLBACK_ON_CTRL_CLASS if you define it, you'll need to include these routines in your code: void usb_handle_class_ctrl_read_callback(); void usb_handle_class_ctrl_write_callback(uns8 *data, uns16 count); void usb_handle_class_request_callback(setup_data_packet sdp);

Use this define if you would like to get notified when data has arrived or been succesfully sent define USB_EP_DATA_CALLBACK if you define it, you'll need to include these routines in your code: void usb_ep_data_out_callback(uns8 end_point, uns8 *buffer_location, uns16 byte_count); void usb_ep_data_in_callback(uns8 end_point, uns16 byte_count);

Put the following in your ISR

usb_handle_isr();

Put the following in your system setup routine

Setup USB usb_setup();

Turn on USB interrupts void turn_usb_ints_on();

Turn on global interrupts turn_global_ints_on();

When you're ready to start the USB subsystem and negotiate address, send descriptors etc, call

usb_enable_module();

Definition in file pic_usb.h.

---

## Define Documentation

### #define BC8 0

Definition at line 198 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), and usb_stall_on_in().

**#define BC9 1**

Definition at line 197 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), and usb_stall_on_in().

**#define BSTALL 2**

Definition at line 196 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_stall(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), usb_stall_ep0(), and usb_stall_on_in().

**#define DATA_STAGE_DIR 7**

Definition at line 178 of file pic_usb.h.

Referenced by usb_handle_transaction().

**#define dt_CONFIGURATION 0x02**

Definition at line 278 of file pic_usb.h.

**#define dt_CS_INTERFACE 0x24**

Definition at line 291 of file pic_usb.h.

**#define dt_DEBUG 0x0a**

Definition at line 286 of file pic_usb.h.

**#define dt_DEVICE 0x01**

Definition at line 277 of file pic_usb.h.

**#define dt_DEVICE_QUALIFIER 0x06**

Definition at line 282 of file pic_usb.h.

**#define dt_ENDPOINT 0x05**

Definition at line 281 of file pic_usb.h.

**#define dt_HID 0x21**

Definition at line 288 of file pic_usb.h.

**#define dt_HID_REPORT 0x22**

Definition at line 289 of file pic_usb.h.

**#define dt_INTERFACE 0x04**

Definition at line 280 of file pic_usb.h.

**#define dt_INTERFACE_ASSOC 0x0b**

Definition at line 287 of file pic_usb.h.

**#define dt_INTERFACE_POWER 0x08**

Definition at line 284 of file pic_usb.h.

**#define dt_OTG 0x09**

Definition at line 285 of file pic_usb.h.

**#define dt_OTHER_SPEED_CONFIG 0x07**

Definition at line 283 of file pic_usb.h.

**#define dt_STRING 0x03**

Definition at line 279 of file pic_usb.h.

**#define DTS 6**

Definition at line 192 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_standard_request(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), usb_send_empty_data_pkt(), usb_send_one_byte(), and usb_stall_on_in().

**#define DTSEN 3**

Definition at line 195 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), and usb_stall_on_in().

**#define INCDIS 4**

Definition at line 194 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), and usb_stall_on_in().

**#define KEN 5**

Definition at line 193 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), and usb_stall_on_in().

**#define PID0 2**

Definition at line 206 of file pic_usb.h.

**#define PID1 3**

Definition at line 205 of file pic_usb.h.

**#define PID2 4**

Definition at line 204 of file pic_usb.h.

**#define PID3 5**

Definition at line 203 of file pic_usb.h.

**#define pid_ACK 0b00000010**

Definition at line 255 of file pic_usb.h.

Referenced by usb_handle_transaction().

**#define pid_DATA0 0b00000011**

Definition at line 248 of file pic_usb.h.

**#define pid_DATA1 0b00001011**

Definition at line 249 of file pic_usb.h.

**#define pid_DATA2 0b00000111**

Definition at line 250 of file pic_usb.h.

**#define pid_IN 0b00001001**

Definition at line 242 of file pic_usb.h.
Referenced by usb_handle_transaction().

**#define pid_MDATA 0b00001111**

Definition at line 251 of file pic_usb.h.

**#define pid_NAK 0b00001010**

Definition at line 256 of file pic_usb.h.

**#define pid_NYET 0b00000110**

Definition at line 258 of file pic_usb.h.

**#define pid_OUT 0b00000001**

Definition at line 241 of file pic_usb.h.
Referenced by usb_handle_transaction().

**#define pid_SETUP 0b00001101**

Definition at line 244 of file pic_usb.h.
Referenced by usb_handle_transaction().

**#define pid_SOF 0b00000101**

Definition at line 243 of file pic_usb.h.

**#define pid_STALL 0b00001110**

Definition at line 257 of file pic_usb.h.

**#define req_Clear_Feature 0x01**

Definition at line 264 of file pic_usb.h.

**#define req_Get_Configuration 0x08**

Definition at line 269 of file pic_usb.h.

**#define req_Get_Descriptor 0x06**

Definition at line 267 of file pic_usb.h.
Referenced by usb_handle_standard_request().

**#define req_Get_Interface 0x0a**

Definition at line 271 of file pic_usb.h.
Referenced by usb_handle_standard_request().

**#define req_Get_Status 0x00**

Definition at line 263 of file pic_usb.h.
Referenced by usb_handle_standard_request().

**#define req_Set_Address 0x05**

Definition at line 266 of file pic_usb.h.
Referenced by usb_handle_standard_request().

**#define req_Set_Configuration 0x09**

Definition at line 270 of file pic_usb.h.
Referenced by usb_handle_standard_request().

**#define req_Set_Descriptor 0x07**

Definition at line 268 of file pic_usb.h.

**#define req_Set_Feature 0x03**

Definition at line 265 of file pic_usb.h.

**#define req_Set_Interface 0x0b**

Definition at line 272 of file pic_usb.h.

**#define req_Synch_Frame 0x0c**

Definition at line 273 of file pic_usb.h.

**#define REQUEST_TYPE0 5**

Definition at line 181 of file pic_usb.h.

Referenced by usb_handle_transaction().

#### #define REQUEST_TYPE1 6

Definition at line 180 of file pic_usb.h.

Referenced by usb_handle_transaction().

#### #define UOWN 7

Definition at line 191 of file pic_usb.h.

Referenced by usb_cdc_handle_tx(), usb_configure_endpoints(), usb_handle_reset(), usb_handle_stall(), usb_handle_transaction(), usb_prime_ep0_out_e(), usb_prime_ep0_out_o(), usb_send_data(), usb_send_data_chunk(), usb_stall_ep0(), and usb_stall_on_in().

#### #define usb_send_status_ack() usb_send_empty_data_pkt()

Send a status acknowledge by sending an empty data packet

Definition at line 397 of file pic_usb.h.

Referenced by usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), and usb_handle_standard_request().

---

## Enumeration Type Documentation

#### enum control_mode_type

Describe the state of the control transfer

**Enumerator:**

***cm_IDLE*** No control transfer taking place

***cm_CTRL_WRITE_DATA_STAGE*** Device receiving data during the data stage

***cm_CTRL_WRITE_DATA_STAGE_CLASS*** Device receiving data during the data stage destined for the class

***cm_CTRL_READ_DATA_STAGE*** Device sending data during the data stage

***cm_CTRL_READ_DATA_STAGE_CLASS*** Device class is sending data during the data stage

***cm_CTRL_READ_AWAITING_STATUS*** Device is awaiting reception of status after sending data

***cm_CTRL_WRITE_SENDING_STATUS*** Device is sending status after receiving data

Definition at line 211 of file pic_usb.h.

#### enum usb_state_type

Handle the different states that USB device can be in

**Enumerator:**

***st_POWERED*** USB device is powered up, ready to start negotiating

***st_DEFAULT*** USB device is now negotiating

***st_ADDRESS*** USB device now has an address

***st_CONFIGURED*** USB device is completely configured and ready to rock and roll

Definition at line 140 of file pic_usb.h.

**enum usb_status_type**

Handle the special case of when we send a status ack and THEN change the address. So we need to know that the USB device is in that micro state (ie, received the new address but not yet sent the status

**Enumerator:**

***us_IDLE***
***us_SET_ADDRESS***

Definition at line 232 of file pic_usb.h.

---

## Function Documentation

### void turn_usb_ints_on ()

If you are using interrupt-driven code (generally the best way of doing things) you can turn on USB interrupts using turn_usb_ints_on(). Don't forget that you will also need to call turn_global_ints_on() as well. Typically this is called in your system setup routine.

Definition at line 962 of file pic_usb.c.

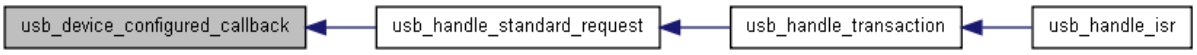
### void usb_device_configured_callback ()

Once descriptors have been received by the host and the host has selected a configuration to use, this routine is triggered. Typically this means that negotiations have completed successfully and an appropriate driver has been loaded.

In order for this callback to be triggered, you must define USB_CALLBACK_ON_DEVICE_CONFIGURED in your config.h

Referenced by usb_handle_standard_request().

Here is the caller graph for this function:




### void usb_enable_module ()

After you've called usb_setup(), you can call usb_enable_module() whenever you're ready for USB negotiations to occur. Normally, this would need to occur relatively quickly after power-up if your PIC is powered by USB and it's purpose is to talk over USB. This is normally called from your main() routine once all other configuration is done.

Once the USB module has successfully negotiated a connection with the host, usb_device_configured_callback() will be called if you have requested this in your config.h file. This will indicate a successful connection. Because of the way USB works, there is no way to tell that it

*hasn't* worked, except via a timer - if you haven't had a good connection in several seconds, you can assume it has failed (although this may just mean the user is hunting for a driver disk etc).

Definition at line 1033 of file pic_usb.c.

References st_DEFAULT, and usb_state.

### void usb_ep_data_in_callback (uns8 *end_point*, uns16 *byte_count*)

If you have called usb_send_data to transfer data to the host, this routine will be fired once this data has been transferred. You may send more data by using usb_send_data(). Since the current PicPack USB library supports only single buffering, transfer speed is limited by how quickly you can refill the buffer again. In the future, we may support double buffering (ping poing buffering) which will most likely improve transfer speeds (to be fair, transfer performance has not been a limiting factor in tests so far).

In order for this callback to be triggered, you must define USB_EP_DATA_CALLBACK in your config.h

**Parameters:**

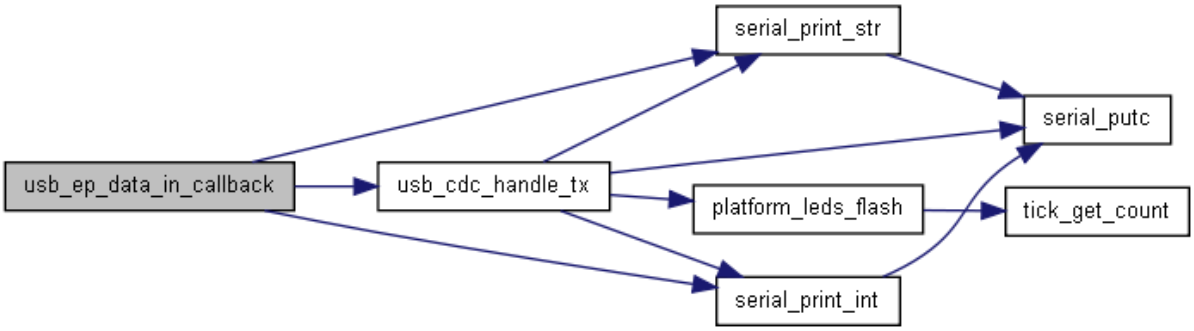
*end_point* The endpoint on which the data was transferred
*byte_count* The number of bytes that were actually transferred

Definition at line 323 of file usb_cdc_class.c.

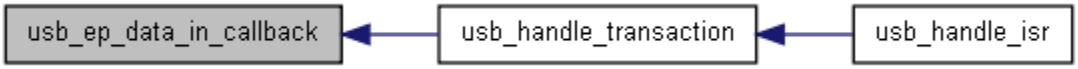
References serial_print_int(), serial_print_str(), and usb_cdc_handle_tx().

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




### void usb_ep_data_out_callback (uns8 *end_point*, uns8 * *buffer_location*, uns16 *byte_count*)

If data is sent to the device and the endpoint is not endpoint 0 (the control transfer endpoint) then this routine is called. Since the routine is passed the actual hardware buffer location, it is important to pull data out of the buffer as soon as possible in order to free up the buffer to receive more data. The buffer is re-primed only once this routine completes since PicPack only supports single-buffered mode. In the future, we may look at supporting double buffering (ping-pong buffering) in order to be able to receive more data even while this routine is being called.

In order for this callback to be triggered, you must define USB_EP_DATA_CALLBACK in your config.h

**Parameters:**

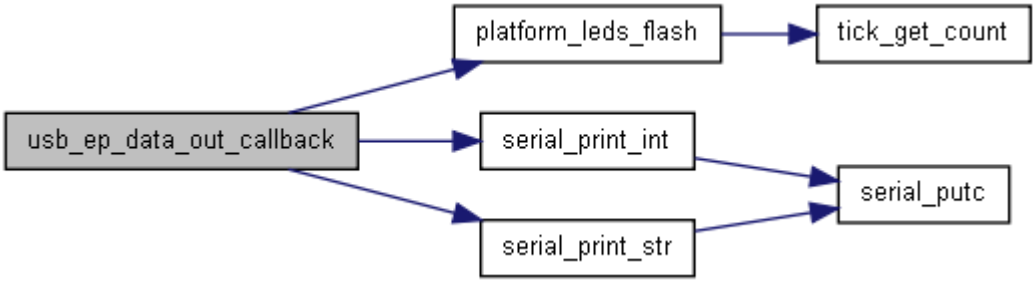
*end_point* The endpoint the data was sent do
*buffer_lcoation* The memory location of the USB buffer where the data was received into
*byte_count* The number of bytes received

Definition at line 283 of file usb_cdc_class.c.

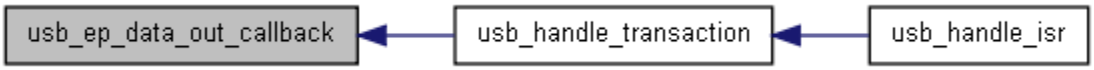
References cdc_rx_buffer, cdc_rx_end, cdc_rx_start, platform_leds_flash(), serial_print_int(), serial_print_str(), and uns8.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




**void usb_get_descriptor_callback (uns8 *descriptor_type*, uns8 *descriptor_num*, uns8 ** *rtn_descriptor_ptr*, uns16 * *rtn_descriptor_size*)**

Once negotiations start, descriptors are quested by the host. The device must be able to respond to these requests. Typically, this routine consists of a switch statement depending on the descriptor_type parameter. The descriptor_num is used to specify which of the descriptor_type descriptors are required, since they may be several (for example, string descriptors).

Since descriptors are specific to a particular device (and project), this callback routine and the associated descriptors are put in a file called usb_config_xxxx.c and placed in the project workspace. This is because while the descriptors could have been provided as part of the PicPack library, you will almost always want to change them to suit your application, even if only for changing the vendor and device IDs and serial numbers.

At present, descriptors are required to be in RAM.

Since descriptor requests are an essential part of the USB protocol, this callback routine is mandatory.

Referenced by usb_handle_standard_request().

Here is the caller graph for this function:


**usb_state_type usb_get_state ()**

Returns the USB state, either powered, default, address or connect. This is updated as the negotiation progresses. It may be useful to query this if, after a suitable time-out, the connection has not been made.

Definition at line 1039 of file pic_usb.c.

References usb_state.

### void usb_handle_class_ctrl_read_callback ()

When a control transfer is taking place, this routine is called to indicate that a control read for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been transferred to the host. If you wish to send more data to the host, use usb_send_data(), or if your control read has sent all the data required, you will need to indicate that the state has changed by setting the control_mode variable to cm_CTRL_READ_AWAITING_STATUS. This will indicate to the stack that it should now wait for the status packet before completing the control transfer.

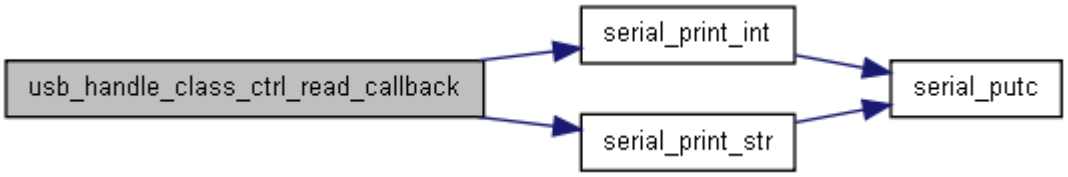
To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 268 of file usb_cdc_class.c.

References setup_data_packet::bRequest, cm_CTRL_READ_AWAITING_STATUS, control_mode, req_GET_LINE_CODING, serial_print_int(), serial_print_str(), and usb_sdp.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:


### void usb_handle_class_ctrl_write_callback (uns8 * *data*, uns16 *count*)

When a control transfer is taking place, this routine is called to indicate that a control write for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been received by the device. If you expect more data from the host, it will arrive in due course since endpoint 0 will be primed for more data automatically. If you have received all the data from the host, you will need to set the control_mode state variable to cm_CTRL_WRITE_SENDING_STATUS and then actually send the status by calling usb_send_status_ack(). Once the status has actually been sent, the control_mode state will automatically change to cm_IDLE to indicate the transfer has completed.

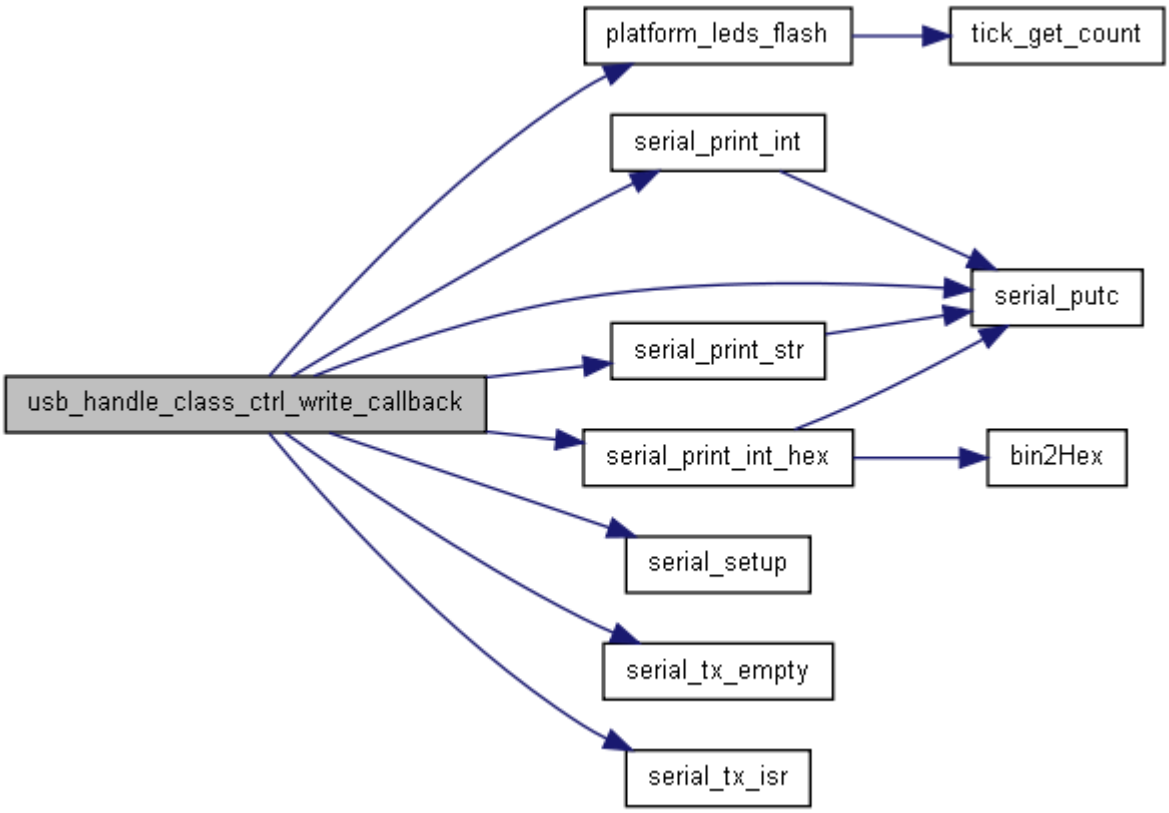
To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 168 of file usb_cdc_class.c.

References long_union::as_byte_array, long_union::as_long, setup_data_packet::bRequest, class_data, cm_CTRL_WRITE_SENDING_STATUS, control_mode, current_bit_rate, line_coding::data_bits, line_coding::dte_rate, line_coding::parity, platform_leds_flash(), req_SET_LINE_CODING, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), serial_setup(), serial_tx_empty(), serial_tx_isr(), line_coding::stop_bits, usb_sdp, and usb_send_status_ack.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:


**void usb_handle_class_request_callback (setup_data_packet *sdp*)**

After receiving a setup packet, where the request is placed on the class, this routine is called. In usb_handle_class_request_callback, you can set up ready for the data stage of the control transfer. The direction of the data stage can be determined by examining test_bit(sdp.bRequest, DATA_STAGE_DIR) although generally it appears to be obvious from the request. The request is stored in sdp.bRequest.

Typically, if it is a control read transfer (that is, it is a request by the host for data), then you will need to move the control_mode state variable to cm_CTRL_READ_DATA_STAGE_CLASS and send data using usb_send_data(). If you only intend to send one packet, you can immediately move the control_mode state variable to cm_CTRL_READ_AWAITING_STATUS to indicate you are waiting for the status to arrive. You could wait for the usb_handle_class_ctrl_read callback and do it (move to cm_CTROL_READ_AWAITING_STATUS) but the PicPack USB stack can handle the control read event for you if you've already switched states.

If it is a control write transfer (that is, it is a request by the host to send data to the device), then you will need to move the control_mode state variable to cm_CTRL_WRITE_DATA_STAGE_CLASS. Then, the usb_handle_class_ctrl_write will be fired when data is received by the device in the data stage.

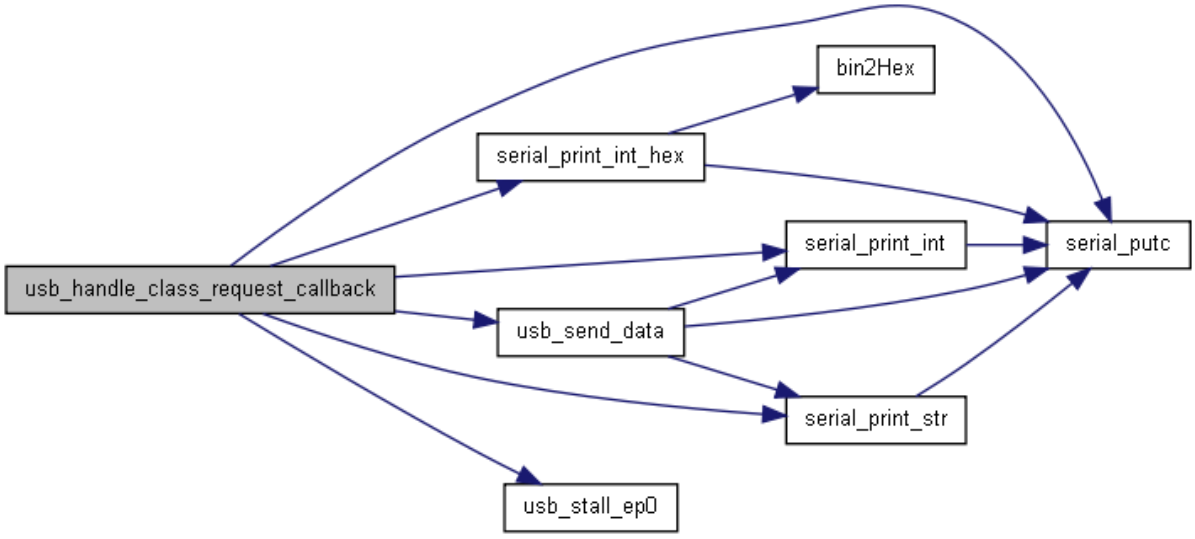
To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 112 of file usb_cdc_class.c.

References long_union::as_byte_array, setup_data_packet::bRequest, cm_CTRL_READ_AWAITING_STATUS, cm_CTRL_READ_DATA_STAGE_CLASS, cm_CTRL_WRITE_DATA_STAGE_CLASS, cm_CTRL_WRITE_SENDING_STATUS, control_mode, line_coding::data_bits, line_coding::dte_rate, line_coding::parity, req_GET_IDLE, req_GET_LINE_CODING, req_GET_PROTOCOL, req_GET_REPORT, req_SET_CONTROL_LINE_STATE, req_SET_IDLE, req_SET_LINE_CODING, req_SET_PROTOCOL, req_SET_REPORT, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), line_coding::stop_bits, uns8, usb_send_data(), usb_send_status_ack, usb_stall_ep0(), setup_data_packet::wLength, and setup_data_packet::wValue.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:


### void usb_handle_isr ()

usb_handle_isr() should be inserted in your interrupt service routine. Alternatively, if you have reason not to want to do interrupt-driven USB, for example, a bootloader, you can poll this routine.

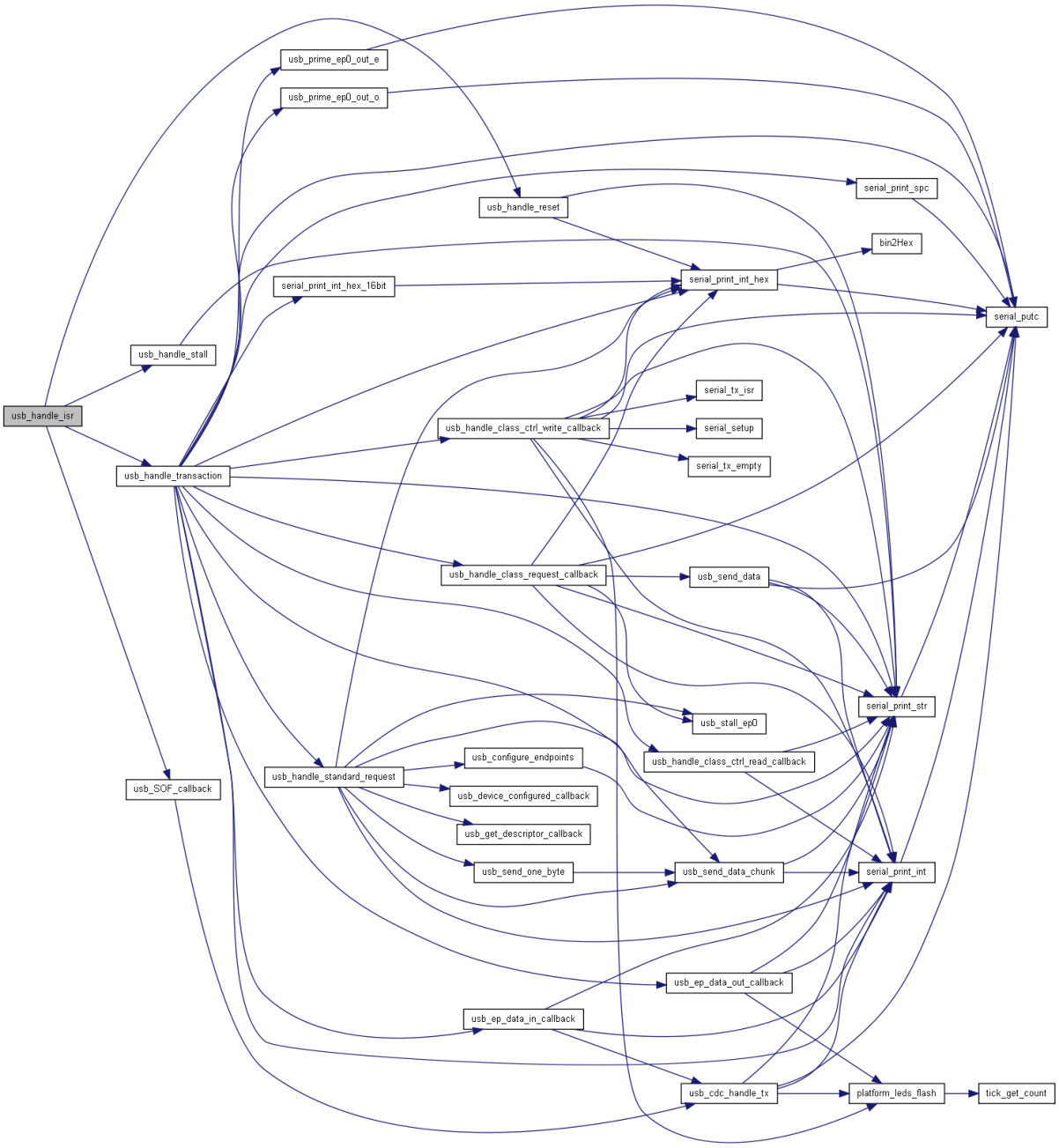
Make sure you call turn_usb_ints() and turn_global_ints_on() to ensure interrupts occur.

It will check for any of the USB interrupt flags and handle: USB transactions, USB reset, USB stall, USB Start Of Frame (including calling usb_SOF_callback() if configured in your config.h and most importantly USB transaction, which is where all the hard work is done.

Definition at line 928 of file pic_usb.c.

References uns8, usb_handle_reset(), usb_handle_stall(), usb_handle_transaction(), and usb_SOF_callback().

Here is the call graph for this function:




**void usb_send_data (uns8 *ep*, uns8 * *data*, uns8 *send_count*, bit *first*)**

Use this routine to send data across the USB pipe.

**Parameters:**

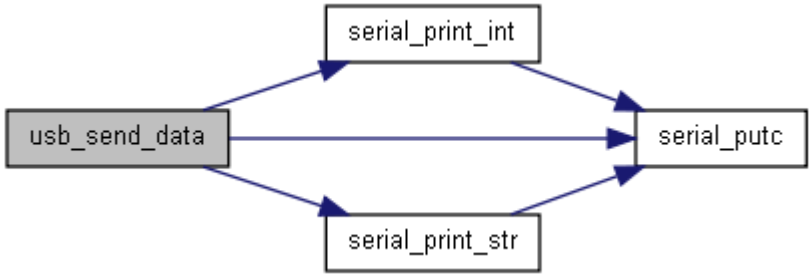
*ep* Endpoint that the data should be sent from
*data* pointer to the data
*send_count* the number of bytes to send
*first* True if this is the first in a series of sends. Generally, this can be set to False, since it will automatically be set to the right value on endpoint creation. However, in the case of control transfers, the data stage needs to have the first parameter set to True to ensure the DTS bit is set correctly.

Definition at line 211 of file pic_usb.c.

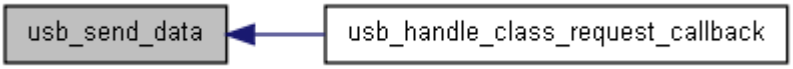
References buffer_descriptor::addr, BC8, BC9, BSTALL, buffer_descriptor::count, DTS, DTSEN, ep_in_bd_location, ep_in_buffer_location, INCDIS, KEN, serial_print_int(), serial_print_str(), serial_putc(), buffer_descriptor::stat, uns16, uns8, and UOWN.

Referenced by usb_handle_class_request_callback().

Here is the call graph for this function:




Here is the caller graph for this function:



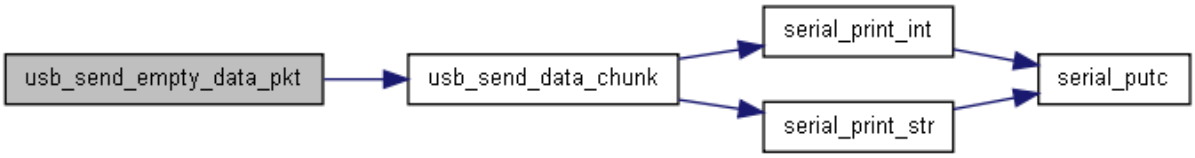

### void usb_send_empty_data_pkt ()

Use this routine to send an data across the USB pipe on endpoint 0. This is the equivalent of sending a status acknowledge.

Definition at line 341 of file pic_usb.c.

References bd0in, buffer_0_in, delivery_buffer, delivery_buffer_size, delivery_bytes_max_send, delivery_bytes_sent, delivery_bytes_to_send, delivery_ptr, DTS, buffer_descriptor::stat, uns8, and usb_send_data_chunk().

Here is the call graph for this function:




### void usb_setup ()

usb_setup() configures the PIC USB hardware ready for use and prepares the internal data structures used to keep track of where the endpoint buffers are.

After calling usb_setup(), you are ready to call usb_enable_module() to actually start USB negotiations. Ensure that you have usb_handle_isr() in your interrupt service routine.

Definition at line 973 of file pic_usb.c.

References bd0in, bd0out_e, bd1in, bd1out, bd2in, bd2out, bd3in, bd3out, bd4in, bd4out, ep_in_bd_location, ep_out_bd_location, st_POWERED, and usb_state.

### void usb_SOF_callback (uns16 *frame*)

Frames in USB occur each 1ms. A SOF packet is sent to each device at the start of each frame. This is a really neat way of getting a 1ms timer without any further work.

**Parameters:**

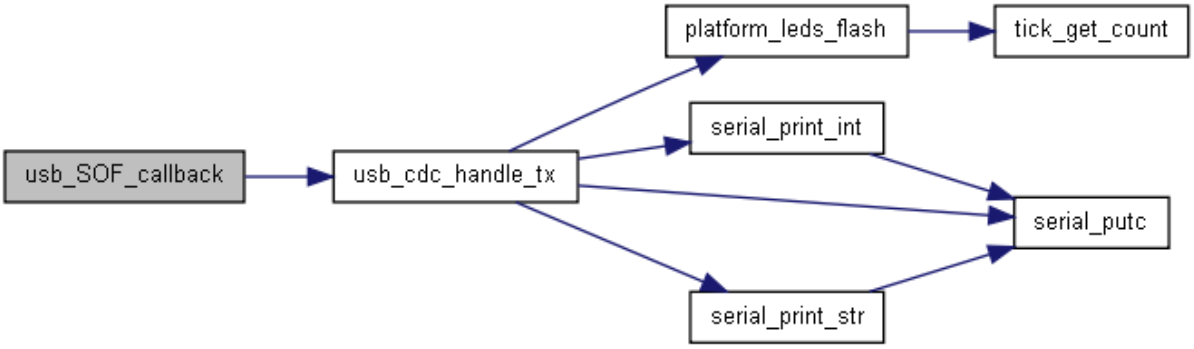
*frame* The frame number. Frames will wrap at 65535.

Definition at line 478 of file usb_cdc_class.c.

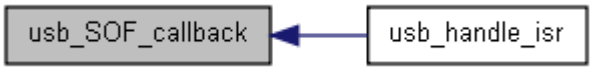
References usb_cdc_handle_tx().

Referenced by usb_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:




### void usb_stall_ep0 ()

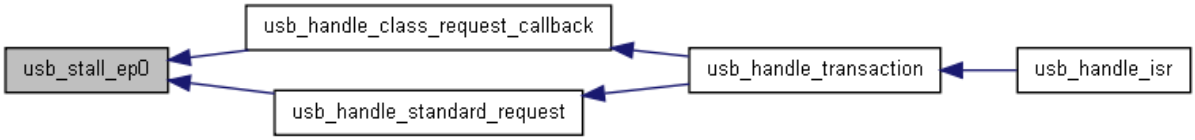
Use this routine to send a stall on the control transfer endpoint - usually used to indicate that the requested function is not available.

Definition at line 203 of file pic_usb.c.

References bd0in, BSTALL, buffer_descriptor::stat, and UOWN.

Referenced by usb_handle_class_request_callback(), and usb_handle_standard_request().

Here is the caller graph for this function:




---

## Variable Documentation

### control_mode_type control_mode

Store the control mode state

Definition at line 55 of file pic_usb.c.

Referenced by usb_handle_class_ctrl_read_callback(), usb_handle_class_ctrl_write_callback(), usb_handle_class_request_callback(), usb_handle_reset(), usb_handle_standard_request(), usb_handle_transaction(), and usb_send_data_chunk().

**uns8 usb_address**

Store the usb address

Definition at line 54 of file pic_usb.c.

Referenced by usb_handle_reset(), usb_handle_standard_request(), and usb_handle_transaction().

**setup_data_packet usb_sdp**

Store the last setup data packet

Definition at line 52 of file pic_usb.c.

Referenced by usb_handle_class_ctrl_read_callback(), and usb_handle_class_ctrl_write_callback().

**usb_state_type usb_state**

Store the current USB device state

Definition at line 50 of file pic_usb.c.

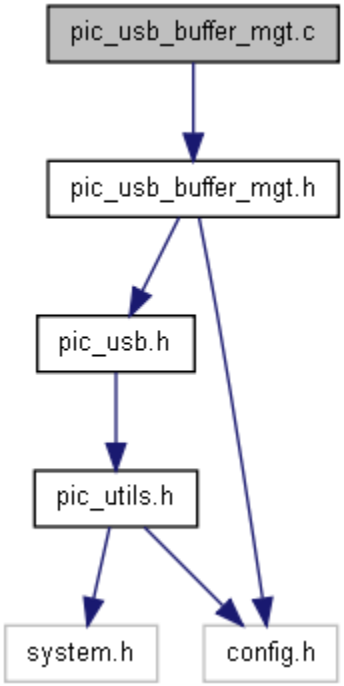
Referenced by usb_enable_module(), usb_get_state(), usb_handle_standard_request(), usb_handle_transaction(), and usb_setup().

---

# pic_usb_buffer_mgt.c File Reference

```
#include "pic_usb_buffer_mgt.h"
```

Include dependency graph for pic_usb_buffer_mgt.c:




## Variables

- buffer_descriptor bd0in

- buffer_descriptor bd0out_e
- buffer_descriptor bd0out_o
- buffer_descriptor bd1in
- buffer_descriptor bd1out
- buffer_descriptor bd2in
- buffer_descriptor bd2out
- buffer_descriptor bd3in
- buffer_descriptor bd3out
- buffer_descriptor bd4in
- buffer_descriptor bd4out
- buffer_descriptor bd5in
- buffer_descriptor bd5out
- buffer_descriptor bd6in
- buffer_descriptor bd6out
- buffer_descriptor bd7in
- buffer_descriptor bd7out
- buffer_descriptor * ep_in_bd_location [USB_HIGHEST_EP+1]
- uns8 * ep_in_buffer_location [USB_HIGHEST_EP+1]
- uns16 ep_in_buffer_size [USB_HIGHEST_EP+1]
- buffer_descriptor * ep_out_bd_location [USB_HIGHEST_EP+1]
- uns8 * ep_out_buffer_location [USB_HIGHEST_EP+1]
- uns16 ep_out_buffer_size [USB_HIGHEST_EP+1]
- uns8 buffer_0_in[USB_EP0_IN_SIZE] USB_EP0_IN_ADDR
- uns8 buffer_0_out_e[USB_EP0_OUT_E_SIZE] USB_EP0_OUT_E_ADDR
- uns8 buffer_0_out_o[USB_EP0_OUT_O_SIZE] USB_EP0_OUT_O_ADDR

---

## Variable Documentation

### buffer_descriptor bd0in

Definition at line 42 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_stall(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data_chunk(), usb_send_empty_data_pkt(), usb_send_one_byte(), usb_setup(), usb_stall_ep0(), and usb_stall_on_in().

### buffer_descriptor bd0out_e

Definition at line 40 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), and usb_setup().

### buffer_descriptor bd0out_o

Definition at line 41 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_transaction(), and usb_prime_ep0_out_o().

### buffer_descriptor bd1in

Definition at line 44 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd1out**

Definition at line 43 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd2in**

Definition at line 46 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd2out**

Definition at line 45 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd3in**

Definition at line 48 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd3out**

Definition at line 47 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd4in**

Definition at line 50 of file pic_usb_buffer_mgt.c.
Referenced by usb_setup().

**buffer_descriptor bd4out**

Definition at line 49 of file pic_usb_buffer_mgt.c.
Referenced by usb_setup().

**buffer_descriptor bd5in**

Definition at line 52 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd5out**

Definition at line 51 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd6in**

Definition at line 54 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd6out**

Definition at line 53 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd7in**

Definition at line 56 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd7out**

Definition at line 55 of file pic_usb_buffer_mgt.c.

**buffer_descriptor* ep_in_bd_location[USB_HIGHEST_EP+1]**

Definition at line 101 of file pic_usb_buffer_mgt.c.
Referenced by usb_cdc_handle_tx(), usb_handle_transaction(), usb_send_data(), and usb_setup().

**uns8* ep_in_buffer_location[USB_HIGHEST_EP+1]**

Definition at line 104 of file pic_usb_buffer_mgt.c.
Referenced by usb_cdc_handle_tx(), and usb_send_data().

**uns16 ep_in_buffer_size[USB_HIGHEST_EP+1]**

Definition at line 181 of file pic_usb_buffer_mgt.c.
Referenced by usb_cdc_handle_tx().

**buffer_descriptor* ep_out_bd_location[USB_HIGHEST_EP+1]**

Definition at line 102 of file pic_usb_buffer_mgt.c.
Referenced by usb_handle_transaction(), and usb_setup().

**uns8* ep_out_buffer_location[USB_HIGHEST_EP+1]**

Definition at line 142 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_transaction().

**uns16 ep_out_buffer_size[USB_HIGHEST_EP+1]**

Definition at line 219 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_transaction().

**uns8 buffer_0_in [USB_EP0_IN_SIZE] USB_EP0_IN_ADDR**

Definition at line 80 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), and usb_handle_standard_request().

**uns8 buffer_0_out_e [USB_EP0_OUT_E_SIZE] USB_EP0_OUT_E_ADDR**

Definition at line 77 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), and usb_prime_ep0_out_e().

**uns8 buffer_0_out_o [USB_EP0_OUT_O_SIZE] USB_EP0_OUT_O_ADDR**

Definition at line 78 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), and usb_prime_ep0_out_o().

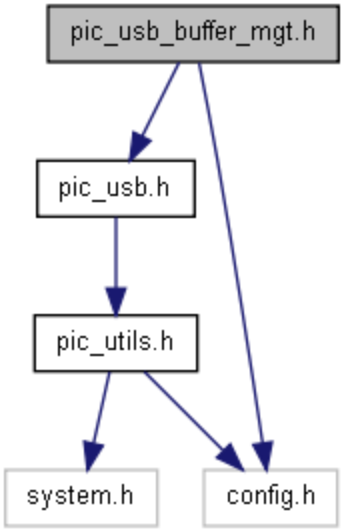
---

## pic_usb_buffer_mgt.h File Reference

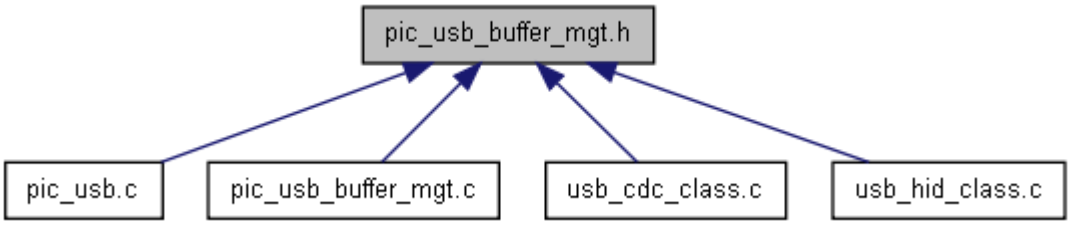
Pic USB buffer routines.

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

Include dependency graph for pic_usb_buffer_mgt.h:




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




## Defines

- #define __pic_ubs_buffer_mgt_h

## Variables

- buffer_descriptor bd0in
- buffer_descriptor bd0out_e
- buffer_descriptor bd0out_o
- buffer_descriptor bd1in
- buffer_descriptor bd1out
- buffer_descriptor bd2in
- buffer_descriptor bd2out
- buffer_descriptor bd3in
- buffer_descriptor bd3out
- buffer_descriptor bd4in
- buffer_descriptor bd4out
- buffer_descriptor bd5in
- buffer_descriptor bd5out
- buffer_descriptor bd6in
- buffer_descriptor bd6out
- buffer_descriptor bd7in
- buffer_descriptor bd7out
- uns8 buffer_0_in [USB_EP0_IN_SIZE]
- uns8 buffer_0_out_e [USB_EP0_OUT_E_SIZE]

- uns8 buffer_0_out_o [USB_EP0_OUT_O_SIZE]
- buffer_descriptor * ep_in_bd_location [USB_HIGHEST_EP+1]
- uns8 * ep_in_buffer_location [USB_HIGHEST_EP+1]
- uns16 ep_in_buffer_size [USB_HIGHEST_EP+1]
- buffer_descriptor * ep_out_bd_location [USB_HIGHEST_EP+1]
- uns8 * ep_out_buffer_location [USB_HIGHEST_EP+1]
- uns16 ep_out_buffer_size [USB_HIGHEST_EP+1]

---

## Detailed Description

Buffer data structures for USB transfers

Definition in file pic_usb_buffer_mgt.h.

---

## Define Documentation

### #define __pic_ubs_buffer_mgt_h

Definition at line 46 of file pic_usb_buffer_mgt.h.

---

## Variable Documentation

### buffer_descriptor bd0in

Definition at line 42 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_stall(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data_chunk(), usb_send_empty_data_pkt(), usb_send_one_byte(), usb_setup(), usb_stall_ep0(), and usb_stall_on_in().

### buffer_descriptor bd0out_e

Definition at line 40 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_transaction(), usb_prime_ep0_out_e(), and usb_setup().

### buffer_descriptor bd0out_o

Definition at line 41 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_reset(), usb_handle_transaction(), and usb_prime_ep0_out_o().

### buffer_descriptor bd1in

Definition at line 44 of file pic_usb_buffer_mgt.c.

Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd1out**

Definition at line 43 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd2in**

Definition at line 46 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd2out**

Definition at line 45 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd3in**

Definition at line 48 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd3out**

Definition at line 47 of file pic_usb_buffer_mgt.c.
Referenced by usb_configure_endpoints(), and usb_setup().

**buffer_descriptor bd4in**

Definition at line 50 of file pic_usb_buffer_mgt.c.
Referenced by usb_setup().

**buffer_descriptor bd4out**

Definition at line 49 of file pic_usb_buffer_mgt.c.
Referenced by usb_setup().

**buffer_descriptor bd5in**

Definition at line 52 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd5out**

Definition at line 51 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd6in**

Definition at line 54 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd6out**

Definition at line 53 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd7in**

Definition at line 56 of file pic_usb_buffer_mgt.c.

**buffer_descriptor bd7out**

Definition at line 55 of file pic_usb_buffer_mgt.c.

**uns8 buffer_0_in[USB_EP0_IN_SIZE]**

Referenced by usb_handle_transaction(), usb_send_data_chunk(), usb_send_empty_data_pkt(), and usb_send_one_byte().

**uns8 buffer_0_out_e[USB_EP0_OUT_E_SIZE]**

Referenced by usb_handle_transaction().

**uns8 buffer_0_out_o[USB_EP0_OUT_O_SIZE]**

Referenced by usb_handle_transaction().

**buffer_descriptor* ep_in_bd_location[USB_HIGHEST_EP+1]**

Definition at line 101 of file pic_usb_buffer_mgt.c.

Referenced by usb_cdc_handle_tx(), usb_handle_transaction(), usb_send_data(), and usb_setup().

**uns8* ep_in_buffer_location[USB_HIGHEST_EP+1]**

Definition at line 104 of file pic_usb_buffer_mgt.c.

Referenced by usb_cdc_handle_tx(), and usb_send_data().

**uns16 ep_in_buffer_size[USB_HIGHEST_EP+1]**

Definition at line 181 of file pic_usb_buffer_mgt.c.

Referenced by usb_cdc_handle_tx().

**buffer_descriptor* ep_out_bd_location[USB_HIGHEST_EP+1]**

Definition at line 102 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_transaction(), and usb_setup().

**uns8* ep_out_buffer_location[USB_HIGHEST_EP+1]**

Definition at line 142 of file pic_usb_buffer_mgt.c.

Referenced by usb_handle_transaction().

**uns16 ep_out_buffer_size[USB_HIGHEST_EP+1]**

Definition at line 219 of file pic_usb_buffer_mgt.c.

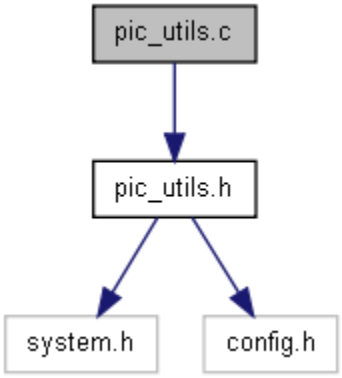
Referenced by usb_handle_transaction().

---

## pic_utils.c File Reference

```
#include "pic_utils.h"
```

Include dependency graph for pic_utils.c:



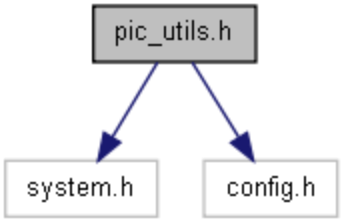

---

## pic_utils.h File Reference

Generic PIC helper routines.

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

Include dependency graph for pic_utils.h:




## Defines

- #define change_pin(port, pin, value)
- #define change_pin_var(port, pin, value) change_pin(port, pin, value)
- #define clear_pin(port, pin) clear_bit(port_array[port - PORTA], pin);
- #define clear_pin_var(port, pin) clear_pin(port, pin)
- #define end_crit_sec() intcon.GIE = store_gie
- #define int16 int
- #define int32 long
- #define int8 char
- #define kill_interrupts()
- #define MAGIC_BOOSTBLOADER_REQUEST 4
- #define make_input(port, pin) set_bit(tris_array[port - PORTA], pin)
- #define make_output(port, pin) clear_bit(tris_array[port - PORTA], pin)
- #define NUMBER_PORTS 1
- #define set_pin(port, pin) set_bit(port_array[port - PORTA], pin);
- #define set_pin_var(port, pin) set_pin(port, pin)
- #define start_crit_sec()
- #define test_output_pin(port, pin) ((port_array[port - PORTA] & (1 << pin)) != 0)
- #define test_pin(port, pin) ((port_in_array[port - PORTA] & (1 << pin)) != 0)
- #define test_pin_var(port, pin) test_pin(port, pin)
- #define toggle_pin(port, pin) port_array[port - PORTA] ^= (1 << (pin));
- #define toggle_pin_var(port, pin) toggle_pin(port, pin)
- #define turn_global_ints_off() clear_bit(intcon, GIE)
- #define turn_global_ints_on() set_bit(intcon, GIE)
- #define turn_peripheral_ints_off() clear_bit(intcon, PEIE)
- #define turn_peripheral_ints_on() set_bit(intcon, PEIE)
- #define uns16 unsigned int
- #define uns32 unsigned long
- #define uns8 unsigned char

---

## Detailed Description

Defines datatypes, port/pin access helpers

Definition in file pic_utils.h.

---

## Define Documentation

### #define change_pin(port, pin, value)

```
Value:if (value) { \
        set_pin(port, pin); \
```

```
    } else { \
        clear_pin(port, pin); \
    }
```

Definition at line 251 of file pic_utils.h.

Referenced by i2c_send_byte(), lcd_write_nibble(), ms5540_get_config(), ms5540_reset(), pcd8544_send_byte(), pic_rf_send_byte(), pic_rf_send_byte_int(), pic_rf_set_mode(), sht15_send_byte(), spi_pulse_0(), spi_pulse_1(), spi_write(), spi_write_lsb(), and spi_write_sure().

### #define change_pin_var(port, pin, value) change_pin(port, pin, value)

Definition at line 265 of file pic_utils.h.

Referenced by cat4016_write_data(), ht1632_write(), and sure_2416_write().

### #define clear_pin(port, pin) clear_bit(port_array[port - PORTA], pin);

Definition at line 238 of file pic_utils.h.

Referenced by cat4016_enable_display(), cat4016_latch_data(), cat4016_setup_io(), cat4016_write_data(), drv_paint(), drv_refresh(), ea_ldp6416_setup_io(), ea_ldp6432_setup_io(), ea_ldp8008_setup_io(), ht1632_fill2(), ht1632_send_command(), ht1632_set_pixel(), ht1632_write(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), lcd_setup(), lcd_toggle_e(), lcd_wait_busy(), lcd_write_command(), lcd_write_data(), lcd_write_data_str(), mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), ms5540_pulse_sclk(), ms5540_send_stop(), ms5540_setup_io(), pcd8544_init(), pcd8544_send_byte(), pcd8544_send_command(), pcd8544_send_data(), pcd8544_setup_io(), pic_rf_init(), pic_rf_quick_init(), pic_rf_read_register(), pic_rf_read_register_inline(), pic_rf_read_register_int(), pic_rf_receive(), pic_rf_send_byte(), pic_rf_send_byte_int(), pic_rf_send_command(), pic_rf_send_command_inline(), pic_rf_send_command_single(), pic_rf_set_channel(), pic_rf_set_mode(), pic_rf_setup(), pic_rf_transmit(), platform_leds_setup_io(), pwm_handle(), sht15_read_byte16(), sht15_send_byte(), sht15_setup_io(), sht15_start(), somo_14d_reset(), somo_14d_send_data(), somo_14d_standby(), spi_pulse_0(), spi_pulse_1(), spi_write(), spi_write_lsb(), spi_write_sure(), sure_2416_fill2(), sure_2416_send_command(), sure_2416_set_pixel(), sure_2416_write(), and sure_7seg_write_str().

### #define clear_pin_var(port, pin) clear_pin(port, pin)

Definition at line 260 of file pic_utils.h.

### #define end_crit_sec() intcon.GIE = store_gie

Definition at line 300 of file pic_utils.h.

Referenced by audio_queue_clear(), drv_paint(), pic_rf_init(), pic_rf_set_channel(), pic_rf_set_mode(), pic_rf_transmit(), pkt_process_rf_data(), serial_getc(), tick_get_count(), usb_cdc_getc(), usb_cdc_handle_tx(), and usb_cdc_putc().

### #define int16 int

Definition at line 82 of file pic_utils.h.

Referenced by ms5540_calc_temp_and_pressure(), and sht15_fix_temperature_h().

### #define int32 long

Definition at line 84 of file pic_utils.h.

Referenced by ms5540_calc_temp_and_pressure(), and sht15_fix_humidity_r().

### #define int8 char

Definition at line 79 of file pic_utils.h.

### #define kill_interrupts()

```
Value:clear_bit(intcon, GIE); \
 nop(); \
 nop(); \
 nop(); \
 nop()
```

Definition at line 288 of file pic_utils.h.

Referenced by pic_rf_receive(), pic_rf_set_channel(), pic_rf_set_mode(), pic_rf_transmit(), and serial_putc().

### #define MAGIC_BOOSTBLOADER_REQUEST 4

Definition at line 375 of file pic_utils.h.

Referenced by term_process().

### #define make_input(port, pin) set_bit(tris_array[port - PORTA], pin)

Definition at line 279 of file pic_utils.h.

Referenced by ht1632_set_pixel(), i2c_setup_io(), mrf24j40_setup_io(), ms5540_setup_io(), pic_rf_receive(), pic_rf_setup(), somo_14d_setup_io(), spi_hw_setup_io(), and sure_2416_set_pixel().

### #define make_output(port, pin) clear_bit(tris_array[port - PORTA], pin)

Definition at line 278 of file pic_utils.h.

Referenced by cat4016_setup_io(), ea_ldp6416_setup_io(), ea_ldp6432_setup_io(), ea_ldp8008_setup_io(), ht1632_set_pixel(), ht1632_setup_io(), i2c_setup_io(), lcd_setup(), mrf24j40_setup_io(), ms5540_setup_io(), pcd8544_setup_io(), pic_rf_init(), pic_rf_quick_init(), pic_rf_set_mode(), pic_rf_setup(), pic_rf_transmit(), platform_leds_setup_io(), pwm_setup_io(), sht15_setup_io(), somo_14d_setup_io(), spi_hw_setup_io(), sure_2416_set_pixel(), and sure_2416_setup().

### #define NUMBER_PORTS 1

Definition at line 51 of file pic_utils.h.

### #define set_pin(port, pin) set_bit(port_array[port - PORTA], pin);

Definition at line 235 of file pic_utils.h.

Referenced by cat4016_enable_display(), cat4016_latch_data(), cat4016_setup_io(), cat4016_write_data(), drv_paint(), drv_refresh(), ea_ldp6416_setup_io(), ea_ldp6432_setup_io(), ht1632_fill2(),

ht1632_send_command(), ht1632_set_pixel(), ht1632_setup_io(), ht1632_write(), i2c_read_eeprom(), i2c_read_eeprom_16bit(), i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), lcd_toggle_e(), lcd_wait_busy(), lcd_write_data(), lcd_write_data_str(), mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_setup_io(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), ms5540_pulse_sclk(), ms5540_send_start(), pcd8544_init(), pcd8544_send_byte(), pcd8544_send_command(), pcd8544_send_data(), pcd8544_setup_io(), pic_rf_init(), pic_rf_read_register(), pic_rf_read_register_inline(), pic_rf_read_register_int(), pic_rf_receive(), pic_rf_send_byte(), pic_rf_send_byte_int(), pic_rf_send_command(), pic_rf_send_command_inline(), pic_rf_send_command_single(), pic_rf_set_channel(), pic_rf_set_mode(), pic_rf_setup(), pic_rf_transmit(), pwm_handle(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), somo_14d_reset(), somo_14d_send_data(), somo_14d_setup_io(), somo_14d_wake(), spi_pulse_0(), spi_pulse_1(), spi_write(), spi_write_lsb(), spi_write_sure(), sure_2416_fill2(), sure_2416_send_command(), sure_2416_set_pixel(), sure_2416_setup(), sure_2416_write(), and sure_7seg_write_str().

### #define set_pin_var(port, pin) set_pin(port, pin)

Definition at line 258 of file pic_utils.h.

### #define start_crit_sec()

```
Value:bit store_gie = intcon.GIE; \
    kill interrupts()
```

Definition at line 295 of file pic_utils.h.

Referenced by audio_queue_clear(), drv_paint(), pic_rf_init(), pic_rf_set_channel(), pic_rf_set_mode(), pic_rf_transmit(), pkt_process_rf_data(), serial_getc(), tick_get_count(), usb_cdc_getc(), usb_cdc_handle_tx(), and usb_cdc_putc().

### #define test_output_pin(port, pin) ((port_array[port - PORTA] & (1 << pin)) != 0)

Definition at line 248 of file pic_utils.h.

### #define test_pin(port, pin) ((port_in_array[port - PORTA] & (1 << pin)) != 0)

Definition at line 245 of file pic_utils.h.

Referenced by ht1632_set_pixel(), i2c_ack_polling(), i2c_receive_byte(), lcd_wait_busy(), ms5540_get_config(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), pic_rf_receive(), pic_rf_send_byte(), pic_rf_send_byte_int(), sht15_read_byte16(), sht15_send_byte(), somo_14d_is_busy(), and sure_2416_set_pixel().

### #define test_pin_var(port, pin) test_pin(port, pin)

Definition at line 264 of file pic_utils.h.

### #define toggle_pin(port, pin) port_array[port - PORTA] ^= (1 << (pin));

Definition at line 241 of file pic_utils.h.

### #define toggle_pin_var(port, pin) toggle_pin(port, pin)

Definition at line 262 of file pic_utils.h.

### #define turn_global_ints_off()  clear_bit(intcon, GIE)

Definition at line 285 of file pic_utils.h.

### #define turn_global_ints_on()  set_bit(intcon, GIE)

Definition at line 284 of file pic_utils.h.

### #define turn_peripheral_ints_off()  clear_bit(intcon, PEIE)

Definition at line 282 of file pic_utils.h.

### #define turn_peripheral_ints_on()  set_bit(intcon, PEIE)

Definition at line 281 of file pic_utils.h.

### #define uns16  unsigned int

Definition at line 81 of file pic_utils.h.

Referenced by ar1000_seek(), ar1000_seek_more(), ar1000_set_volume(), ar1000_test(), ar1000_tune(), draw_length_str(), draw_print_buffer(), draw_print_str(), draw_rect(), draw_set_pixel(), drv_paint(), hmc6352_get_data(), ht1632_fill2(), i2c_read_eeprom_16bit(), its1_device_process(), its2_device_process(), its2_process_tx_queue(), its2_router_process(), its_transmit_to_sa(), mrf24j40_active_channel_scan(), mrf24j40_receive(), mrf24j40_scan_for_lowest_channel_ed(), ms5540_get_config(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), ms5540_init(), ms5540_reset(), pkt_init(), pkt_process_rf_data(), pkt_process_tx_queue(), platform_leds_process(), sht15_fix_humidity_l(), sht15_read(), sht15_read_byte16(), sht15_read_humidity(), sht15_read_temperature(), sure_2416_fill2(), tick_get_count(), timer_handle_1_isr(), tmp75_read_16bit(), usb_cdc_handle_tx(), usb_handle_transaction(), usb_send_data(), usb_send_data_chunk(), and wpan_data_received_callback().

### #define uns32  unsigned long

Definition at line 83 of file pic_utils.h.

### #define uns8  unsigned char

Definition at line 80 of file pic_utils.h.

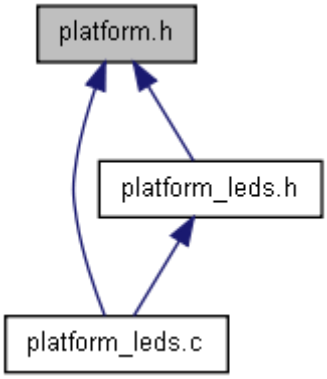
Referenced by ar1000_init(), ar1000_set_volume(), ar1000_write_registers(), audio_queue_add(), audio_queue_process(), cat4016_write_data(), draw_bitmap(), draw_clear_screen(), draw_length_str(), draw_print_buffer(), draw_print_str(), draw_set_pixel(), draw_tests_run(), drv_paint(), drv_print_buffer(), drv_refresh(), hc4led_write_str(), hmc6352_read_eeprom(), hmc6352_read_ram(), hmc6352_set_mode(), ht1632_fill(), ht1632_send_command(), ht1632_set_pixel(), ht1632_write(), i2c_read_eeprom(), i2c_receive_byte(), i2c_send_byte(), its1_controller_init(), its1_device_init(), its1_device_process(), its2_find_free_queue_slot(), its2_forward_routed_packet(), its2_print_packet(), its2_print_queue(), its2_process_tx_queue(), its2_rebroadcast_net_discover_req(), its2_router_init(), its2_router_queue_packet(),

its2_transmit(), its_add_local_device(), its_add_net_device(), its_get_device_handle(), its_init(), its_print_devices(), its_transmit_to_ea(), its_transmit_to_sa(), mrf24j40_active_channel_scan(), mrf24j40_flush_receive_buffer(), mrf24j40_handle_isr(), mrf24j40_init(), mrf24j40_init_coordinator(), mrf24j40_long_addr_read(), mrf24j40_receive(), mrf24j40_scan_for_lowest_channel_ed(), mrf24j40_set_extended_address(), mrf24j40_short_addr_read(), mrf24j40_transmit(), mrf24j40_transmit_to_extended_address(), mrf24j40_transmit_to_short_address(), ms5540_get_config(), ms5540_get_raw_pressure(), ms5540_get_raw_temp(), ms5540_reset(), pcd8544_send_byte(), pic_rf_init(), pic_rf_quick_init(), pic_rf_read_register(), pic_rf_read_register_inline(), pic_rf_read_register_int(), pic_rf_receive(), pic_rf_send_byte(), pic_rf_send_byte_int(), pic_rf_send_bytes(), pic_rf_send_bytes_inline(), pic_rf_send_command(), pic_rf_send_command_inline(), pic_rf_send_command_single(), pic_rf_set_mode(), pic_rf_transmit(), pkt_calc_check_byte(), pkt_check_check_byte(), pkt_init(), pkt_print_packet(), pkt_process_rf_data(), pkt_process_tx_queue(), pkt_queue_packet(), pkt_seen(), pkt_send_packet(), pkt_send_payload(), pwm_handle(), serial_getc(), serial_print_int(), serial_print_str(), serial_putc(), serial_rx_isr(), serial_tx_isr(), sht15_read_byte16(), sht15_send_byte(), somo_14d_send_data(), spi_write(), spi_write_lsb(), spi_write_sure(), sure_2416_fill(), sure_2416_send_command(), sure_2416_set_pixel(), sure_2416_write(), sure_7seg_write_str(), term_process(), tmp75_read(), usb_cdc_getc(), usb_cdc_handle_tx(), usb_cdc_print_int(), usb_cdc_print_str(), usb_cdc_putc(), usb_ep_data_out_callback(), usb_handle_class_request_callback(), usb_handle_isr(), usb_handle_standard_request(), usb_handle_transaction(), usb_send_data(), usb_send_data_chunk(), usb_send_empty_data_pkt(), usb_send_one_byte(), wpan_data_received_callback(), wpan_handle_receive(), wpan_print_address(), and wpan_print_mac_command().

---

# platform.h File Reference

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




## Defines

- #define EA_LED_PANEL_DRIVER 7
- #define EA_PLT1001 7
- #define EA_PLT1002 8
- #define EA_PLT1003 10
- #define EA_PLT_1001 7
- #define EA_PLT_1002 8
- #define EA_PLT_1003 10
- #define EA_USB2SERIAL 10

- #define EA_WEATHER_STATION 9
- #define EA_WIRELESS_TEMP_SENSOR 8
- #define OLIMEX_BOARD 2
- #define OLIMEX_PIC_LCD3310 5
- #define SCHMARTBOARD 6
- #define SFE_TDN_V1 3
- #define SURE_PICDEM_2 1
- #define TECH_TOYS_PIC18F4550 4

---

## Detailed Description

Definition in file platform.h.

---

## Define Documentation

### #define EA_LED_PANEL_DRIVER 7

EA Clock board

Definition at line 58 of file platform.h.

### #define EA_PLT1001 7

Definition at line 59 of file platform.h.

### #define EA_PLT1002 8

Definition at line 63 of file platform.h.

### #define EA_PLT1003 10

Definition at line 69 of file platform.h.

### #define EA_PLT_1001 7

Definition at line 60 of file platform.h.

### #define EA_PLT_1002 8

Definition at line 64 of file platform.h.

### #define EA_PLT_1003 10

Definition at line 70 of file platform.h.

**#define EA_USB2SERIAL 10**

EA USB2TTL board

Definition at line 68 of file platform.h.

**#define EA_WEATHER_STATION 9**

EA weather station

Definition at line 66 of file platform.h.

**#define EA_WIRELESS_TEMP_SENSOR 8**

EA wireless temp sensor board

Definition at line 62 of file platform.h.

**#define OLIMEX_BOARD 2**

Standard Olimex board

Definition at line 48 of file platform.h.

**#define OLIMEX_PIC_LCD3310 5**

Olimex PIC LCD with Nokia 3310 LCD dispaly

Definition at line 54 of file platform.h.

**#define SCHMARTBOARD 6**

Schmartboard

Definition at line 56 of file platform.h.

**#define SFE_TDN_V1 3**

SparkFun RF Terminal Development Node

Definition at line 50 of file platform.h.

**#define SURE_PICDEM_2 1**

Sure Electronics PicDem 2 demo board

Definition at line 46 of file platform.h.

**#define TECH_TOYS_PIC18F4550 4**

Tech Toys HK Pic 18f4550 Eval Dev 4a

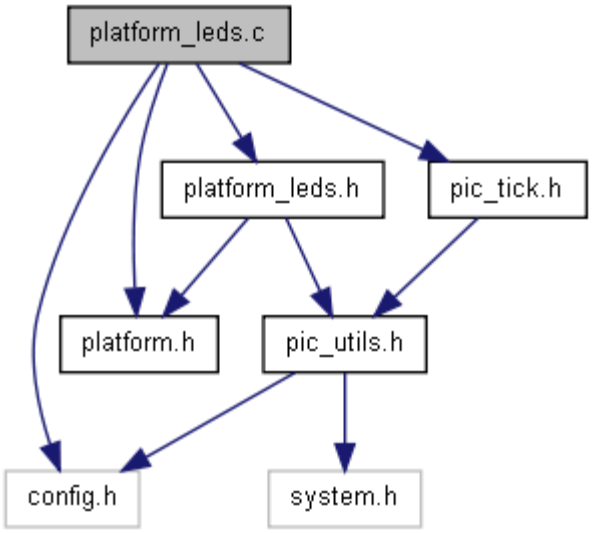
Definition at line 52 of file platform.h.

---

## platform_leds.c File Reference

```
#include "config.h"
#include "platform_leds.h"
#include "platform.h"
```

```
#include "pic_tick.h"
```
Include dependency graph for platform_leds.c:




## Defines

- #define PLATFORM_LEDS_FLASH_TICKS 250

## Functions

- void platform_leds_flash (uns8 led)
- void platform_leds_flashing (uns8 led, uns8 enable)
- void platform_leds_process ()
- void platform_leds_setup_io ()

---

## Define Documentation

### #define PLATFORM_LEDS_FLASH_TICKS 250

Definition at line 43 of file platform_leds.c.

Referenced by platform_leds_process().

---

## Function Documentation

### void platform_leds_flash (uns8 *led*)

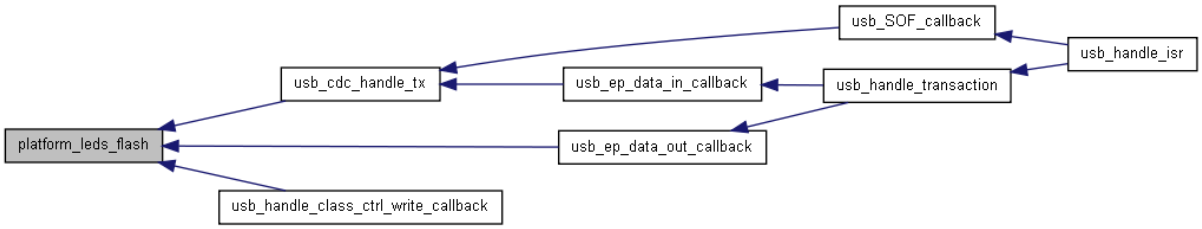
Definition at line 81 of file platform_leds.c.

References tick_get_count().

Referenced by usb_cdc_handle_tx(), usb_ep_data_out_callback(), and usb_handle_class_ctrl_write_callback().

Here is the call graph for this function:


Here is the caller graph for this function:




**void platform_leds_flashing (uns8 *led*, uns8 *enable*)**

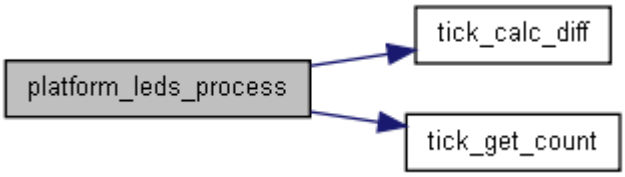
Definition at line 106 of file platform_leds.c.

**void platform_leds_process ()**

Definition at line 124 of file platform_leds.c.

References PLATFORM_LEDS_FLASH_TICKS, tick_calc_diff(), tick_get_count(), and uns16.

Here is the call graph for this function:




**void platform_leds_setup_io ()**

Definition at line 64 of file platform_leds.c.

References clear_pin, and make_output.

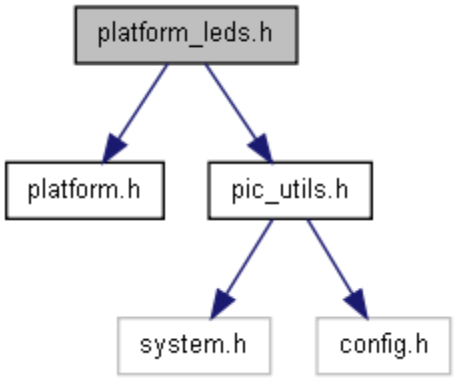
---

# platform_leds.h File Reference

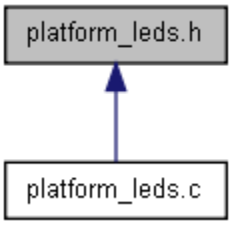
Easy access to the LEDs on your platform.

```
#include "platform.h"
#include "pic_utils.h"
```

Include dependency graph for platform_leds.h:




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




## Defines

- #define __platform_heds_h

## Functions

- void platform_leds_flash (uns8 led)
- void platform_leds_flashing (uns8 led, uns8 enable)
- void platform_leds_process ()
- void platform_leds_setup_io ()

---

## Detailed Description

Definition in file platform_leds.h.

---

## Define Documentation

### #define __platform_heds_h

Definition at line 43 of file platform_leds.h.

---

## Function Documentation

### void platform_leds_flash (uns8 *led*)

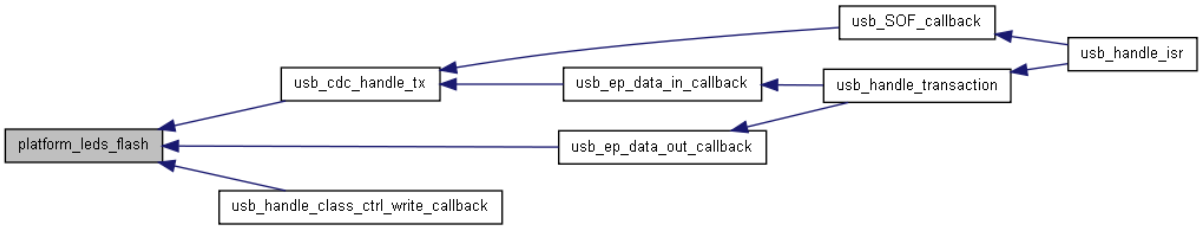
Definition at line 81 of file platform_leds.c.

References tick_get_count().

Referenced by usb_cdc_handle_tx(), usb_ep_data_out_callback(), and usb_handle_class_ctrl_write_callback().

Here is the call graph for this function:


Here is the caller graph for this function:




### void platform_leds_flashing (uns8 *led*, uns8 *enable*)

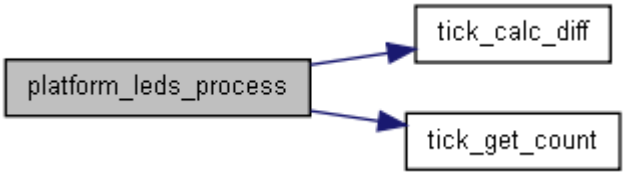
Definition at line 106 of file platform_leds.c.

### void platform_leds_process ()

Definition at line 124 of file platform_leds.c.

References PLATFORM_LEDS_FLASH_TICKS, tick_calc_diff(), tick_get_count(), and uns16.

Here is the call graph for this function:




### void platform_leds_setup_io ()

Definition at line 64 of file platform_leds.c.

References clear_pin, and make_output.

# protocol.h File Reference

Protocol definitions for Pkt mesh network.

## Defines

- #define CAPS_INFO1_DATE 1
- #define CAPS_INFO1_TIME 0
- #define CAPS_INPUTS_SWITCH1 0
- #define CAPS_INPUTS_SWITCH2 1
- #define CAPS_INPUTS_SWITCH3 2
- #define CAPS_INPUTS_SWITCH4 3
- #define CAPS_INPUTS_SWITCH5 4
- #define CAPS_INPUTS_SWITCH6 5
- #define CAPS_INPUTS_SWITCH7 6
- #define CAPS_INPUTS_SWITCH8 7
- #define CAPS_OUTPUTS_DIMMER1 4
- #define CAPS_OUTPUTS_DIMMER2 5
- #define CAPS_OUTPUTS_DIMMER3 6
- #define CAPS_OUTPUTS_DIMMER4 7
- #define CAPS_OUTPUTS_RELAY1 0
- #define CAPS_OUTPUTS_RELAY2 1
- #define CAPS_OUTPUTS_RELAY3 2
- #define CAPS_OUTPUTS_RELAY4 3
- #define CAPS_SENSOR_AIR_PRESSURE 2
- #define CAPS_SENSOR_HUMIDITY 1
- #define CAPS_SENSOR_LIGHT 3
- #define CAPS_SENSOR_PRESENCE 4
- #define CAPS_SENSOR_TEMP 0
- #define EE_MY_ADDR_H 0x00
- #define EE_MY_ADDR_L 0x01
- #define EE_MY_INFO1 0x07
- #define EE_MY_INPUTS 0x05
- #define EE_MY_LAST_PKT_ID_H 0x02
- #define EE_MY_LAST_PKT_ID_L 0x03
- #define EE_MY_OUTPUTS 0x06
- #define EE_MY_SENSORS 0x04
- #define PL_ADDR_RESPONSE 5
- #define PL_CAPS_RESPONSE 4
- #define PL_CHANGE_RESPONSE 9
- #define PL_OTHER_RESPONSE 11
- #define PL_REQ_ADDR 2
- #define PL_REQ_CAPS 3
- #define PL_REQ_INFO1 10
- #define PL_REQ_INFORM_ON_CHANGE 8
- #define PL_REQ_SENSOR 6
- #define PL_SENSOR_RESPONSE 7
- #define PL_SET_ADDR 1
- #define PL_SET_OUTPUT 9

## Detailed Description

Definition in file protocol.h.

---

## Define Documentation

### #define CAPS_INFO1_DATE 1

Definition at line 136 of file protocol.h.

### #define CAPS_INFO1_TIME 0

Definition at line 135 of file protocol.h.

### #define CAPS_INPUTS_SWITCH1 0

Definition at line 116 of file protocol.h.

### #define CAPS_INPUTS_SWITCH2 1

Definition at line 117 of file protocol.h.

### #define CAPS_INPUTS_SWITCH3 2

Definition at line 118 of file protocol.h.

### #define CAPS_INPUTS_SWITCH4 3

Definition at line 119 of file protocol.h.

### #define CAPS_INPUTS_SWITCH5 4

Definition at line 120 of file protocol.h.

### #define CAPS_INPUTS_SWITCH6 5

Definition at line 121 of file protocol.h.

### #define CAPS_INPUTS_SWITCH7 6

Definition at line 122 of file protocol.h.

**#define CAPS_INPUTS_SWITCH8 7**

Definition at line 123 of file protocol.h.

**#define CAPS_OUTPUTS_DIMMER1 4**

Definition at line 130 of file protocol.h.

**#define CAPS_OUTPUTS_DIMMER2 5**

Definition at line 131 of file protocol.h.

**#define CAPS_OUTPUTS_DIMMER3 6**

Definition at line 132 of file protocol.h.

**#define CAPS_OUTPUTS_DIMMER4 7**

Definition at line 133 of file protocol.h.

**#define CAPS_OUTPUTS_RELAY1 0**

Definition at line 126 of file protocol.h.

**#define CAPS_OUTPUTS_RELAY2 1**

Definition at line 127 of file protocol.h.

**#define CAPS_OUTPUTS_RELAY3 2**

Definition at line 128 of file protocol.h.

**#define CAPS_OUTPUTS_RELAY4 3**

Definition at line 129 of file protocol.h.

**#define CAPS_SENSOR_AIR_PRESSURE 2**

Definition at line 111 of file protocol.h.

**#define CAPS_SENSOR_HUMIDITY 1**

Definition at line 110 of file protocol.h.

**#define CAPS_SENSOR_LIGHT 3**

Definition at line 112 of file protocol.h.

**#define CAPS_SENSOR_PRESENCE 4**

Definition at line 113 of file protocol.h.

**#define CAPS_SENSOR_TEMP 0**

Definition at line 109 of file protocol.h.

**#define EE_MY_ADDR_H 0x00**

Definition at line 95 of file protocol.h.

**#define EE_MY_ADDR_L 0x01**

Definition at line 96 of file protocol.h.

**#define EE_MY_INFO1 0x07**

Definition at line 104 of file protocol.h.

**#define EE_MY_INPUTS 0x05**

Definition at line 102 of file protocol.h.

**#define EE_MY_LAST_PKT_ID_H 0x02**

Definition at line 98 of file protocol.h.

**#define EE_MY_LAST_PKT_ID_L 0x03**

Definition at line 99 of file protocol.h.

**#define EE_MY_OUTPUTS 0x06**

Definition at line 103 of file protocol.h.

**#define EE_MY_SENSORS 0x04**

Definition at line 101 of file protocol.h.

**#define PL_ADDR_RESPONSE 5**

Definition at line 60 of file protocol.h.

**#define PL_CAPS_RESPONSE 4**

Definition at line 52 of file protocol.h.

**#define PL_CHANGE_RESPONSE 9**

Definition at line 75 of file protocol.h.

**#define PL_OTHER_RESPONSE 11**

Definition at line 88 of file protocol.h.

**#define PL_REQ_ADDR 2**

Definition at line 50 of file protocol.h.

**#define PL_REQ_CAPS 3**

Definition at line 51 of file protocol.h.

**#define PL_REQ_INFO1 10**

Definition at line 85 of file protocol.h.

**#define PL_REQ_INFORM_ON_CHANGE 8**

Definition at line 71 of file protocol.h.

**#define PL_REQ_SENSOR 6**

Definition at line 63 of file protocol.h.

**#define PL_SENSOR_RESPONSE 7**

Definition at line 66 of file protocol.h.

**#define PL_SET_ADDR 1**

Definition at line 47 of file protocol.h.

#### #define PL_SET_OUTPUT 9

Definition at line 80 of file protocol.h.

---

# sfe_tdn_v1.h File Reference

## Defines

- #define __sfe_tdn_v1_h defined
- #define stat1 1
- #define stat2 3
- #define stat3 4

---

## Define Documentation

#### #define __sfe_tdn_v1_h defined

Definition at line 42 of file sfe_tdn_v1.h.

#### #define stat1 1

Definition at line 45 of file sfe_tdn_v1.h.

#### #define stat2 3

Definition at line 46 of file sfe_tdn_v1.h.

#### #define stat3 4

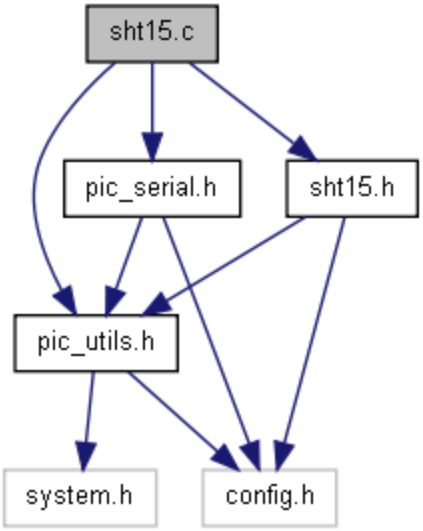
Definition at line 47 of file sfe_tdn_v1.h.

---

# sht15.c File Reference

```
#include "sht15.h"
#include "pic_utils.h"
#include "pic_serial.h"
```

Include dependency graph for sht15.c:




## Defines

- #define CHECK_HUMD 0b00000101
- #define CHECK_STAT 0b00000111
- #define CHECK_TEMP 0b00000011
- #define sht15_read_sda() set_bit(tris_array[sht15_sda_port - PORTA], sht15_sda_pin);
- #define sht15_write_sda() clear_bit(tris_array[sht15_sda_port - PORTA], sht15_sda_pin);
- #define WRITE_STAT 0b00000110

## Functions

- uns16 sht15_fix_humidity (uns16 sensor_out)
- uns16 sht15_fix_humidity_l (uns8 sensor_out)
- uns16 sht15_fix_humidity_r (uns16 sensor_out)
- int16 sht15_fix_temperature_h (uns16 sensor_out)
- void sht15_read (void)
- uns16 sht15_read_byte16 (void)
- uns16 sht15_read_humidity (void)
- uns16 sht15_read_temperature (void)
- void sht15_send_byte (uns8 sht15_command)
- void sht15_setup_io (void)
- void sht15_start (void)

---

## Define Documentation

### #define CHECK_HUMD 0b00000101

Definition at line 46 of file sht15.c.

Referenced by sht15_read(), and sht15_read_humidity().

### #define CHECK_STAT 0b00000111

Definition at line 47 of file sht15.c.

**#define CHECK_TEMP 0b00000011**

Definition at line 45 of file sht15.c.

Referenced by sht15_read(), and sht15_read_temperature().

**#define sht15_read_sda() set_bit(tris_array[sht15_sda_port - PORTA], sht15_sda_pin);**

Definition at line 42 of file sht15.c.

Referenced by sht15_read_byte16(), and sht15_send_byte().

**#define sht15_write_sda() clear_bit(tris_array[sht15_sda_port - PORTA], sht15_sda_pin);**

Definition at line 41 of file sht15.c.

Referenced by sht15_read_byte16(), sht15_send_byte(), and sht15_start().

**#define WRITE_STAT 0b00000110**

Definition at line 48 of file sht15.c.

---

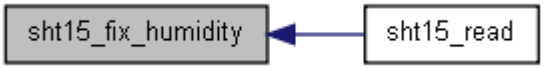
## Function Documentation

**uns16 sht15_fix_humidity (uns16 *sensor_out*)**

Definition at line 327 of file sht15.c.

Referenced by sht15_read().

Here is the caller graph for this function:




**uns16 sht15_fix_humidity_l (uns8 *sensor_out*)**

Definition at line 347 of file sht15.c.

References uns16.

Referenced by sht15_read_humidity().

Here is the caller graph for this function:

### uns16 sht15_fix_humidity_r (uns16 *sensor_out*)

Definition at line 305 of file sht15.c.

References c1, c2, c3, and int32.

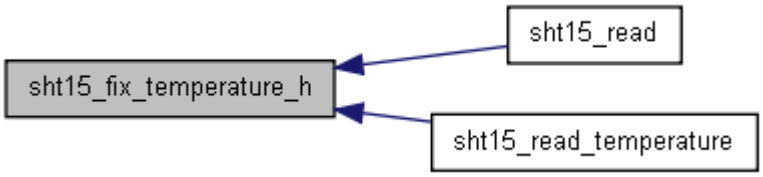
### int16 sht15_fix_temperature_h (uns16 *sensor_out*)

Definition at line 366 of file sht15.c.

References int16.

Referenced by sht15_read(), and sht15_read_temperature().

Here is the caller graph for this function:

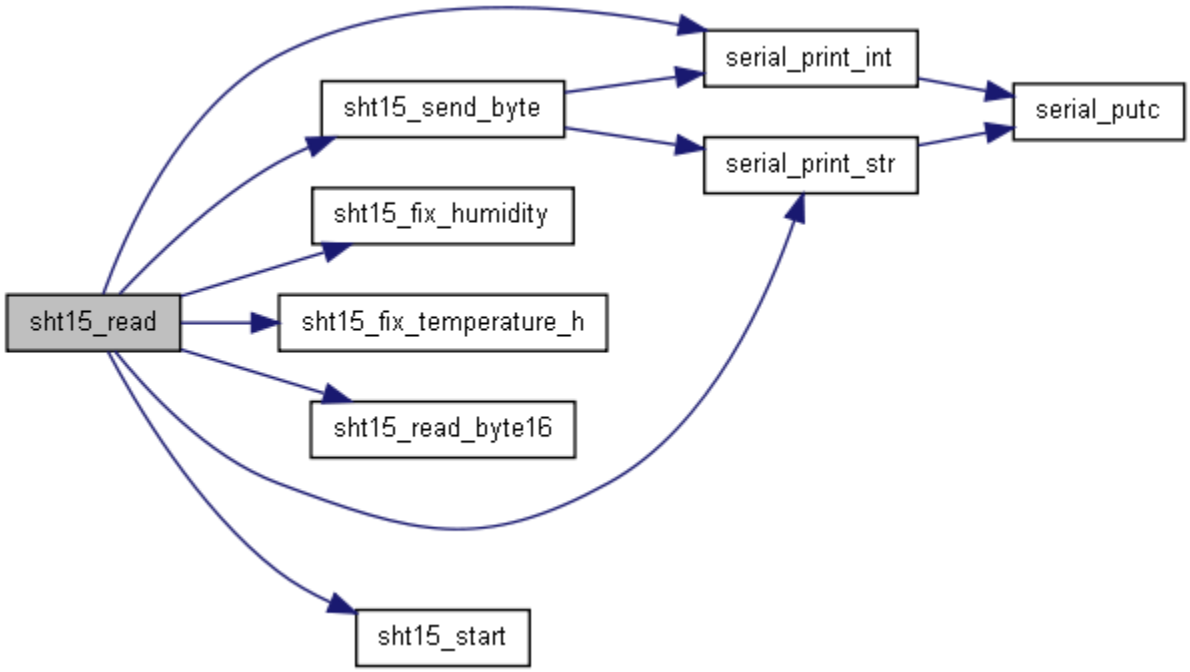



### void sht15_read (void)

Definition at line 87 of file sht15.c.

References CHECK_HUMD, CHECK_TEMP, serial_print_int(), serial_print_str(), sht15_fix_humidity(), sht15_fix_temperature_h(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:



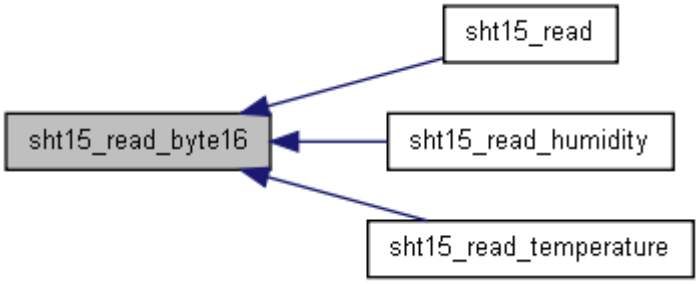
### uns16 sht15_read_byte16 (void)

Definition at line 216 of file sht15.c.

References clear_pin, set_pin, sht15_read_sda, sht15_write_sda, test_pin, uns16, and uns8.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the caller graph for this function:

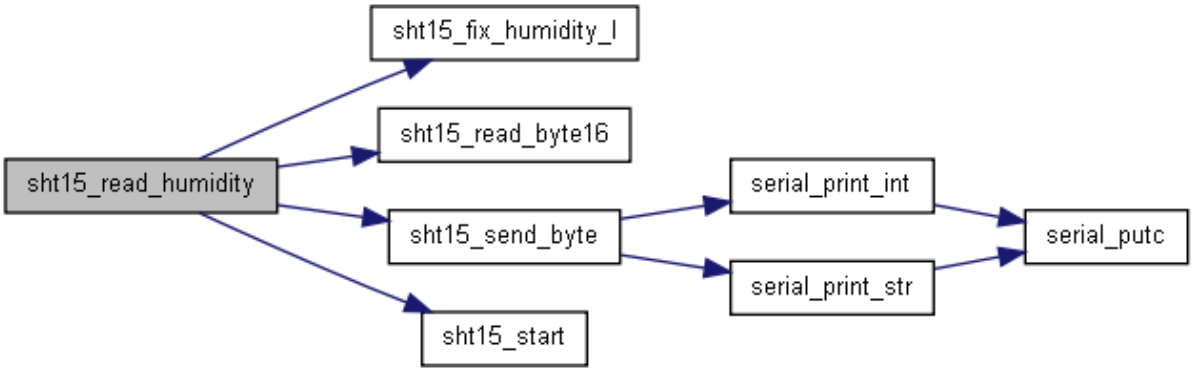



### uns16 sht15_read_humidity (void)

Definition at line 59 of file sht15.c.

References CHECK_HUMD, sht15_fix_humidity_l(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:

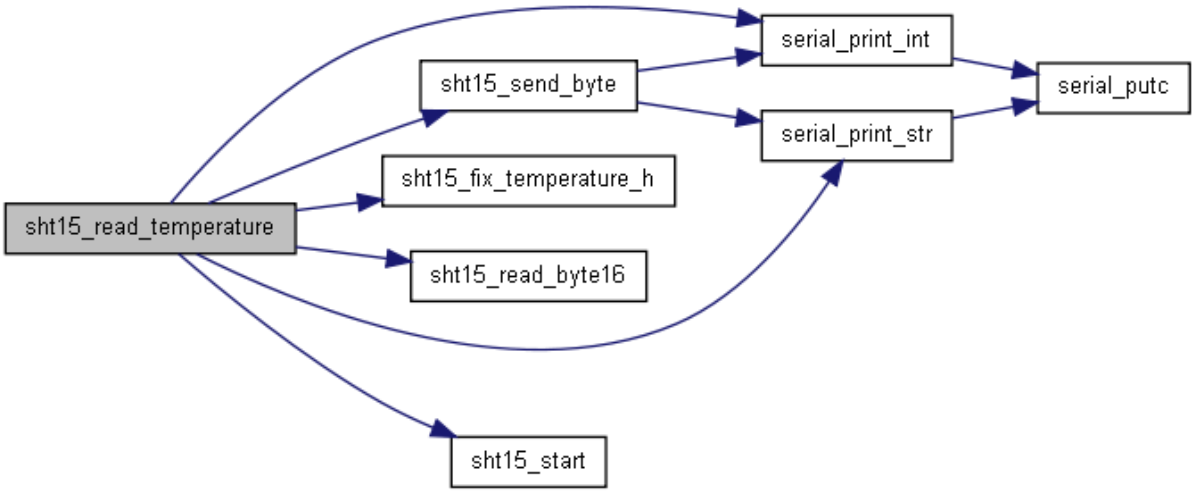



### uns16 sht15_read_temperature (void)

Definition at line 71 of file sht15.c.

References CHECK_TEMP, serial_print_int(), serial_print_str(), sht15_fix_temperature_h(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:



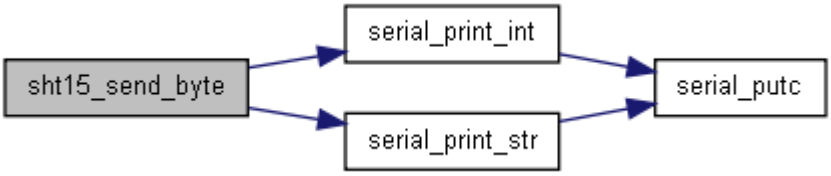

### void sht15_send_byte (uns8 *sht15_command*)

Definition at line 118 of file sht15.c.

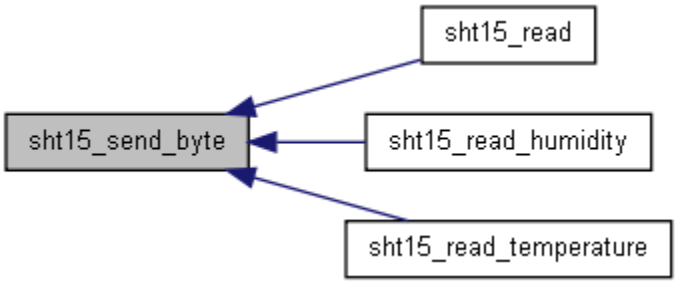
References change_pin, clear_pin, serial_print_int(), serial_print_str(), set_pin, sht15_read_sda, sht15_write_sda, test_pin, and uns8.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the call graph for this function:




Here is the caller graph for this function:




### void sht15_setup_io (void)

Definition at line 51 of file sht15.c.

References clear_pin, and make_output.

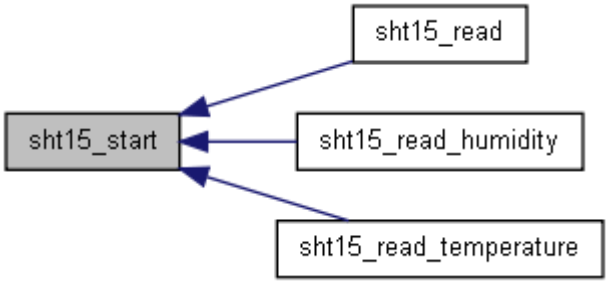
**void sht15_start (void)**

Definition at line 172 of file sht15.c.

References clear_pin, set_pin, and sht15_write_sda.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the caller graph for this function:



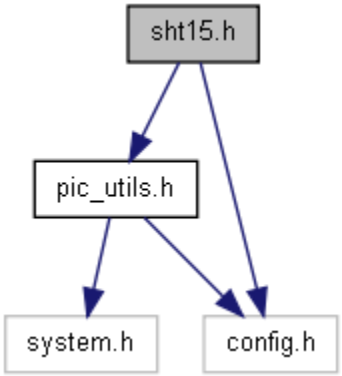

---

## sht15.h File Reference

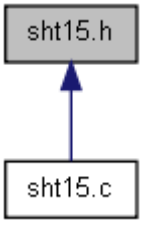
Support for SHT15 and SHT11 digital humidity sensors.

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

Include dependency graph for sht15.h:




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



## Defines

- #define __sht15_h defined

## Functions

- uns16 sht15_fix_humidity (uns16 sensor_out)
- uns16 sht15_fix_humidity_l (uns8 sensor_out)
- uns16 sht15_fix_humidity_r (uns16 sensor_out)
- int16 sht15_fix_temperature_h (uns16 sensor_out)
- void sht15_read (void)
- uns16 sht15_read_byte16 (void)
- uns16 sht15_read_humidity (void)
- uns16 sht15_read_temperature (void)
- void sht15_send_byte (uns8 sht15_command)
- *void sht15_setup_io (void)
- void sht15_start (void)

---

## Detailed Description

Include the following in your config.h:

- ---------- sht15 defines
- ----------

define sht15_sck_port PORTA define sht15_sck_pin 1 define sht15_sda_port PORTA define sht15_sda_pin 0

- ----------

Definition in file sht15.h.

---

## Define Documentation

### #define __sht15_h defined

Definition at line 56 of file sht15.h.

---

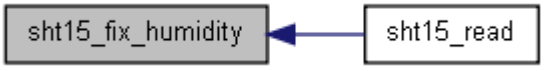
## Function Documentation

### uns16 sht15_fix_humidity (uns16 *sensor_out*)

Definition at line 327 of file sht15.c.

Referenced by sht15_read().

Here is the caller graph for this function:



### uns16 sht15_fix_humidity_l (uns8 *sensor_out*)

Definition at line 347 of file sht15.c.

References uns16.

Referenced by sht15_read_humidity().

Here is the caller graph for this function:


### uns16 sht15_fix_humidity_r (uns16 *sensor_out*)

Definition at line 305 of file sht15.c.

References c1, c2, c3, and int32.

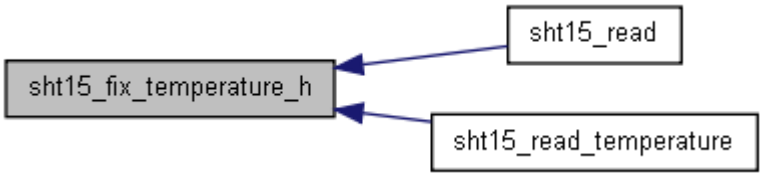
### int16 sht15_fix_temperature_h (uns16 *sensor_out*)

Definition at line 366 of file sht15.c.

References int16.

Referenced by sht15_read(), and sht15_read_temperature().

Here is the caller graph for this function:

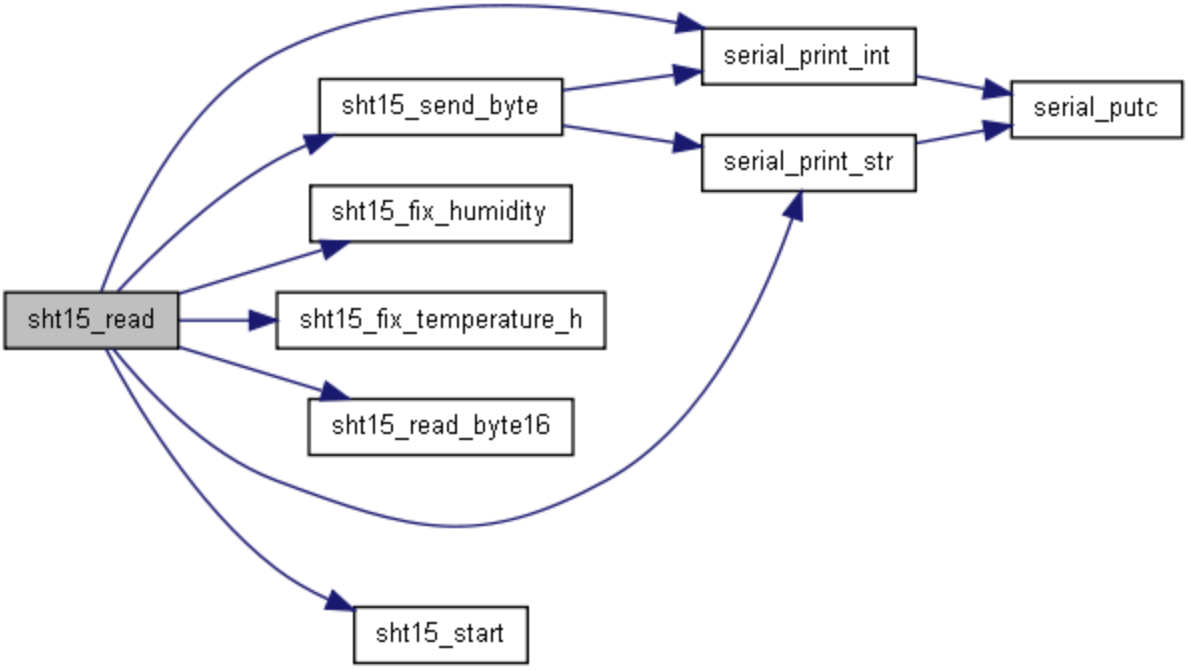



### void sht15_read (void)

Definition at line 87 of file sht15.c.

References CHECK_HUMD, CHECK_TEMP, serial_print_int(), serial_print_str(), sht15_fix_humidity(), sht15_fix_temperature_h(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:



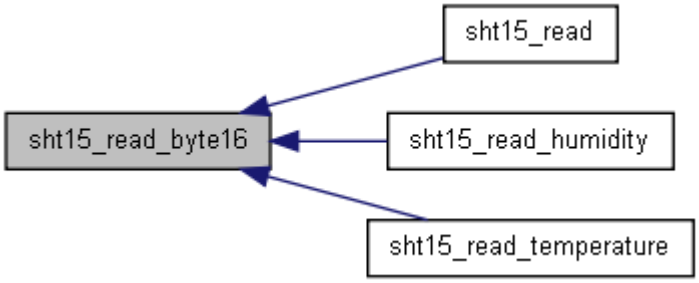

**uns16 sht15_read_byte16 (void)**

Definition at line 216 of file sht15.c.

References clear_pin, set_pin, sht15_read_sda, sht15_write_sda, test_pin, uns16, and uns8.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the caller graph for this function:

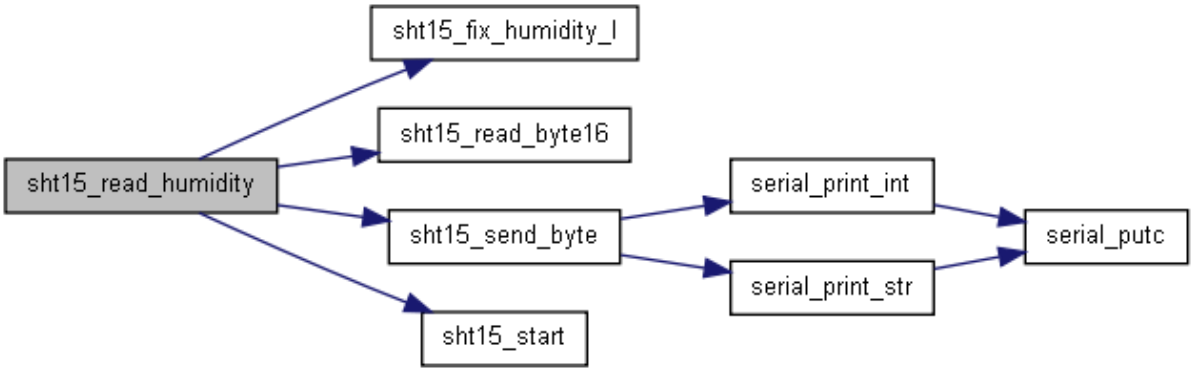



**uns16 sht15_read_humidity (void)**

Definition at line 59 of file sht15.c.

References CHECK_HUMD, sht15_fix_humidity_l(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:

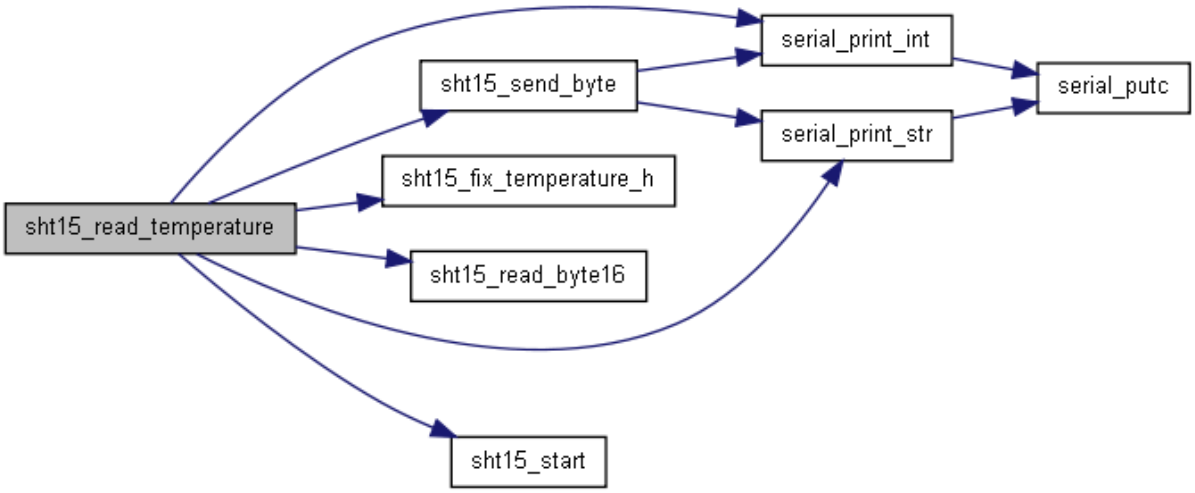



**uns16 sht15_read_temperature (void)**

Definition at line 71 of file sht15.c.

References CHECK_TEMP, serial_print_int(), serial_print_str(), sht15_fix_temperature_h(), sht15_read_byte16(), sht15_send_byte(), sht15_start(), and uns16.

Here is the call graph for this function:



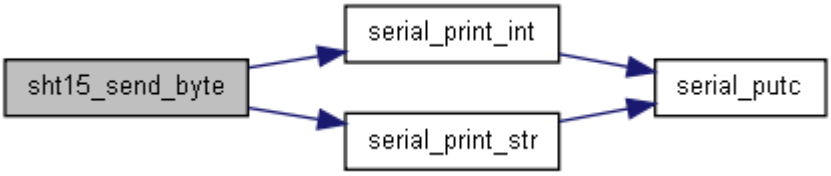

**void sht15_send_byte (uns8 *sht15_command*)**

Definition at line 118 of file sht15.c.

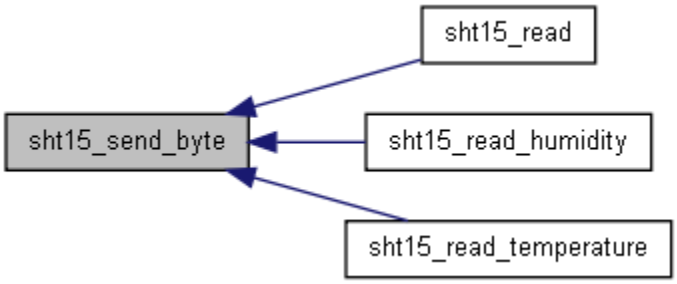
References change_pin, clear_pin, serial_print_int(), serial_print_str(), set_pin, sht15_read_sda, sht15_write_sda, test_pin, and uns8.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the call graph for this function:



Here is the caller graph for this function:




*** void sht15_setup_io (void)**

Definition at line 51 of file sht15.c.

References clear_pin, and make_output.

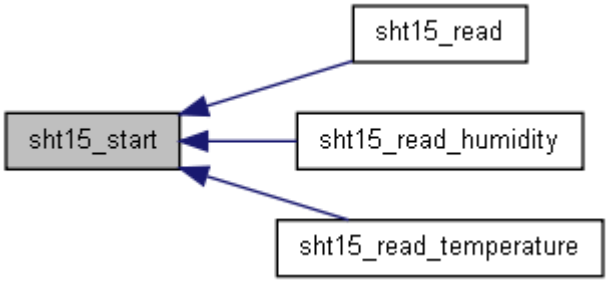
**void sht15_start (void)**

Definition at line 172 of file sht15.c.

References clear_pin, set_pin, and sht15_write_sda.

Referenced by sht15_read(), sht15_read_humidity(), and sht15_read_temperature().

Here is the caller graph for this function:

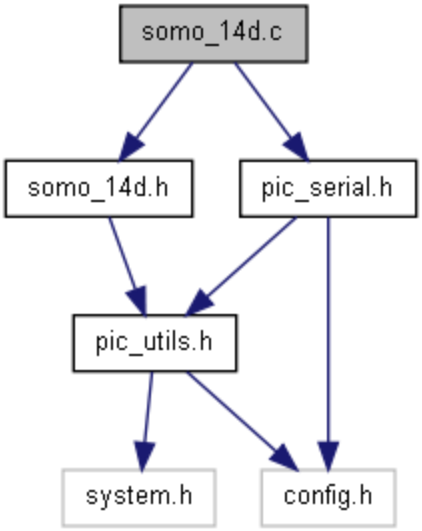



---

## somo_14d.c File Reference

```
#include "somo_14d.h"
#include "pic_serial.h"
```

Include dependency graph for somo_14d.c:




## Functions

- uns8 somo_14d_is_busy ()
- void somo_14d_play_pause ()
- void somo_14d_reset ()
- void somo_14d_send_data (uns16 data)
- void somo_14d_set_file_id (uns16 file_id)
- void somo_14d_set_volume (uns8 level)
- void somo_14d_setup_io ()
- void somo_14d_standby ()
- void somo_14d_stop ()
- void somo_14d_wake ()

---

## Function Documentation

### uns8 somo_14d_is_busy ()

Definition at line 123 of file somo_14d.c.

References test_pin.

Referenced by audio_queue_clear().

Here is the caller graph for this function:


### void somo_14d_play_pause ()

Definition at line 113 of file somo_14d.c.

References SOMO_14D_PLAY_PAUSE_CMD, and somo_14d_send_data().

Here is the call graph for this function:


### void somo_14d_reset ()

Definition at line 57 of file somo_14d.c.

References clear_pin, and set_pin.

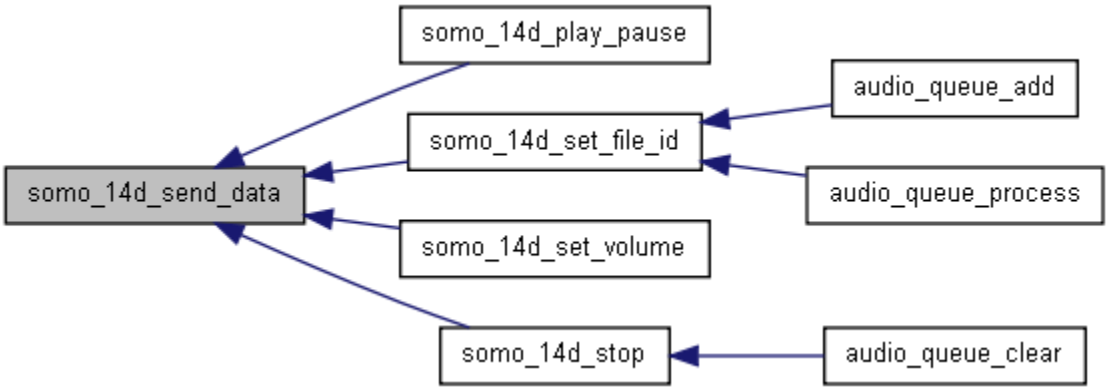
### void somo_14d_send_data (uns16 *data*)

Definition at line 79 of file somo_14d.c.

References clear_pin, set_pin, and uns8.

Referenced by somo_14d_play_pause(), somo_14d_set_file_id(), somo_14d_set_volume(), and somo_14d_stop().

Here is the caller graph for this function:




### void somo_14d_set_file_id (uns16 *file_id*)

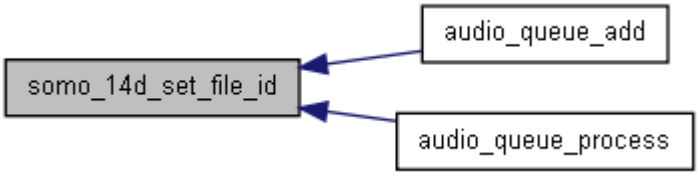
Definition at line 103 of file somo_14d.c.

References somo_14d_send_data().

Referenced by audio_queue_add(), and audio_queue_process().

Here is the call graph for this function:


Here is the caller graph for this function:



### void somo_14d_set_volume (uns8 *level*)

Definition at line 108 of file somo_14d.c.

References somo_14d_send_data(), and SOMO_14D_VOLUME_CMD.

Here is the call graph for this function:


### void somo_14d_setup_io ()

Definition at line 41 of file somo_14d.c.

References make_input, make_output, and set_pin.

### void somo_14d_standby ()

Definition at line 65 of file somo_14d.c.

References clear_pin.

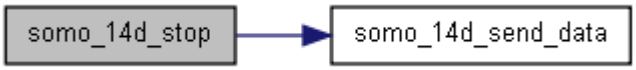
### void somo_14d_stop ()

Definition at line 117 of file somo_14d.c.

References somo_14d_send_data(), and SOMO_14D_STOP_CMD.

Referenced by audio_queue_clear().

Here is the call graph for this function:




Here is the caller graph for this function:

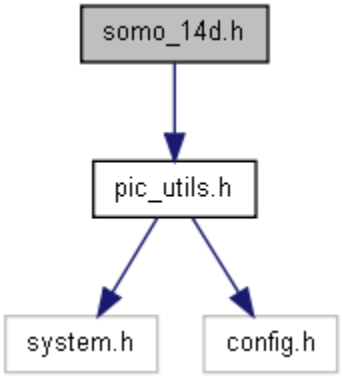

### void somo_14d_wake ()

Definition at line 73 of file somo_14d.c.

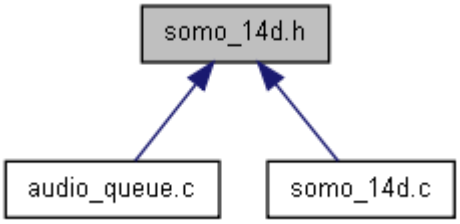
References set_pin.

# somo_14d.h File Reference

Somo-14D audio player interface.
`#include "pic_utils.h"`
Include dependency graph for somo_14d.h:




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




## Defines

- #define SOMO_14D_PLAY_PAUSE_CMD 0xfffe
- #define SOMO_14D_STOP_CMD 0xffff
- #define SOMO_14D_VOLUME_CMD 0xfff0

## Functions

- uns8 somo_14d_is_busy ()
- void somo_14d_play_pause ()
- void somo_14d_reset ()
- void somo_14d_set_file_id (uns16 file_id)
- void somo_14d_set_volume (uns8 level)
- void somo_14d_setup_io ()
- void somo_14d_standby ()
- void somo_14d_stop ()
- void somo_14d_wake ()

---

## Detailed Description

Library for accessing the functionality of the 4D systems .ad4 audio player

Put the following into your config.h

- - - - - - - - - - - - - - - - - - - - somo_14d defines
- - - - - - - - - - - - - - - - - - -

define somo_14d_clk_port PORTA define somo_14d_clk_pin 1

define somo_14d_data_port PORTA define somo_14d_data_pin 2

don't define these if you don't want to use them

define somo_14d_reset_port PORTA define somo_14d_reset_pin 3

define somo_14d_busy_port PORTA define somo_14d_busy_pin 4

- - - - - - - - - - - - - - - - - - -

Definition in file somo_14d.h.

---

## Define Documentation

### #define SOMO_14D_PLAY_PAUSE_CMD 0xfffe

Definition at line 69 of file somo_14d.h.

Referenced by somo_14d_play_pause().

### #define SOMO_14D_STOP_CMD 0xffff

Definition at line 70 of file somo_14d.h.

Referenced by somo_14d_stop().

### #define SOMO_14D_VOLUME_CMD 0xfff0

Definition at line 71 of file somo_14d.h.

Referenced by somo_14d_set_volume().

---

## Function Documentation

### uns8 somo_14d_is_busy ()

Definition at line 123 of file somo_14d.c.

References test_pin.

Referenced by audio_queue_clear().

Here is the caller graph for this function:

### void somo_14d_play_pause ()

Definition at line 113 of file somo_14d.c.

References SOMO_14D_PLAY_PAUSE_CMD, and somo_14d_send_data().

Here is the call graph for this function:


### void somo_14d_reset ()

Definition at line 57 of file somo_14d.c.

References clear_pin, and set_pin.

### void somo_14d_set_file_id (uns16 *file_id*)

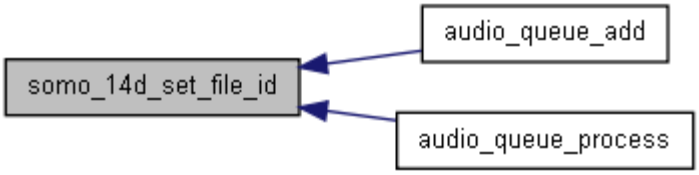
Definition at line 103 of file somo_14d.c.

References somo_14d_send_data().

Referenced by audio_queue_add(), and audio_queue_process().

Here is the call graph for this function:


Here is the caller graph for this function:




### void somo_14d_set_volume (uns8 *level*)

Definition at line 108 of file somo_14d.c.

References somo_14d_send_data(), and SOMO_14D_VOLUME_CMD.

Here is the call graph for this function:


### void somo_14d_setup_io ()

Definition at line 41 of file somo_14d.c.

References make_input, make_output, and set_pin.

### void somo_14d_standby ()

Definition at line 65 of file somo_14d.c.

References clear_pin.

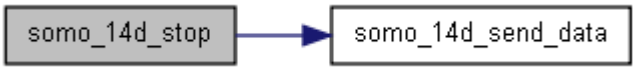
### void somo_14d_stop ()

Definition at line 117 of file somo_14d.c.

References somo_14d_send_data(), and SOMO_14D_STOP_CMD.

Referenced by audio_queue_clear().

Here is the call graph for this function:




Here is the caller graph for this function:


### void somo_14d_wake ()

Definition at line 73 of file somo_14d.c.

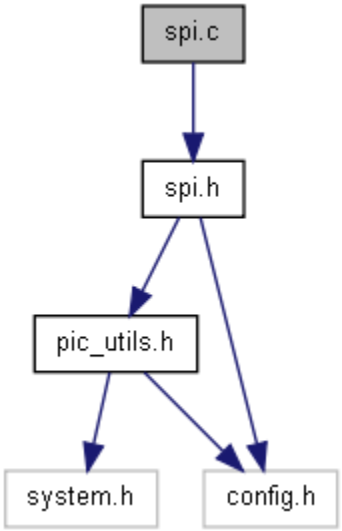
References set_pin.

---

## spi.c File Reference

```
#include "spi.h"
```

Include dependency graph for spi.c:



## Functions

- void spi_pulse_0 ()
- *SPI test routine.* void spi_pulse_1 ()
- *SPI test routine.* void spi_setup ()
- *Setup ports and pins for SPI output.* void spi_write (uns8 data)
- *Send a byte of data using software spi.* void spi_write_lsb (uns8 data)
- *Send a byte of data using software spi.* void spi_write_sure (uns8 data)

*SPI write for Sure devices.*

---

## Function Documentation

### void spi_pulse_0 ()

Definition at line 90 of file spi.c.

References change_pin, clear_pin, and set_pin.

### void spi_pulse_1 ()

Definition at line 96 of file spi.c.

References change_pin, clear_pin, and set_pin.

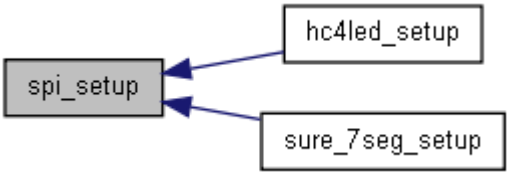
### void spi_setup ()

Setup ports and pins for SPI output

Definition at line 69 of file spi.c.

Referenced by hc4led_setup(), and sure_7seg_setup().

Here is the caller graph for this function:




### void spi_write (uns8 *data*)

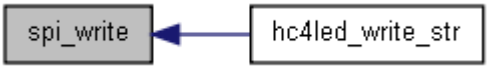
Sends a byte of data MSB first, data only changes on clock low

Definition at line 41 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by hc4led_write_str().

Here is the caller graph for this function:



**void spi_write_lsb (uns8 *data*)**

Sends a byte of data LSB first, data only changes on clock low

Definition at line 55 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

**void spi_write_sure (uns8 *data*)**

SPI write byte for Sure devices. Sure devices do things a little differently. Data goes LSB first but data changes on clock high.

Definition at line 75 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by sure_7seg_write_str().

Here is the caller graph for this function:

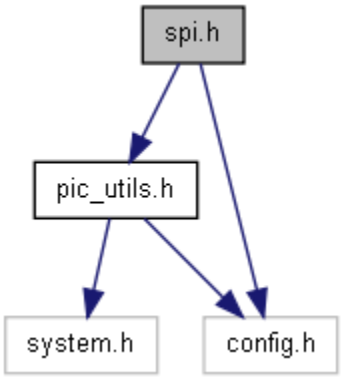

---

## spi.h File Reference

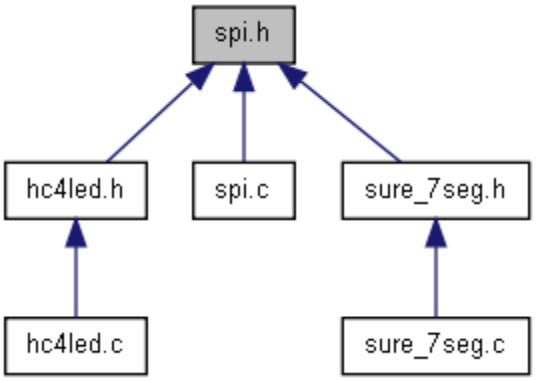
Outputs SPI-like interfaces (clock+data).

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

Include dependency graph for spi.h:




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




## Functions

- void spi_pulse_0 ()
- *SPI test routine.* void spi_pulse_1 ()
- *SPI test routine.* void spi_setup ()
- *Setup ports and pins for SPI output.* void spi_write (uns8 data)
- *Send a byte of data using software spi.* void spi_write_lsb (uns8 data)
- *Send a byte of data using software spi.* void spi_write_sure (uns8 data)

*SPI write for Sure devices.*

---

## Detailed Description

Covers standard SPI-like interfaces (clock + data) and Sure Electronics displays which are a little different

Definition in file spi.h.

---

## Function Documentation

### void spi_pulse_0 ()

Definition at line 90 of file spi.c.

References change_pin, clear_pin, and set_pin.

### void spi_pulse_1 ()

Definition at line 96 of file spi.c.

References change_pin, clear_pin, and set_pin.

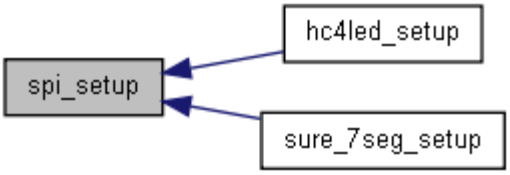
### void spi_setup ()

Setup ports and pins for SPI output

Definition at line 69 of file spi.c.

Referenced by hc4led_setup(), and sure_7seg_setup().

Here is the caller graph for this function:




**void spi_write (uns8 *data*)**

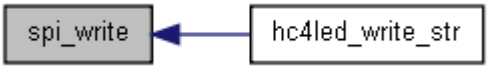
Sends a byte of data MSB first, data only changes on clock low

Definition at line 41 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by hc4led_write_str().

Here is the caller graph for this function:




**void spi_write_lsb (uns8 *data*)**

Sends a byte of data LSB first, data only changes on clock low

Definition at line 55 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

**void spi_write_sure (uns8 *data*)**

SPI write byte for Sure devices. Sure devices do things a little differently. Data goes LSB first but data changes on clock high.

Definition at line 75 of file spi.c.

References change_pin, clear_pin, set_pin, and uns8.

Referenced by sure_7seg_write_str().

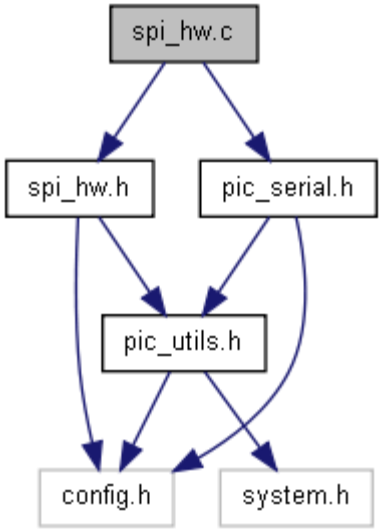
Here is the caller graph for this function:


---

## spi_hw.c File Reference

```
#include "spi_hw.h"
#include "pic_serial.h"
```

Include dependency graph for spi_hw.c:




## Functions

- void spi_hw_init ()
- uns8 spi_hw_receive ()
- void spi_hw_setup_io ()
- void spi_hw_transmit (uns8 data)

---

## Function Documentation

### void spi_hw_init ()

Definition at line 82 of file spi_hw.c.

### uns8 spi_hw_receive ()

Definition at line 106 of file spi_hw.c.

References spi_hw_transmit().

Here is the call graph for this function:


### void spi_hw_setup_io ()

Definition at line 40 of file spi_hw.c.

References make_input, and make_output.

### void spi_hw_transmit (uns8 *data*)

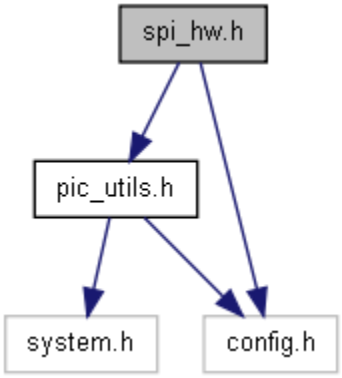
Definition at line 91 of file spi_hw.c.

---

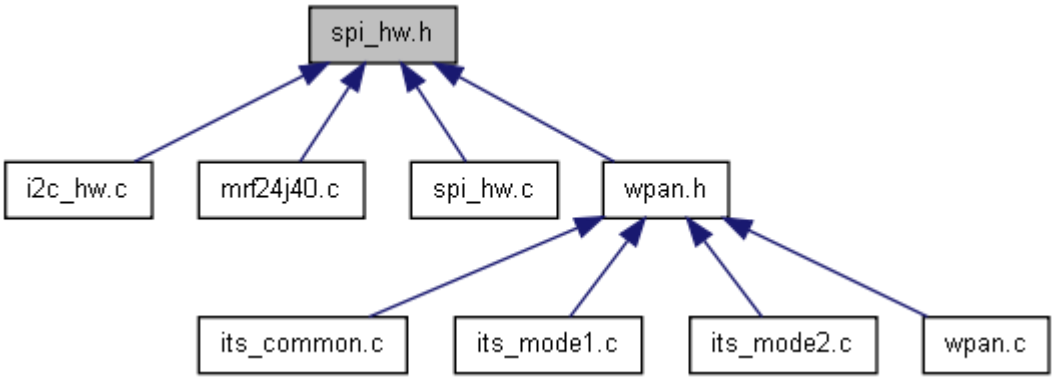
# spi_hw.h File Reference

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

Include dependency graph for spi_hw.h:




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




## Functions

- void spi_hw_init ()
- uns8 spi_hw_receive ()
- void spi_hw_setup_io ()
- void spi_hw_transmit (uns8 data)

---

## Detailed Description

Serial Peripheral Interface (HW) routines

Put the following into your config.h

- - - - - - - - - - - - - - - - - - - - spi_hw defines
- - - - - - - - - - - - - - - - - - -

define SPI_HW_MASTER_MODE or define SPI_HW_SLAVE_MODE

In slave mode, we can use ss: define SPI_HW_USE_SS

In master mode, we need to define clock define SPI_HW_MASTER_CLOCK_TMR2_DIV_2 define SPI_HW_MASTER_CLOCK_FOSC_DIV_64 define SPI_HW_MASTER_CLOCK_FOSC_DIV_16 define SPI_HW_MASTER_CLOCK_FOSC_DIV_4

- - - - - - - - - - - - - - - - - - -

Definition in file spi_hw.h.

---

## Function Documentation

### void spi_hw_init ()

Definition at line 82 of file i2c_hw.c.

### uns8 spi_hw_receive ()

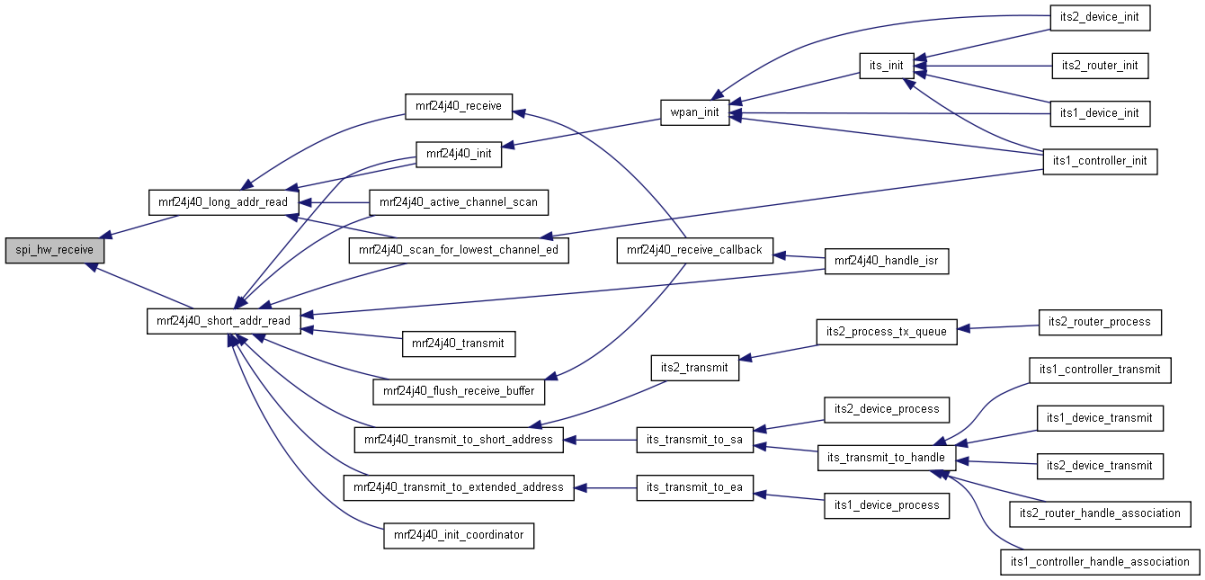
Definition at line 106 of file i2c_hw.c.

References spi_hw_transmit().

Referenced by mrf24j40_long_addr_read(), and mrf24j40_short_addr_read().

Here is the call graph for this function:


Here is the caller graph for this function:



### void spi_hw_setup_io ()

Definition at line 40 of file i2c_hw.c.

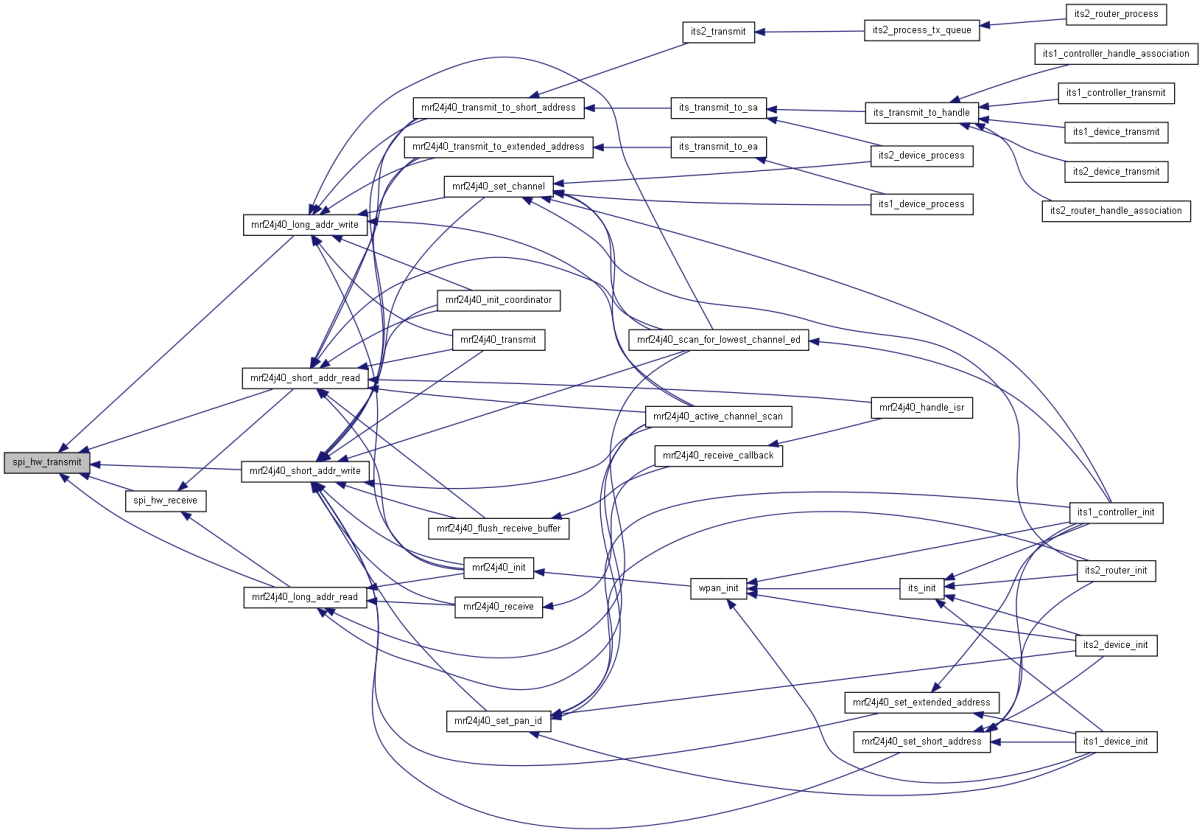
References make_input, and make_output.

### void spi_hw_transmit (uns8 *data*)

Definition at line 91 of file i2c_hw.c.

Referenced by mrf24j40_long_addr_read(), mrf24j40_long_addr_write(), mrf24j40_short_addr_read(), mrf24j40_short_addr_write(), and spi_hw_receive().

Here is the caller graph for this function:



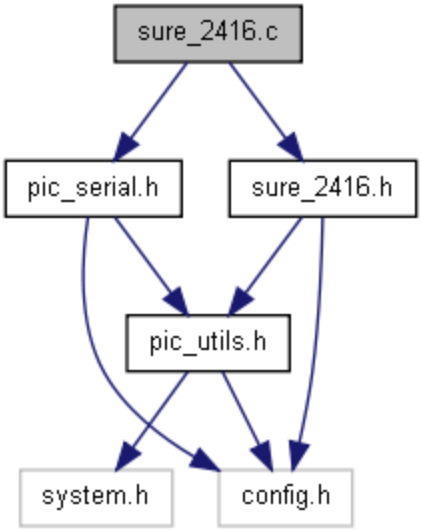

---

## sure_2416.c File Reference

Sure 2416 led matrix display routines.

```
#include "sure_2416.h"
#include "pic_serial.h"
```

Include dependency graph for sure_2416.c:




## Functions

- void sure_2416_fill (uns8 colour)
- void sure_2416_fill2 (uns8 colour)
- void sure_2416_init ()
- void sure_2416_send_command (uns8 command)
- void sure_2416_set_brightness (uns8 brightness)
- void sure_2416_set_pixel (uns8 x, uns8 y, uns8 colour)
- void sure_2416_setup ()
- void sure_2416_write (uns8 mem_addr, uns8 data)

---

## Detailed Description

Definition in file sure_2416.c.

---

## Function Documentation

### void sure_2416_fill (uns8 *colour*)

Definition at line 308 of file sure_2416.c.

References sure_2416_write(), and uns8.

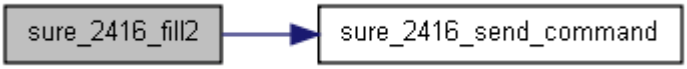
Here is the call graph for this function:


### void sure_2416_fill2 (uns8 *colour*)

Definition at line 323 of file sure_2416.c.

References clear_pin, set_pin, SURE_2416_CMD_LEDS_OFF, SURE_2416_CMD_LEDS_ON, sure_2416_send_command(), and uns16.

Here is the call graph for this function:

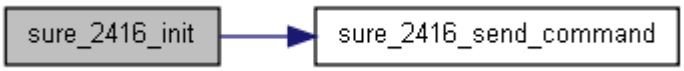



### void sure_2416_init ()

Definition at line 58 of file sure_2416.c.

References SURE_2416_CMD_CLK_MASTER_MODE, SURE_2416_CMD_LEDS_ON, SURE_2416_CMD_PMOS_16_COMMON, SURE_2416_CMD_SYS_DISABLE, SURE_2416_CMD_SYS_ENABLE, and sure_2416_send_command().

Here is the call graph for this function:



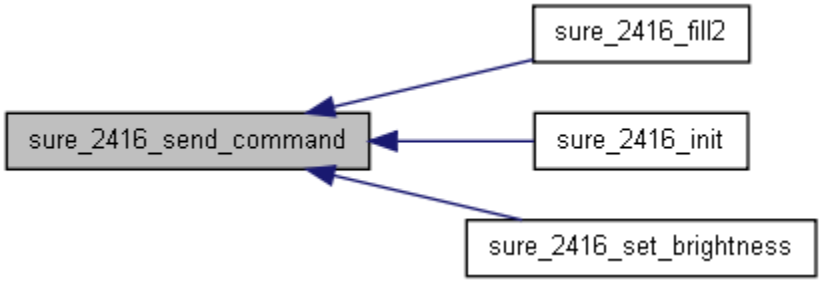

### void sure_2416_send_command (uns8 *command*)

Definition at line 68 of file sure_2416.c.

References clear_pin, set_pin, and uns8.

Referenced by sure_2416_fill2(), sure_2416_init(), and sure_2416_set_brightness().

Here is the caller graph for this function:




### void sure_2416_set_brightness (uns8 *brightness*)

Definition at line 188 of file sure_2416.c.

References sure_2416_send_command().

Here is the call graph for this function:

**void sure_2416_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 193 of file sure_2416.c.

References clear_pin, make_input, make_output, set_pin, test_pin, and uns8.

**void sure_2416_setup ()**

Definition at line 43 of file sure_2416.c.

References make_output, and set_pin.

**void sure_2416_write (uns8 *mem_addr*, uns8 *data*)**

Definition at line 124 of file sure_2416.c.

References change_pin_var, clear_pin, set_pin, and uns8.

Referenced by sure_2416_fill().

Here is the caller graph for this function:

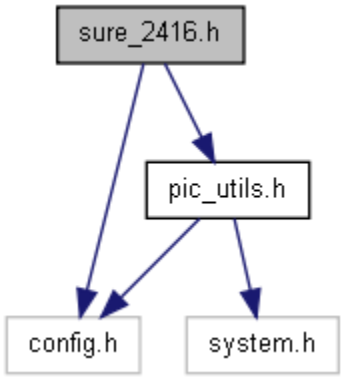

---

## sure_2416.h File Reference

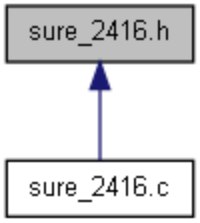
Sure 2416 led matrix display routines.

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

Include dependency graph for sure_2416.h:




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




## Defines

- #define SURE_2416_CMD_BLINK_OFF 0b00001000
- #define SURE_2416_CMD_BLINK_ON 0b00001001
- #define SURE_2416_CMD_CLK_MASTER_MODE 0b00010100
- #define SURE_2416_CMD_CLK_SLAVE_MODE 0b00010000
- #define SURE_2416_CMD_CLK_SOURCE_EXT 0b00011100
- #define SURE_2416_CMD_CLK_SOURCE_INT_RC 0b00011000
- #define SURE_2416_CMD_LEDS_OFF 0b00000010
- #define SURE_2416_CMD_LEDS_ON 0b00000011
- #define SURE_2416_CMD_NMOS_16_COMMON 0b00100100
- #define SURE_2416_CMD_NMOS_8_COMMON 0b00100000
- #define SURE_2416_CMD_PMOS_16_COMMON 0b00101100
- #define SURE_2416_CMD_PMOS_8_COMMON 0b00101000
- #define SURE_2416_CMD_SYS_DISABLE 0b00000000
- #define SURE_2416_CMD_SYS_ENABLE 0b00000001

## Functions

- void sure_2416_clear ()
- void sure_2416_fill (uns8 colour)
- void sure_2416_fill2 (uns8 colour)
- uns8 sure_2416_get_pixel (uns8 x, uns8 y)
- void sure_2416_horizontal_line (uns8 x, uns8 y, uns8 length, uns8 colour)
- void sure_2416_init ()
- void sure_2416_send_command (uns8 command)
- void sure_2416_set_brightness (uns8 brightness)
- void sure_2416_set_pixel (uns8 x, uns8 y, uns8 colour)
- void sure_2416_setup ()
- void sure_2416_vertical_line (uns8 x, uns8 y, uns8 length, uns8 colour)
- void sure_2416_write (uns8 mem_addr, uns8 data)

---

## Detailed Description

Definition in file sure_2416.h.

---

## Define Documentation

### #define SURE_2416_CMD_BLINK_OFF 0b00001000

Definition at line 86 of file sure_2416.h.

**#define SURE_2416_CMD_BLINK_ON 0b00001001**

Definition at line 87 of file sure_2416.h.

**#define SURE_2416_CMD_CLK_MASTER_MODE 0b00010100**

Definition at line 77 of file sure_2416.h.
Referenced by sure_2416_init().

**#define SURE_2416_CMD_CLK_SLAVE_MODE 0b00010000**

Definition at line 78 of file sure_2416.h.

**#define SURE_2416_CMD_CLK_SOURCE_EXT 0b00011100**

Definition at line 81 of file sure_2416.h.

**#define SURE_2416_CMD_CLK_SOURCE_INT_RC 0b00011000**

Definition at line 80 of file sure_2416.h.

**#define SURE_2416_CMD_LEDS_OFF 0b00000010**

Definition at line 83 of file sure_2416.h.
Referenced by sure_2416_fill2().

**#define SURE_2416_CMD_LEDS_ON 0b00000011**

Definition at line 84 of file sure_2416.h.
Referenced by sure_2416_fill2(), and sure_2416_init().

**#define SURE_2416_CMD_NMOS_16_COMMON 0b00100100**

Definition at line 73 of file sure_2416.h.

**#define SURE_2416_CMD_NMOS_8_COMMON 0b00100000**

Definition at line 72 of file sure_2416.h.

**#define SURE_2416_CMD_PMOS_16_COMMON 0b00101100**

Definition at line 75 of file sure_2416.h.

Referenced by sure_2416_init().

**#define SURE_2416_CMD_PMOS_8_COMMON 0b00101000**

Definition at line 74 of file sure_2416.h.

**#define SURE_2416_CMD_SYS_DISABLE 0b00000000**

Definition at line 69 of file sure_2416.h.

Referenced by sure_2416_init().

**#define SURE_2416_CMD_SYS_ENABLE 0b00000001**

Definition at line 70 of file sure_2416.h.

Referenced by sure_2416_init().

---

## Function Documentation

**void sure_2416_clear ()**

**void sure_2416_fill (uns8 *colour*)**

Definition at line 308 of file sure_2416.c.

References sure_2416_write(), and uns8.

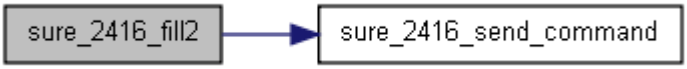
Here is the call graph for this function:


**void sure_2416_fill2 (uns8 *colour*)**

Definition at line 323 of file sure_2416.c.

References clear_pin, set_pin, SURE_2416_CMD_LEDS_OFF, SURE_2416_CMD_LEDS_ON, sure_2416_send_command(), and uns16.

Here is the call graph for this function:



**uns8 sure_2416_get_pixel (uns8 *x*, uns8 *y*)**

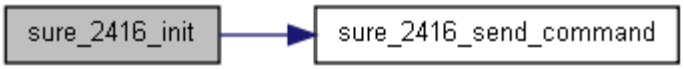
**void sure_2416_horizontal_line (uns8 *x*, uns8 *y*, uns8 *length*, uns8 *colour*)**

**void sure_2416_init ()**

Definition at line 58 of file sure_2416.c.

References SURE_2416_CMD_CLK_MASTER_MODE, SURE_2416_CMD_LEDS_ON, SURE_2416_CMD_PMOS_16_COMMON, SURE_2416_CMD_SYS_DISABLE, SURE_2416_CMD_SYS_ENABLE, and sure_2416_send_command().

Here is the call graph for this function:



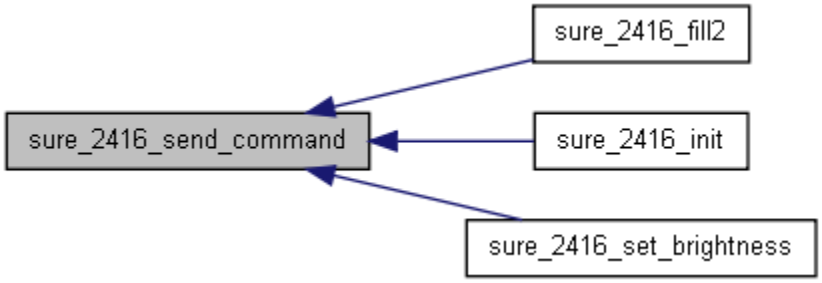

**void sure_2416_send_command (uns8 *command*)**

Definition at line 68 of file sure_2416.c.

References clear_pin, set_pin, and uns8.

Referenced by sure_2416_fill2(), sure_2416_init(), and sure_2416_set_brightness().

Here is the caller graph for this function:




**void sure_2416_set_brightness (uns8 *brightness*)**

Definition at line 188 of file sure_2416.c.

References sure_2416_send_command().

Here is the call graph for this function:


**void sure_2416_set_pixel (uns8 *x*, uns8 *y*, uns8 *colour*)**

Definition at line 193 of file sure_2416.c.

References clear_pin, make_input, make_output, set_pin, test_pin, and uns8.

**void sure_2416_setup ()**

Definition at line 43 of file sure_2416.c.

References make_output, and set_pin.

**void sure_2416_vertical_line (uns8 *x*, uns8 *y*, uns8 *length*, uns8 *colour*)**

**void sure_2416_write (uns8 *mem_addr*, uns8 *data*)**

Definition at line 124 of file sure_2416.c.

References change_pin_var, clear_pin, set_pin, and uns8.

Referenced by sure_2416_fill().

Here is the caller graph for this function:

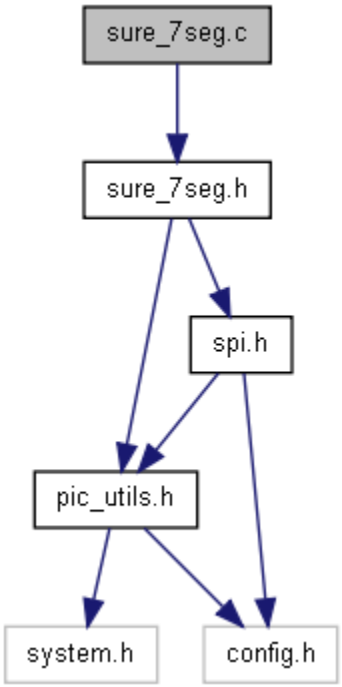

---

## sure_7seg.c File Reference

Routines to talk to Sure electronics seven segment displays.

```
#include "sure_7seg.h"
```

Include dependency graph for sure_7seg.c:



## Functions

- uns8 sure_7seg_convert (uns8 digit)
- void sure_7seg_setup ()
- *Setup ports and pins to communicate to Sure display.* void sure_7seg_write_str (char *data)

*Display ASCII string to 7 segment displays.*

---

## Detailed Description

Definition in file sure_7seg.c.

---

## Function Documentation

### uns8 sure_7seg_convert (uns8 *digit*)

Definition at line 48 of file sure_7seg.c.

Referenced by sure_7seg_write_str().

Here is the caller graph for this function:

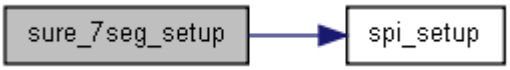

### void sure_7seg_setup ()

Set up ports and pins as appropriate to communicate via SPI to Sure 7 segment displays

Definition at line 43 of file sure_7seg.c.

References spi_setup().

Here is the call graph for this function:




### void sure_7seg_write_str (char * *data*)

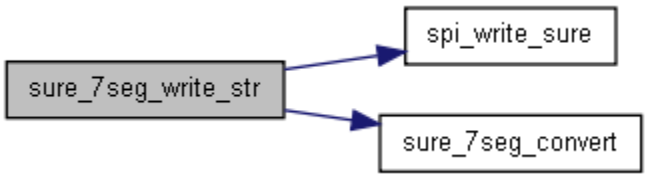
Converts ASCII to the appropriate magic characters to display on a Sure 7 segment display. Only numbers 0-9 and space are implented for now.

To create your own characters, add together: 0x40 -------- | | 0x02 | | 0x20 | 0x01 | |------| | | 0x04 | | 0x10 | | -------- 0x08

Definition at line 64 of file sure_7seg.c.

References clear_pin, set_pin, spi_write_sure(), sure_7seg_convert(), and uns8.

Here is the call graph for this function:

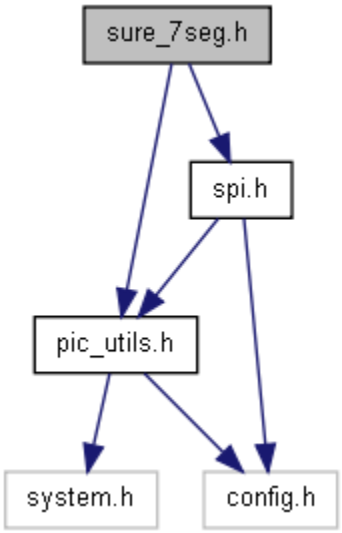


# sure_7seg.h File Reference

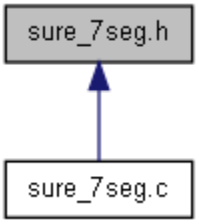
Routines to talk to Sure electronics seven segment displays.

```
#include "spi.h"
#include "pic_utils.h"
```

Include dependency graph for sure_7seg.h:




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




## Functions

- void sure_7seg_setup ()
- *Setup ports and pins to communicate to Sure display.* void sure_7seg_write_str (char *data)

*Display ASCII string to 7 segment displays.*

## Detailed Description

Definition in file sure_7seg.h.

## Function Documentation

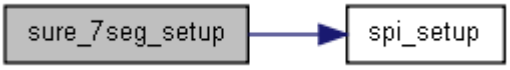
### void sure_7seg_setup ()

Set up ports and pins as appropriate to communicate via SPI to Sure 7 segment displays

Definition at line 43 of file sure_7seg.c.

References spi_setup().

Here is the call graph for this function:




### void sure_7seg_write_str (char * *data*)

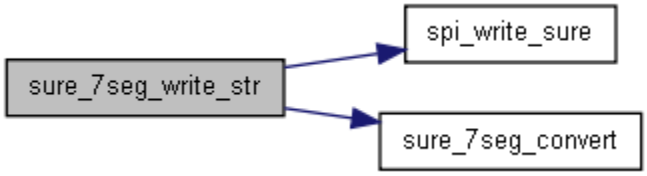
Converts ASCII to the appropriate magic characters to display on a Sure 7 segment display. Only numbers 0-9 and space are implented for now.

To create your own characters, add together: 0x40 -------- | | 0x02 | | 0x20 | 0x01 | |------| | | 0x04 | | 0x10 | | -------- 0x08

Definition at line 64 of file sure_7seg.c.

References clear_pin, set_pin, spi_write_sure(), sure_7seg_convert(), and uns8.

Here is the call graph for this function:



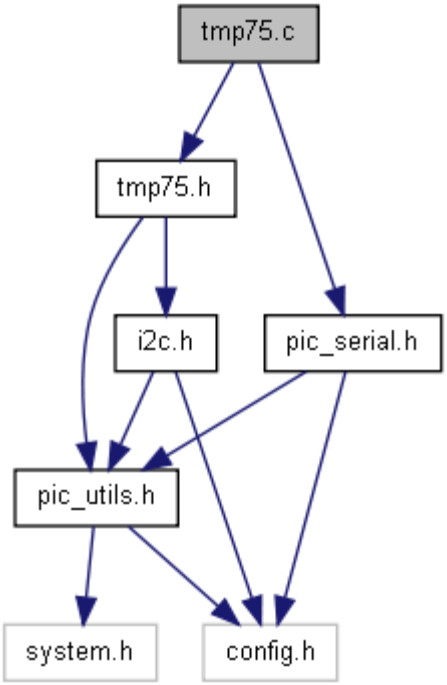

---

# tmp75.c File Reference

Routines to access TMP75 temperature sensor.

```
#include "tmp75.h"
#include "pic_serial.h"
```

Include dependency graph for tmp75.c:




## Functions

- void tmp75_convert_temp (uns8 addr)
- *Start temperature conversion on tmp75.* uns8 tmp75_get_config (uns8 addr)
- *Get tmp75 config register.* uns16 tmp75_get_temp (uns8 addr)
- *Read temperature from tmp75.* uns8 tmp75_read (uns8 addr, uns8 pointer)
- uns16 tmp75_read_16bit (uns8 addr, uns8 pointer)
- void tmp75_set_config (uns8 addr, uns8 config)
- *Set tmp75 config register.* void tmp75_setup ()
- uns8 tmp75_write (uns8 addr, uns8 pointer, uns8 data)

---

## Detailed Description

Definition in file tmp75.c.

---

## Function Documentation

### void tmp75_convert_temp (uns8 *addr*)

This routine starts the temperature conversion in the tmp75. Issue this command before actually reading the temperature.

needs fixing

Definition at line 107 of file tmp75.c.

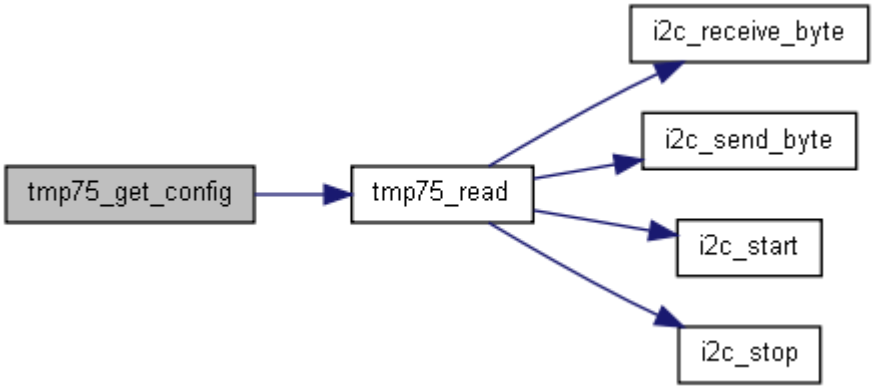
**uns8 tmp75_get_config (uns8 *addr*)**

Gets the tmp75 config register (memory location 0x01)

Definition at line 102 of file tmp75.c.

References TMP75_CONFIG_REGISTER, and tmp75_read().

Here is the call graph for this function:



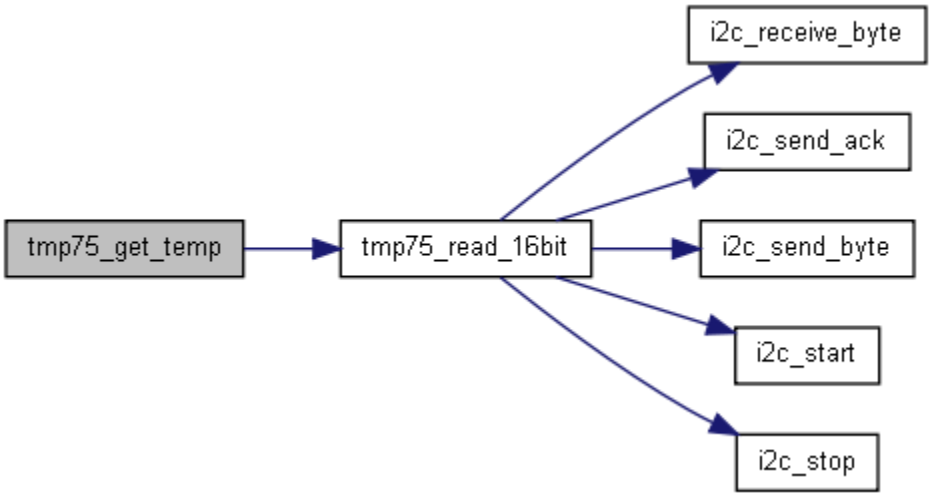

**uns16 tmp75_get_temp (uns8 *addr*)**

Returns 16bit raw temperature register from tmp75.

Definition at line 115 of file tmp75.c.

References tmp75_read_16bit(), and TMP75_TEMP_REGISTER.

Here is the call graph for this function:



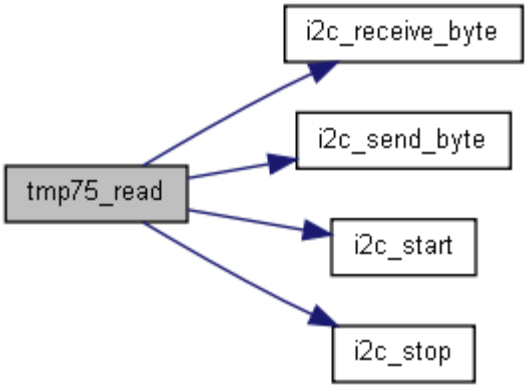

**uns8 tmp75_read (uns8 *addr*, uns8 *pointer*)**

Definition at line 42 of file tmp75.c.

References i2c_receive_byte(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns8.

Referenced by tmp75_get_config().

Here is the call graph for this function:




Here is the caller graph for this function:

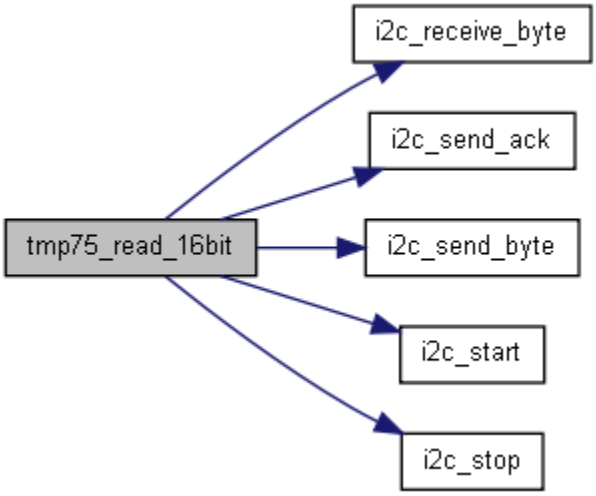

**uns16 tmp75_read_16bit (uns8 *addr*, uns8 *pointer*)**

Definition at line 73 of file tmp75.c.

References i2c_receive_byte(), i2c_send_ack(), i2c_send_byte(), i2c_start(), i2c_stop(), and uns16.

Referenced by tmp75_get_temp().

Here is the call graph for this function:




Here is the caller graph for this function:

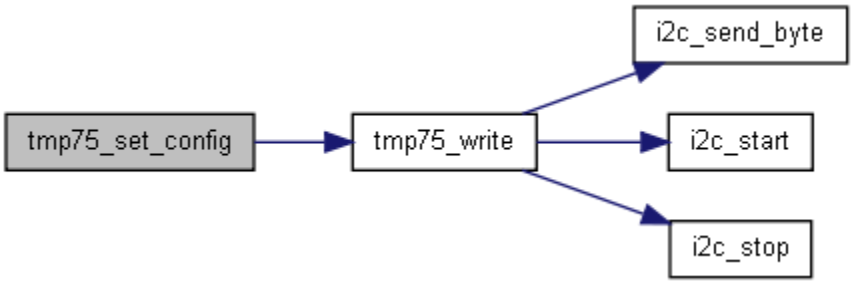

**void tmp75_set_config (uns8 *addr*, uns8 *config*)**

Sets the tmp75 config register

Definition at line 97 of file tmp75.c.

References TMP75_CONFIG_REGISTER, and tmp75_write().

Here is the call graph for this function:




**void tmp75_setup ()**

Definition at line 93 of file tmp75.c.

References i2c_setup.

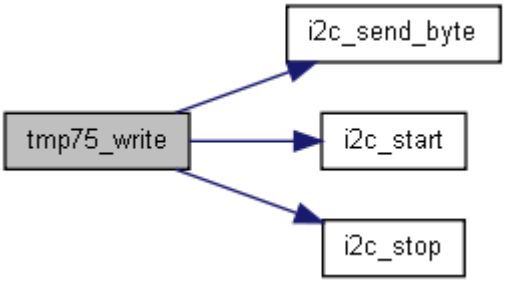
**uns8 tmp75_write (uns8 *addr*, uns8 *pointer*, uns8 *data*)**

Definition at line 61 of file tmp75.c.

References i2c_send_byte(), i2c_start(), and i2c_stop().

Referenced by tmp75_set_config().

Here is the call graph for this function:




Here is the caller graph for this function:

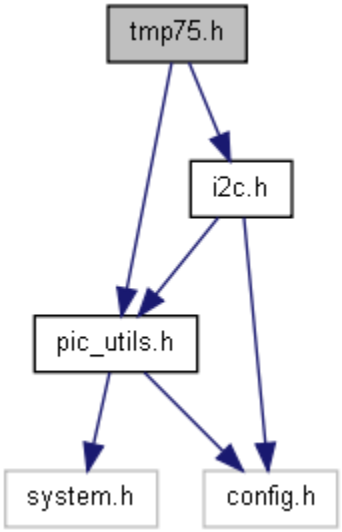

---

## tmp75.h File Reference

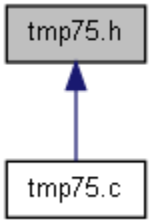
Routines to access TMP75 temperature sensor.

```
#include "pic_utils.h"
#include "i2c.h"
```

Include dependency graph for tmp75.h:




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




## Defines

- #define TMP75_CONF_F0 3
- #define TMP75_CONF_F1 4
- #define TMP75_CONF_OS 7
- #define TMP75_CONF_POL 2
- #define TMP75_CONF_R0 5
- #define TMP75_CONF_R1 6
- #define TMP75_CONF_SD 0
- #define TMP75_CONF_TM 1
- #define TMP75_CONFIG_REGISTER 0b00000001
- #define tmp75_setup() tmp75_setup_io()
- *Setup tmp75 ports and pins.* #define TMP75_TEMP_REGISTER 0b00000000
- #define TMP75_THI_REGISTER 0b00000011
- #define TMP75_TLOW_REGISTER 0b00000010

## Functions

- void tmp75_convert_temp (uns8 addr)
- *Start temperature conversion on tmp75.* uns8 tmp75_get_config (uns8 addr)
- *Get tmp75 config register.* uns16 tmp75_get_temp (uns8 addr)
- *Read temperature from tmp75.* void tmp75_set_config (uns8 addr, uns8 config)
- *Set tmp75 config register.* void tmp75_setup_io (void)

## Detailed Description

Put the following in your config.h

```
// - - - - - - - - - - - - - - - - - - - -
// TMP75 defines
// - - - - - - - - - - - - - - - - - - - -

#define TMP75_ADDR  0x00
```

Definition in file tmp75.h.

---

## Define Documentation

### #define TMP75_CONF_F0 3

Definition at line 62 of file tmp75.h.

### #define TMP75_CONF_F1 4

Definition at line 61 of file tmp75.h.

### #define TMP75_CONF_OS 7

Definition at line 58 of file tmp75.h.

### #define TMP75_CONF_POL 2

Definition at line 63 of file tmp75.h.

### #define TMP75_CONF_R0 5

Definition at line 60 of file tmp75.h.

### #define TMP75_CONF_R1 6

Definition at line 59 of file tmp75.h.

### #define TMP75_CONF_SD 0

Definition at line 65 of file tmp75.h.

### #define TMP75_CONF_TM 1

Definition at line 64 of file tmp75.h.

**#define TMP75_CONFIG_REGISTER 0b00000001**

Definition at line 54 of file tmp75.h.

Referenced by tmp75_get_config(), and tmp75_set_config().

**#define tmp75_setup() tmp75_setup_io()**

Definition at line 74 of file tmp75.h.

**#define TMP75_TEMP_REGISTER 0b00000000**

Definition at line 53 of file tmp75.h.

Referenced by tmp75_get_temp().

**#define TMP75_THI_REGISTER 0b00000011**

Definition at line 56 of file tmp75.h.

**#define TMP75_TLOW_REGISTER 0b00000010**

Definition at line 55 of file tmp75.h.

---

## Function Documentation

**void tmp75_convert_temp (uns8 *addr*)**

This routine starts the temperature conversion in the tmp75. Issue this command before actually reading the temperature.

needs fixing

Definition at line 107 of file tmp75.c.

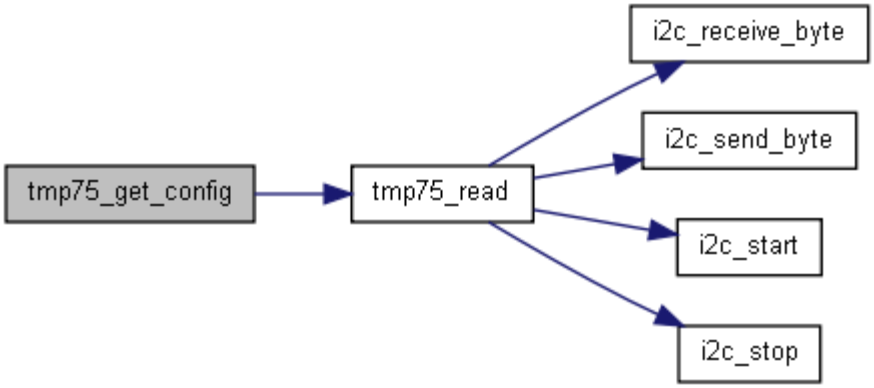
**uns8 tmp75_get_config (uns8 *addr*)**

Gets the tmp75 config register (memory location 0x01)

Definition at line 102 of file tmp75.c.

References TMP75_CONFIG_REGISTER, and tmp75_read().

Here is the call graph for this function:



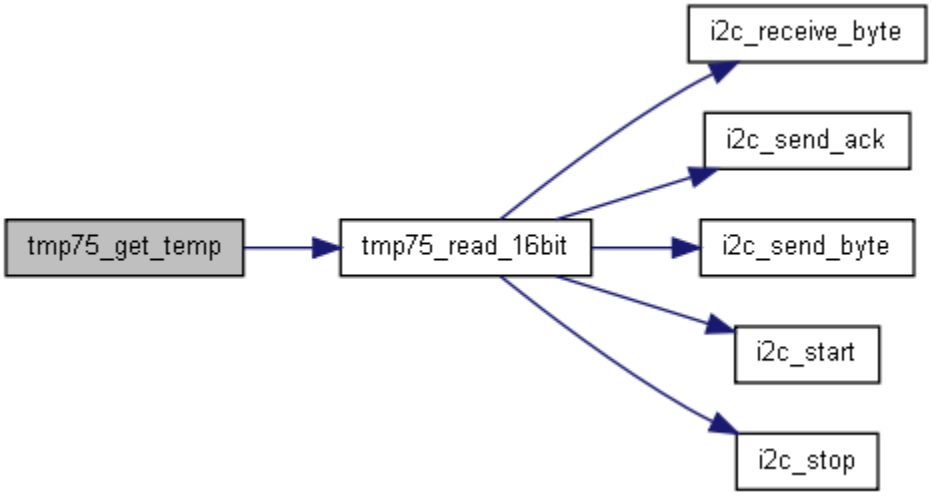

**uns16 tmp75_get_temp (uns8 *addr*)**

Returns 16bit raw temperature register from tmp75.

Definition at line 115 of file tmp75.c.

References tmp75_read_16bit(), and TMP75_TEMP_REGISTER.

Here is the call graph for this function:



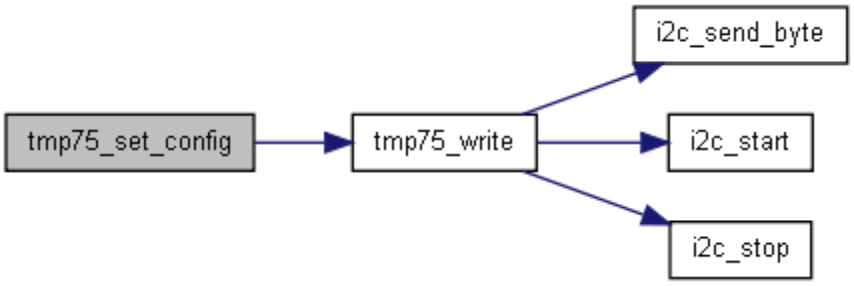

**void tmp75_set_config (uns8 *addr*, uns8 *config*)**

Sets the tmp75 config register

Definition at line 97 of file tmp75.c.

References TMP75_CONFIG_REGISTER, and tmp75_write().

Here is the call graph for this function:




**void tmp75_setup_io (void)**

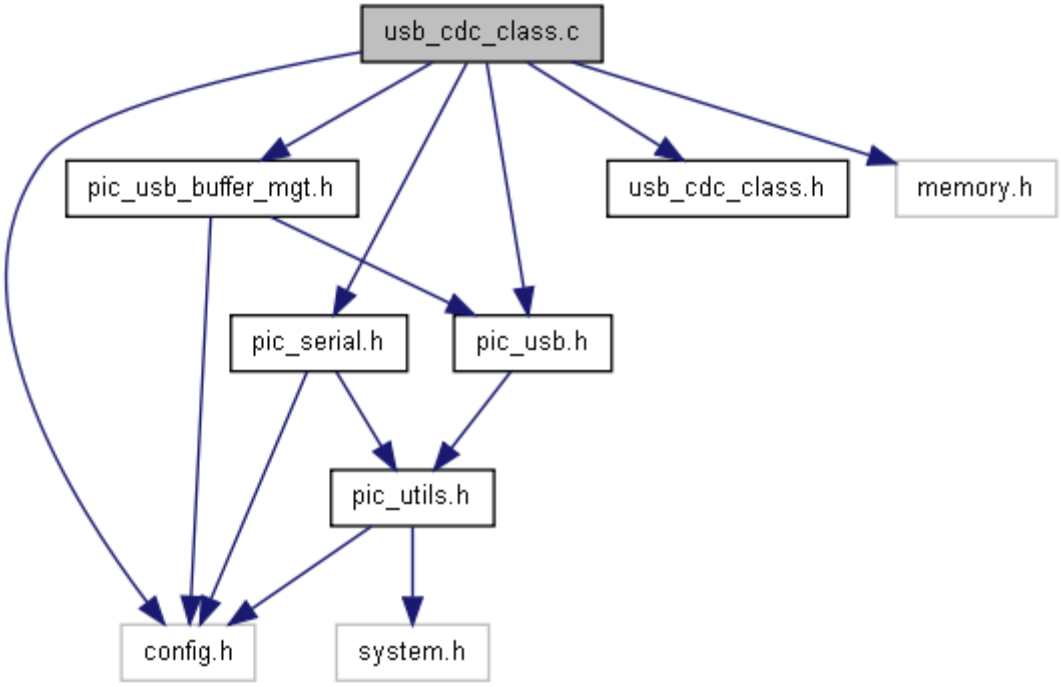
---

## usb_cdc_class.c File Reference

Pic CDC USB routines.

```
#include "config.h"
#include "pic_usb.h"
#include "pic_usb_buffer_mgt.h"
#include "pic_serial.h"
#include "usb_cdc_class.h"
#include "memory.h"
```

Include dependency graph for usb_cdc_class.c:



## Data Structures

- struct line_coding
- union long_union

## Defines

- #define not_SERIAL_STATE 0xa1
- #define req_CLEAR_COMM_FEATURE 0x04
- #define req_GET_COMM_FEATURE 0x03
- #define req_GET_ENCAPSULATED_RESPONSE 0x01
- #define req_GET_LINE_CODING 0x21
- #define req_SEND_BREAK 0x23
- #define req_SEND_ENCAPSULATED_COMMAND 0x00
- #define req_SET_COMM_FEATURE 0x02
- #define req_SET_CONTROL_LINE_STATE 0x22
- #define req_SET_LINE_CODING 0x20

## Functions

- uns8 usb_cdc_getc (void)
- *Retrieve a received character from the USB serial port.* void usb_cdc_handle_tx ()
- *Transmit any buffered characters over ther USB virtual serial port.* void usb_cdc_print_int (uns16 i)
- *Print a 16 bit number to the USB virtual serial port.* void usb_cdc_print_str (char *str)
- *Print a string out to the USB virtual serial port.* void usb_cdc_putc (uns8 c)
- *Sends a single character to the USB serial port.* uns8 usb_cdc_rx_avail ()
- *Check to see if a character is available in the receive buffer.* void usb_cdc_set_dcd ()
- void usb_cdc_set_dsr ()
- void usb_cdc_setup ()
- *Set up data structures ready for USB CDC use.* uns8 usb_cdc_tx_empty ()
- *Check to see if the transmit buffer is empty.* void usb_ep_data_in_callback (uns8 end_point, uns16 byte_count)
- *Callback routine triggered when data has been sent to the host.* void usb_ep_data_out_callback (uns8 end_point, uns8 *buffer, uns16 byte_count)
- *Callback routine triggered when data has been sent to the device.* void usb_handle_class_ctrl_read_callback ()
- *Callback routine for a class control read.* void usb_handle_class_ctrl_write_callback (uns8 *data, uns16 count)
- *Callback routine for a class control write.* void usb_handle_class_request_callback (setup_data_packet sdp)
- *Callback routine for a control transfer request that is placed on the class.* void usb_SOF_callback (uns16 frame)

## *Callback routine triggered each time a start of frame (SOF) has been received.*

## Variables

- uns8 cdc_rx_buffer [USB_CDC_RX_BUFFER_SIZE]
- uns8 cdc_rx_end = 0
- uns8 cdc_rx_start = 0
- uns8 cdc_tx_buffer [USB_CDC_TX_BUFFER_SIZE]
- uns8 cdc_tx_end = 0
- uns8 cdc_tx_start = 0
- uns8 class_data [8]
- uns16 current_bit_rate
- long_union dte_rate

---

## Detailed Description

Communication Device Class (Serial Port) USB routines

Definition in file usb_cdc_class.c.

---

## Define Documentation

### #define not_SERIAL_STATE 0xa1

Definition at line 74 of file usb_cdc_class.c.

### #define req_CLEAR_COMM_FEATURE 0x04

Definition at line 68 of file usb_cdc_class.c.

### #define req_GET_COMM_FEATURE 0x03

Definition at line 67 of file usb_cdc_class.c.

### #define req_GET_ENCAPSULATED_RESPONSE 0x01

Definition at line 65 of file usb_cdc_class.c.

### #define req_GET_LINE_CODING 0x21

Definition at line 70 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_read_callback(), and usb_handle_class_request_callback().

### #define req_SEND_BREAK 0x23

Definition at line 72 of file usb_cdc_class.c.

### #define req_SEND_ENCAPSULATED_COMMAND 0x00

Definition at line 64 of file usb_cdc_class.c.

### #define req_SET_COMM_FEATURE 0x02

Definition at line 66 of file usb_cdc_class.c.

### #define req_SET_CONTROL_LINE_STATE 0x22

Definition at line 71 of file usb_cdc_class.c.

Referenced by usb_handle_class_request_callback().

#### #define req_SET_LINE_CODING 0x20

Definition at line 69 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback(), and usb_handle_class_request_callback().

---

## Function Documentation

### uns8 usb_cdc_getc ()

Receive a character from the USB serial port. If a character has not been received, this routine will wait indefinately.

**Returns:**

Byte from the receive buffer.

Definition at line 371 of file usb_cdc_class.c.

References cdc_rx_buffer, cdc_rx_end, cdc_rx_start, end_crit_sec, start_crit_sec, and uns8.

### void usb_cdc_handle_tx ()

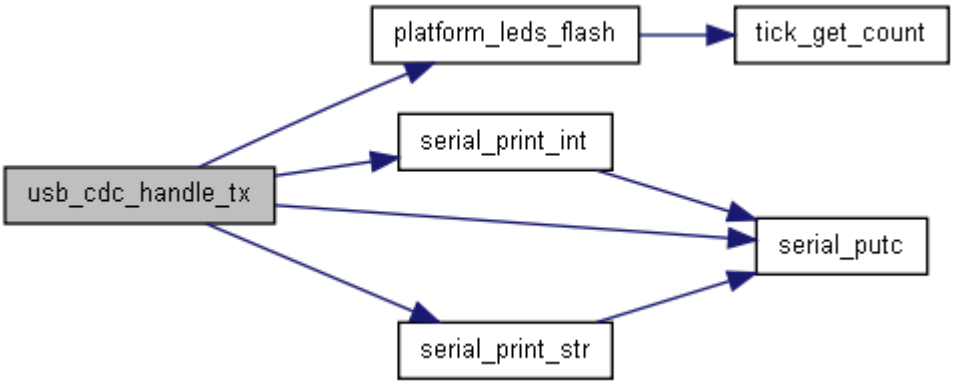
Generally only used internally, this routine will attempt to place any characters in the transmit queue into the USB buffers. It will fail gracefully if the USB buffer is already owned by the SIE.

Definition at line 392 of file usb_cdc_class.c.

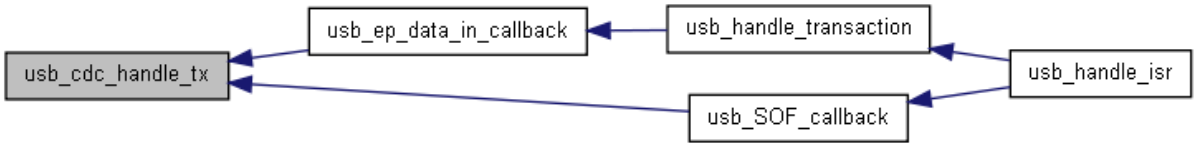
References buffer_descriptor::addr, BC8, BC9, BSTALL, cdc_tx_buffer, cdc_tx_end, cdc_tx_start, buffer_descriptor::count, DTS, DTSEN, end_crit_sec, ep_in_bd_location, ep_in_buffer_location, ep_in_buffer_size, INCDIS, KEN, platform_leds_flash(), serial_print_int(), serial_print_str(), serial_putc(), start_crit_sec, buffer_descriptor::stat, uns16, uns8, and UOWN.

Referenced by usb_ep_data_in_callback(), and usb_SOF_callback().

Here is the call graph for this function:




Here is the caller graph for this function:



### void usb_cdc_print_int (uns16 *i*)

Print a 16 bit unsigned number in decimal to the USB virtual serial port

**Parameters:**

*i* the 16 bit number to be printed

Definition at line 491 of file usb_cdc_class.c.

References uns8, and usb_cdc_putc().

Here is the call graph for this function:


### void usb_cdc_print_str (char * *str*)

Send a null terminated string out the virtual serial port

**Parameters:**

*str* the string to be sent

Definition at line 459 of file usb_cdc_class.c.

References uns8, and usb_cdc_putc().

Here is the call graph for this function:


### void usb_cdc_putc (uns8 *c*)

Deliver a single character out the virtual serial port. This routine will add the character to the transmit buffer. The actual buffer will be physically sent on the Start Of Frame (SOF) interrupt (each 1ms) or on the end point interrupt - ie, when the last character or chunk of characters was sent.

**Parameters:**

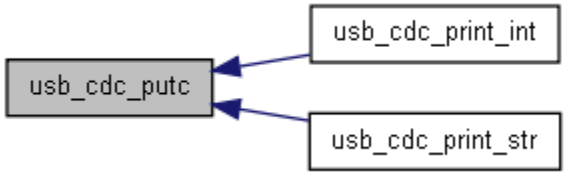
*c* The 8 bit byte to be transmitted.

Definition at line 343 of file usb_cdc_class.c.

References cdc_tx_buffer, cdc_tx_end, cdc_tx_start, end_crit_sec, start_crit_sec, and uns8.

Referenced by usb_cdc_print_int(), and usb_cdc_print_str().

Here is the caller graph for this function:




### uns8 usb_cdc_rx_avail ()

If one or more bytes are available in the USB serial port receive buffer, this routine will return true. If there are no bytes available, it will return false.

**Returns:**

True if buffer is not empty, False if buffer is empty.

Definition at line 456 of file usb_cdc_class.c.

References cdc_rx_end, and cdc_rx_start.

### void usb_cdc_set_dcd ()

Definition at line 335 of file usb_cdc_class.c.

### void usb_cdc_set_dsr ()

Definition at line 338 of file usb_cdc_class.c.

### void usb_cdc_setup ()

Configures the default DTE rate, stop bits etc.

Definition at line 483 of file usb_cdc_class.c.

References long_union::as_long, and current_bit_rate.

### uns8 usb_cdc_tx_empty ()

Sometimes it is useful to see if the transmit buffer is empty, since then you can be sure your data is well on its way. In the case of USB, this means that the data has been at least placed into the outbound USB buffer; it's not possible to tell until after the fact if the data has actually been squirted out the USB port.

**Returns:**

True if transmit buffer is empty, False if buffer still has data in it.

Definition at line 457 of file usb_cdc_class.c.

References cdc_tx_end, and cdc_tx_start.

### void usb_ep_data_in_callback (uns8 *end_point*, uns16 *byte_count*)

If you have called usb_send_data to transfer data to the host, this routine will be fired once this data has been transferred. You may send more data by using usb_send_data(). Since the current PicPack USB library supports only single buffering, transfer speed is limited by how quickly you can refill the buffer again. In the future, we may support double buffering (ping poing buffering) which will most likely improve transfer speeds (to be fair, transfer performance has not been a limiting factor in tests so far).

In order for this callback to be triggered, you must define USB_EP_DATA_CALLBACK in your config.h

**Parameters:**

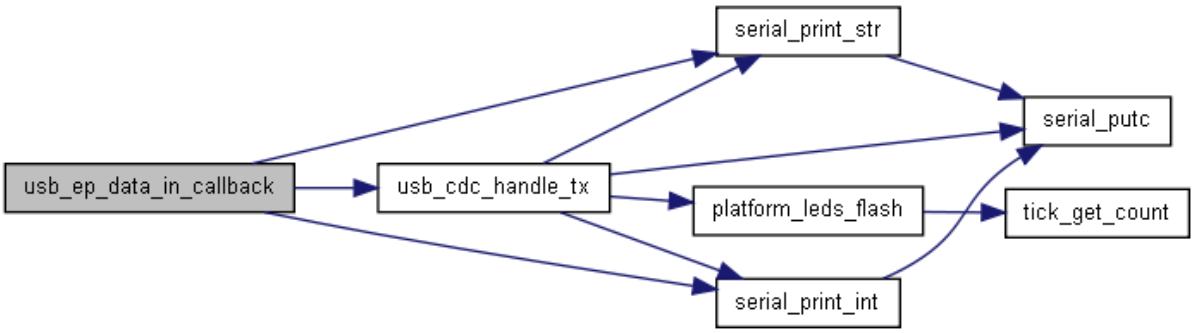
*end_point* The endpoint on which the data was transferred
*byte_count* The number of bytes that were actually transferred

Definition at line 323 of file usb_cdc_class.c.

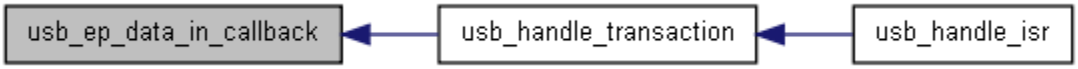
References serial_print_int(), serial_print_str(), and usb_cdc_handle_tx().

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:




**void usb_ep_data_out_callback (uns8 *end_point*, uns8 * *buffer_location*, uns16 *byte_count*)**

If data is sent to the device and the endpoint is not endpoint 0 (the control transfer endpoint) then this routine is called. Since the routine is passed the actual hardware buffer location, it is important to pull data out of the buffer as soon as possible in order to free up the buffer to receive more data. The buffer is re-primed only once this routine completes since PicPack only supports single-buffered mode. In the future, we may look at supporting double buffering (ping-pong buffering) in order to be able to receive more data even while this routine is being called.

In order for this callback to be triggered, you must define USB_EP_DATA_CALLBACK in your config.h

**Parameters:**

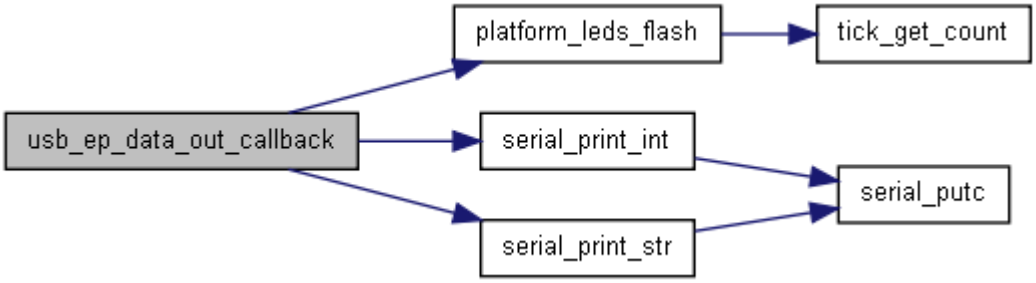
*end_point* The endpoint the data was sent do
*buffer_lcoation* The memory location of the USB buffer where the data was received into
*byte_count* The number of bytes received

Definition at line 283 of file usb_cdc_class.c.

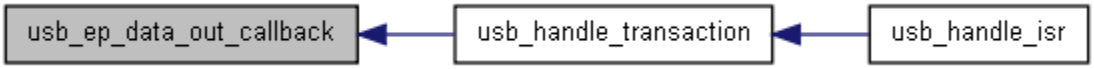
References cdc_rx_buffer, cdc_rx_end, cdc_rx_start, platform_leds_flash(), serial_print_int(), serial_print_str(), and uns8.

Referenced by usb_handle_transaction().

Here is the call graph for this function:




Here is the caller graph for this function:



**void usb_handle_class_ctrl_read_callback ()**

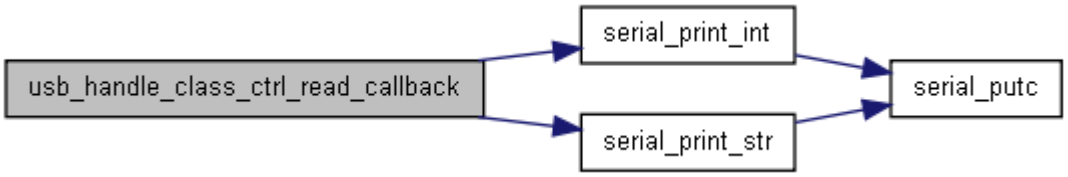
When a control transfer is taking place, this routine is called to indicate that a control read for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been transferred to the host. If you wish to send more data to the host, use usb_send_data(), or if your control read has sent all the data required, you will need to indicate that the state has changed by setting the control_mode variable to cm_CTRL_READ_AWAITING_STATUS. This will indicate to the stack that it should now wait for the status packet before completing the control transfer.

To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 268 of file usb_cdc_class.c.

References setup_data_packet::bRequest, cm_CTRL_READ_AWAITING_STATUS, control_mode, req_GET_LINE_CODING, serial_print_int(), serial_print_str(), and usb_sdp.

Here is the call graph for this function:




**void usb_handle_class_ctrl_write_callback (uns8 * *data*, uns16 *count*)**

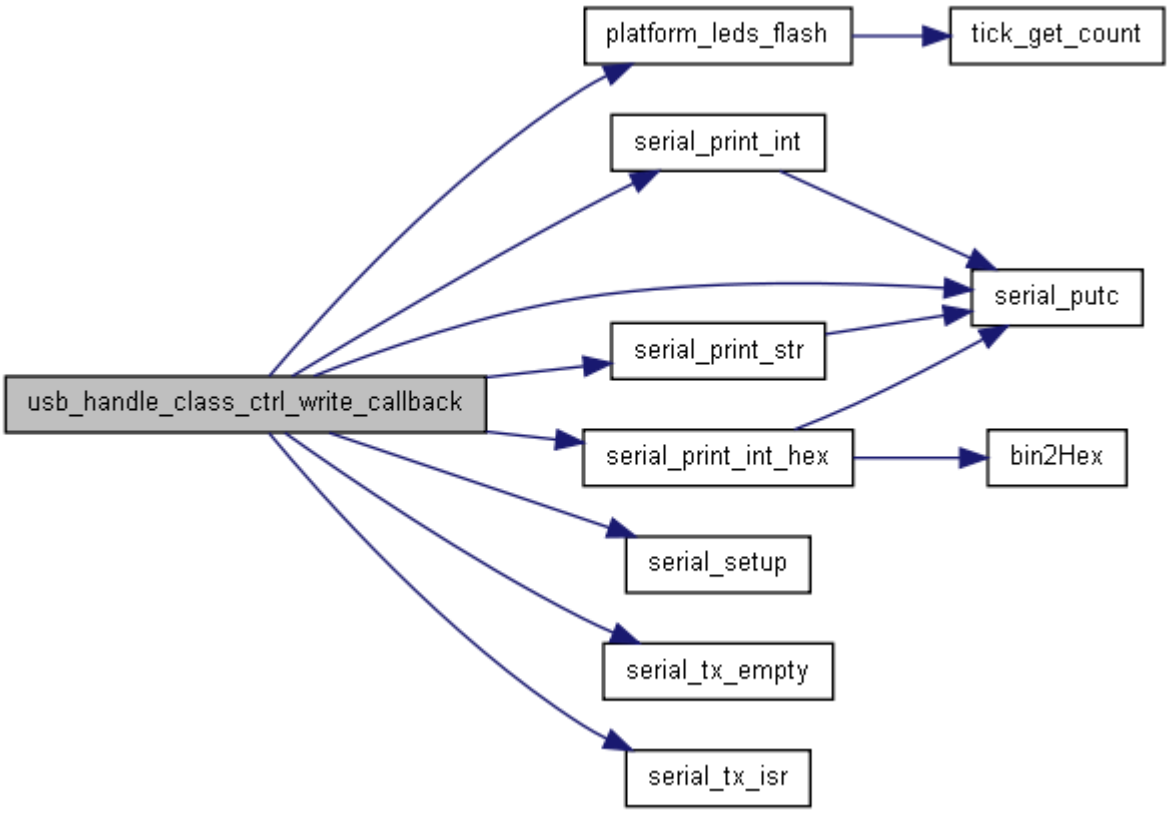
When a control transfer is taking place, this routine is called to indicate that a control write for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been received by the device. If you expect more data from the host, it will arrive in due course since endpoint 0 will be primed for more data automatically. If you have received all the data from the host, you will need to set the control_mode state variable to cm_CTRL_WRITE_SENDING_STATUS and then actually send the status by calling usb_send_status_ack(). Once the status has actually been sent, the control_mode state will automatically change to cm_IDLE to indicate the transfer has completed.

To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 168 of file usb_cdc_class.c.

References long_union::as_byte_array, long_union::as_long, setup_data_packet::bRequest, class_data, cm_CTRL_WRITE_SENDING_STATUS, control_mode, current_bit_rate, line_coding::data_bits, line_coding::dte_rate, line_coding::parity, platform_leds_flash(), req_SET_LINE_CODING, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), serial_setup(), serial_tx_empty(), serial_tx_isr(), line_coding::stop_bits, usb_sdp, and usb_send_status_ack.

Here is the call graph for this function:




**void usb_handle_class_request_callback (setup_data_packet *sdp*)**

After receiving a setup packet, where the request is placed on the class, this routine is called. In usb_handle_class_request_callback, you can set up ready for the data stage of the control transfer. The direction of the data stage can be determined by examining test_bit(sdp.bRequest, DATA_STAGE_DIR) although generally it appears to be obvious from the request. The request is stored in sdp.bRequest.

Typically, if it is a control read transfer (that is, it is a request by the host for data), then you will need to move the control_mode state variable to cm_CTRL_READ_DATA_STAGE_CLASS and send data using usb_send_data(). If you only intend to send one packet, you can immediately move the control_mode state variable to cm_CTRL_READ_AWAITING_STATUS to indicate you are waiting for the status to arrive. You could wait for the usb_handle_class_ctrl_read callback and do it (move to cm_CTROL_READ_AWAITING_STATUS) but the PicPack USB stack can handle the control read event for you if you've already switched states.

If it is a control write transfer (that is, it is a request by the host to send data to the device), then you will need to move the control_mode state variable to cm_CTRL_WRITE_DATA_STAGE_CLASS. Then, the usb_handle_class_ctrl_write will be fired when data is received by the device in the data stage.

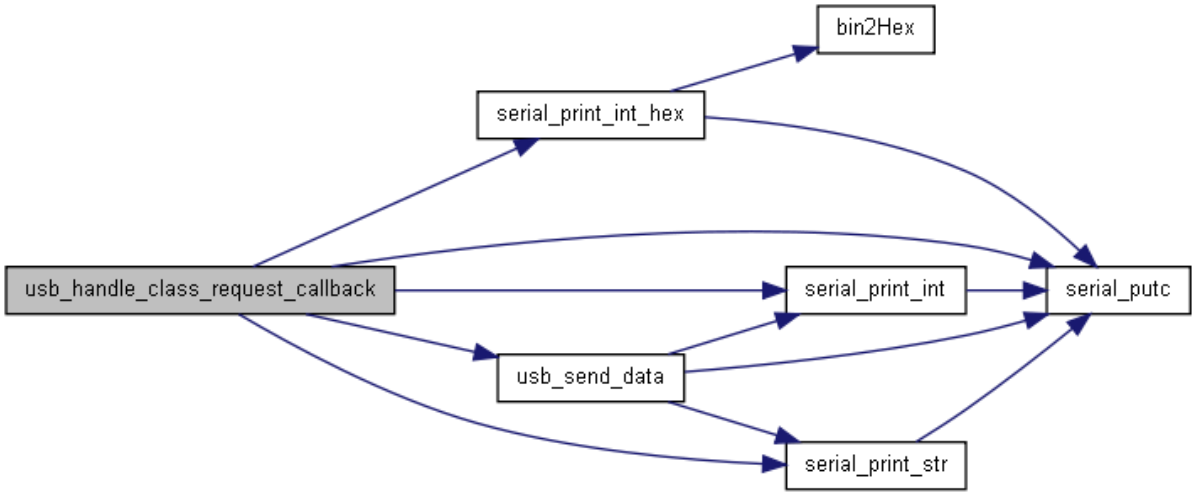
To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 112 of file usb_cdc_class.c.

References long_union::as_byte_array, setup_data_packet::bRequest, cm_CTRL_READ_AWAITING_STATUS, cm_CTRL_READ_DATA_STAGE_CLASS,

cm_CTRL_WRITE_DATA_STAGE_CLASS, cm_CTRL_WRITE_SENDING_STATUS, control_mode, line_coding::data_bits, line_coding::dte_rate, line_coding::parity, req_GET_LINE_CODING, req_SET_CONTROL_LINE_STATE, req_SET_LINE_CODING, serial_print_int(), serial_print_int_hex(), serial_print_str(), serial_putc(), line_coding::stop_bits, uns8, usb_send_data(), usb_send_status_ack, setup_data_packet::wLength, and setup_data_packet::wValue.

Here is the call graph for this function:




### void usb_SOF_callback (uns16 *frame*)

Frames in USB occur each 1ms. A SOF packet is sent to each device at the start of each frame. This is a really neat way of getting a 1ms timer without any further work.

**Parameters:**

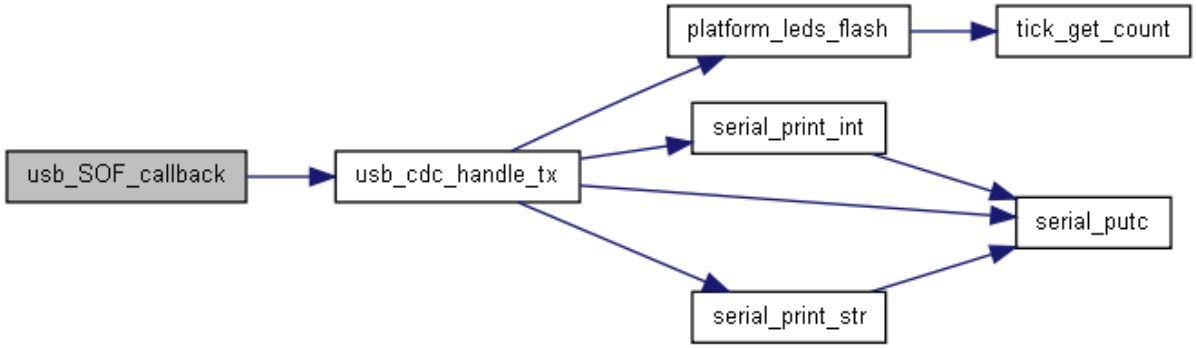
*frame* The frame number. Frames will wrap at 65535.

Definition at line 478 of file usb_cdc_class.c.

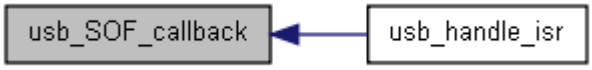
References usb_cdc_handle_tx().

Referenced by usb_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:



## Variable Documentation

**uns8 cdc_rx_buffer[USB_CDC_RX_BUFFER_SIZE]**

Receive fifo

Definition at line 102 of file usb_cdc_class.c.

Referenced by usb_cdc_getc(), and usb_ep_data_out_callback().

**uns8 cdc_rx_end = 0**

Receive fifo end point

Definition at line 106 of file usb_cdc_class.c.

Referenced by usb_cdc_getc(), usb_cdc_rx_avail(), and usb_ep_data_out_callback().

**uns8 cdc_rx_start = 0**

Receive fifo start point

Definition at line 104 of file usb_cdc_class.c.

Referenced by usb_cdc_getc(), usb_cdc_rx_avail(), and usb_ep_data_out_callback().

**uns8 cdc_tx_buffer[USB_CDC_TX_BUFFER_SIZE]**

Transmit fifo

Definition at line 95 of file usb_cdc_class.c.

Referenced by usb_cdc_handle_tx(), and usb_cdc_putc().

**uns8 cdc_tx_end = 0**

Transmit fifo end point

Definition at line 99 of file usb_cdc_class.c.

Referenced by usb_cdc_handle_tx(), usb_cdc_putc(), and usb_cdc_tx_empty().

**uns8 cdc_tx_start = 0**

Transmit fifo start point

Definition at line 97 of file usb_cdc_class.c.

Referenced by usb_cdc_handle_tx(), usb_cdc_putc(), and usb_cdc_tx_empty().

**uns8 class_data[8]**

Definition at line 109 of file usb_cdc_class.c.

Referenced by usb_handle_class_ctrl_write_callback().

**uns16 current_bit_rate**

Definition at line 91 of file usb_cdc_class.c.

Referenced by usb_cdc_setup(), and usb_handle_class_ctrl_write_callback().

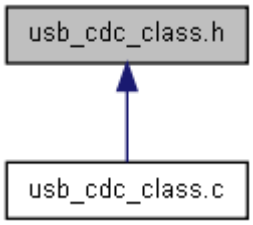
**long_union dte_rate**

Definition at line 92 of file usb_cdc_class.c.

---

# usb_cdc_class.h File Reference

USB Communications Device Class (Serial Port) routines.
This graph shows which files directly or indirectly include this file:




## Functions

- uns8 usb_cdc_getc ()
- *Retrieve a received character from the USB serial port.* void usb_cdc_handle_tx ()
- *Transmit any buffered characters over ther USB virtual serial port.* void usb_cdc_print_int (uns16 i)
- *Print a 16 bit number to the USB virtual serial port.* void usb_cdc_print_str (char *str)
- *Print a string out to the USB virtual serial port.* void usb_cdc_putc (uns8 c)
- *Sends a single character to the USB serial port.* uns8 usb_cdc_rx_avail ()
- *Check to see if a character is available in the receive buffer.* void usb_cdc_setup ()
- *Set up data structures ready for USB CDC use.* uns8 usb_cdc_tx_empty ()

*Check to see if the transmit buffer is empty.*

---

## Detailed Description

Definition in file usb_cdc_class.h.

---

## Function Documentation

### uns8 usb_cdc_getc ()

Receive a character from the USB serial port. If a character has not been received, this routine will wait indefinately.

**Returns:**

Byte from the receive buffer.

Definition at line 371 of file usb_cdc_class.c.

References cdc_rx_buffer, cdc_rx_end, cdc_rx_start, end_crit_sec, start_crit_sec, and uns8.

### void usb_cdc_handle_tx ()

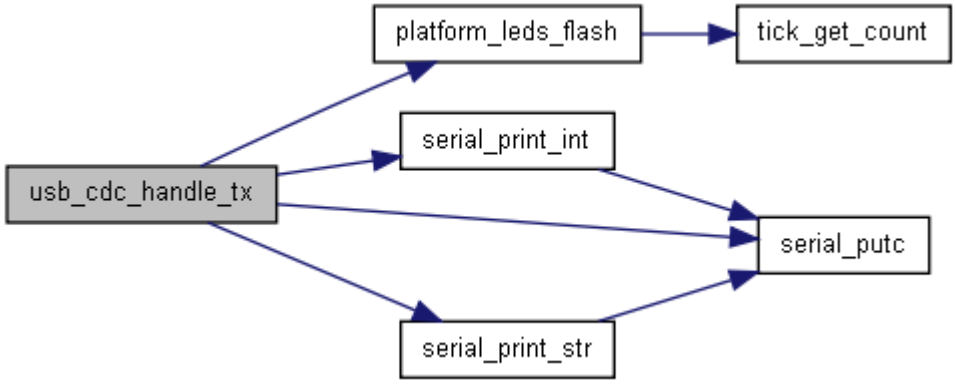
Generally only used internally, this routine will attempt to place any characters in the transmit queue into the USB buffers. It will fail gracefully if the USB buffer is already owned by the SIE.

Definition at line 392 of file usb_cdc_class.c.

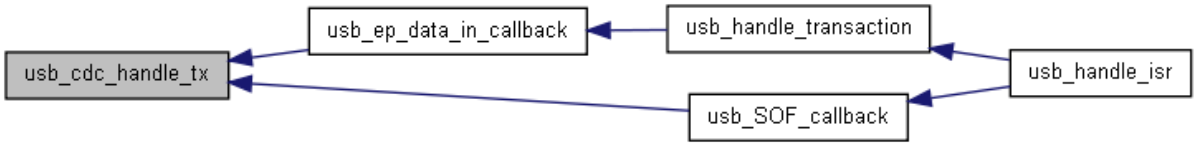
References buffer_descriptor::addr, BC8, BC9, BSTALL, cdc_tx_buffer, cdc_tx_end, cdc_tx_start, buffer_descriptor::count, DTS, DTSEN, end_crit_sec, ep_in_bd_location, ep_in_buffer_location, ep_in_buffer_size, INCDIS, KEN, platform_leds_flash(), serial_print_int(), serial_print_str(), serial_putc(), start_crit_sec, buffer_descriptor::stat, uns16, uns8, and UOWN.

Referenced by usb_ep_data_in_callback(), and usb_SOF_callback().

Here is the call graph for this function:




Here is the caller graph for this function:




### void usb_cdc_print_int (uns16 *i*)

Print a 16 bit unsigned number in decimal to the USB virtual serial port

**Parameters:**

*i* the 16 bit number to be printed

Definition at line 491 of file usb_cdc_class.c.

References uns8, and usb_cdc_putc().

Here is the call graph for this function:


### void usb_cdc_print_str (char * *str*)

Send a null terminated string out the virtual serial port

**Parameters:**

*str* the string to be sent

Definition at line 459 of file usb_cdc_class.c.

References uns8, and usb_cdc_putc().

Here is the call graph for this function:


### void usb_cdc_putc (uns8 *c*)

Deliver a single character out the virtual serial port. This routine will add the character to the transmit buffer. The actual buffer will be physically sent on the Start Of Frame (SOF) interrupt (each 1ms) or on the end point interrupt - ie, when the last character or chunk of characters was sent.

**Parameters:**

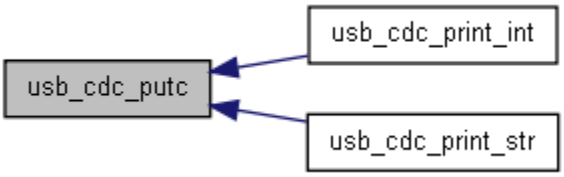
*c* The 8 bit byte to be transmitted.

Definition at line 343 of file usb_cdc_class.c.

References cdc_tx_buffer, cdc_tx_end, cdc_tx_start, end_crit_sec, start_crit_sec, and uns8.

Referenced by usb_cdc_print_int(), and usb_cdc_print_str().

Here is the caller graph for this function:




### uns8 usb_cdc_rx_avail ()

If one or more bytes are available in the USB serial port receive buffer, this routine will return true. If there are no bytes available, it will return false.

**Returns:**

True if buffer is not empty, False if buffer is empty.

Definition at line 456 of file usb_cdc_class.c.

References cdc_rx_end, and cdc_rx_start.

### void usb_cdc_setup ()

Configures the default DTE rate, stop bits etc.

Definition at line 483 of file usb_cdc_class.c.

References long_union::as_long, and current_bit_rate.

### uns8 usb_cdc_tx_empty ()

Sometimes it is useful to see if the transmit buffer is empty, since then you can be sure your data is well on its way. In the case of USB, this means that the data has been at least placed into the outbound USB buffer; it's not possible to tell until after the fact if the data has actually been squirted out the USB port.

**Returns:**

True if transmit buffer is empty, False if buffer still has data in it.

Definition at line 457 of file usb_cdc_class.c.

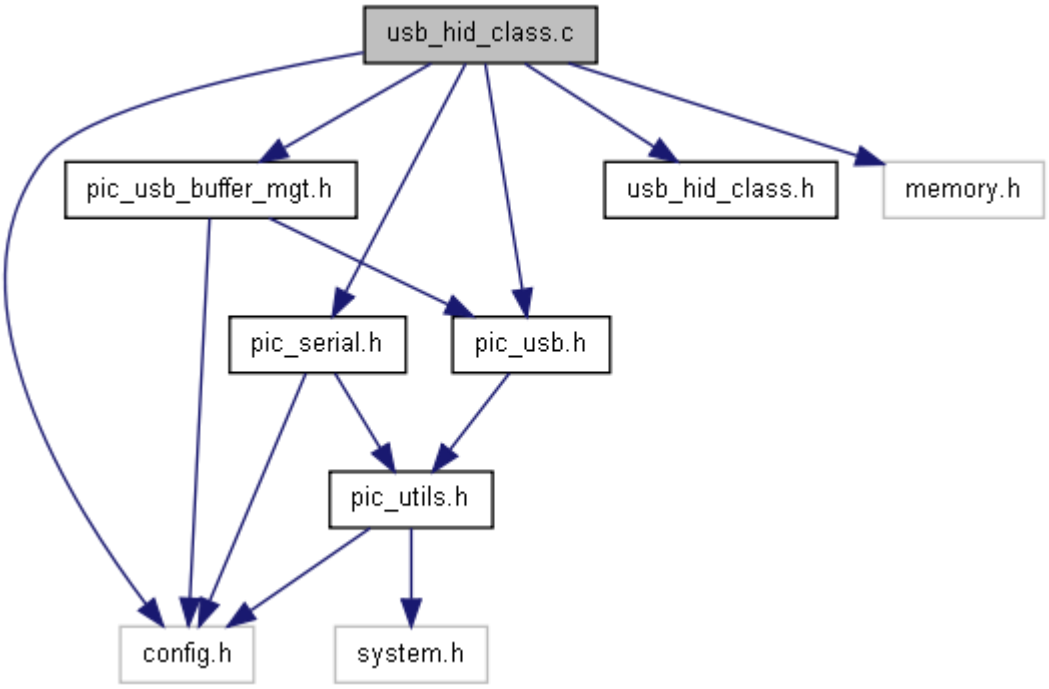
References cdc_tx_end, and cdc_tx_start.

# usb_hid_class.c File Reference

Callbacks for implementing USB HID class.

```
#include "config.h"
#include "pic_usb.h"
#include "pic_usb_buffer_mgt.h"
#include "pic_serial.h"
#include "usb_hid_class.h"
#include "memory.h"
```

Include dependency graph for usb_hid_class.c:




## Functions

- void usb_handle_class_ctrl_read_callback ()
- *Callback routine for a class control read.* void usb_handle_class_ctrl_write_callback (uns8 *data, uns16 count)
- *Callback routine for a class control write.* void usb_handle_class_request_callback (setup_data_packet sdp)

*Callback routine for a control transfer request that is placed on the class.*

## Detailed Description

Definition in file usb_hid_class.c.

## Function Documentation

### void usb_handle_class_ctrl_read_callback ()

When a control transfer is taking place, this routine is called to indicate that a control read for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been transferred to the host. If you wish to send more data to the host, use usb_send_data(), or if your control read has sent all the data required, you will need to indicate that the state has changed by setting the control_mode variable to cm_CTRL_READ_AWAITING_STATUS. This will indicate to the stack that it should now wait for the status packet before completing the control transfer.

To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 54 of file usb_hid_class.c.

Referenced by usb_handle_transaction().

Here is the caller graph for this function:


### void usb_handle_class_ctrl_write_callback (uns8 * *data*, uns16 *count*)

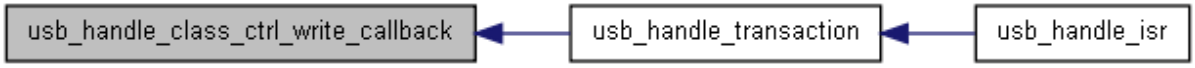
When a control transfer is taking place, this routine is called to indicate that a control write for the class has taken place. Since everything in USB land is all about what has just happened, this callback will occur after data has been received by the device. If you expect more data from the host, it will arrive in due course since endpoint 0 will be primed for more data automatically. If you have received all the data from the host, you will need to set the control_mode state variable to cm_CTRL_WRITE_SENDING_STATUS and then actually send the status by calling usb_send_status_ack(). Once the status has actually been sent, the control_mode state will automatically change to cm_IDLE to indicate the transfer has completed.

To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 57 of file usb_hid_class.c.

Referenced by usb_handle_transaction().

Here is the caller graph for this function:




### void usb_handle_class_request_callback (setup_data_packet *sdp*)

After receiving a setup packet, where the request is placed on the class, this routine is called. In usb_handle_class_request_callback, you can set up ready for the data stage of the control transfer. The direction of the data stage can be determined by examining test_bit(sdp.bRequest, DATA_STAGE_DIR) although generally it appears to be obvious from the request. The request is stored in sdp.bRequest.

Typically, if it is a control read transfer (that is, it is a request by the host for data), then you will need to move the control_mode state variable to cm_CTRL_READ_DATA_STAGE_CLASS and send data using usb_send_data(). If you only intend to send one packet, you can immediately move the

control_mode state variable to cm_CTRL_READ_AWAITING_STATUS to indicate you are waiting for the status to arrive. You could wait for the usb_handle_class_ctrl_read callback and do it (move to cm_CTROL_READ_AWAITING_STATUS) but the PicPack USB stack can handle the control read event for you if you've already switched states.

If it is a control write transfer (that is, it is a request by the host to send data to the device), then you will need to move the control_mode state variable to cm_CTRL_WRITE_DATA_STAGE_CLASS. Then, the usb_handle_class_ctrl_write will be fired when data is received by the device in the data stage.

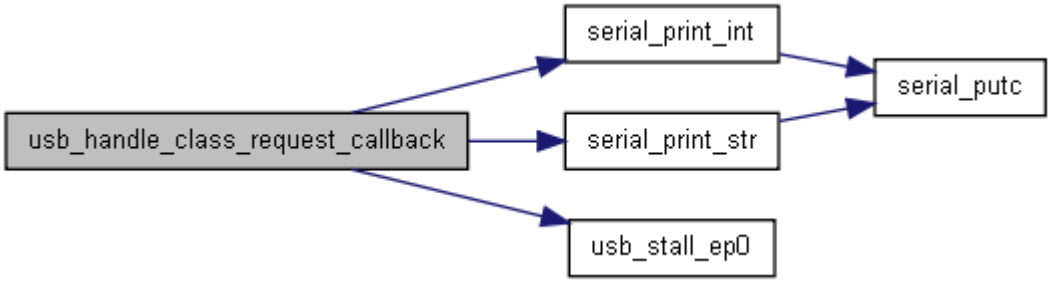
To allow this callback to trigger, ensure you define USB_CALLBACK_ON_CLASS_CTRL in your config.h

Definition at line 60 of file usb_hid_class.c.

References setup_data_packet::bRequest, req_GET_IDLE, req_GET_PROTOCOL, req_GET_REPORT, req_SET_IDLE, req_SET_PROTOCOL, req_SET_REPORT, serial_print_int(), serial_print_str(), and usb_stall_ep0().

Referenced by usb_handle_transaction().

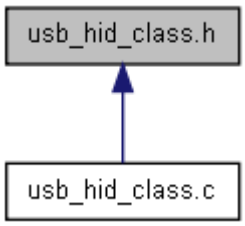
Here is the call graph for this function:




Here is the caller graph for this function:


## usb_hid_class.h File Reference

Helper definitions for USB HID class.
This graph shows which files directly or indirectly include this file:



## Defines

- #define req_GET_IDLE 0x02
- #define req_GET_PROTOCOL 0x03
- #define req_GET_REPORT 0x01
- #define req_SET_IDLE 0x0a
- #define req_SET_PROTOCOL 0x0b
- #define req_SET_REPORT 0x09

---

## Detailed Description

Definition in file usb_hid_class.h.

---

## Define Documentation

### #define req_GET_IDLE 0x02

Definition at line 46 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

### #define req_GET_PROTOCOL 0x03

Definition at line 47 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

### #define req_GET_REPORT 0x01

Definition at line 45 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

### #define req_SET_IDLE 0x0a

Definition at line 49 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

### #define req_SET_PROTOCOL 0x0b

Definition at line 50 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

### #define req_SET_REPORT 0x09

Definition at line 48 of file usb_hid_class.h.

Referenced by usb_handle_class_request_callback().

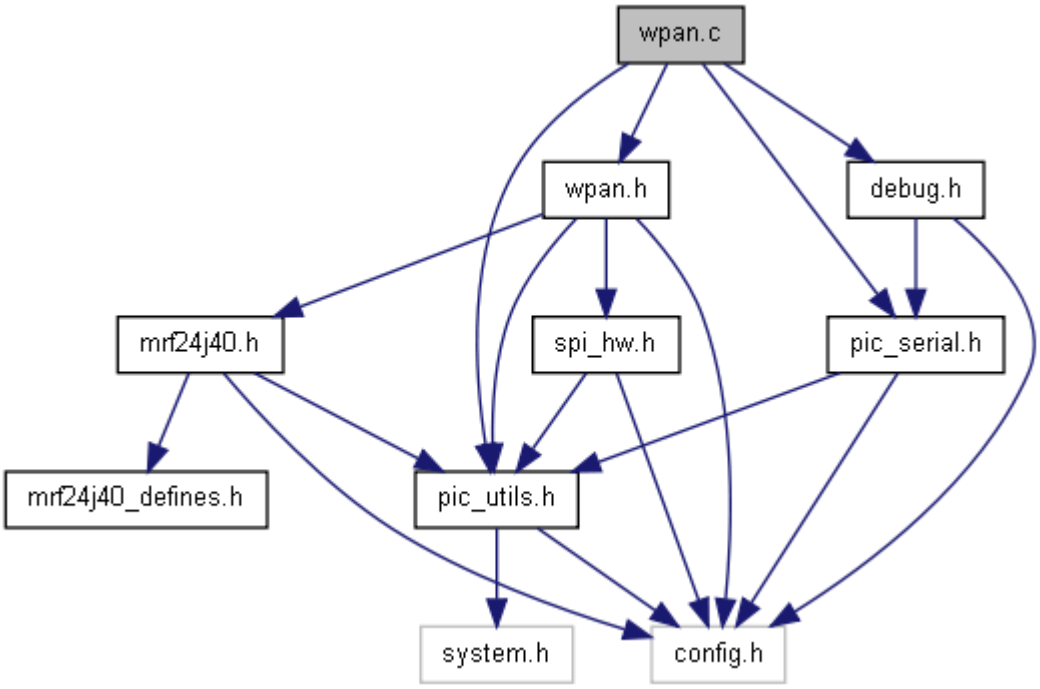
---

## wpan.c File Reference

Wireless Personal Area Network routines.

```
#include "wpan.h"
#include "pic_utils.h"
#include "pic_serial.h"
#include "debug.h"
```

Include dependency graph for wpan.c:




### Defines

- #define debug_on

### Functions

- void mrf24j40_receive_callback ()
- *Callback is actioned when mrf24j40 has a packet received off air.* void mrf24j40_transmit_callback (uns8 status, uns8 retries, uns8 channel_busy)
- *Callback is actioned when mrf24j40 has finished transmitting a packet, or failed to do so.* void wpan_handle_receive ()
- *Handle reception of packet.* void wpan_init ()
- *Initialise this and lower layers ready for use.* void wpan_print_address (wpan_address *addr)
- *Print the address of the packet.* void wpan_print_frame_type (uns8 frame_type)
- *Print information about the WPAN frame type.* void wpan_print_mac_command (uns8 *data)
- *Debug routine that prints out the mac command type.* void wpan_process ()

- *Process WPAN layer.* void wpan_setup_io ()

### *Setup IO as requred.* Variables

- uns8 pkt_received = 0
- uns8 receive_lost = 0
- uns8 wpan_rx_buffer [50]
- uns8 wpan_rx_count

---

## Detailed Description

Definition in file wpan.c.

---

## Define Documentation

### #define debug_on

Definition at line 40 of file wpan.c.

---

## Function Documentation

### void mrf24j40_receive_callback ()

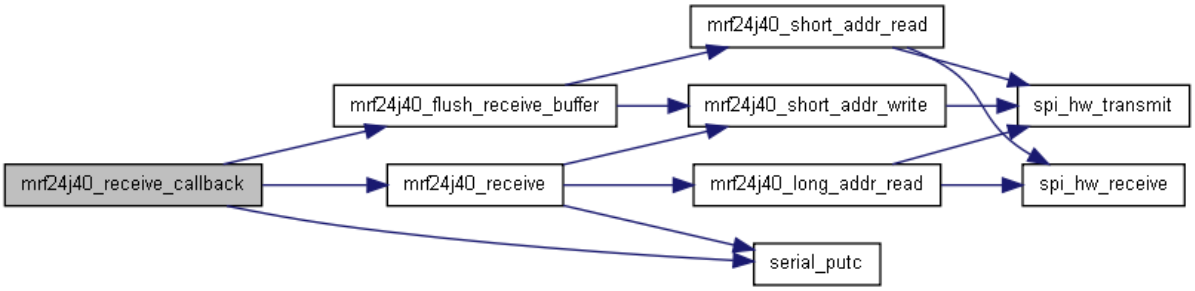
Callback indicating the mrf24j40 has a packet ready.

Definition at line 56 of file wpan.c.

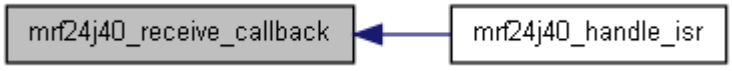
References debug_int, debug_str, mrf24j40_flush_receive_buffer(), mrf24j40_receive(), pkt_received, receive_lost, serial_putc(), wpan_rx_buffer, and wpan_rx_count.

Referenced by mrf24j40_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:



**void mrf24j40_transmit_callback (uns8 *status*, uns8 *retries*, uns8 *channel_busy*)**

Callback indicating the mrf24j40 has finished the transmission sequence.

**Parameters:**

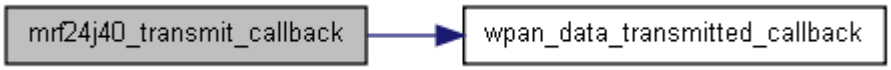
*status* Set to 0 for success or 1 for failure
*retries* Set to the number of retries
*channel_busy* Set to 1 if failure was due to the channel being busy.

Definition at line 75 of file wpan.c.

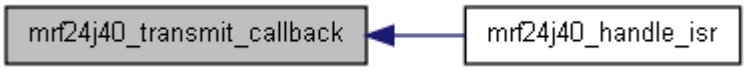
References debug_int, debug_str, and wpan_data_transmitted_callback().

Referenced by mrf24j40_handle_isr().

Here is the call graph for this function:




Here is the caller graph for this function:




**void wpan_handle_receive ()**

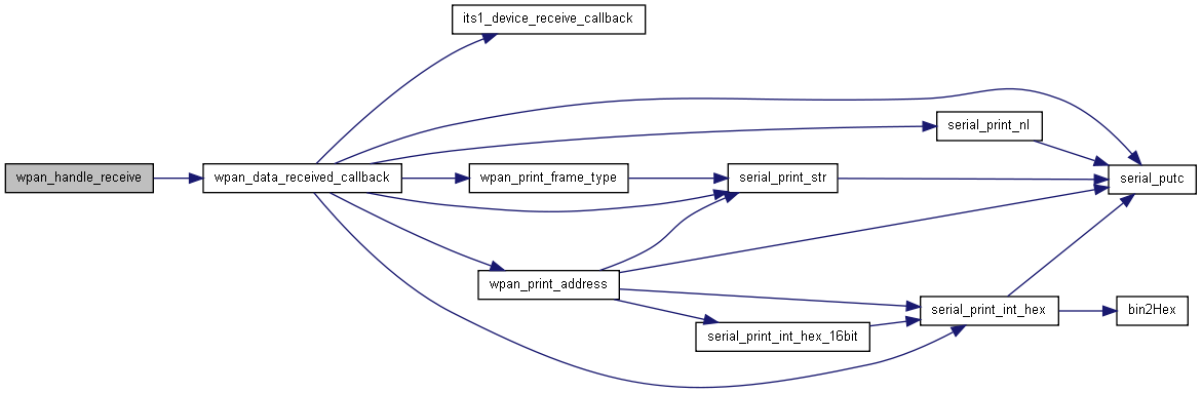
Internal routine for WPAN layer to handle the reception of a packet.

Definition at line 225 of file wpan.c.

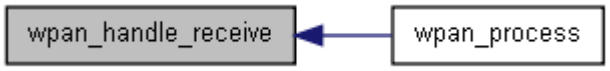
References wpan_address::dest_address_type, wpan_address::dest_ea, wpan_address::dest_pan_id, wpan_address::dest_sa, FRAME_TYPE_BEACON, FRAME_TYPE_DATA, FRAME_TYPE_MAC_COMMAND, pkt_received, wpan_address::source_address_type, wpan_address::source_ea, wpan_address::source_pan_id, wpan_address::source_sa, uns8, WPAN_ADDR_TYPE_EXTENDED, WPAN_ADDR_TYPE_NONE, WPAN_ADDR_TYPE_SHORT, wpan_data_received_callback(), wpan_rx_buffer, and wpan_rx_count.

Referenced by wpan_process().

Here is the call graph for this function:




Here is the caller graph for this function:



**void wpan_init ()**

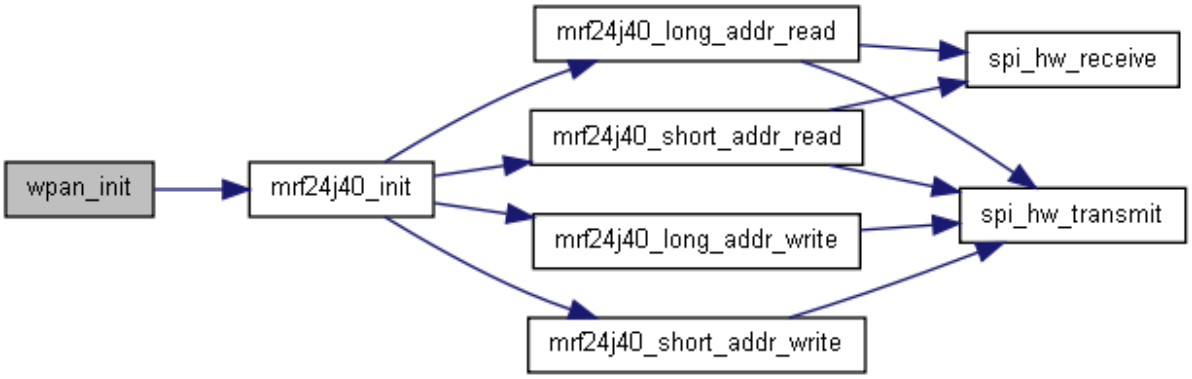
Initialise underlying hardware (typically mrf24j40)

Definition at line 336 of file wpan.c.

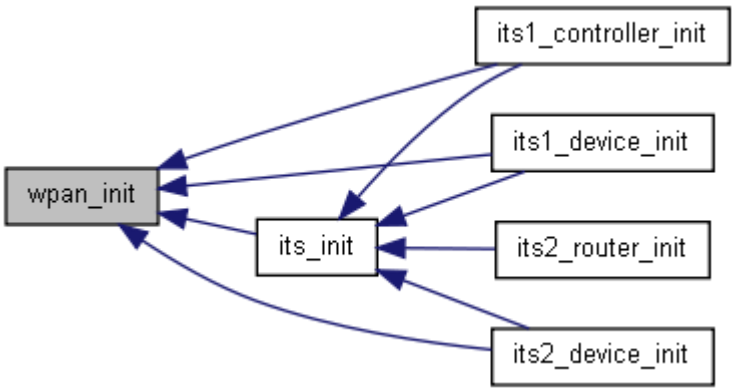
References mrf24j40_init().

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void wpan_print_address (wpan_address * *addr*)**

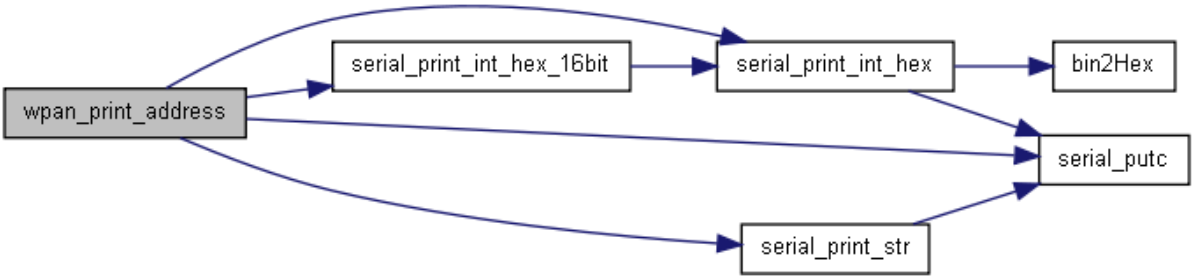
Print out the source and destination address

Definition at line 96 of file wpan.c.

References debug_str, wpan_address::dest_address_type, wpan_address::dest_ea, wpan_address::dest_pan_id, wpan_address::dest_sa, serial_print_int_hex(), serial_print_int_hex_16bit(), serial_print_str(), serial_putc(), wpan_address::source_address_type, wpan_address::source_ea, wpan_address::source_pan_id, wpan_address::source_sa, uns8, WPAN_ADDR_TYPE_EXTENDED, and WPAN_ADDR_TYPE_SHORT.

Referenced by wpan_data_received_callback().

Here is the call graph for this function:




Here is the caller graph for this function:

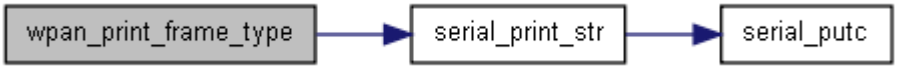

### void wpan_print_frame_type (uns8 *frame_type*)

Definition at line 148 of file wpan.c.

References FRAME_TYPE_ACK, FRAME_TYPE_BEACON, FRAME_TYPE_DATA, FRAME_TYPE_MAC_COMMAND, and serial_print_str().

Referenced by wpan_data_received_callback().

Here is the call graph for this function:




Here is the caller graph for this function:

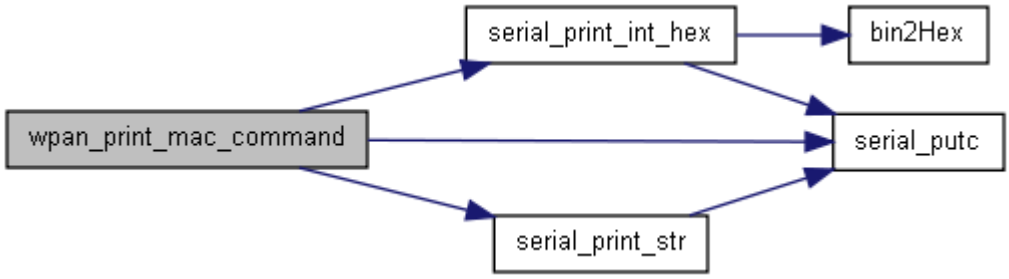

### void wpan_print_mac_command (uns8 * *data*)

Print the MAC layer command details

Definition at line 166 of file wpan.c.

References MAC_CMD_ASSOC_REQ, MAC_CMD_ASSOC_RES, MAC_CMD_BEACON_REQ, MAC_CMD_COORD_REALIGN, MAC_CMD_DATA_REQ, MAC_CMD_DISASSOC, MAC_CMD_GTS_REQ, MAC_CMD_ORPHAN, MAC_CMD_PAN_ID_CONFLICT, serial_print_int_hex(), serial_print_str(), serial_putc(), and uns8.

Here is the call graph for this function:



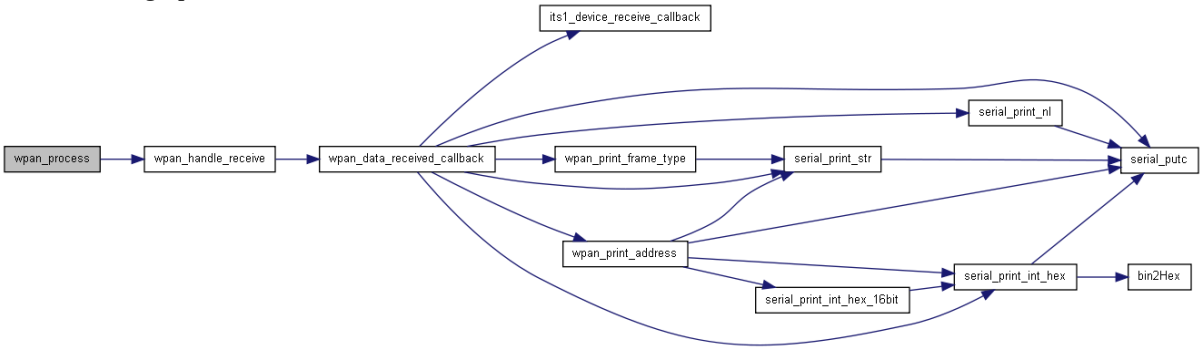
### void wpan_process ()

Call wpan_process() regularly to handle events at the WPAN layer. Typically this includes handling any received packets, processing them, and handing up to a higher layer.

Definition at line 214 of file wpan.c.

References pkt_received, and wpan_handle_receive().

Here is the call graph for this function:




### void wpan_setup_io ()

Set up ports and pins for WPAN use

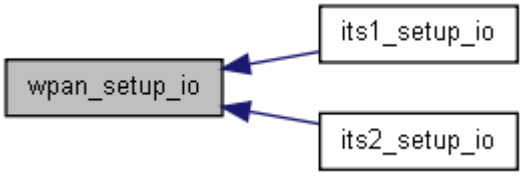
Definition at line 332 of file wpan.c.

References mrf24j40_setup_io().

Referenced by its1_setup_io(), and its2_setup_io().

Here is the call graph for this function:


Here is the caller graph for this function:




---

## Variable Documentation

### uns8 pkt_received = 0

Definition at line 53 of file wpan.c.

Referenced by mrf24j40_receive_callback(), wpan_handle_receive(), and wpan_process().

### uns8 receive_lost = 0

Definition at line 51 of file wpan.c.

Referenced by mrf24j40_receive_callback().

**uns8 wpan_rx_buffer[50]**

Definition at line 49 of file wpan.c.

Referenced by mrf24j40_receive_callback(), and wpan_handle_receive().

**uns8 wpan_rx_count**

Definition at line 50 of file wpan.c.

Referenced by mrf24j40_receive_callback(), and wpan_handle_receive().

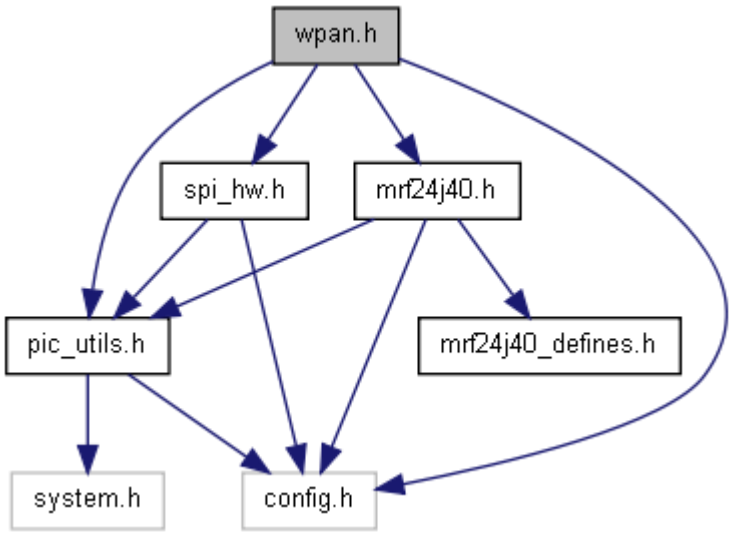
---

## wpan.h File Reference

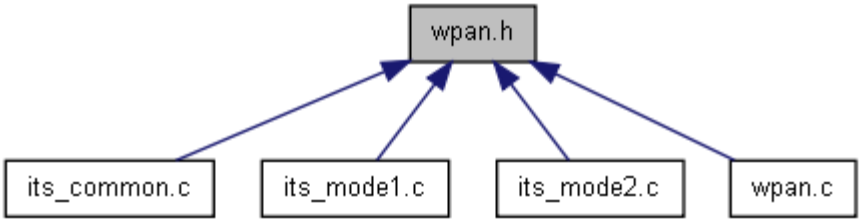
Wireless Personal Area Network routines.

```
#include "pic_utils.h"
#include "config.h"
#include "mrf24j40.h"
#include "spi_hw.h"
```

Include dependency graph for wpan.h:




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




## Data Structures

- struct wpan_address

## Defines

- #define FRAME_TYPE_ACK 0b010
- #define FRAME_TYPE_BEACON 0b000
- #define FRAME_TYPE_DATA 0b001
- #define FRAME_TYPE_MAC_COMMAND 0b011
- #define MAC_CMD_ASSOC_REQ 0x01
- #define MAC_CMD_ASSOC_RES 0x02
- #define MAC_CMD_BEACON_REQ 0x07
- #define MAC_CMD_COORD_REALIGN 0x08
- #define MAC_CMD_DATA_REQ 0x04
- #define MAC_CMD_DISASSOC 0x03
- #define MAC_CMD_GTS_REQ 0x09
- #define MAC_CMD_ORPHAN 0x06
- #define MAC_CMD_PAN_ID_CONFLICT 0x05
- #define WPAN_ADDR_TYPE_EXTENDED 0b00000011
- #define WPAN_ADDR_TYPE_NONE 0b00000000
- #define WPAN_ADDR_TYPE_SHORT 0b00000010

## Functions

- void wpan_data_received_callback (wpan_address *addr, uns8 *data, uns8 data_size, uns8 lqi, uns8 rssi)
- *Callback for data received.* void wpan_data_transmitted_callback (uns8 status, uns8 retries, uns8 channel_busy)
- *Callback for data transmitted.* void wpan_handle_receive ()
- *Handle reception of packet.* void wpan_init ()
- *Initialise this and lower layers ready for use.* void wpan_print_address (wpan_address *addr)
- *Print the address of the packet.* void wpan_print_frame_type (uns8 frame_type)
- *Print information about the WPAN frame type.* void wpan_print_mac_command (uns8 *data)
- *Debug routine that prints out the mac command type.* void wpan_process ()
- *Process WPAN layer.* void wpan_setup_io ()

## *Setup IO as requred.* Variables

- uns8 pkt_received

---

## Detailed Description

Put this in your config.h

```
// ------------------------
// WPAN (802.15.4) wireless
// ------------------------
```

```
#define WPAN_USE_MRF24J50
// ------------------------
```

Definition in file wpan.h.

---

## Define Documentation

### #define FRAME_TYPE_ACK 0b010

Definition at line 79 of file wpan.h.

Referenced by wpan_print_frame_type().

### #define FRAME_TYPE_BEACON 0b000

Definition at line 77 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_frame_type().

### #define FRAME_TYPE_DATA 0b001

Definition at line 78 of file wpan.h.

Referenced by its2_transmit(), its_transmit_to_ea(), its_transmit_to_sa(), wpan_handle_receive(), and wpan_print_frame_type().

### #define FRAME_TYPE_MAC_COMMAND 0b011

Definition at line 80 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_frame_type().

### #define MAC_CMD_ASSOC_REQ 0x01

Definition at line 67 of file wpan.h.

Referenced by wpan_print_mac_command().

### #define MAC_CMD_ASSOC_RES 0x02

Definition at line 68 of file wpan.h.

Referenced by wpan_print_mac_command().

### #define MAC_CMD_BEACON_REQ 0x07

Definition at line 73 of file wpan.h.

Referenced by wpan_print_mac_command().

**#define MAC_CMD_COORD_REALIGN 0x08**

Definition at line 74 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define MAC_CMD_DATA_REQ 0x04**

Definition at line 70 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define MAC_CMD_DISASSOC 0x03**

Definition at line 69 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define MAC_CMD_GTS_REQ 0x09**

Definition at line 75 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define MAC_CMD_ORPHAN 0x06**

Definition at line 72 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define MAC_CMD_PAN_ID_CONFLICT 0x05**

Definition at line 71 of file wpan.h.
Referenced by wpan_print_mac_command().

**#define WPAN_ADDR_TYPE_EXTENDED 0b00000011**

Definition at line 63 of file wpan.h.
Referenced by wpan_handle_receive(), and wpan_print_address().

**#define WPAN_ADDR_TYPE_NONE 0b00000000**

Definition at line 65 of file wpan.h.
Referenced by wpan_handle_receive().

**#define WPAN_ADDR_TYPE_SHORT 0b00000010**

Definition at line 64 of file wpan.h.

Referenced by wpan_handle_receive(), and wpan_print_address().

---

## Function Documentation

### void wpan_data_received_callback (wpan_address * *addr*, uns8 * *data*, uns8 *data_size*, uns8 *lqi*, uns8 *rssi*)

After the packet has been processed and the address extracted, it is delivered to the callback function.

### void wpan_data_transmitted_callback (uns8 *status*, uns8 *retries*, uns8 *channel_busy*)

Once the lower layers have attempted to transmit the packet, the results will be presented to the callback.

**Parameters:**

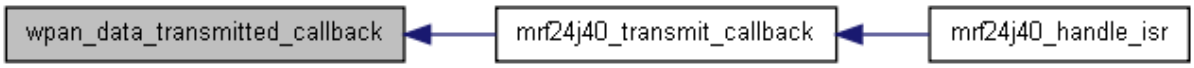
*status* Set to 0 for success or 1 for failure
*retries* Set to the number of retries
*channel_busy* Set to 1 if failure was due to the channel being busy.

Definition at line 804 of file its_mode2.c.

References its2_transmitting.

Referenced by mrf24j40_transmit_callback().

Here is the caller graph for this function:




### void wpan_handle_receive ()

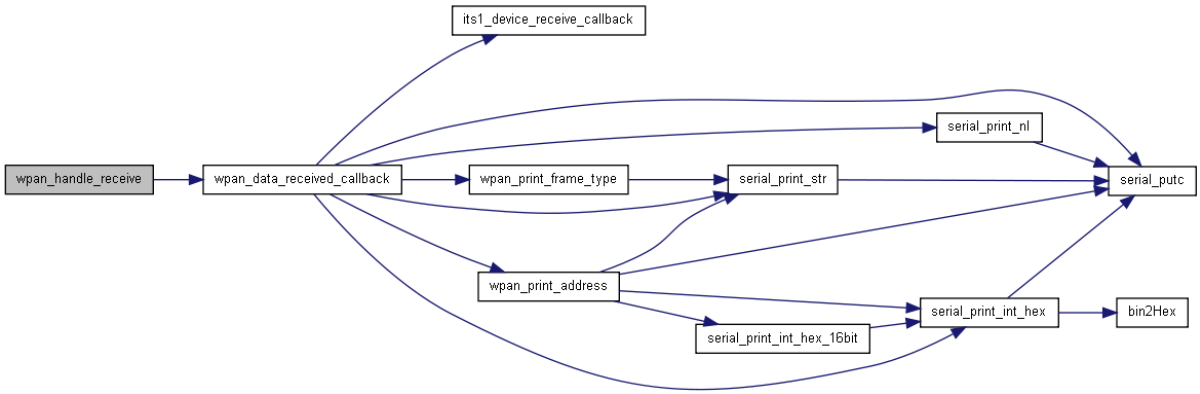
Internal routine for WPAN layer to handle the reception of a packet.

Definition at line 225 of file wpan.c.

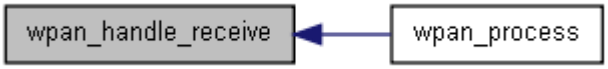
References wpan_address::dest_address_type, wpan_address::dest_ea, wpan_address::dest_pan_id, wpan_address::dest_sa, FRAME_TYPE_BEACON, FRAME_TYPE_DATA, FRAME_TYPE_MAC_COMMAND, pkt_received, wpan_address::source_address_type, wpan_address::source_ea, wpan_address::source_pan_id, wpan_address::source_sa, uns8, WPAN_ADDR_TYPE_EXTENDED, WPAN_ADDR_TYPE_NONE, WPAN_ADDR_TYPE_SHORT, wpan_data_received_callback(), wpan_rx_buffer, and wpan_rx_count.

Referenced by wpan_process().

Here is the call graph for this function:



Here is the caller graph for this function:




**void wpan_init ()**

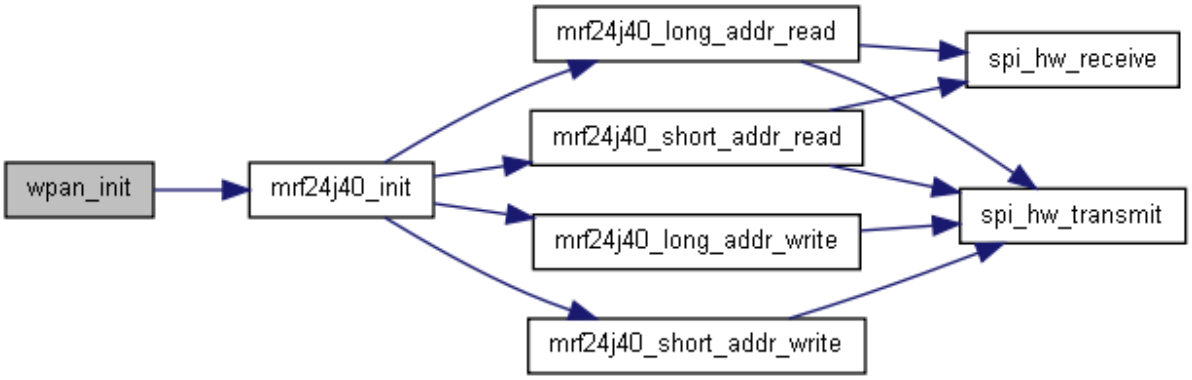
Initialise underlying hardware (typically mrf24j40)

Definition at line 336 of file wpan.c.

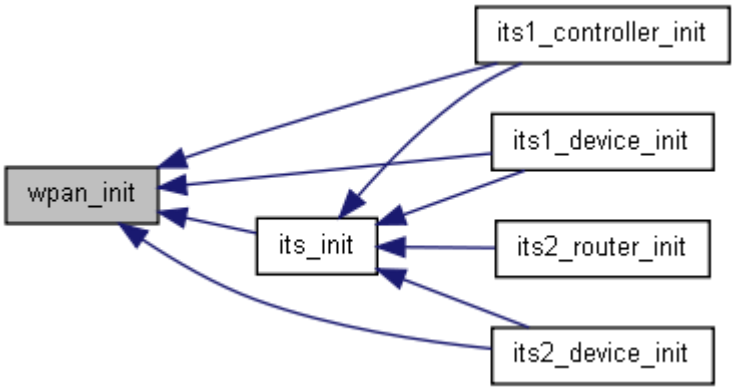
References mrf24j40_init().

Referenced by its1_controller_init(), its1_device_init(), its2_device_init(), and its_init().

Here is the call graph for this function:




Here is the caller graph for this function:




**void wpan_print_address (wpan_address * *addr*)**

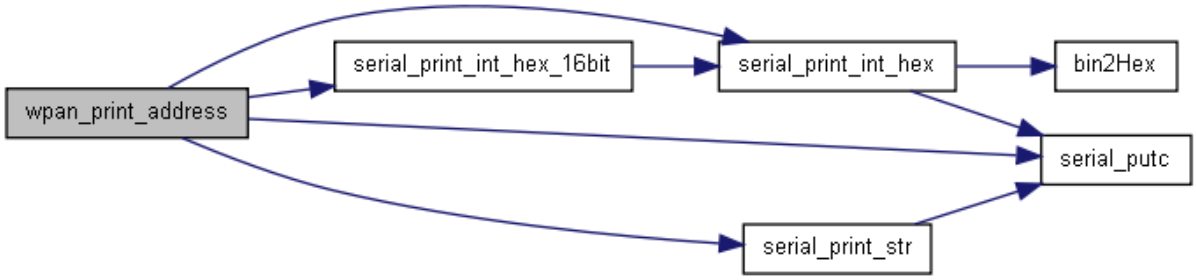
Print out the source and destination address

Definition at line 96 of file wpan.c.

References debug_str, wpan_address::dest_address_type, wpan_address::dest_ea, wpan_address::dest_pan_id, wpan_address::dest_sa, serial_print_int_hex(), serial_print_int_hex_16bit(), serial_print_str(), serial_putc(), wpan_address::source_address_type, wpan_address::source_ea, wpan_address::source_pan_id, wpan_address::source_sa, uns8, WPAN_ADDR_TYPE_EXTENDED, and WPAN_ADDR_TYPE_SHORT.

Referenced by wpan_data_received_callback().

Here is the call graph for this function:




Here is the caller graph for this function:

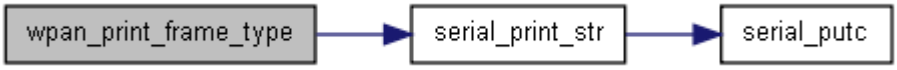

### void wpan_print_frame_type (uns8 *frame_type*)

Definition at line 148 of file wpan.c.

References FRAME_TYPE_ACK, FRAME_TYPE_BEACON, FRAME_TYPE_DATA, FRAME_TYPE_MAC_COMMAND, and serial_print_str().

Referenced by wpan_data_received_callback().

Here is the call graph for this function:




Here is the caller graph for this function:

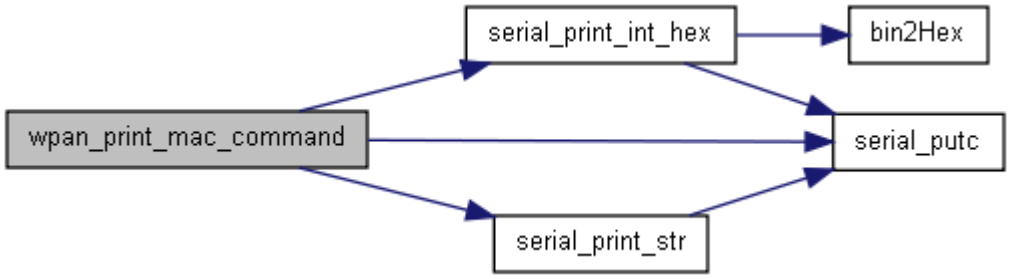

### void wpan_print_mac_command (uns8 * *data*)

Print the MAC layer command details

Definition at line 166 of file wpan.c.

References MAC_CMD_ASSOC_REQ, MAC_CMD_ASSOC_RES, MAC_CMD_BEACON_REQ, MAC_CMD_COORD_REALIGN, MAC_CMD_DATA_REQ, MAC_CMD_DISASSOC, MAC_CMD_GTS_REQ, MAC_CMD_ORPHAN, MAC_CMD_PAN_ID_CONFLICT, serial_print_int_hex(), serial_print_str(), serial_putc(), and uns8.

Here is the call graph for this function:



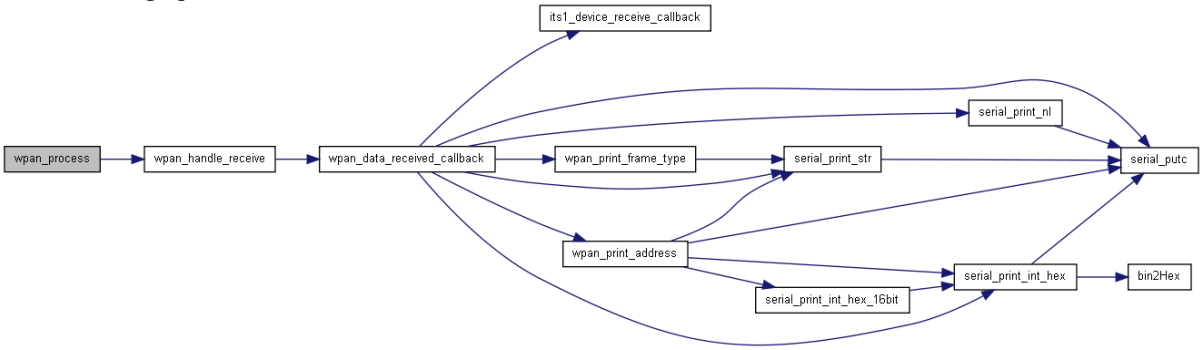
### void wpan_process ()

Call wpan_process() regularly to handle events at the WPAN layer. Typically this includes handling any received packets, processing them, and handing up to a higher layer.

Definition at line 214 of file wpan.c.

References pkt_received, and wpan_handle_receive().

Here is the call graph for this function:




### void wpan_setup_io ()

Set up ports and pins for WPAN use

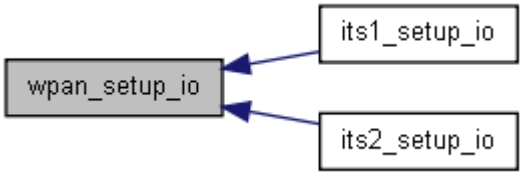
Definition at line 332 of file wpan.c.

References mrf24j40_setup_io().

Referenced by its1_setup_io(), and its2_setup_io().

Here is the call graph for this function:


Here is the caller graph for this function:




---

## Variable Documentation

### uns8 pkt_received

Definition at line 53 of file wpan.c.

Referenced by mrf24j40_receive_callback(), wpan_handle_receive(), and wpan_process().

# Index