CMOS 24GHz Radar Sensor Library source code release

Socionext Inc.

v1.5

Thu Jun 25 17:17:20 2020 +0900

Contents

Chapter 1

CMOS 24GHz Radar Sensor Library Reading and Porting Guides

About source code

This source code is "CMOS 24GHz Radar Sensor Library" based on SC1233AR3 Evaluation Kit (EVK). As Evaluation Kit, the code is intended to be used for many purposes. Then the code may not be optimized for the particular usage. Please refer the specification in the datasheet [13], and use this code as guides when writing a code.

Descriptions of CMOS 24GHz Radar Sensor Library are belows,

- Basic Control Flow
- Driver Adaptation Layer, a kind of porting layer, but it may not be suitable for final products and will be re-written entirely.
- Low-Level Device Layer, specific SPI and I2C interface to implement Driver Adaptation Layer

Warning

The code may be re-written without any notice.

Others

- Others

Chapter 2

Basic Control Flow

From SC1233AR3 Control API point of view, Brief Control Flow is written in "rs24g_sample_sc1233.c" like belows,

Brief Control Flow (API view)

- rs_open() opens a sensor device
- rs_setup_param() sets up sensor parameters
  - rs_setup_param_local()
    - motion_wide_getcode() or distance_wide_getcode() prepares setup parameters from pre-defined Sequencer Code Data
  - rs_load_seqcode(), it mainly works as follows,
    - rs_ctl_cmd_chipboot_sc1233() booting up ("Start-Up") the sensor [1]
    - rs_setup_seq() sets up the sensor with the setup parameters
  - rs_update_param_local()
    - motion_wide_update_param() or distance_wide_update_param() modifies parameters from user settings
- rs_start() starts sensing
- rs_get_motion() or rs_get_distance() gets sensing results. Looping here to get sensing results is available
  - get_devdata()
    - rs_ctl_cmd_wait_and_get_sensor_data()
      - rs_ctl_cmd_get_sensor_data() gets distance data from Sensor FIFO or registers
- rs_stop() stops sensing
- rs_shutdown() shuts down the sensor
- rs_close() closes the sensor device

The code for EVK works as "Timer operation mode". There are other operation mode, but it is out of scope.

Warning

This code includes Sequencer Codes for Sensor. Those are dedicated for SC1233AR3 Evaluation Kit only. Use proper Sequencer Code released with Sensor device or Host platform.

Chapter 3

Driver Adaptation Layer

Driver Adaptation Layer is defined in rs_ctl_dev.h which controls a Sensor through Low-Level Device Layer.

Implement these documented functions to fit Host platform on, it may work at least almost the same as EVK. But even this layer is exists, the code may have some overheads. So, it may be required to refine code to reduce those overheads.

Driver Adaptation Functions

- General functions
  - rs_ctl_dev_open()
  - rs_ctl_dev_close()
- Read Write Sensors through communication device
  - rs_ctl_dev_write()
  - rs_ctl_dev_read()
- GPIO related functions
  - rs_ctl_dev_term_set()
  - rs_ctl_dev_term_get()
  - rs_ctl_dev_term_trig_clear()
  - rs_ctl_dev_term_trig_set()
  - rs_ctl_dev_term_trig_wait()

Note

Especially current implementation of rs_ctl_dev_term_trig_wait() is a kind of polling base, so it may have timing variation from the trigger signal. To reduce the timing variation, it may be required to re-write using directly call back or others on Host platform.

Chapter 4

Low-Level Device Layer

Sample driver implementation on MBED

SPI

defined in rs_spi.h

I2C

defined in rs_i2c.h

GPIO

defined in rs_term.h

Warning

SPI and I2C drivers are kind of hardware adapters, so usually there is no difference out of adapters. But some differences are there between I2C and SPI to use by hardware restrictions. Please check the code and specifications [2] carefully.

Chapter 5

Others

Options

- RS_CPUIF_USE_I2C
  select I2C instead of SPI to accessing Sensor as an adapter.
- RS_ROM_LIMIT_32KB
- RS_ROM_LIMIT_64KB
- RS_ROM_LIMIT_128KB
- RS_ROM_LIMIT_192KB
- RS_ROM_LIMIT_256KB
  limit ROM size. The size value is not accurate because it depends on the target platform.
- RS_WRITE_MEM_LIMIT
  limit data size per a single transaction at writing Sequencer program code and FFT window function. The value to be defined must be a multiple of 16 bytes, because the word size of Sequencer program code is 16 bytes.

Chapter 6

Important Notice

Request for your special attention and precautions in using the technical information and semiconductors described in this book

(1) If any of the products or technical information described in this book is to be exported or provided to non-residents, the laws and regulations of the exporting country, especially, those with regard to security export control, must be observed.

(2) The technical information described in this book is intended only to show the main characteristics and application circuit examples of the products. No license is granted in and to any intellectual property right or other right owned by Socionext Inc. or any other company. Therefore, no responsibility is assumed by our company as to the infringement upon any such right owned by any other company which may arise as a result of the use of technical information described in this book.

(3) The products described in this book are intended to be used for general applications (such as office equipment, communications equipment, measuring instruments and household appliances), or for specific applications as expressly stated in this book. Consult our sales staff in advance for information on the following applications:

- Special applications (such as for airplanes, aerospace, automotive equipment, traffic signaling equipment, combustion equipment, life support systems and safety devices) in which exceptional quality and reliability are required, or if the failure or malfunction of the products may directly jeopardize life or harm the human body. It is to be understood that our company shall not be held responsible for any damage incurred as a result of or in connection with your using the products described in this book for any special application, unless our company agrees to your using the products in this book for any special application.

(4) The products and product specifications described in this book are subject to change without notice for modification and/or improvement. At the final stage of your design, purchasing, or use of the products, therefore, ask for the most up-to-date Product Standards in advance to make sure that the latest specifications satisfy your requirements.

(5) When designing your equipment, comply with the range of absolute maximum rating and the guaranteed operating conditions (operating power supply voltage and operating environment etc.). Especially, please be careful not to exceed the range of absolute maximum rating on the transient state, such as power-on, power-off and mode-switching. Otherwise, we will not be liable for any defect which may arise later in your equipment. Even when the products are used within the guaranteed values, take into the consideration of incidence of break down and failure mode, possible to occur to semiconductor products. Measures on the systems such as redundant design, arresting the spread of fire or preventing glitch are recommended in order to prevent physical injury, fire, social damages, for example, by using the products.

(6) Comply with the instructions for use in order to prevent breakdown and characteristics change due to external factors (ESD, EOS, thermal stress and mechanical stress) at the time of handling, mounting or at customer's process. When using products for which damp-proof packing is required, satisfy the conditions, such as shelf life and the elapsed time since first opening the packages.

(7) This book may be not reprinted or reproduced whether wholly or partially, without the prior written permission of our company.

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 9

File Index

9.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 10

Module Documentation

10.1 Sensor terminal ID

Macros

- #define RS_TERM_NRST (RS_TERM_TYPE_IN | (0x00000000))
- #define RS_TERM_CE (RS_TERM_TYPE_IN | (0x00000001))
- #define RS_TERM_OR (RS_TERM_TYPE_OUT | (0x00000000))
- #define RS_TERM_DETOUT (RS_TERM_TYPE_OUT | (0x00000001))

10.1.1 Detailed Description

10.1.2 Macro Definition Documentation

10.1.2.1 RS_TERM_CE

```
#define RS_TERM_CE (RS_TERM_TYPE_IN | (0x00000001))
```

ID for CE pin

Definition at line 24 of file rs_dev_term.h.

10.1.2.2 RS_TERM_DETOUT

```
#define RS_TERM_DETOUT (RS_TERM_TYPE_OUT | (0x00000001))
```

ID for DETOUT pin

Definition at line 28 of file rs_dev_term.h.

10.1.2.3 RS_TERM_NRST

```
#define RS_TERM_NRST (RS_TERM_TYPE_IN | (0x00000000))
```

ID for NRST pin

Definition at line 22 of file rs_dev_term.h.

10.1.2.4 RS_TERM_OR

```
#define RS_TERM_OR (RS_TERM_TYPE_OUT | (0x00000000))
```

ID for OR pin

Definition at line 26 of file rs_dev_term.h.

10.2 type of trigger

Macros

- #define RS_TERM_TRIGGER_RISING (0x00000001)
- #define RS_TERM_TRIGGER_FALLING (0x00000002)

10.2.1 Detailed Description

10.2.2 Macro Definition Documentation

10.2.2.1 RS_TERM_TRIGGER_FALLING

```
#define RS_TERM_TRIGGER_FALLING (0x00000002)
```

Falling edge

Definition at line 44 of file rs_dev_term.h.

10.2.2.2 RS_TERM_TRIGGER_RISING

```
#define RS_TERM_TRIGGER_RISING (0x00000001)
```

Rising edge

Definition at line 42 of file rs_dev_term.h.

Chapter 11

Class Documentation

11.1 rs_i2c Class Reference

Public Member Functions

- rs_ret_t write (const uint8_t ∗wdata, size_t size, rs_bool_t end=RS_TRUE)
- rs_ret_t read (const uint8_t ∗wdata, size_t wsize, uint8_t ∗rdata, size_t rsize=1)

11.1.1 Detailed Description

Sensor driver for I2C on MBED@HRM1017

Version

(PRELIMINARY)

Definition at line 24 of file rs_i2c.h.

11.1.2 Member Function Documentation

11.1.2.1 read()

```
rs_ret_t rs_i2c::read (
             const uint8_t ∗ wdata,
             size_t wsize,
             uint8_t ∗ rdata,
             size_t rsize = 1 )
```

raw I2C read operation

support only to read status registers

Parameters

in	*wdata*	pointer of write data
in	*wsize*	number of write data in bytes
out	*rdata*	pointer of read data
in	*rsize*	number of read data in bytes

11.1.2.2 write()

```
rs_ret_t rs_i2c::write (
             const uint8_t * wdata,
             size_t size,
             rs_bool_t end = RS_TRUE )
```

raw I2C write operation with terminate option

Parameters

in	*wdata*	pointer of write data, might be NULL when read
in	*size*	number of read data in bytes
in	*end*	option to terminate a I2C transaction

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

- lib/rs_ctl_dev_poc/include/rs_i2c.h

11.2 rs_spi Class Reference

Public Member Functions

- rs_ret_t write (const void ∗wdata, size_t size, bool end=true)
- rs_ret_t read (const void ∗wdata, size_t wsize, void ∗rdata, size_t rsize)
- rs_ret_t spi_read (const void ∗wdata, void ∗rdata, size_t size)

Private Member Functions

- void readwrite (const void ∗wdata, size_t wsize, void ∗rdata, size_t rsize, bool end=true)

11.2.1 Detailed Description

Sensor driver for SPI on MBED@HRM1017

Version

(PRELIMINARY)

Definition at line 24 of file rs_spi.h.

11.2.2 Member Function Documentation

11.2.2.1 read()

```
rs_ret_t rs_spi::read (
             const void * wdata,
             size_t wsize,
             void * rdata,
             size_t rsize )  [inline]
```

SPI read operation

Parameters

`in`	*wdata*	pointer of write data
`in`	*wsize*	number of write data in bytes
`out`	*rdata*	pointer of read data
`in`	*rsize*	number of read data in bytes

See also

rs_spi::write(const void ∗, size_t, void ∗, size_t, bool)

Definition at line 82 of file rs_spi.h.

11.2.2.2 readwrite()

```
void rs_spi::readwrite (
             const void * wdata,
             size_t wsize,
             void * rdata,
             size_t rsize,
             bool end = true )  [inline], [private]
```

raw SPI write/read operation with terminate option

Parameters

`in`	*wdata*	pointer of write data, might be NULL when read
`in`	*wsize*	number of write data in bytes
`out`	*rdata*	pointer of read data, might be NULL when write
`in`	*rsize*	number of read data in bytes
`in`	*end*	option to terminate a SPI transaction

Definition at line 39 of file rs_spi.h.

11.2.2.3 spi_read()

```
rs_ret_t rs_spi::spi_read (
             const void * wdata,
             void * rdata,
             size_t size )  [inline]
```

SPI raw read operation

Parameters

`in`	*wdata*	pointer of write data
`out`	*rdata*	pointer of read data
`in`	*size*	number of read data in bytes

See also

rs_spi::write(const void ∗, size_t, void ∗, size_t, bool)

Definition at line 99 of file rs_spi.h.

11.2.2.4 write()

```
rs_ret_t rs_spi::write (
            const void * wdata,
            size_t size,
            bool end = true )  [inline]
```

SPI write operation with terminate option

Parameters

in	*wdata*	pointer of write data
in	*size*	number of write data in bytes
in	*end*	option to terminate a SPI transaction

Definition at line 61 of file rs_spi.h.

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

- lib/rs_ctl_dev_poc/include/rs_spi.h

11.3 rs_term Class Reference

Public Member Functions

- rs_ret_t get (uint32_t term, rs_bool_t ∗val)
- rs_ret_t set (uint32_t term, rs_bool_t val)
- rs_ret_t set_trigger (uint32_t term, uint32_t trigger)
- rs_ret_t clear_trigger (uint32_t term)
- rs_ret_t wait_trigger (uint32_t term, uint32_t timeout, uint32_t trigger, rs_bool_t ∗val)

11.3.1 Detailed Description

Sensor driver for GPIO on MBED@HRM1017

Version

(PRLIMINARY)

Definition at line 126 of file rs_term.h.

11.3.2 Member Function Documentation

11.3.2.1 clear_trigger()

```
rs_ret_t rs_term::clear_trigger (
            uint32_t term )  [inline]
```

Clear triggered flag

Parameters

in	*term*	terminal ID for Sensor

Definition at line 198 of file rs_term.h.

11.3.2.2 get()

```
rs_ret_t rs_term::get (
             uint32_t term,
             rs_bool_t * val )   [inline]
```

read GPIO

Parameters

in	*term*	terminal ID for Sensor
out	*val*	pointer of read value

Definition at line 144 of file rs_term.h.

11.3.2.3 set()

```
rs_ret_t rs_term::set (
             uint32_t term,
             rs_bool_t val )   [inline]
```

write(set) GPIO

Parameters

in	*term*	ternial ID for Sensor
out	*val*	value to write(set)

Definition at line 165 of file rs_term.h.

11.3.2.4 set_trigger()

```
rs_ret_t rs_term::set_trigger (
             uint32_t term,
             uint32_t trigger )   [inline]
```

Setup GPIO as trigger

Parameters

in	*term*	terminal ID for Sensor
in	*trigger*	type of trigger

Definition at line 182 of file rs_term.h.

11.3.2.5 wait_trigger()

```
rs_ret_t rs_term::wait_trigger (
            uint32_t term,
            uint32_t timeout,
            uint32_t trigger,
            rs_bool_t * val )  [inline]
```

Wait trigger event

Parameters

in	*term*	terminal ID for Sensor
in	*timeout*	timeout in ms.
in	*trigger*	type of trigger to wait
out	*val*	pointer of value at triggerd

Definition at line 217 of file rs_term.h.

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

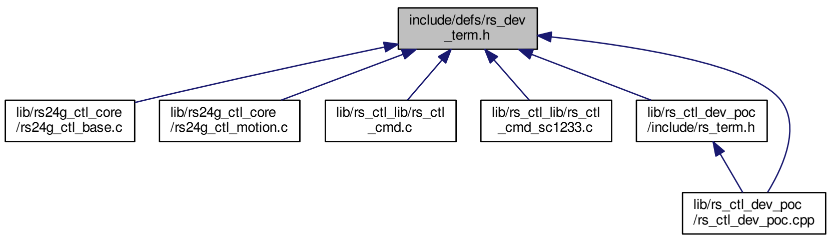
- lib/rs_ctl_dev_poc/include/rs_term.h

Chapter 12

File Documentation

12.1 include/defs/rs_dev_term.h File Reference

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


include/defs/rs_dev _term.h
lib/rs24g_ctl_core /rs24g_ctl_base.c
lib/rs24g_ctl_core /rs24g_ctl_motion.c
lib/rs_ctl_lib/rs_ctl _cmd.c
lib/rs_ctl_lib/rs_ctl _cmd_sc1233.c
lib/rs_ctl_dev_poc /include/rs_term.h
lib/rs_ctl_dev_poc /rs_ctl_dev_poc.cpp

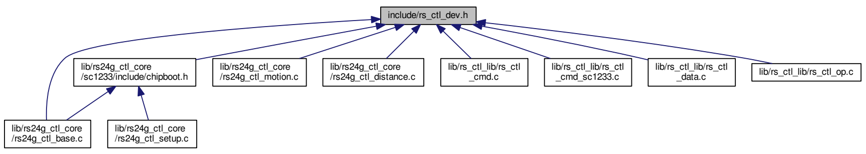
Macros

- #define RS_TERM_NRST (RS_TERM_TYPE_IN | (0x00000000))
- #define RS_TERM_CE (RS_TERM_TYPE_IN | (0x00000001))
- #define RS_TERM_OR (RS_TERM_TYPE_OUT | (0x00000000))
- #define RS_TERM_DETOUT (RS_TERM_TYPE_OUT | (0x00000001))
- #define RS_TERM_TRIGGER_RISING (0x00000001)
- #define RS_TERM_TRIGGER_FALLING (0x00000002)

12.2 include/rs_ctl_dev.h File Reference

device level adaptation layer for Sensor

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


include/rs_ctl_dev.h
lib/rs24g_ctl_core /sc1233/include/chipboot.h
lib/rs24g_ctl_core /rs24g_ctl_motion.c
lib/rs24g_ctl_core /rs24g_ctl_distance.c
lib/rs_ctl_lib/rs_ctl _cmd.c
lib/rs_ctl_lib/rs_ctl _cmd_sc1233.c
lib/rs_ctl_lib/rs_ctl _data.c
lib/rs_ctl_lib/rs_ctl_op.c
lib/rs24g_ctl_core /rs24g_ctl_base.c
lib/rs24g_ctl_core /rs24g_ctl_setup.c

Functions

- RS_IF rs_ret_t rs_ctl_dev_open (rs_ctl_dev_t ∗dev, const void ∗attr)
- RS_IF rs_ret_t rs_ctl_dev_close (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_dev_write (rs_ctl_dev_t dev, const uint8_t ∗wdata, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_dev_read (rs_ctl_dev_t dev, const uint8_t ∗wdata, rs_size_t wsize, uint8_t ∗rdata, rs↩_size_t rsize)
- RS_IF rs_ret_t rs_ctl_dev_term_set (rs_ctl_dev_t dev, uint32_t term, rs_bool_t val)
- RS_IF rs_ret_t rs_ctl_dev_term_get (rs_ctl_dev_t dev, uint32_t term, rs_bool_t ∗val)
- RS_IF rs_ret_t rs_ctl_dev_term_trig_clear (rs_ctl_dev_t dev, uint32_t term)
- RS_IF rs_ret_t rs_ctl_dev_term_trig_set (rs_ctl_dev_t dev, uint32_t term, uint32_t trigger)
- RS_IF rs_ret_t rs_ctl_dev_term_trig_wait (rs_ctl_dev_t dev, uint32_t timeout, uint32_t term, uint32_t trigger, rs_bool_t ∗val)

12.2.1 Detailed Description

device level adaptation layer for Sensor

Version

(PRELIMINARY)

Warning

This is a part of sensor library source code for Evaluation Kit.

12.2.2 Function Documentation

12.2.2.1 rs_ctl_dev_close()

```
RS_IF rs_ret_t rs_ctl_dev_close (
             rs_ctl_dev_t dev )
```

close device to communicate with Sensor

Parameters

in	*dev*	device handle

Return values

RS_OK	everything is OK,
others	something wrong

See also

rs_dev_close()

12.2.2.2 rs_ctl_dev_open()

```
RS_IF rs_ret_t rs_ctl_dev_open (
              rs_ctl_dev_t * dev,
              const void * attr )
```

open device to communicate with Sensor

Parameters

out	*dev*	pointer of device handle (handle should be allocated at upper level)
in	*attr*	attributes (not used now)

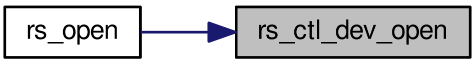
Return values

RS_OK	everything is OK
others	something wrong

See also

rs_dev_open()

Here is the caller graph for this function:



12.2.2.3 rs_ctl_dev_read()

```
RS_IF rs_ret_t rs_ctl_dev_read (
              rs_ctl_dev_t dev,
              const uint8_t * wdata,
              rs_size_t wsize,
              uint8_t * rdata,
              rs_size_t rsize )
```

read Status Register, Sensor Register and Sensor RAM [3] [4]

Sensor RAM contains FIFO

Parameters

in	*dev*	device handle
in	*wdata*	pointer of write data
in	*wsize*	number of write data in bytes
out	*rdata*	pointer of read data
in	*rsize*	number of read data in bytes

Return values

RS_OK	everything is OK,
others	something wrong

See also

rs_dev_read()

12.2.2.4 rs_ctl_dev_term_get()

```
RS_IF rs_ret_t rs_ctl_dev_term_get (
            rs_ctl_dev_t dev,
            uint32_t term,
            rs_bool_t * val )
```

get Sensor Terminal(GPIO)

Parameters

in	*dev*	device handle
in	*term*	Sensor terminal ID
out	*val*	pointer of getting bool value

Return values

RS_OK	everything is OK,
others	something wrong

See also

rs_dev_term_get()

12.2.2.5 rs_ctl_dev_term_set()

```
RS_IF rs_ret_t rs_ctl_dev_term_set (
            rs_ctl_dev_t dev,
            uint32_t term,
            rs_bool_t val )
```

set Sensor Terminal(GPIO)

Parameters

in	*dev*	device handle
in	*term*	Sensor terminal ID
in	*val*	setting bool value

Return values

RS_OK	everything is OK,

Return values

others	something wrong

See also

rs_dev_term_set()

12.2.2.6 rs_ctl_dev_term_trig_clear()

```
RS_IF rs_ret_t rs_ctl_dev_term_trig_clear (
            rs_ctl_dev_t dev,
            uint32_t term )
```

clear Trigger flag for Sensor Terminal(GPIO)

Parameters

in	*dev*	device handle
in	*term*	Sensor terminal ID

Return values

RS_OK	everything is OK,
others	something wrong

See also

rs_dev_term_clear_trigger()

12.2.2.7 rs_ctl_dev_term_trig_set()

```
RS_IF rs_ret_t rs_ctl_dev_term_trig_set (
            rs_ctl_dev_t dev,
            uint32_t term,
            uint32_t trigger )
```

setup Trigger for Sensor Terminal(GPIO)

Parameters

in	*dev*	device handle
in	*term*	Sensor terminal ID
in	*trigger*	type of trigger as RISING or FALLING edge

Return values

RS_OK	everything is OK,
others	something wrong

See also

rs_dev_term_set_trigger()

12.2.2.8 rs_ctl_dev_term_trig_wait()

```
RS_IF rs_ret_t rs_ctl_dev_term_trig_wait (
             rs_ctl_dev_t dev,
             uint32_t timeout,
             uint32_t term,
             uint32_t trigger,
             rs_bool_t * val )
```

wait Trigger is fired for Sensor Terminal(GPIO)

Parameters

in	*dev*	device handle
in	*timeout*	timeout in msec
in	*term*	Sensor terminal ID
in	*trigger*	type of trigger to wait (RISE or FALL)
out	*val*	pointer of getting bool value at fire

Return values

RS_OK	everything is OK,
RS_ETOUT	when timeout,
others	something wrong

See also

rs_dev_term_wait_trigger()

12.2.2.9 rs_ctl_dev_write()

```
RS_IF rs_ret_t rs_ctl_dev_write (
             rs_ctl_dev_t dev,
             const uint8_t * wdata,
             rs_size_t size )
```

write Fast Control, Status Register, Sensor Register and Sensor RAM [5] [6], [7]

Sensor RAM contains Sequencer program code and FFT window function

Parameters

in	*dev*	device handle
in	*wdata*	pointer of write data
in	*size*	number of write data in bytes

Return values

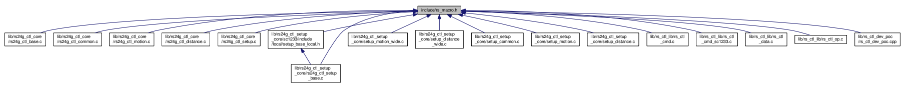
RS_OK	everything is OK,
others	something wrong

See also

rs_dev_write()

12.3 include/rs_macro.h File Reference

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


include/rs_macro.h
lib/rs24g_ctl_core /rs24g_ctl_base.c
lib/rs24g_ctl_core /rs24g_ctl_common.c
lib/rs24g_ctl_core /rs24g_ctl_motion.c
lib/rs24g_ctl_core /rs24g_ctl_distance.c
lib/rs24g_ctl_core /rs24g_ctl_setup.c
lib/rs24g_ctl_setup _core/sc1233/include /local/setup_base_local.h
lib/rs24g_ctl_setup _core/rs24g_ctl_setup _base.c
lib/rs24g_ctl_setup _core/setup_motion_wide.c
lib/rs24g_ctl_setup _core/setup_distance _wide.c
lib/rs24g_ctl_setup _core/setup_common.c
lib/rs24g_ctl_setup _core/setup_motion.c
lib/rs24g_ctl_setup _core/setup_distance.c
lib/rs_ctl_lib/rs_ctl _cmd.c
lib/rs_ctl_lib/rs_ctl _cmd_sc1233.c
lib/rs_ctl_lib/rs_ctl _data.c
lib/rs_ctl_lib/rs_ctl_op.c
lib/rs_ctl_dev_poc /rs_ctl_dev_poc.cpp

Macros

- #define RET_CONV(type, api, refval, retval)
- #define RET_CHECK(api, retval) RET_CONV(int, api, !0, retval)
- #define RET_ORG(type, api, refval)
- #define RS_ASSERT(api, retval) RET_CONV(int, api, !0, retval)
- #define RS_CALL(api) RET_ORG(rs_ret_t, api, RS_OK)

12.3.1 Detailed Description

call function and return depends on return value from the function.

12.3.2 Macro Definition Documentation

12.3.2.1 RET_CHECK

```
#define RET_CHECK(
              api,
              retval ) RET_CONV(int, api, !0, retval)
```

evaluate *api*, return *retval* if the evaluation value is false.

Parameters

in	*api*	expression to evaluate
in	*retval*	return vale on false

Definition at line 72 of file rs_macro.h.

12.3.2.2 RET_CONV

```
#define RET_CONV(
              type,
              api,
              refval,
              retval )
```

Value:

```
{                                                                                    \
    const type macroret = (api);                                                     \
    if(macroret != refval){                                                          \
        rs_macro_printf(                                                             \
            "(%s) [error] ref=%x ret=%x\n",                                          \
            __FUNCTION__,                                                            \
            (uint32_t) refval, (uint32_t) macroret                                   \
        );                                                                           \
        return retval;                                                               \
    }                                                                                \
}
```

call *api*, return *retval* if return value of *api* is not *refval*.

Parameters

in	*type*	type of return value (if available)
in	*api*	callee function
in	*refval*	expected return value from api
in	*retval*	return value if return value is not refval

Definition at line 49 of file rs_macro.h.

12.3.2.3 RET_ORG

```
#define RET_ORG(
                type,
                api,
                refval )
```

Value:

```
{                                                                                    \
    const type macroret = (api);                                                     \
    if(macroret != refval){                                                          \
        rs_macro_printf(                                                             \
            "(%s) [error] ref=%x ret=%x\n",                                          \
            __FUNCTION__,                                                            \
            (uint32_t) refval, (uint32_t) macroret                                   \
        );                                                                           \
        return macroret;                                                             \
    }                                                                                \
}
```

call *api*, return return value of *api* if the return value is not *refval*.

Parameters

in	*type*	type of return value (if available)
in	*api*	callee function
in	*refval*	expected return value from api

Definition at line 96 of file rs_macro.h.

12.3.2.4 RS_ASSERT

```
#define RS_ASSERT(
                api,
                retval ) RET_CONV(int, api, !0, retval)
```

evaluate *api*, return *retval* if the evaluation value is false.

Parameters

in	*api*	expression to evaluate
in	*retval*	return vale on false

Definition at line 124 of file rs_macro.h.

12.3.2.5 RS_CALL

```
#define RS_CALL(
              api ) RET_ORG(rs_ret_t, api, RS_OK)
```

call *api*, return return value of *api* if the return value is not RS_OK.

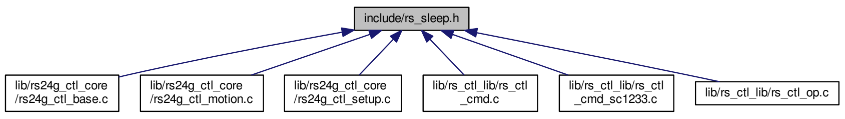
Parameters

in	*api*	callee function

Definition at line 135 of file rs_macro.h.

12.4 include/rs_sleep.h File Reference

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


include/rs_sleep.h
lib/rs24g_ctl_core /rs24g_ctl_base.c
lib/rs24g_ctl_core /rs24g_ctl_motion.c
lib/rs24g_ctl_core /rs24g_ctl_setup.c
lib/rs_ctl_lib/rs_ctl _cmd.c
lib/rs_ctl_lib/rs_ctl _cmd_sc1233.c
lib/rs_ctl_lib/rs_ctl_op.c

Functions

- void rs_usleep (uint32_t usec)

12.4.1 Function Documentation

12.4.1.1 rs_usleep()

```
void rs_usleep (
             uint32_t usec )
```

sleep *usec* microseconds.

Parameters

in	*usec*	sleep time in microseconds

12.5 lib/rs24g_ctl_core/rs24g_ctl_base.c File Reference

Functions

- RS_IF rs_ret_t rs_open (rs_handle_t ∗handle)
- RS_IF rs_ret_t rs_close (rs_handle_t handle)
- RS_IF rs_ret_t rs_shutdown (rs_handle_t handle)
- RS_IF rs_ret_t rs_start (rs_handle_t handle)
- RS_IF rs_ret_t rs_stop (rs_handle_t handle)
- RS_IF rs_ret_t rs_resume (rs_handle_t handle)

12.5.1 Function Documentation

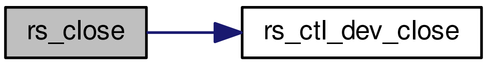
12.5.1.1 rs_close()

```
RS_IF rs_ret_t rs_close (
             rs_handle_t handle )
```

close Sensor

Definition at line 117 of file rs24g_ctl_base.c.

Here is the call graph for this function:



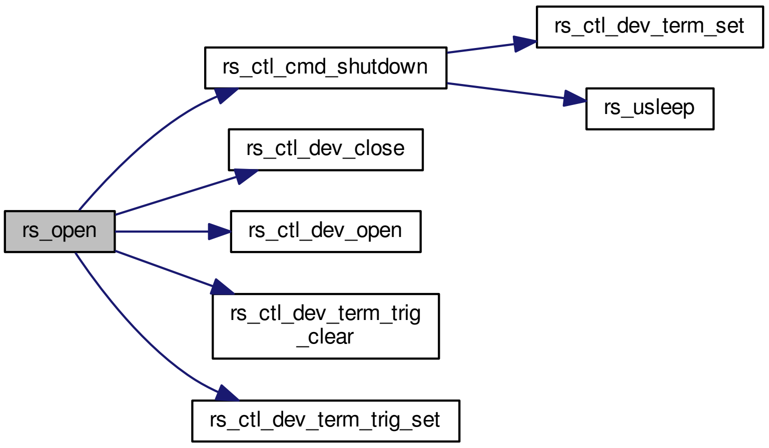
12.5.1.2 rs_open()

```
RS_IF rs_ret_t rs_open (
             rs_handle_t * handle )
```

open Sensor

Definition at line 41 of file rs24g_ctl_base.c.

Here is the call graph for this function:



12.5.1.3 rs_resume()

```
RS_IF rs_ret_t rs_resume (
            rs_handle_t handle )
```

resume sensing

Definition at line 409 of file rs24g_ctl_base.c.

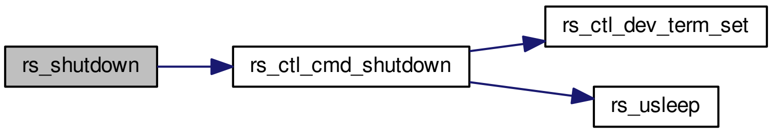
12.5.1.4 rs_shutdown()

```
RS_IF rs_ret_t rs_shutdown (
            rs_handle_t handle )
```

shutdown Sensor

Definition at line 144 of file rs24g_ctl_base.c.

Here is the call graph for this function:



12.5.1.5 rs_start()

```
RS_IF rs_ret_t rs_start (
             rs_handle_t handle )
```

start sensing

Definition at line 270 of file rs24g_ctl_base.c.

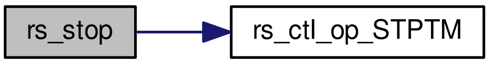
12.5.1.6 rs_stop()

```
RS_IF rs_ret_t rs_stop (
             rs_handle_t handle )
```

stop sensing

Definition at line 329 of file rs24g_ctl_base.c.

Here is the call graph for this function:



12.6 lib/rs24g_ctl_core/rs24g_ctl_common.c File Reference

Functions

- rs_ret_t get_devdata (rs_handle_t handle, uint32_t timeout, const uint16_t ∗reg_addr, rs_size_t reg_size, uint32_t ∗reg)
- rs_ret_t get_devfifo (rs_handle_t handle, uint32_t timeout, rs_size_t fifo_size, uint8_t ∗fifo_data, struct rs_↩ fifo_info ∗fifo_info)
- rs_ret_t get_size_fifo (rs_handle_t handle, rs_size_t ∗size)

12.6.1 Function Documentation

12.6.1.1 get_devdata()

```
rs_ret_t get_devdata (
             rs_handle_t handle,
             uint32_t timeout,
             const uint16_t * reg_addr,
             rs_size_t reg_size,
             uint32_t * reg )
```

get data from Sensor

Definition at line 30 of file rs24g_ctl_common.c.

12.6.1.2 get_devfifo()

```
rs_ret_t get_devfifo (
             rs_handle_t handle,
             uint32_t timeout,
             rs_size_t fifo_size,
             uint8_t * fifo_data,
             struct rs_fifo_info * fifo_info )
```

get FIFO data from Sensor

Definition at line 74 of file rs24g_ctl_common.c.

12.6.1.3 get_size_fifo()

```
rs_ret_t get_size_fifo (
             rs_handle_t handle,
             rs_size_t * size )
```

get frame size in FIFO

Definition at line 138 of file rs24g_ctl_common.c.

12.7 lib/rs24g_ctl_core/rs24g_ctl_distance.c File Reference

Functions

- RS_IF rs_ret_t rs_get_distance (rs_handle_t handle, uint32_t timeout, struct rs_distance_data *data)
- RS_IF rs_ret_t rs_set_peak_level_lower (rs_handle_t handle, uint8_t level)
- RS_IF rs_ret_t rs_get_peak_level_lower (rs_handle_t handle, uint8_t *level)

12.7.1 Function Documentation

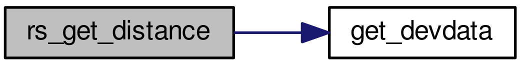
12.7.1.1 rs_get_distance()

```
RS_IF rs_ret_t rs_get_distance (
             rs_handle_t handle,
             uint32_t timeout,
             struct rs_distance_data * data )
```

get Distance

Definition at line 32 of file rs24g_ctl_distance.c.

Here is the call graph for this function:



12.7.1.2 rs_get_peak_level_lower()

```
RS_IF rs_ret_t rs_get_peak_level_lower (
            rs_handle_t handle,
            uint8_t * level )
```

get lower limit for peak level

Definition at line 115 of file rs24g_ctl_distance.c.

12.7.1.3 rs_set_peak_level_lower()

```
RS_IF rs_ret_t rs_set_peak_level_lower (
            rs_handle_t handle,
            uint8_t level )
```

set lower limit for peak level

Definition at line 91 of file rs24g_ctl_distance.c.

12.8 lib/rs24g_ctl_core/rs24g_ctl_motion.c File Reference

Functions

- RS_IF rs_ret_t rs_get_motion (rs_handle_t handle, rs_bool_t ∗motion)
- RS_IF rs_ret_t rs_wait_motion_change (rs_handle_t handle, uint32_t timeout, rs_bool_t ∗motion)
- RS_IF rs_ret_t rs_setup_smoothed_level (rs_handle_t handle, rs_smoothedope_t smoothed_level_ope, uint32_t fft_point, uint16_t addr, const uint32_t ∗val, rs_size_t num)
- RS_IF rs_ret_t rs_read_smoothed_level (rs_handle_t handle, uint32_t fft_point, uint16_t addr, uint32_t ∗val, rs_size_t num)

12.8.1 Function Documentation

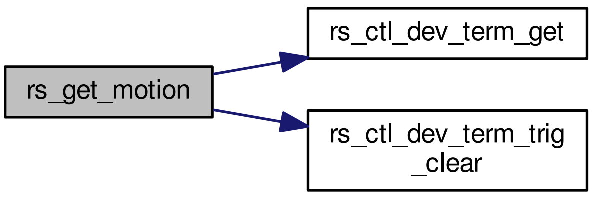
12.8.1.1 rs_get_motion()

```
RS_IF rs_ret_t rs_get_motion (
            rs_handle_t handle,
            rs_bool_t * motion )
```

get Motion

Definition at line 31 of file rs24g_ctl_motion.c.

Here is the call graph for this function:



12.8.1.2 rs_read_smoothed_level()

```
RS_IF rs_ret_t rs_read_smoothed_level (
             rs_handle_t handle,
             uint32_t fft_point,
             uint16_t addr,
             uint32_t * val,
             rs_size_t num )
```

read smoothed level

Definition at line 217 of file rs24g_ctl_motion.c.

12.8.1.3 rs_setup_smoothed_level()

```
RS_IF rs_ret_t rs_setup_smoothed_level (
             rs_handle_t handle,
             rs_smoothedope_t smoothed_level_ope,
             uint32_t fft_point,
             uint16_t addr,
             const uint32_t * val,
             rs_size_t num )
```

setup smoothed level

Definition at line 161 of file rs24g_ctl_motion.c.

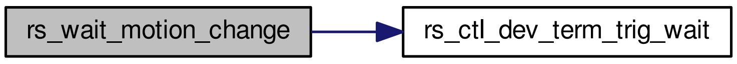
12.8.1.4 rs_wait_motion_change()

```
RS_IF rs_ret_t rs_wait_motion_change (
             rs_handle_t handle,
             uint32_t timeout,
             rs_bool_t * motion )
```

wait changeing motion

Definition at line 73 of file rs24g_ctl_motion.c.

Here is the call graph for this function:



12.9 lib/rs24g_ctl_core/rs24g_ctl_setup.c File Reference

Functions

- static rs_ret_t rs_setup_seq (rs_handle_t handle)
- RS_IF rs_ret_t rs_load_seqcode (rs_handle_t handle, rs_mode_t mode)

12.9.1 Function Documentation

12.9.1.1 rs_load_seqcode()

```
RS_IF rs_ret_t rs_load_seqcode (
             rs_handle_t handle,
             rs_mode_t mode )
```

Loading Sequencer Code

Definition at line 154 of file rs24g_ctl_setup.c.

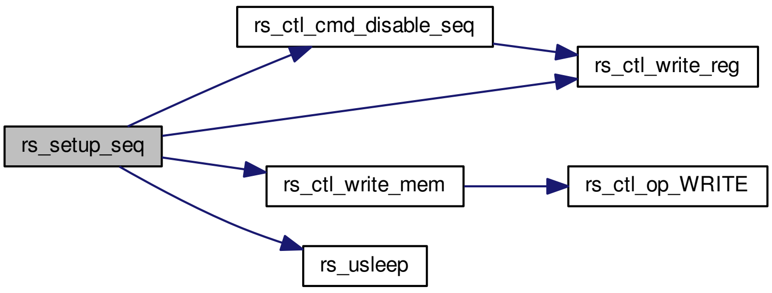
12.9.1.2 rs_setup_seq()

```
static rs_ret_t rs_setup_seq (
             rs_handle_t handle )  [static]
```

Setting registers, Sequencer Code and FFT parameters [8], [9], [10]

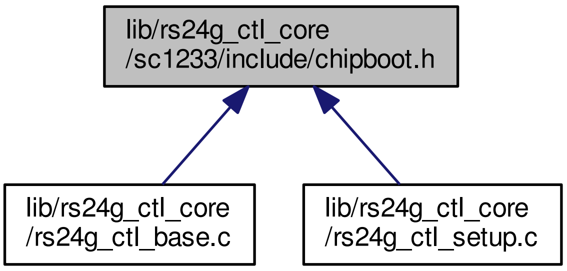
Definition at line 68 of file rs24g_ctl_setup.c.

Here is the call graph for this function:



12.10 lib/rs24g_ctl_core/sc1233/include/chipboot.h File Reference

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


lib/rs24g_ctl_core /sc1233/include/chipboot.h
lib/rs24g_ctl_core /rs24g_ctl_base.c
lib/rs24g_ctl_core /rs24g_ctl_setup.c

Macros

- #define chipboot(...) rs_ctl_cmd_chipboot_sc1233(__VA_ARGS__)

12.10.1 Macro Definition Documentation

12.10.1.1 chipboot

```
#define chipboot(
              ...   ) rs_ctl_cmd_chipboot_sc1233(__VA_ARGS__)
```

boot ("Start-Up") command

Definition at line 22 of file chipboot.h.

12.11 lib/rs24g_ctl_setup_core/rs24g_ctl_setup_base.c File Reference

Functions

- RS_IF rs_ret_t rs_setup_param (rs_handle_t handle, rs_mode_t mode, const void ∗param)
- RS_IF rs_ret_t rs_update_param (rs_handle_t handle, rs_mode_t mode, const void ∗param)

12.11.1 Function Documentation

12.11.1.1 rs_setup_param()

```
RS_IF rs_ret_t rs_setup_param (
             rs_handle_t handle,
             rs_mode_t mode,
             const void * param )
```

Setup Parameters

Parameters

`in`	*handle*	Sensor Handle
`in`	*mode*	Sensor Operation Mode
`in`	*param*	Sensor Parameters

Definition at line 42 of file rs24g_ctl_setup_base.c.

12.11.1.2 rs_update_param()

```
RS_IF rs_ret_t rs_update_param (
            rs_handle_t handle,
            rs_mode_t mode,
            const void * param )
```

Update Parameters

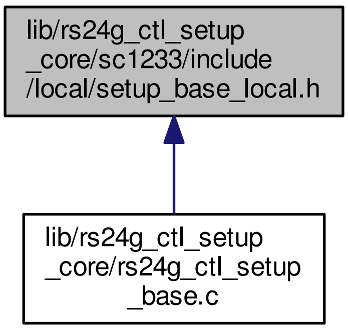
Parameters

`in`	*handle*	Sensor Handle
`in`	*mode*	Sensor Operation Mode
`in`	*param*	Sensor Parameters for updates

Definition at line 69 of file rs24g_ctl_setup_base.c.

12.12 lib/rs24g_ctl_setup_core/sc1233/include/local/setup_base_local.h File Reference

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


lib/rs24g_ctl_setup_core/sc1233/include/local/setup_base_local.h
lib/rs24g_ctl_setup_core/rs24g_ctl_setup_base.c

Functions

- static __inline rs_ret_t rs_setup_param_local (rs_mode_t mode, const void ∗param, rs_code_ref_t code, rs_resource_holder_t res)
- static __inline rs_ret_t rs_get_ctl_mode_from_setup_mode (rs_mode_t setup_mode, rs_resource_holder_t res, rs_mode_t ∗p_ctl_mode)
- static __inline rs_ret_t rs_update_param_local (rs_handle_t handle, rs_mode_t mode, const void ∗param)

12.12.1 Function Documentation

12.12.1.1 rs_get_ctl_mode_from_setup_mode()

```
static __inline rs_ret_t rs_get_ctl_mode_from_setup_mode (
            rs_mode_t setup_mode,
            rs_resource_holder_t res,
            rs_mode_t * p_ctl_mode )  [static]
```

get mode for control

Definition at line 111 of file setup_base_local.h.

12.12.1.2 rs_setup_param_local()

```
static __inline rs_ret_t rs_setup_param_local (
            rs_mode_t mode,
            const void * param,
            rs_code_ref_t code,
            rs_resource_holder_t res )  [static]
```

prepare parameters

Definition at line 55 of file setup_base_local.h.

12.12.1.3 rs_update_param_local()

```
static __inline rs_ret_t rs_update_param_local (
            rs_handle_t handle,
            rs_mode_t mode,
            const void * param )  [static]
```

update parameters

Definition at line 128 of file setup_base_local.h.

12.13 lib/rs24g_ctl_setup_core/setup_common.c File Reference

Functions

- rs_ret_t rs_setup_interval (rs_handle_t handle, uint32_t interval)
- rs_ret_t rs_setup_hpf (rs_handle_t handle, rs_hpf_t hpf)

12.13.1 Function Documentation

12.13.1.1 rs_setup_hpf()

```
rs_ret_t rs_setup_hpf (
            rs_handle_t handle,
            rs_hpf_t hpf )
```

Update HPF register

Definition at line 180 of file setup_common.c.

12.13.1.2 rs_setup_interval()

```
rs_ret_t rs_setup_interval (
            rs_handle_t handle,
            uint32_t interval )
```

Update interval register

Definition at line 170 of file setup_common.c.

12.14 lib/rs24g_ctl_setup_core/setup_distance.c File Reference

Functions

- rs_ret_t rs_setup_beta (rs_handle_t handle, uint8_t beta)
- rs_ret_t rs_setup_range_peak (rs_handle_t handle, uint32_t upper, uint32_t lower)

12.14.1 Function Documentation

12.14.1.1 rs_setup_beta()

```
rs_ret_t rs_setup_beta (
            rs_handle_t handle,
            uint8_t beta )
```

Update smoothing factor register

Definition at line 42 of file setup_distance.c.

12.14.1.2 rs_setup_range_peak()

```
rs_ret_t rs_setup_range_peak (
            rs_handle_t handle,
            uint32_t upper,
            uint32_t lower )
```

Update distance measurement frequency index register

Definition at line 52 of file setup_distance.c.

12.15 lib/rs24g_ctl_setup_core/setup_distance_wide.c File Reference

Functions

- rs_ret_t distance_wide_getcode (const struct rs_distance_param ∗lp, rs_code_ref_t code)
- rs_ret_t distance_wide_update_param (rs_handle_t handle, const struct rs_distance_param ∗lp)

12.15.1 Function Documentation

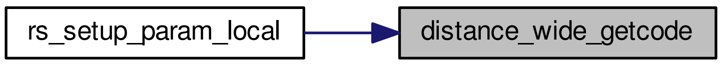
12.15.1.1 distance_wide_getcode()

```
rs_ret_t distance_wide_getcode (
            const struct rs_distance_param * lp,
            rs_code_ref_t code )
```

prepare parameters for distance detection

Definition at line 33 of file setup_distance_wide.c.

Here is the caller graph for this function:



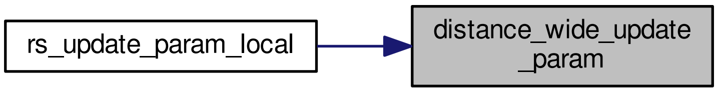
12.15.1.2 distance_wide_update_param()

```
rs_ret_t distance_wide_update_param (
             rs_handle_t handle,
             const struct rs_distance_param * lp )
```

update parameters for distance detection

Definition at line 252 of file setup_distance_wide.c.

Here is the caller graph for this function:



12.16 lib/rs24g_ctl_setup_core/setup_motion.c File Reference

Functions

- rs_ret_t rs_setup_alpha (rs_handle_t handle, uint8_t alpha)
- rs_ret_t rs_setup_motion_threshold (rs_handle_t handle, uint16_t motion_threshold)
- rs_ret_t rs_setup_startup_count (rs_handle_t handle, uint8_t startup_count)
- rs_ret_t rs_setup_range_motion (rs_handle_t handle, uint32_t upper, uint32_t lower)

12.16.1 Function Documentation

12.16.1.1 rs_setup_alpha()

```
rs_ret_t rs_setup_alpha (
             rs_handle_t handle,
             uint8_t alpha )
```

Update smoothing factor register

Definition at line 45 of file setup_motion.c.

12.16.1.2 rs_setup_motion_threshold()

```
rs_ret_t rs_setup_motion_threshold (
             rs_handle_t handle,
             uint16_t motion_threshold )
```

Update entry motion detection threshold register

Definition at line 55 of file setup_motion.c.

12.16.1.3 rs_setup_range_motion()

```
rs_ret_t rs_setup_range_motion (
             rs_handle_t handle,
             uint32_t upper,
             uint32_t lower )
```

Update entry motion detection frequency index register

Definition at line 75 of file setup_motion.c.

12.16.1.4 rs_setup_startup_count()

```
rs_ret_t rs_setup_startup_count (
             rs_handle_t handle,
             uint8_t startup_count )
```

Update start-up counter register

Definition at line 65 of file setup_motion.c.

12.17 lib/rs24g_ctl_setup_core/setup_motion_wide.c File Reference

Functions

- rs_ret_t motion_wide_getcode (const struct rs_motion_param ∗lp, rs_code_ref_t code)
- rs_ret_t motion_wide_update_param (rs_handle_t handle, const struct rs_motion_param ∗lp)

12.17.1 Function Documentation

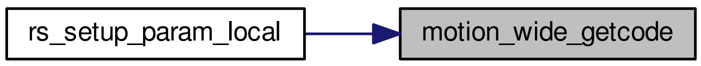
12.17.1.1 motion_wide_getcode()

```
rs_ret_t motion_wide_getcode (
             const struct rs_motion_param * lp,
             rs_code_ref_t code )
```

prepare parameters for motion detection

Definition at line 34 of file setup_motion_wide.c.

Here is the caller graph for this function:



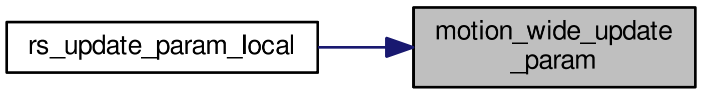
12.17.1.2 motion_wide_update_param()

```
rs_ret_t motion_wide_update_param (
             rs_handle_t handle,
             const struct rs_motion_param * lp )
```

update parameters for motion detection

Definition at line 104 of file setup_motion_wide.c.

Here is the caller graph for this function:



12.18 lib/rs_ctl_dev_poc/include/rs_i2c.h File Reference

Sensor driver implemantation for I2C on MBED@HRM1017.

Classes

- class rs_i2c

12.18.1 Detailed Description

Sensor driver implemantation for I2C on MBED@HRM1017.

Version

(PRELIMINARY)

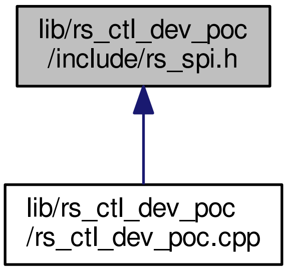
Warning

This is just a sample source code.

12.19 lib/rs_ctl_dev_poc/include/rs_spi.h File Reference

Sensor driver implemantation for SPI on MBED@HRM1017.

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


lib/rs_ctl_dev_poc /include/rs_spi.h
lib/rs_ctl_dev_poc /rs_ctl_dev_poc.cpp

Classes

- class rs_spi

12.19.1 Detailed Description

Sensor driver implemantation for SPI on MBED@HRM1017.

Version

(PRELIMINARY)

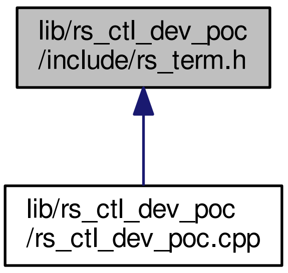
Warning

This is just a sample source code.

12.20 lib/rs_ctl_dev_poc/include/rs_term.h File Reference

Sensor driver implemantation for GPIO on MBED@HRM1017.

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



Classes

- class rs_term

12.20.1 Detailed Description

Sensor driver implemantation for GPIO on MBED@HRM1017.

Version

(PRELIMINARY)

Warning

This is just a sample source code.

12.21 lib/rs_ctl_dev_poc/rs_ctl_dev_poc.cpp File Reference

Functions

- RS_IF rs_ret_t rs_dev_open (rs_ctl_dev_t ∗dev)
- RS_IF rs_ret_t rs_dev_close (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_dev_write (rs_ctl_dev_t dev, const uint8_t ∗wdata, size_t size)
- RS_IF rs_ret_t rs_dev_write_low_memory (rs_ctl_dev_t dev, const uint8_t ∗hdata, size_t hsize, const uint8↩_t ∗wdata, size_t wsize)
- RS_IF rs_ret_t rs_dev_read (rs_ctl_dev_t dev, const uint8_t ∗wdata, size_t wsize, uint8_t ∗rdata, size_t rsize)
- RS_IF rs_ret_t rs_dev_term_get (rs_ctl_dev_t dev, uint32_t term, rs_bool_t ∗val)
- RS_IF rs_ret_t rs_dev_term_set (rs_ctl_dev_t dev, uint32_t term, rs_bool_t val)
- RS_IF rs_ret_t rs_dev_term_set_trigger (rs_ctl_dev_t dev, uint32_t term, uint32_t trigger)
- RS_IF rs_ret_t rs_dev_term_clear_trigger (rs_ctl_dev_t dev, uint32_t term)
- RS_IF rs_ret_t rs_dev_term_wait_trigger (rs_ctl_dev_t dev, uint32_t term, uint32_t timeout, uint32_t trigger, rs_bool_t ∗val)

12.21.1 Function Documentation

12.21.1.1 rs_dev_close()

```
RS_IF rs_ret_t rs_dev_close (
             rs_ctl_dev_t dev )
```

close device file

Definition at line 56 of file rs_ctl_dev_poc.cpp.

12.21.1.2 rs_dev_open()

```
RS_IF rs_ret_t rs_dev_open (
             rs_ctl_dev_t * dev )
```

open device file

Definition at line 42 of file rs_ctl_dev_poc.cpp.

12.21.1.3 rs_dev_read()

```
RS_IF rs_ret_t rs_dev_read (
             rs_ctl_dev_t dev,
             const uint8_t * wdata,
             size_t wsize,
             uint8_t * rdata,
             size_t rsize )
```

read data from device

See also

rs_spi::read(), rs_i2c::read()

Definition at line 81 of file rs_ctl_dev_poc.cpp.

12.21.1.4 rs_dev_term_clear_trigger()

```
RS_IF rs_ret_t rs_dev_term_clear_trigger (
             rs_ctl_dev_t dev,
             uint32_t term )
```

clear triggered flag

See also

rs_term::clear_trigger()

Definition at line 117 of file rs_ctl_dev_poc.cpp.

12.21.1.5 rs_dev_term_get()

```
RS_IF rs_ret_t rs_dev_term_get (
             rs_ctl_dev_t dev,
             uint32_t term,
             rs_bool_t * val )
```

read device terminal (GPIO) value

See also

rs_term::get()

Definition at line 93 of file rs_ctl_dev_poc.cpp.

12.21.1.6 rs_dev_term_set()

```
RS_IF rs_ret_t rs_dev_term_set (
              rs_ctl_dev_t dev,
              uint32_t term,
              rs_bool_t val )
```

write device terminal (GPIO) value

See also

rs_term::set()

Definition at line 101 of file rs_ctl_dev_poc.cpp.

12.21.1.7 rs_dev_term_set_trigger()

```
RS_IF rs_ret_t rs_dev_term_set_trigger (
              rs_ctl_dev_t dev,
              uint32_t term,
              uint32_t trigger )
```

setup terminal (GPIO) as trigger

See also

rs_term::set_trigger()

Definition at line 109 of file rs_ctl_dev_poc.cpp.

12.21.1.8 rs_dev_term_wait_trigger()

```
RS_IF rs_ret_t rs_dev_term_wait_trigger (
              rs_ctl_dev_t dev,
              uint32_t term,
              uint32_t timeout,
              uint32_t trigger,
              rs_bool_t * val )
```

wait trigger event

See also

rs_term::wait_trigger()

Definition at line 125 of file rs_ctl_dev_poc.cpp.

12.21.1.9 rs_dev_write()

```
RS_IF rs_ret_t rs_dev_write (
              rs_ctl_dev_t dev,
              const uint8_t * wdata,
              size_t size )
```

write data to device

See also

rs_spi::write(), rs_i2c::write()

Definition at line 65 of file rs_ctl_dev_poc.cpp.

12.21.1.10 rs_dev_write_low_memory()

```
RS_IF rs_ret_t rs_dev_write_low_memory (
              rs_ctl_dev_t dev,
              const uint8_t * hdata,
              size_t hsize,
              const uint8_t * wdata,
              size_t wsize )
```

write data to device for low memory usage

See also

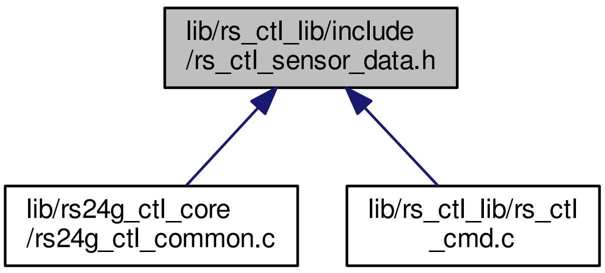
rs_spi::write_low_memory(), rs_i2c::write_low_memory()

Definition at line 73 of file rs_ctl_dev_poc.cpp.

12.22 lib/rs_ctl_lib/include/rs_ctl_sensor_data.h File Reference

sensor data

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



12.22.1 Detailed Description

sensor data

Version

(PRELIMINARY)

Warning

This is a part of sensor library source code for Evaluation Kit.

12.23 lib/rs_ctl_lib/rs_ctl_cmd.c File Reference

Functions

- RS_IF rs_ret_t rs_ctl_cmd_shutdown (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_cmd_disable_seq (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_cmd_enable_seq (rs_ctl_dev_t dev)
- static rs_ret_t read_status (rs_ctl_dev_t dev, const struct rs_ctl_sensor_data_set ∗set, struct rs_ctl_sensor↩_data ∗data)
- static rs_ret_t read_registers (rs_ctl_dev_t dev, const struct rs_ctl_sensor_data_set ∗set, struct rs_ctl_↩sensor_data ∗data)
- static rs_ret_t read_fifo (rs_ctl_dev_t dev, const struct rs_ctl_sensor_data_set ∗set, struct rs_ctl_sensor_data ∗data)
- RS_IF rs_ret_t rs_ctl_cmd_get_sensor_data (rs_ctl_dev_t dev, const struct rs_ctl_sensor_data_set ∗set, struct rs_ctl_sensor_data ∗data)
- RS_IF rs_ret_t rs_ctl_cmd_wait_and_get_sensor_data (rs_ctl_dev_t dev, uint32_t timeout, const struct rs_↩ctl_sensor_data_set ∗set, struct rs_ctl_sensor_data ∗data)

12.23.1 Function Documentation

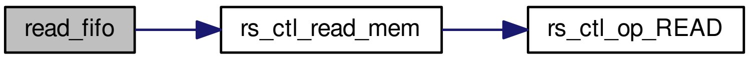
12.23.1.1 read_fifo()

```
static rs_ret_t read_fifo (
            rs_ctl_dev_t dev,
            const struct rs_ctl_sensor_data_set ∗ set,
            struct rs_ctl_sensor_data ∗ data )  [static]
```

read FIFO

Definition at line 213 of file rs_ctl_cmd.c.

Here is the call graph for this function:



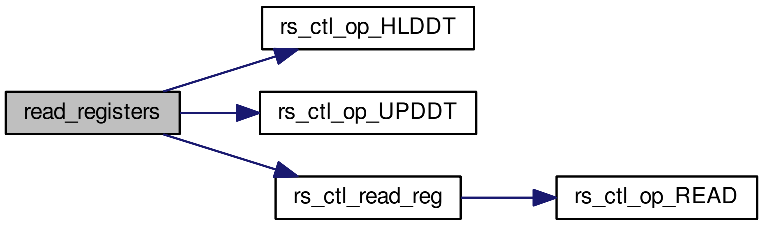
12.23.1.2 read_registers()

```
static rs_ret_t read_registers (
            rs_ctl_dev_t dev,
            const struct rs_ctl_sensor_data_set * set,
            struct rs_ctl_sensor_data * data )  [static]
```

read registers

Definition at line 190 of file rs_ctl_cmd.c.

Here is the call graph for this function:



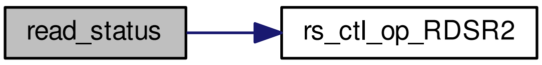
12.23.1.3 read_status()

```
static rs_ret_t read_status (
            rs_ctl_dev_t dev,
            const struct rs_ctl_sensor_data_set * set,
            struct rs_ctl_sensor_data * data )  [static]
```

read status register

Definition at line 178 of file rs_ctl_cmd.c.

Here is the call graph for this function:



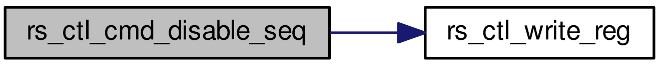
12.23.1.4 rs_ctl_cmd_disable_seq()

```
RS_IF rs_ret_t rs_ctl_cmd_disable_seq (
            rs_ctl_dev_t dev )
```

"disable Sequencer" to write Sequencer Code [9]

Definition at line 94 of file rs_ctl_cmd.c.

Here is the call graph for this function:



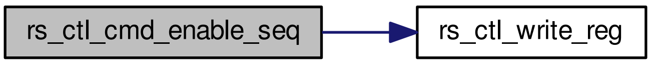
12.23.1.5 rs_ctl_cmd_enable_seq()

```
RS_IF rs_ret_t rs_ctl_cmd_enable_seq (
             rs_ctl_dev_t dev )
```

"enable Sequencer" (makes not possible to write Sequencer Code) [10]

Definition at line 107 of file rs_ctl_cmd.c.

Here is the call graph for this function:



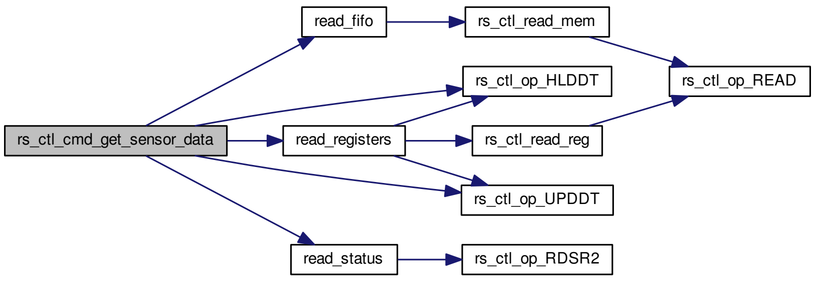
12.23.1.6 rs_ctl_cmd_get_sensor_data()

```
RS_IF rs_ret_t rs_ctl_cmd_get_sensor_data (
             rs_ctl_dev_t dev,
             const struct rs_ctl_sensor_data_set * set,
             struct rs_ctl_sensor_data * data )
```

get distance data from Sensor FIFO or registers [11]

Definition at line 126 of file rs_ctl_cmd.c.

Here is the call graph for this function:



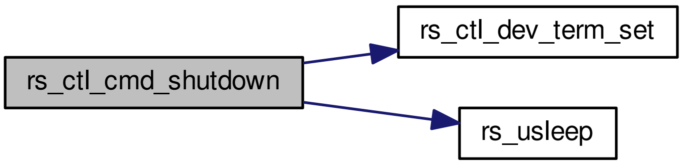
12.23.1.7 rs_ctl_cmd_shutdown()

```
RS_IF rs_ret_t rs_ctl_cmd_shutdown (
              rs_ctl_dev_t dev )
```

"shutdown" command [12]

Definition at line 80 of file rs_ctl_cmd.c.

Here is the call graph for this function:



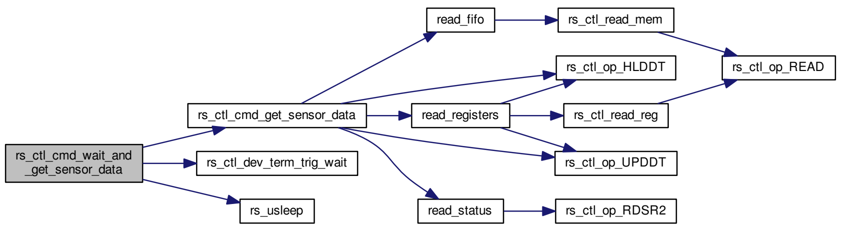
12.23.1.8 rs_ctl_cmd_wait_and_get_sensor_data()

```
RS_IF rs_ret_t rs_ctl_cmd_wait_and_get_sensor_data (
              rs_ctl_dev_t dev,
              uint32_t timeout,
              const struct rs_ctl_sensor_data_set * set,
              struct rs_ctl_sensor_data * data )
```

wait trigger of OR pin and get sensor data

Definition at line 151 of file rs_ctl_cmd.c.

Here is the call graph for this function:



12.24 lib/rs_ctl_lib/rs_ctl_cmd_sc1233.c File Reference

Functions

- RS_IF rs_ret_t rs_ctl_cmd_chipboot_sc1233 (rs_ctl_dev_t dev)

12.24.1 Function Documentation

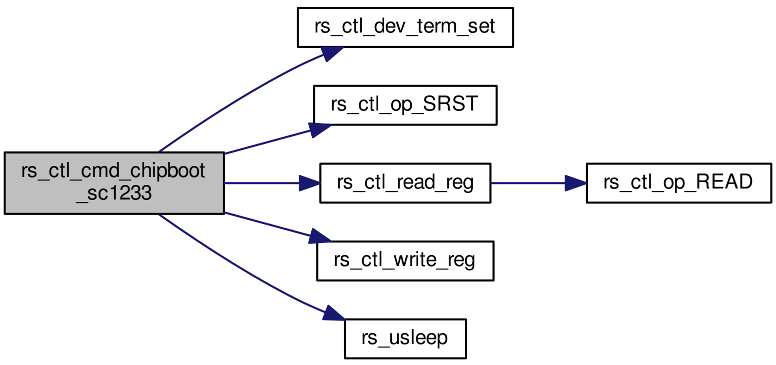
12.24.1.1 rs_ctl_cmd_chipboot_sc1233()

```
RS_IF rs_ret_t rs_ctl_cmd_chipboot_sc1233 (
            rs_ctl_dev_t dev )
```

boot ("Start-Up") command [1]

Definition at line 27 of file rs_ctl_cmd_sc1233.c.

Here is the call graph for this function:



12.25 lib/rs_ctl_lib/rs_ctl_data.c File Reference

Functions

- RS_IF uint16_t rs_calc_crc16 (uint16_t crc16, const uint8_t ∗data, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_write_mem (rs_ctl_dev_t dev, uint32_t addr, const uint8_t ∗data, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_read_mem (rs_ctl_dev_t dev, uint32_t addr, uint8_t ∗data, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_write_reg (rs_ctl_dev_t dev, uint32_t addr, uint32_t data)
- RS_IF rs_ret_t rs_ctl_read_reg (rs_ctl_dev_t dev, uint32_t addr, uint32_t ∗data)
- RS_IF rs_ret_t rs_ctl_write_regs (rs_ctl_dev_t dev, uint32_t addr, const uint32_t ∗data, rs_size_t num)
- RS_IF rs_ret_t rs_ctl_read_regs (rs_ctl_dev_t dev, uint32_t addr, uint32_t ∗data, rs_size_t num)

12.25.1 Function Documentation

12.25.1.1 rs_calc_crc16()

```
RS_IF uint16_t rs_calc_crc16 (
             uint16_t crc16,
             const uint8_t ∗ data,
             rs_size_t size )
```

calculate CRC-16

Definition at line 25 of file rs_ctl_data.c.

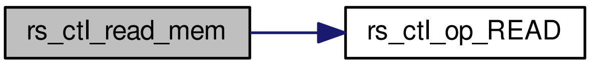
12.25.1.2 rs_ctl_read_mem()

```
RS_IF rs_ret_t rs_ctl_read_mem (
             rs_ctl_dev_t dev,
             uint32_t addr,
             uint8_t ∗ data,
             rs_size_t size )
```

read Sensor Memory (FIFO memory)

Definition at line 78 of file rs_ctl_data.c.

Here is the call graph for this function:



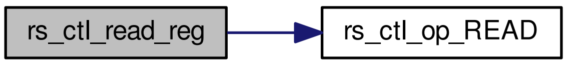
12.25.1.3 rs_ctl_read_reg()

```
RS_IF rs_ret_t rs_ctl_read_reg (
             rs_ctl_dev_t dev,
             uint32_t addr,
             uint32_t ∗ data )
```

read a Sensor Register

Definition at line 102 of file rs_ctl_data.c.

Here is the call graph for this function:



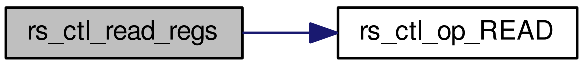
12.25.1.4 rs_ctl_read_regs()

```
RS_IF rs_ret_t rs_ctl_read_regs (
              rs_ctl_dev_t dev,
              uint32_t addr,
              uint32_t * data,
              rs_size_t num )
```

read Sensor Registers

Definition at line 141 of file rs_ctl_data.c.

Here is the call graph for this function:



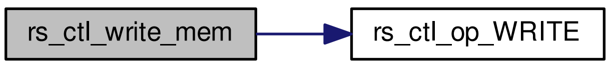
12.25.1.5 rs_ctl_write_mem()

```
RS_IF rs_ret_t rs_ctl_write_mem (
              rs_ctl_dev_t dev,
              uint32_t addr,
              const uint8_t * data,
              rs_size_t size )
```

write Sensor Memory (Sequencer Code or FFT Window Function)

Definition at line 68 of file rs_ctl_data.c.

Here is the call graph for this function:



12.25.1.6 rs_ctl_write_reg()

```
RS_IF rs_ret_t rs_ctl_write_reg (
              rs_ctl_dev_t dev,
              uint32_t addr,
              uint32_t data )
```

write a Sensor Register

Definition at line 88 of file rs_ctl_data.c.

12.25.1.7 rs_ctl_write_regs()

```
RS_IF rs_ret_t rs_ctl_write_regs (
              rs_ctl_dev_t dev,
              uint32_t addr,
              const uint32_t * data,
              rs_size_t num )
```

write Sensor Registers

Definition at line 118 of file rs_ctl_data.c.

12.26 lib/rs_ctl_lib/rs_ctl_op.c File Reference

Functions

- RS_IF rs_ret_t rs_ctl_op_HRST (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_SRST (rs_ctl_dev_t dev, int with_extra)
- RS_IF rs_ret_t rs_ctl_op_DSLEEP (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_WRSR (rs_ctl_dev_t dev, uint8_t data)
- RS_IF rs_ret_t rs_ctl_op_RDSR (rs_ctl_dev_t dev, uint8_t ∗data)
- RS_IF rs_ret_t rs_ctl_op_WRITE (rs_ctl_dev_t dev, uint32_t address, const uint8_t ∗data, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_op_READ (rs_ctl_dev_t dev, uint32_t address, uint8_t ∗data, rs_size_t size)
- RS_IF rs_ret_t rs_ctl_op_RDSR2 (rs_ctl_dev_t dev, uint8_t ∗data)
- RS_IF rs_ret_t rs_ctl_op_ENATM (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_DISTM (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_RUNTM (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_STPTM (rs_ctl_dev_t dev)
- RS_IF rs_ret_t rs_ctl_op_HLDDT (rs_ctl_dev_t dev, int with_timer, int in_deep_sleep)
- RS_IF rs_ret_t rs_ctl_op_UPDDT (rs_ctl_dev_t dev, int with_timer)

12.26.1 Function Documentation

12.26.1.1 rs_ctl_op_DISTM()

```
RS_IF rs_ret_t rs_ctl_op_DISTM (
              rs_ctl_dev_t dev )
```

send DISTM Command

See also

rs_ctl_dev_write()

Definition at line 178 of file rs_ctl_op.c.

12.26.1.2 rs_ctl_op_DSLEEP()

```
RS_IF rs_ret_t rs_ctl_op_DSLEEP (
              rs_ctl_dev_t dev )
```

send DSLEEP Command

See also

rs_ctl_dev_write()

Definition at line 115 of file rs_ctl_op.c.

12.26.1.3 rs_ctl_op_ENATM()

```
RS_IF rs_ret_t rs_ctl_op_ENATM (
              rs_ctl_dev_t dev )
```

send ENATM Command

See also

rs_ctl_dev_write()

Definition at line 169 of file rs_ctl_op.c.

12.26.1.4 rs_ctl_op_HLDDT()

```
RS_IF rs_ret_t rs_ctl_op_HLDDT (
              rs_ctl_dev_t dev,
              int with_timer,
              int in_deep_sleep )
```

send HLDDT Command

See also

rs_ctl_dev_write()

Definition at line 205 of file rs_ctl_op.c.

12.26.1.5 rs_ctl_op_HRST()

```
RS_IF rs_ret_t rs_ctl_op_HRST (
              rs_ctl_dev_t dev )
```

send HRST Command

See also

rs_ctl_dev_write()

Definition at line 87 of file rs_ctl_op.c.

12.26.1.6 rs_ctl_op_RDSR()

```
RS_IF rs_ret_t rs_ctl_op_RDSR (
              rs_ctl_dev_t dev,
              uint8_t * data )
```

send RDSR Command

See also

rs_ctl_dev_read()

Definition at line 133 of file rs_ctl_op.c.

12.26.1.7 rs_ctl_op_RDSR2()

```
RS_IF rs_ret_t rs_ctl_op_RDSR2 (
              rs_ctl_dev_t dev,
              uint8_t * data )
```

send RDSR2 Command

See also

rs_ctl_dev_read()

Definition at line 160 of file rs_ctl_op.c.

12.26.1.8 rs_ctl_op_READ()

```
RS_IF rs_ret_t rs_ctl_op_READ (
              rs_ctl_dev_t dev,
              uint32_t address,
              uint8_t * data,
              rs_size_t size )
```

send READ Command

See also

rs_ctl_dev_read()

Definition at line 151 of file rs_ctl_op.c.

12.26.1.9 rs_ctl_op_RUNTM()

```
RS_IF rs_ret_t rs_ctl_op_RUNTM (
            rs_ctl_dev_t dev )
```

send RUNTM Command

See also

rs_ctl_dev_write()

Definition at line 187 of file rs_ctl_op.c.

12.26.1.10 rs_ctl_op_SRST()

```
RS_IF rs_ret_t rs_ctl_op_SRST (
            rs_ctl_dev_t dev,
            int with_extra )
```

send SRST Command

See also

rs_ctl_dev_write()

Definition at line 96 of file rs_ctl_op.c.

12.26.1.11 rs_ctl_op_STPTM()

```
RS_IF rs_ret_t rs_ctl_op_STPTM (
            rs_ctl_dev_t dev )
```

send STPTM Command

See also

rs_ctl_dev_write()

Definition at line 196 of file rs_ctl_op.c.

12.26.1.12 rs_ctl_op_UPDDT()

```
RS_IF rs_ret_t rs_ctl_op_UPDDT (
            rs_ctl_dev_t dev,
            int with_timer )
```

send UPDDT Command

See also

rs_ctl_dev_write()

Definition at line 228 of file rs_ctl_op.c.

12.26.1.13 rs_ctl_op_WRITE()

```
RS_IF rs_ret_t rs_ctl_op_WRITE (
             rs_ctl_dev_t dev,
             uint32_t address,
             const uint8_t * data,
             rs_size_t size )
```

send WRITE Command

See also

rs_ctl_dev_write()

Definition at line 142 of file rs_ctl_op.c.

12.26.1.14 rs_ctl_op_WRSR()

```
RS_IF rs_ret_t rs_ctl_op_WRSR (
             rs_ctl_dev_t dev,
             uint8_t data )
```

send WRSR Command

See also

rs_ctl_dev_write()

Definition at line 124 of file rs_ctl_op.c.

Bibliography

[1] In Datasheet [13], chapter "5.3.1. Chip boot", page 58. 3, 61

[2] In Datasheet [13], chapter "5.1. SPI and I2C Transaction", pages 40–50. 7

[3] In Datasheet [13], chapter "5.1.6. Read Transaction for register", pages 48–49. 31

[4] In Datasheet [13], chapter "5.1.7. Read Transaction for FIFO memory", pages 49–50. 31

[5] In Datasheet [13], chapter "5.1.3. Write Transaction for register", pages 45–46. 34

[6] In Datasheet [13], chapter "5.1.4. Write Transaction for Sequencer program code", pages 46–47. 34

[7] In Datasheet [13], chapter "5.1.5. Write Transaction for FFT window function", pages 47–48. 34

[8] In Datasheet [13], chapter "5.3.3. Writing FFT window function", page 59. 44

[9] In Datasheet [13], chapter "5.3.4. Writing sequencer codes", pages 59–60. 44, 58

[10] In Datasheet [13], chapter "5.3.5. Enabling sequencer", page 60. 44, 59

[11] In Datasheet [13], chapter "5.3.7. Reading results", pages 61–62. 59

[12] In Datasheet [13], chapter "5.3.9. Shutdown", page 62. 60

[13] Socionext Inc. *Datasheet SC1233AR3 24GHz radar sensor LSI*. Ver. 1.3. June. 2020. 1, 69

Index