miniPhi

1.0.3

Version 1.0

Thu Feb 12 2015 15:22:03

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

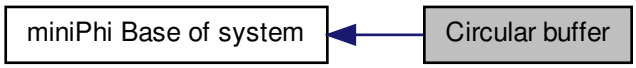
Chapter 2

Module Documentation

2.1 Circular buffer

Circular queueing system.

Collaboration diagram for Circular buffer:



Data Structures

- struct mp_circular_buffer_s
- struct mp_circular_s

Typedefs

- typedef struct mp_circular_s mp_circular_t
- typedef struct mp_circular_buffer_s mp_circular_buffer_t
- typedef void(∗ mp_circular_int_t)(mp_circular_t ∗cir)

Functions

- mp_ret_t mp_circular_init (mp_kernel_t ∗kernel, mp_circular_t ∗cir, mp_circular_int_t enable, mp_circular_int_t disable)

  Initiate circular buffer context.
- void mp_circular_fini (mp_circular_t ∗cir)

  Terminate circular buffer context.
- mp_circular_buffer_t ∗ mp_circular_read (mp_circular_t ∗cir)
- mp_ret_t mp_circular_write (mp_circular_t ∗cir, unsigned char ∗data, int size)
- void mp_circular_rxInterrupt (mp_circular_t ∗cir, unsigned char c)

  Circular buffer RX interrupt handler.

- unsigned char mp_circular_txInterrupt (mp_circular_t *cir, mp_bool_t *done)

  Circular buffer TX interrupt handler.

- int mp_circular_bufferSize ()

  Returns the circular buffer size.

2.1.1 Detailed Description

Circular queueing system.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

This library supports circular buffering used into generaly in communication systems. Common circular buffering respects the heap organisation and the interrupt model.

Example 1: Initiate context for RX

```
// Circular context for RX
ret = mp_circular_init(
        kernel, &hdl->rxCir,
        PHY_rxIntEnable, PHY_rxIntDisable
);
if(!ret) {
        mp_printk("PHY: Can not create RX circular");
        return(FALSE);
}

// place my user pointer
serial->txCir.user = serial;
```

2.1.2 Typedef Documentation

2.1.2.1 typedef struct mp_circular_buffer_s mp_circular_buffer_t

Definition at line 35 of file circular.h.

2.1.2.2 typedef void(* mp_circular_int_t)(mp_circular_t *cir)

Definition at line 37 of file circular.h.

2.1.2.3 typedef struct mp_circular_s mp_circular_t

Definition at line 34 of file circular.h.

2.1.3 Function Documentation

2.1.3.1 int mp_circular_bufferSize ( )

Returns the circular buffer size.

This returns the circular buffer size

Definition at line 293 of file circular.c.

References MP_CIRCULAR_BUFFER_SIZE.

2.1.3.2 void mp_circular_fini (mp_circular_t * *cir*)

Terminate circular buffer context.

This terminate a circular buffer context by freeing all circular buffer.

Parameters

in	*cir*	Circular context

Definition at line 96 of file circular.c.

References mp_circular_s::first, mp_circular_s::kernel, mp_mem_free(), and mp_circular_buffer_s::next.

Here is the call graph for this function:

2.1.3.3 mp_ret_t mp_circular_init (mp_kernel_t * *kernel,* mp_circular_t * *cir,* mp_circular_int_t *enable,* mp_circular_int_t *disable*)

Initiate circular buffer context.

This initiates a circular buffer context

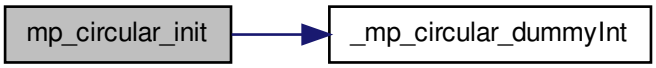
Parameters

in	*kernel*	Kernel handler
in	*cir*	Circular context
in	*enable*	Enable interrupt
in	*disable*	Disable interrupt

Definition at line 73 of file circular.c.

References _mp_circular_dummyInt(), mp_circular_s::disable, mp_circular_s::enable, mp_circular_s::first, mp_circular_s::kernel, mp_circular_s::last, NULL, and TRUE.

Here is the call graph for this function:

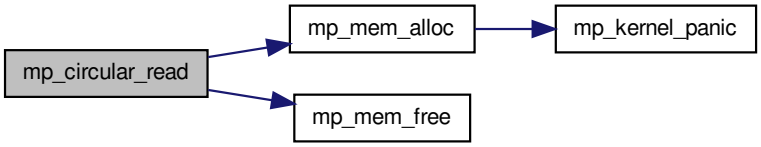


2.1.3.4 mp_circular_buffer_t* mp_circular_read (mp_circular_t * *cir*)

Definition at line 110 of file circular.c.

References mp_circular_s::disable, mp_circular_s::enable, mp_circular_s::first, mp_circular_s::kernel, mp_circular_s::last, mp_mem_alloc(), mp_mem_free(), mp_circular_buffer_s::next, NULL, mp_circular_buffer_s::size, and mp_circular_s::totalSize.

Here is the call graph for this function:



2.1.3.5 void mp_circular_rxInterrupt (mp_circular_t * *cir,* unsigned char *c*)

Circular buffer RX interrupt handler.

This control RX buffer interrupt.

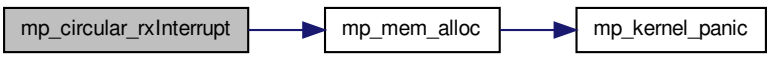
Parameters

in	*cir*	Circular context

Definition at line 206 of file circular.c.

References mp_circular_buffer_s::data, mp_circular_s::kernel, mp_circular_s::last, MP_CIRCULAR_BUFFER_SIZE, mp_mem_alloc(), mp_circular_buffer_s::next, NULL, mp_circular_buffer_s::size, and mp_circular_s::totalSize.

Here is the call graph for this function:



2.1.3.6 unsigned char mp_circular_txInterrupt (mp_circular_t * *cir,* mp_bool_t * *done*)

Circular buffer TX interrupt handler.

This control TX buffer interrupt. When there is no more buffer the *done pointer is set to YES and the returned char is not sent. If *done is set to NO then the function returns the unsigned char which must be sent by the PHY driver.

Parameters

in	*cir*	Circular context
out	*done*	Interrupt status

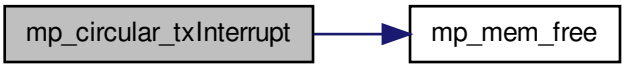
Returns

unsigned char has to be sent

Definition at line 240 of file circular.c.

References mp_circular_buffer_s::data, mp_circular_s::disable, mp_circular_s::first, mp_circular_s::kernel, mp_circular_s::last, mp_mem_free(), mp_circular_buffer_s::next, NO, NULL, mp_circular_buffer_s::pos, mp_circular_buffer_s::size, mp_circular_s::totalSize, and YES.

Here is the call graph for this function:

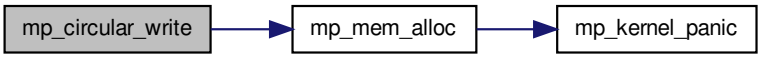


2.1.3.7 mp_ret_t mp_circular_write (mp_circular_t * *cir,* unsigned char * *data,* int *size*)

Definition at line 147 of file circular.c.

References mp_circular_buffer_s::data, mp_circular_s::disable, mp_circular_s::enable, mp_circular_s::first, mp_circular_s::kernel, mp_circular_s::last, MP_CIRCULAR_BUFFER_SIZE, mp_mem_alloc(), mp_circular_buffer_s::next, NULL, mp_circular_buffer_s::pos, mp_circular_buffer_s::size, mp_circular_s::totalSize, and TRUE.

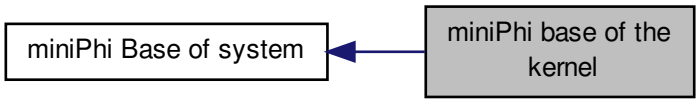
Here is the call graph for this function:



2.2 miniPhi base of the kernel

Provides logical base for an embedded kernel.

Collaboration diagram for miniPhi base of the kernel:



Functions

- void mp_kernel_init (mp_kernel_t ∗kernel, mp_kernel_onBoot_t onBoot, void ∗user)
- void mp_kernel_fini (mp_kernel_t ∗kernel)
- void mp_kernel_state (mp_kernel_t ∗kernel, char number)
- void mp_kernel_loop (mp_kernel_t ∗kernel)

2.2.1 Detailed Description

Provides logical base for an embedded kernel.

Version

1.0.0

Author

2014 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

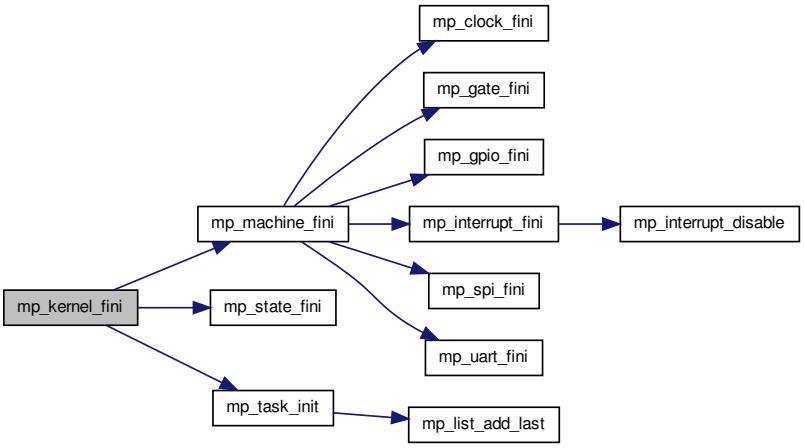
2.2.2 Function Documentation

2.2.2.1 void mp_kernel_fini (mp_kernel_t ∗ *kernel*)

Definition at line 102 of file kernel.c.

References mp_machine_fini(), mp_state_fini(), mp_task_init(), mp_kernel_s::states, and mp_kernel_s::tasks.

Here is the call graph for this function:

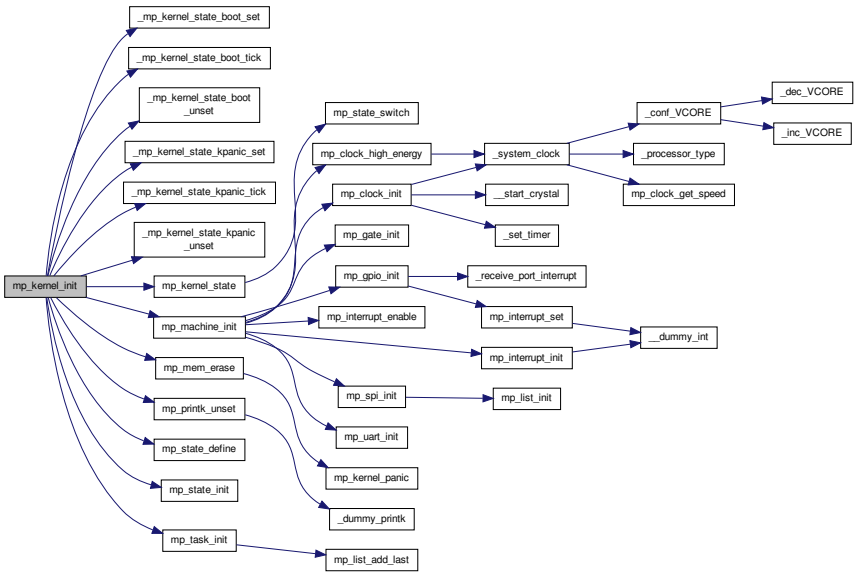


2.2.2.2 void mp_kernel_init (mp_kernel_t * *kernel,* mp_kernel_onBoot_t *onBoot,* void * *user*)

Definition at line 50 of file kernel.c.

References _mp_kernel_state_boot_set(), _mp_kernel_state_boot_tick(), _mp_kernel_state_boot_unset(), _mp_kernel_state_kpanic_set(), _mp_kernel_state_kpanic_tick(), _mp_kernel_state_kpanic_unset(), MP_KERNEL_BOOT, MP_KERNEL_KPANIC, mp_kernel_state(), MP_KERNEL_VERSION, mp_machine_init(), mp_mem_erase(), mp_printk_unset(), mp_state_define(), mp_state_init(), mp_task_init(), mp_kernel_s::onBoot, mp_kernel_s::onBootUser, mp_kernel_s::states, mp_kernel_s::tasks, and mp_kernel_s::version.

Here is the call graph for this function:

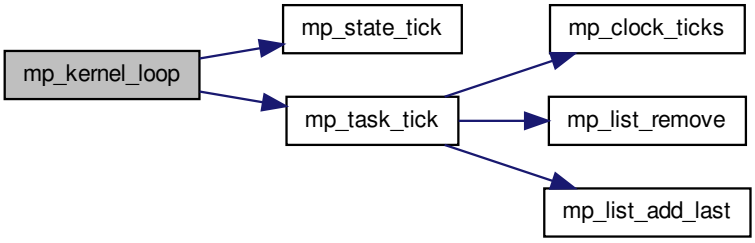


2.2.2.3 void mp_kernel_loop (mp_kernel_t * *kernel*)

Definition at line 118 of file kernel.c.

References mp_state_tick(), mp_task_tick(), MP_TASK_WORKING, mp_kernel_s::states, and mp_kernel_s::tasks.

Here is the call graph for this function:



2.2.2.4 void mp_kernel_state (mp_kernel_t * *kernel*, char *number*)

Definition at line 114 of file kernel.c.

References mp_state_switch(), and mp_kernel_s::states.

Referenced by mp_kernel_init().

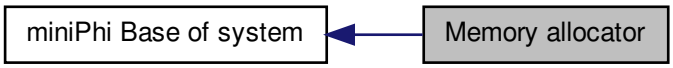
Here is the call graph for this function:

Here is the caller graph for this function:

2.3 Memory allocator

Tiny memory allocation system.

Collaboration diagram for Memory allocator:



Functions

- void * mp_mem_alloc (mp_kernel_t *kernel, int size)
- void mp_mem_free (mp_kernel_t *kernel, void *ptr)
- mp_ret_t mp_mem_erase (mp_kernel_t *kernel)

2.3.1 Detailed Description

Tiny memory allocation system.

Version

1.0.0

Author

2014 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

miniPhi common memory is a library to manage memory allocations. It is a tiny version of malloc() and then without "size hashing" algo. This allocation system doesn't consume too much energy.

Each block has the same size and the size of a block is defined in config.h by MP_MEM_CHUNK (set to 1024 by default)

The predefine memory space is set my the define MP_MEM_SIZE (set to 50 by default) in config.h

It is also possible to specify the section used for the linear memory if you wish (for example) to have an allocator in the Flash. If MP_COMMON_MEM_USE_MALLOC is defined then the whole memory will be allocated using malloc() and if it is not defined the whole memory will be allocated in the Flash.

You can overwrite those values by defining your own config.h file.

SUPPORT_COMMON_MEM must be defined to use this module

2.3.2 Function Documentation

2.3.2.1 void* mp_mem_alloc (mp_kernel_t * *kernel*, int *size*)

HEAP memory chunk allocation

Parameters

kernel	The kernel context
size	Size of chunk, used as informational

Returns

Point to a free space

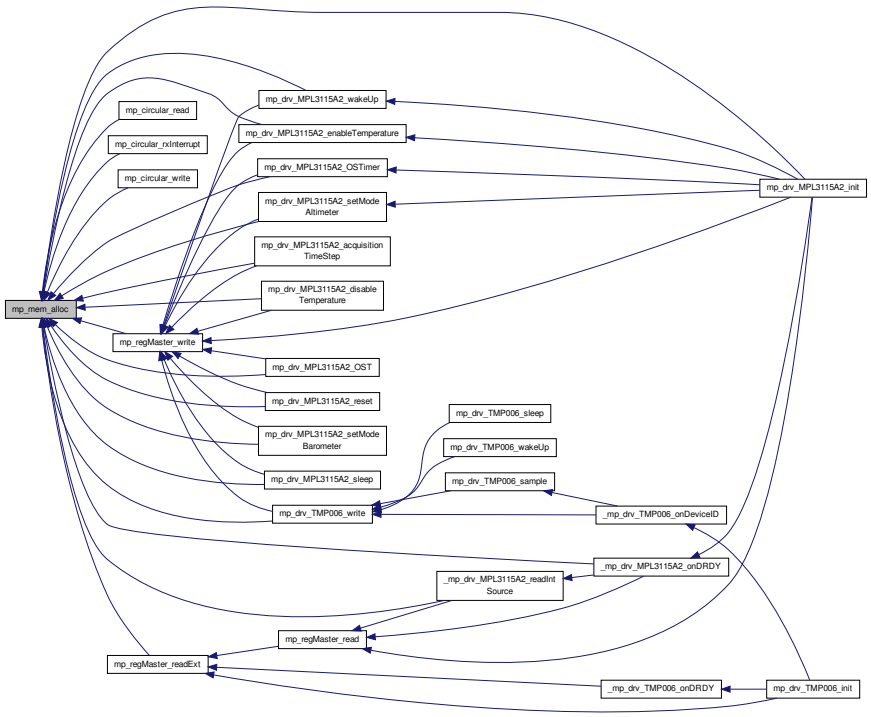
Definition at line 89 of file mem.c.

References __allocated_heap_idx, __line_size, KPANIC_MEM_OOM, KPANIC_MEM_SIZE, mp_kernel_panic(), MP_MEM_CHUNK, MP_MEM_SIZE, MP_MEM_SPACING, mp_printk, and NULL.

Referenced by _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_readIntSource(), mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_write(), mp_drv_MPL3115A2_acquisitionTimeStep(), mp_drv_MPL3115A2_disableTemperature(), mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_init(), mp_drv_MPL3115A2_OST(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_reset(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_setModeBarometer(), mp_drv_MPL3115A2_sleep(), mp_drv_MPL3115A2_wakeUp(), mp_drv_TMP006_write(), mp_regMaster_readExt(), and mp_regMaster_write().

Here is the call graph for this function:

Here is the caller graph for this function:



2.3.2.2 mp_ret_t mp_mem_erase (mp_kernel_t * *kernel*)

erase HEAP memory

Parameters

kernel	The kernel context

Returns

TRUE or FALSE

Definition at line 135 of file mem.c.

References __allocated_heap_idx, __allocated_heap_last, __line, _MP_HEAP_CKSIZE, FALSE, KPANIC_MEM_-OOM, KPANIC_MEM_SIZE, mp_kernel_panic(), MP_MEM_SIZE, mp_printk, NULL, and TRUE.

Referenced by mp_kernel_init().

Here is the call graph for this function:

Here is the caller graph for this function:

2.3.2.3 void mp_mem_free (mp_kernel_t * *kernel*, void * *ptr*)

free HEAP memory chunk

Parameters

kernel	The kernel context
ptr	pointer to free

Returns

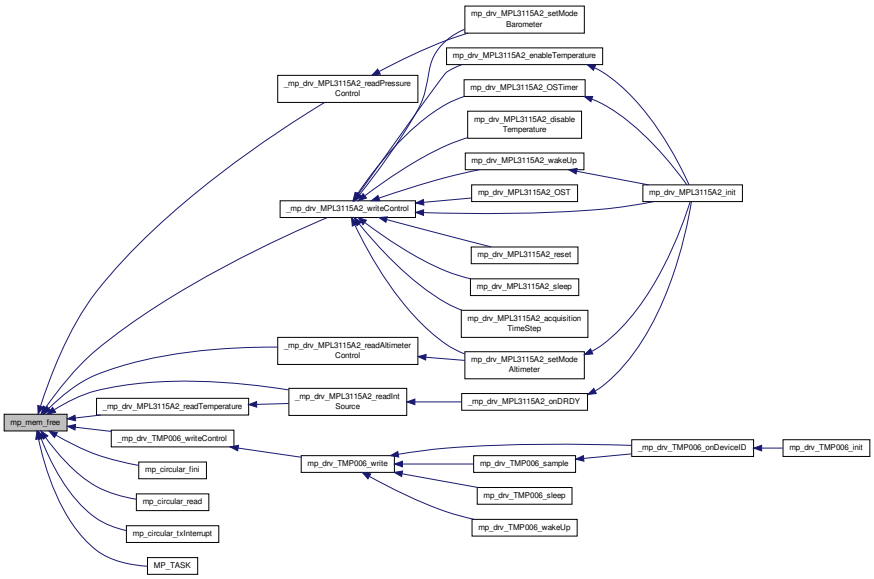
Nothing

Definition at line 120 of file mem.c.

References __allocated_heap_idx, and __allocated_heap_last.

Referenced by _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_readIntSource(), _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_MPL3115A2_readTemperature(), _mp_drv_MPL3115A2_writeControl(), _mp_drv_TMP006_writeControl(), mp_circular_fini(), mp_circular_read(), mp_circular_txInterrupt(), and MP_TASK().

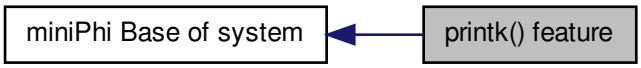
Here is the caller graph for this function:



2.4 printk() feature

Common kernel print.

Collaboration diagram for printk() feature:



Macros

- #define _DP(a,...)
- #define mp_printk(a, args...) mp_printk_call(mp_printk_user, a, ##args)

  miniPhi printk()

Typedefs

- typedef void(∗ mp_printk_call_t)(void ∗user, char ∗fmt,...)

Functions

- mp_ret_t mp_printk_set (mp_printk_call_t call, void ∗user)

  Set output callback for mp_printk()
- mp_ret_t mp_printk_unset ()

  Unset output callback.

2.4.1 Detailed Description

Common kernel print.

Version

1.0.0

Author

2014 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

This module reference functions used to setup mp_printk() output.

Example 1 : Defined printk callback and use circular buffering

```
static void _olimex_printk(void *user, char *fmt, ...) {
        olimex_msp430_t *olimex = user;
        unsigned char *buffer = malloc(256);
        va_list args;
        int size;

        va_start(args, fmt);
        size = vsnprintf((char *)buffer, 256-3, fmt, args);
        va_end(args);

        buffer[size++] = '\n';
        buffer[size++] = '\r';

        mp_serial_write(&olimex->serial, buffer, size);

        free(buffer);

        return;
}
```

Example 2 : initialize serial UART and set printk

```
// setup serial interface
{
        ret = mp_serial_initUART(&olimex->kernel, &olimex->serial, &olimex->proxyUARTDst, "Serial DST UART"
      );
        if(ret == FALSE)
                return;
}

// set printk
mp_printk_set(_olimex_printk, olimex);
```

2.4.2 Macro Definition Documentation

2.4.2.1 #define _DP(*a, ...*)

Definition at line 32 of file printk.h.

2.4.2.2 #define mp_printk(*a, args...*) mp_printk_call(mp_printk_user, a, ##args)

miniPhi printk()

This function is the general call to output messages.

Parameters

in	*a*	Format string
in	*args*	Format argument

Definition at line 43 of file printk.h.

Referenced by _mp_drv_MPL3115A2_onSettings(), _mp_drv_MPL3115A2_onWhoIAm(), _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_TMP006_onDeviceID(), _mp_drv_TMP006_onManufacturerID(), _mp_drv_TMP006_onSettings(), hexdump(), hexdumpf(), log_key(), mp_adc_create(), mp_adc_remove(), mp_drv_LSM9DS0_calibrate(), mp_drv_LSM9DS0_fini(), mp_drv_LSM9DS0_init(), mp_drv_MPL3115A2_fini(), mp_drv_MPL3115A2_init(), mp_drv_TMP006_fini(), mp_drv_TMP006_init(), mp_i2c_open(), mp_i2c_setup(), mp_mem_alloc(), mp_mem_erase(), mp_printk_hexdump(), mp_spi_open(), mp_spi_setup(), mp_uart_open(), print_bd_addr(), and print_UUID128().

2.4.3 Typedef Documentation

2.4.3.1 typedef void(* mp_printk_call_t)(void *user, char *fmt,...)

Definition at line 45 of file printk.h.

2.4.4 Function Documentation

2.4.4.1 mp_ret_t mp_printk_set (mp_printk_call_t *call*, void * *user*)

Set output callback for mp_printk()

This function sets the callback and the user pointer used at each call of mp_printk().

Parameters

in	*call*	Printk callback
in	*user*	Embedded user pointer

Definition at line 92 of file printk.c.

References mp_printk_call, mp_printk_user, and TRUE.

2.4.4.2 mp_ret_t mp_printk_unset ( )

Unset output callback.

This function clear the mp_printk() callback

Definition at line 103 of file printk.c.

References _dummy_printk(), mp_printk_call, and TRUE.

Referenced by mp_kernel_init().

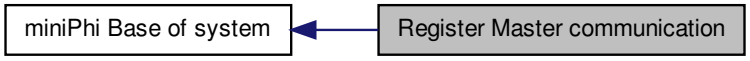
Here is the call graph for this function:

Here is the caller graph for this function:

2.5 Register Master communication

Circular register model for Master node.

Collaboration diagram for Register Master communication:



Data Structures

- struct mp_regMaster_op_s
- struct mp_regMaster_s

Typedefs

- typedef struct mp_regMaster_op_s mp_regMaster_op_t
- typedef struct mp_regMaster_s mp_regMaster_t
- typedef void(∗ mp_regMaster_cb_t)(mp_regMaster_op_t ∗operand, mp_bool_t terminate)
- typedef void(∗ mp_regMaster_int_t)(mp_regMaster_t ∗cirr)

Functions

- mp_ret_t mp_regMaster_init_i2c (mp_kernel_t ∗kernel, mp_regMaster_t ∗cirr, mp_i2c_t ∗i2c, void ∗user, char ∗who)

  Initiate circular register context.
- void mp_regMaster_fini (mp_regMaster_t ∗cirr)

  Terminate circular register context.
- mp_ret_t mp_regMaster_readExt (mp_regMaster_t ∗cirr, unsigned char ∗reg, int regSize, unsigned char ∗wait, int waitSize, mp_regMaster_cb_t callback, void ∗user, mp_bool_t swap)

  Extended circular register read operation.
- mp_ret_t mp_regMaster_write (mp_regMaster_t ∗cirr, unsigned char ∗reg, int regSize, mp_regMaster_cb_t callback, void ∗user)

  Start circular register write operation.
- unsigned char ∗ mp_regMaster_register (unsigned char reg)

  Get HEAP memory for register.
- static mp_ret_t mp_regMaster_read (mp_regMaster_t ∗cirr, unsigned char ∗reg, int regSize, unsigned char ∗wait, int waitSize, mp_regMaster_cb_t callback, void ∗user)

  Start circular register read operation.

2.5.1 Detailed Description

Circular register model for Master node.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

Examples

Driver structure

```
struct mp_drv_MPL3115A2_s {
        mp_kernel_t *kernel;

        mp_i2c_t i2c;
        mp_regMaster_t regMaster;

        // [...]
};
```

Before initializing regMaster you must setup the I2C or SPI interfaces. Here is an example using I2C

```
ret = mp_regMaster_init_i2c(kernel, &MPL3115A2->regMaster,
                &MPL3115A2->i2c, MPL3115A2, "MPL3115A2 I2C");
if(ret == FALSE) {
        mp_printk("MPL3115A2 error while creating regMaster context");
        mp_i2c_close(&MPL3115A2->i2c);
        return(NULL);
}
```

In regMaster a read operation comes with a write before to read. This is how register communication works. regMaster has it own queuing system which is splited in "operands". Each operand has a reg pointer and a wait pointer respectively are the register(s) to write and the data to receive. The allocation of those pointers are not managed by regMaster then you will have to control the buffer by yourself.

There are two ways to control it. Use a .bss pointer which doesn't need to be free as followed :

```
mp_regMaster_read(
        &MPL3115A2->regMaster,
        &_registers[MPL3115A2_WHO_AM_I], 1,
        (unsigned char *)&MPL3115A2->whoIam, 1,
        _mp_drv_MPL3115A2_onWhoIAm, MPL3115A2
);
```

Or use an allocated memory space in the HEAP to receive or emit information and then you will have to the end callback to free the buffer :

```
void _mp_drv_MPL3115A2_writeControl(
      mp_regMaster_op_t *operand, mp_bool_t terminate) {
        mp_drv_MPL3115A2_t *MPL3115A2 = operand->user;
        // free allocated register
        mp_mem_free(MPL3115A2->kernel, operand->reg);
}

// [...]

unsigned char *ptr = mp_mem_alloc(MPL3115A2->kernel, 2);
unsigned char *src = ptr;

// forge the registers to write
 (ptr++) = MPL3115A2_PT_DATA_CFG;
 ptr = 0x07;

mp_regMaster_write(
        &MPL3115A2->regMaster,
        src, 2,
        _mp_drv_MPL3115A2_writeControl, MPL3115A2
);
```

2.5.2 Typedef Documentation

2.5.2.1 typedef void(∗ mp_regMaster_cb_t)(mp_regMaster_op_t ∗operand, mp_bool_t terminate)

Definition at line 36 of file regMaster.h.

2.5.2.2 typedef void(∗ mp_regMaster_int_t)(mp_regMaster_t ∗cirr)

Definition at line 37 of file regMaster.h.

2.5.2.3 typedef struct mp_regMaster_op_s mp_regMaster_op_t

Definition at line 33 of file regMaster.h.

2.5.2.4 typedef struct mp_regMaster_s mp_regMaster_t

Definition at line 34 of file regMaster.h.

2.5.3 Function Documentation

2.5.3.1 void mp_regMaster_fini (mp_regMaster_t ∗ *cirr*)

Terminate circular register context.

This terminate a circular register context

Parameters

in	*cirr*	Circular context.

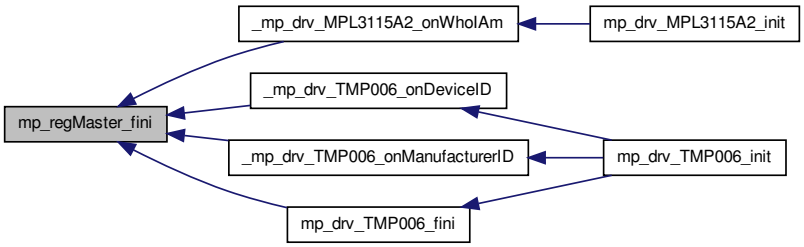
Definition at line 200 of file regMaster.c.

References mp_regMaster_s::asr, mp_regMaster_s::disableRX, mp_regMaster_s::disableTX, and mp_task_destroy().

Referenced by _mp_drv_MPL3115A2_onWhoIAm(), _mp_drv_TMP006_onDeviceID(), _mp_drv_TMP006_onManufacturerID(), and mp_drv_TMP006_fini().

Here is the call graph for this function:

Here is the caller graph for this function:



2.5.3.2 mp_ret_t mp_regMaster_init_i2c (mp_kernel_t * *kernel*, mp_regMaster_t * *cirr*, mp_i2c_t * *i2c*, void * *user*, char * *who*)

Initiate circular register context.

This initiates a circular register context

Parameters

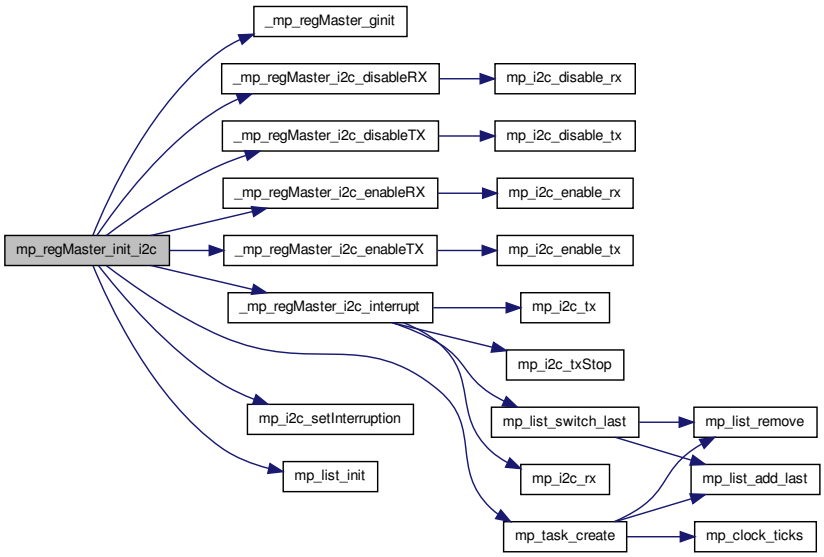
in	*kernel*	Kernel handler
in	*cirr*	Circular context.
in	*user*	User pointer embedded

Definition at line 153 of file regMaster.c.

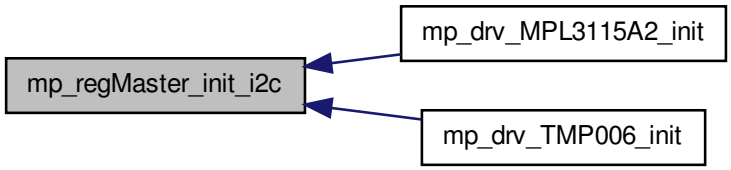
References _mp_regMaster_ginit(), _mp_regMaster_i2c_disableRX(), _mp_regMaster_i2c_disableTX(), _mp_regMaster_i2c_enableRX(), _mp_regMaster_i2c_enableTX(), _mp_regMaster_i2c_interrupt(), mp_regMaster_s::asr, mp_regMaster_s::disableRX, mp_regMaster_s::disableTX, mp_regMaster_s::enableRX, mp_regMaster_s::enableTX, mp_regMaster_s::executing, FALSE, mp_regMaster_s::i2c, mp_regMaster_s::kernel, mp_i2c_setInterruption(), mp_list_init(), mp_task_create(), MP_TASK_SIG_SLEEP, mp_regMaster_s::pending, mp_task_s::signal, mp_kernel_s::tasks, TRUE, mp_i2c_s::user, and mp_regMaster_s::user.

Referenced by mp_drv_MPL3115A2_init(), and mp_drv_TMP006_init().

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.3.3 static mp_ret_t mp_regMaster_read (mp_regMaster_t * *cirr*, unsigned char * *reg*, int *regSize*, unsigned char * *wait*, int *waitSize*, mp_regMaster_cb_t *callback*, void * *user*) [inline], [static]

Start circular register read operation.

This initiates a read operation using circular register.

When callback() is executed you must take care of the allocated pointer (if used). The terminate boolean argument is set to TRUE to notify whether regMaster has been shutdown. In this case you must stop actions and free buffers if needed.

Parameters

in	*cirr*	Circular context.

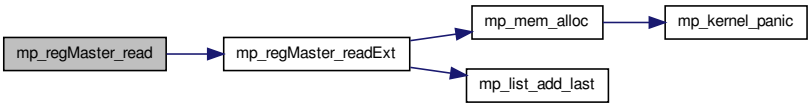
in	*reg*	Registers to write
in	*regSize*	Size of the registers to write
out	*wait*	Buffer to fill
in	*waitSize*	Number of bytes to read. wait allocation must be aligned with waitSize.
in	*callback*	Callback executed on the end of operation
in	*user*	User pointer embedded and passed as argument

Definition at line 123 of file regMaster.h.

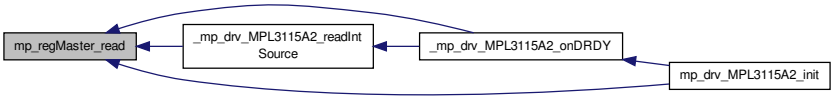
References FALSE, and mp_regMaster_readExt().

Referenced by _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_readIntSource(), and mp_drv_MPL3115A2_init().

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.3.4 mp_ret_t mp_regMaster_readExt (mp_regMaster_t * *cirr*, unsigned char * *reg*, int *regSize*, unsigned char * *wait*, int *waitSize*, mp_regMaster_cb_t *callback*, void * *user*, mp_bool_t *swap*)

Extended circular register read operation.

This initiates a read operation using circular register.

When callback() is executed you must take care of the allocated pointer (if used). The terminate boolean argument is set to TRUE to notify whether regMaster has been shutdown. In this case you must stop actions and free buffers if needed.

Parameters

in	*cirr*	Circular context.
in	*reg*	Registers to write
in	*regSize*	Size of the registers to write
out	*wait*	Buffer to fill
in	*waitSize*	Number of bytes to read. wait allocation must be aligned with waitSize.
in	*callback*	Callback executed on the end of operation

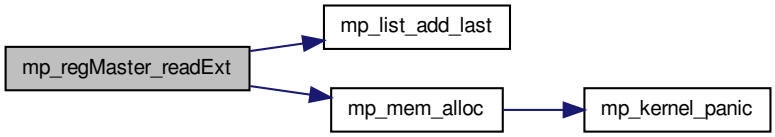
in	*user*	User pointer embedded and passed as argument
in	*swap*	set to TRUE to swap RX buffer

Definition at line 231 of file regMaster.c.

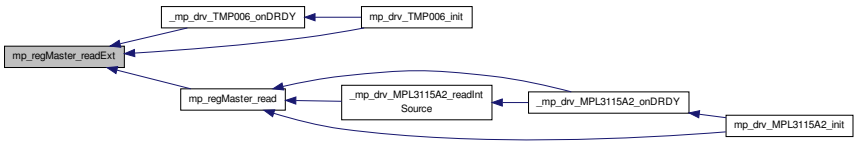
References mp_regMaster_s::asr, mp_regMaster_op_s::callback, mp_regMaster_op_s::item, mp_regMaster_s::kernel, mp_list_add_last(), mp_mem_alloc(), MP_REGMASTER_STATE_TX, MP_TASK_SIG_PENDING, MP_TASK_SIG_SLEEP, mp_regMaster_s::pending, mp_regMaster_op_s::reg, mp_regMaster_op_s::regPos, mp_regMaster_op_s::regSize, mp_task_s::signal, mp_regMaster_op_s::state, mp_regMaster_op_s::swap, TRUE, mp_regMaster_op_s::user, mp_regMaster_op_s::wait, mp_regMaster_op_s::waitPos, and mp_regMaster_op_s::waitSize.

Referenced by _mp_drv_TMP006_onDRDY(), mp_drv_TMP006_init(), and mp_regMaster_read().

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.3.5 unsigned char * mp_regMaster_register (unsigned char *reg*)

Get HEAP memory for register.

This function is very useful because it allows to get a valid HEAP pointer which contains the register to write. In this case you don't need to allocate or free register after its usage. Only limitation register is limited to 8bits.

Parameters

in	*reg*	Register content

Returns

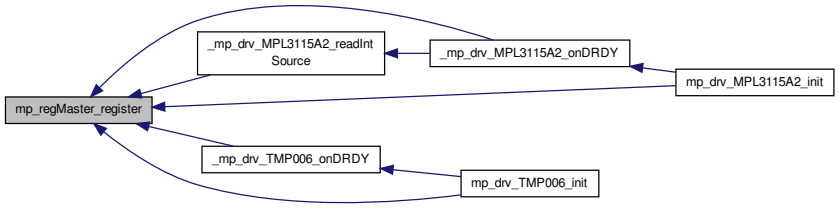
HEAP memory pointer contains the register

Definition at line 318 of file regMaster.c.

References _registers.

Referenced by _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_readIntSource(), _mp_drv_TMP006_onDRDY(), mp_drv_MPL3115A2_init(), and mp_drv_TMP006_init().

Here is the caller graph for this function:



2.5.3.6 mp_ret_t mp_regMaster_write (mp_regMaster_t * *cirr,* unsigned char * *reg,* int *regSize,* mp_regMaster_cb_t *callback,* void * *user*)

Start circular register write operation.

This initiates a write operation using circular register

Parameters

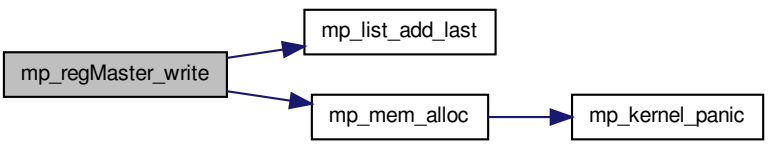
in	*cirr*	Circular context.
in	*reg*	Registers to write
in	*regSize*	Size of the registers to write
in	*callback*	Callback executed on the end of operation
in	*user*	User pointer embedded and passed as argument

Definition at line 280 of file regMaster.c.

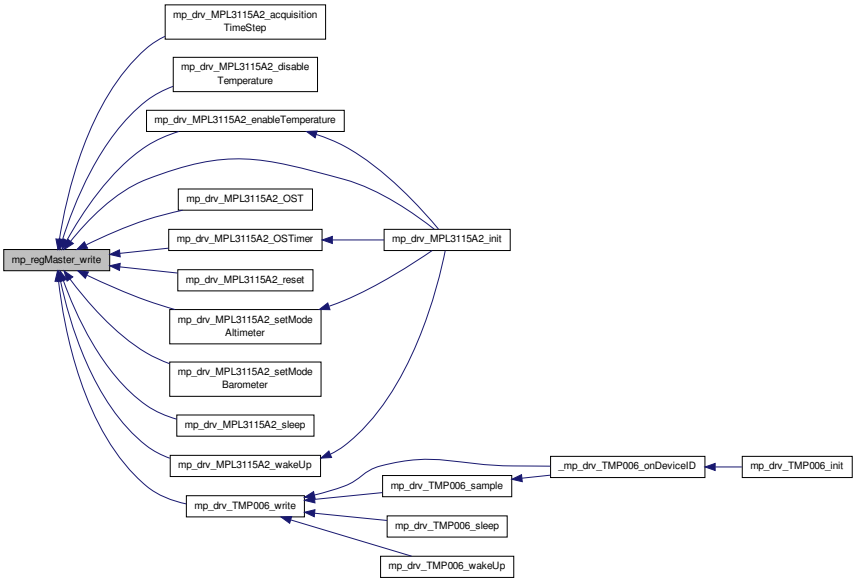
References mp_regMaster_s::asr, mp_regMaster_op_s::callback, mp_regMaster_op_s::item, mp_regMaster_s::kernel, mp_list_add_last(), mp_mem_alloc(), MP_REGMASTER_STATE_TX, MP_TASK_SIG_PENDING, MP_TASK_SIG_SLEEP, mp_regMaster_s::pending, mp_regMaster_op_s::reg, mp_regMaster_op_s::regPos, mp_regMaster_op_s::regSize, mp_task_s::signal, mp_regMaster_op_s::state, TRUE, and mp_regMaster_op_s::user.

Referenced by mp_drv_MPL3115A2_acquisitionTimeStep(), mp_drv_MPL3115A2_disableTemperature(), mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_init(), mp_drv_MPL3115A2_OST(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_reset(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_setModeBarometer(), mp_drv_MPL3115A2_sleep(), mp_drv_MPL3115A2_wakeUp(), and mp_drv_TMP006_write().

Here is the call graph for this function:



Here is the caller graph for this function:



2.6 Common state manager

States machine.

Collaboration diagram for Common state manager:

Data Structures

- struct mp_state_s
- struct mp_state_handler_s

Typedefs

- typedef struct mp_state_handler_s mp_state_handler_t
- typedef struct mp_state_s mp_state_t
- typedef void(∗ mp_state_callback_t)(void ∗user)

Functions

- void mp_state_init (mp_state_handler_t ∗hdl)

  Initialise state machine context.
- void mp_state_fini (mp_state_handler_t ∗hdl)

  Terminate state machine context.
- mp_ret_t mp_state_switch (mp_state_handler_t ∗hdl, char number)

  Change machine state.
- void mp_state_tick (mp_state_handler_t ∗hdl)

  General machine state tick interrupt.
- mp_ret_t mp_state_define (mp_state_handler_t ∗hdl, char number, char ∗name, void ∗user, mp_state_callback_t set, mp_state_callback_t unset, mp_state_callback_t tick)

  Define a machine state.

2.6.1 Detailed Description

States machine.

Version

1.0.0

Author

2014 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

Common states machine is a library used to manage differents states of a machine.

Each state has 3 callbacks which have to defined.

- mp_state_callback_t set : When the state is initialized
- mp_state_callback_t unset : When the state is terminated
- mp_state_callback_t tick : A state tick (corresponding to the loop)

miniPhi machine states is context free then you can use everywhere in your code.

A state machine context has a limited states defined by MP_STATE_MAX (set to 5 by default).

Example 1 shows how to initialize a machine state context

```
// initialize logical machine state
mp_state_init(&kernel->states);

// define KPANIC machine state
mp_state_define(
        &kernel->states,
        MP_KERNEL_KPANIC, "KPANIC", kernel,
        _mp_kernel_state_kpanic_set,
        _mp_kernel_state_kpanic_unset,
        _mp_kernel_state_kpanic_tick
);

// define BOOT machine state
mp_state_define(
        &kernel->states,
        MP_KERNEL_BOOT, "BOOT", kernel,
        _mp_kernel_state_boot_set,
        _mp_kernel_state_boot_unset,
        _mp_kernel_state_boot_tick
);
```

Example 2 shows how to switch machine state n1 :

```
mp_state_switch(&kernel->states, 1);
```

2.6.2 Typedef Documentation

2.6.2.1 typedef void(∗ mp_state_callback_t)(void ∗user)

Definition at line 12 of file state.h.

2.6.2.2 typedef struct mp_state_handler_s mp_state_handler_t

Definition at line 9 of file state.h.

2.6.2.3 typedef struct mp_state_s mp_state_t

Definition at line 10 of file state.h.

2.6.3 Function Documentation

2.6.3.1 mp_ret_t mp_state_define (mp_state_handler_t ∗ *hdl*, char *number*, char ∗ *name*, void ∗ *user*, mp_state_callback_t *set*, mp_state_callback_t *unset*, mp_state_callback_t *tick*)

Define a machine state.

Parameters

in	*hdl*	Context
in	*number*	Machine state number
in	*name*	Name of the state
in	*user*	Embedded user pointer
in	*set*	Setup callback
in	*unset*	Unset callback
in	*tick*	Tick callback

Definition at line 150 of file state.c.

References FALSE, MP_STATE_MAX, mp_state_s::name, mp_state_s::number, mp_state_s::set, mp_state_handler_s::states, mp_state_s::tick, TRUE, mp_state_s::unset, mp_state_s::used, mp_state_s::user, and YES.

Referenced by mp_kernel_init().

Here is the caller graph for this function:

2.6.3.2 void mp_state_fini (mp_state_handler_t * *hdl*)

Terminate state machine context.

Parameters

in	*hdl*	Context

Definition at line 96 of file state.c.

Referenced by mp_kernel_fini().

Here is the caller graph for this function:

2.6.3.3 void mp_state_init (mp_state_handler_t * *hdl*)

Initialise state machine context.

Parameters

in	*hdl*	Context

Definition at line 86 of file state.c.

References mp_state_handler_s::states.

Referenced by mp_kernel_init().

Here is the caller graph for this function:

2.6.3.4 mp_ret_t mp_state_switch (mp_state_handler_t * *hdl*, char *number*)

Change machine state.

Parameters

in	*hdl*	Context
in	*number*	Machine state number

Returns

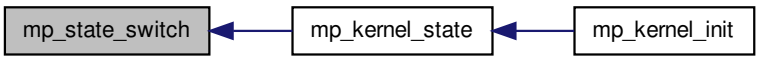
TRUE or FALSE

Definition at line 108 of file state.c.

References mp_state_handler_s::changeState, FALSE, MP_STATE_MAX, and TRUE.

Referenced by mp_kernel_state().

Here is the caller graph for this function:



2.6.3.5 void mp_state_tick (mp_state_handler_t * *hdl*)

General machine state tick interrupt.

Parameters

in	*hdl*	Context

Definition at line 119 of file state.c.

References mp_state_handler_s::changeState, mp_state_handler_s::currentState, mp_state_s::set, mp_state_handler_s::states, mp_state_s::tick, mp_state_s::unset, and mp_state_s::user.

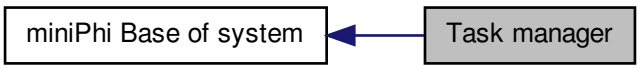
Referenced by mp_kernel_loop().

Here is the caller graph for this function:

2.7 Task manager

Create and control kernel task.

Collaboration diagram for Task manager:



Data Structures

- struct mp_task_s
- struct mp_task_handler_s

Typedefs

- typedef struct mp_task_handler_s mp_task_handler_t
- typedef struct mp_task_s mp_task_t
- typedef void(∗ mp_task_wakeup_t)(mp_task_t ∗task)

Enumerations

- enum mp_task_signal_t { MP_TASK_SIG_OK, MP_TASK_SIG_SLEEP, MP_TASK_SIG_PENDING, MP_TASK_SIG_STOP, MP_TASK_SIG_DEAD }
- enum mp_task_tick_t { MP_TASK_STOPPED, MP_TASK_WORKING, MP_TASK_DESTROYING }

Functions

- void mp_task_init (mp_task_handler_t ∗hdl)
- void mp_task_fini (mp_task_handler_t ∗hdl)
- void mp_task_flush (mp_task_handler_t ∗hdl)
- mp_task_t ∗ mp_task_create (mp_task_handler_t ∗hdl, void ∗name, mp_task_wakeup_t wakeup, void ∗user, unsigned long delay)
- mp_ret_t mp_task_destroy (mp_task_t ∗task)
- mp_task_tick_t mp_task_tick (mp_task_handler_t ∗hdl)

2.7.1 Detailed Description

Create and control kernel task.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

2.7.2 Typedef Documentation

2.7.2.1 typedef struct mp_task_handler_s mp_task_handler_t

Definition at line 29 of file task.h.

2.7.2.2 typedef struct mp_task_s mp_task_t

Definition at line 30 of file task.h.

2.7.2.3 typedef void(∗ mp_task_wakeup_t)(mp_task_t ∗task)

Definition at line 46 of file task.h.

2.7.3 Enumeration Type Documentation

2.7.3.1 enum mp_task_signal_t

Enumerator

MP_TASK_SIG_OK

MP_TASK_SIG_SLEEP

MP_TASK_SIG_PENDING

MP_TASK_SIG_STOP

MP_TASK_SIG_DEAD

Definition at line 32 of file task.h.

2.7.3.2 enum mp_task_tick_t

Enumerator

MP_TASK_STOPPED

MP_TASK_WORKING

MP_TASK_DESTROYING

Definition at line 40 of file task.h.

2.7.4 Function Documentation

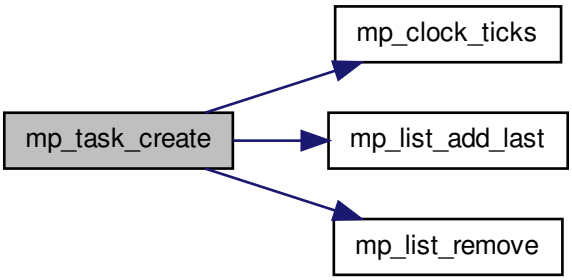
2.7.4.1 mp_task_t∗ mp_task_create (mp_task_handler_t ∗ *hdl*, void ∗ *name*, mp_task_wakeup_t *wakeup*, void ∗ *user*, unsigned long *delay*)

Definition at line 91 of file task.c.

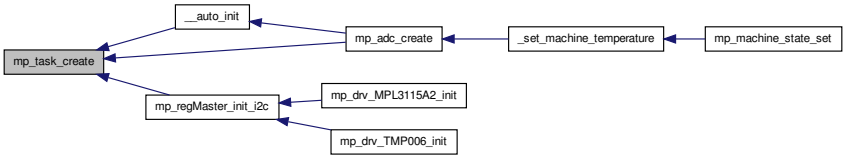
References mp_task_s::check, mp_task_s::delay, mp_list_s::first, mp_task_handler_s::freeList, mp_task_s::item, mp_list_s::last, mp_clock_ticks(), mp_list_add_last(), mp_list_remove(), MP_TASK_SIG_OK, mp_task_s::name, NULL, mp_task_s::signal, mp_task_handler_s::usedList, mp_task_handler_s::usedNumber, mp_list_item_s::user, mp_task_s::user, and mp_task_s::wakeup.

Referenced by __auto_init(), mp_adc_create(), and mp_regMaster_init_i2c().

Here is the call graph for this function:



Here is the caller graph for this function:



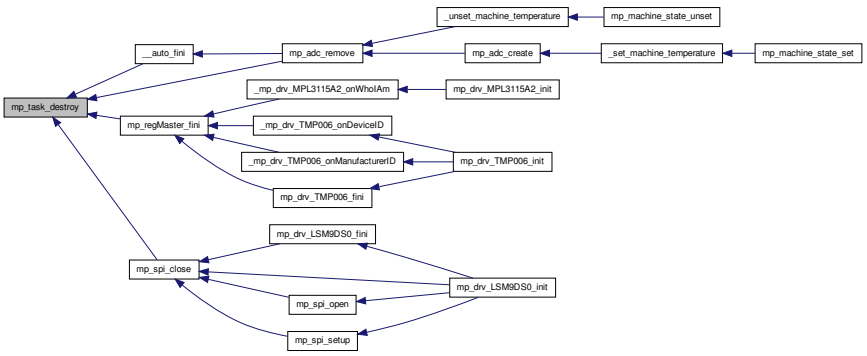
2.7.4.2 mp_ret_t mp_task_destroy (mp_task_t * *task*)

Definition at line 121 of file task.c.

References MP_TASK_SIG_STOP, mp_task_s::signal, and TRUE.

Referenced by __auto_fini(), mp_adc_remove(), mp_regMaster_fini(), and mp_spi_close().

Here is the caller graph for this function:



2.7.4.3 void mp_task_fini (mp_task_handler_t * *hdl*)

Definition at line 58 of file task.c.

References mp_list_s::first, mp_task_flush(), MP_TASK_SIG_STOP, NULL, mp_task_handler_s::signal, and mp_task_handler_s::usedList.

Here is the call graph for this function:

2.7.4.4 void mp_task_flush (mp_task_handler_t * *hdl*)

Definition at line 70 of file task.c.

References mp_list_s::first, mp_task_s::item, MP_TASK_SIG_STOP, mp_list_item_s::next, NULL, mp_task_s::signal, mp_task_handler_s::usedList, and mp_list_item_s::user.

Referenced by mp_task_fini().

Here is the caller graph for this function:

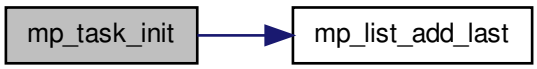
2.7.4.5 void mp_task_init (mp_task_handler_t * *hdl*)

Definition at line 41 of file task.c.

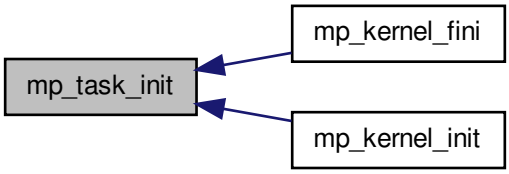
References mp_task_handler_s::freeList, mp_task_s::item, mp_list_add_last(), MP_TASK_MAX, MP_TASK_SIG_OK, mp_task_handler_s::signal, and mp_task_handler_s::tasks.

Referenced by mp_kernel_fini(), and mp_kernel_init().

Here is the call graph for this function:



Here is the caller graph for this function:



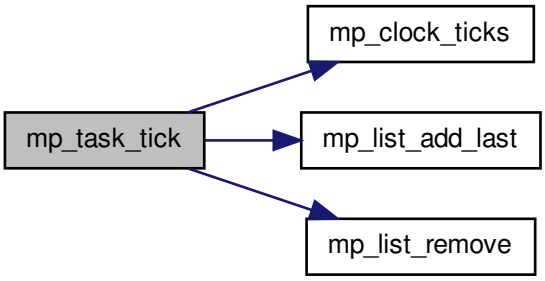
2.7.4.6 mp_task_tick_t mp_task_tick (mp_task_handler_t * *hdl*)

Definition at line 126 of file task.c.

References mp_task_s::check, mp_task_s::delay, mp_task_handler_s::freeList, mp_task_s::item, mp_list_s::last, mp_clock_ticks(), mp_list_add_last(), mp_list_remove(), MP_TASK_DESTROYING, MP_TASK_SIG_DEAD, MP_TASK_SIG_OK, MP_TASK_SIG_PENDING, MP_TASK_SIG_SLEEP, MP_TASK_SIG_STOP, MP_TASK_STOPPED, MP_TASK_WORKING, NO, NULL, mp_list_item_s::prev, mp_task_s::signal, mp_task_handler_s::signal, mp_task_handler_s::usedList, mp_task_handler_s::usedNumber, mp_list_item_s::user, mp_task_s::wakeup, and YES.

Referenced by mp_kernel_loop().

Here is the call graph for this function:



Here is the caller graph for this function:

2.8 ST LSM9DS0

ST LSM9DS0 3-axis Accelerometer / Gyroscope / Magneto.

Collaboration diagram for ST LSM9DS0:

Functions

- mp_ret_t mp_drv_LSM9DS0_init (mp_kernel_t ∗kernel, mp_drv_LSM9DS0_t ∗LSM9DS0, mp_options_t ∗options, char ∗who)
- mp_ret_t mp_drv_LSM9DS0_fini (mp_drv_LSM9DS0_t ∗LSM9DS0)

2.8.1 Detailed Description

ST LSM9DS0 3-axis Accelerometer / Gyroscope / Magneto.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

2.8.2 Function Documentation

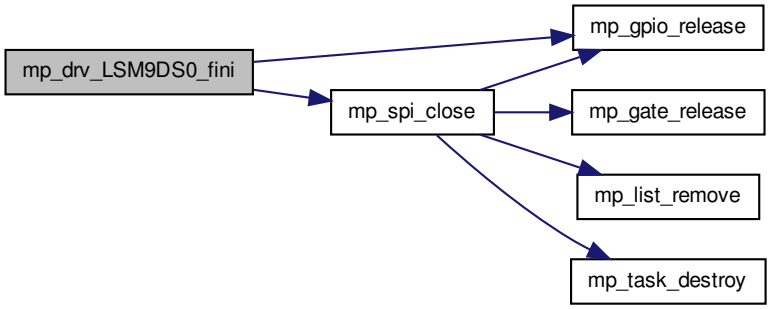
2.8.2.1 mp_ret_t mp_drv_LSM9DS0_fini (mp_drv_LSM9DS0_t ∗ *LSM9DS0*)

Definition at line 201 of file LSM9DS0.c.

References mp_drv_LSM9DS0_s::csG, mp_drv_LSM9DS0_s::csXM, mp_drv_LSM9DS0_s::drdy, mp_drv_LSM9DS0_s::int1, mp_drv_LSM9DS0_s::int2, mp_drv_LSM9DS0_s::intG, mp_gpio_release(), mp_printk, mp_spi_close(), mp_drv_LSM9DS0_s::spi, and TRUE.

Referenced by mp_drv_LSM9DS0_init().

Here is the call graph for this function:



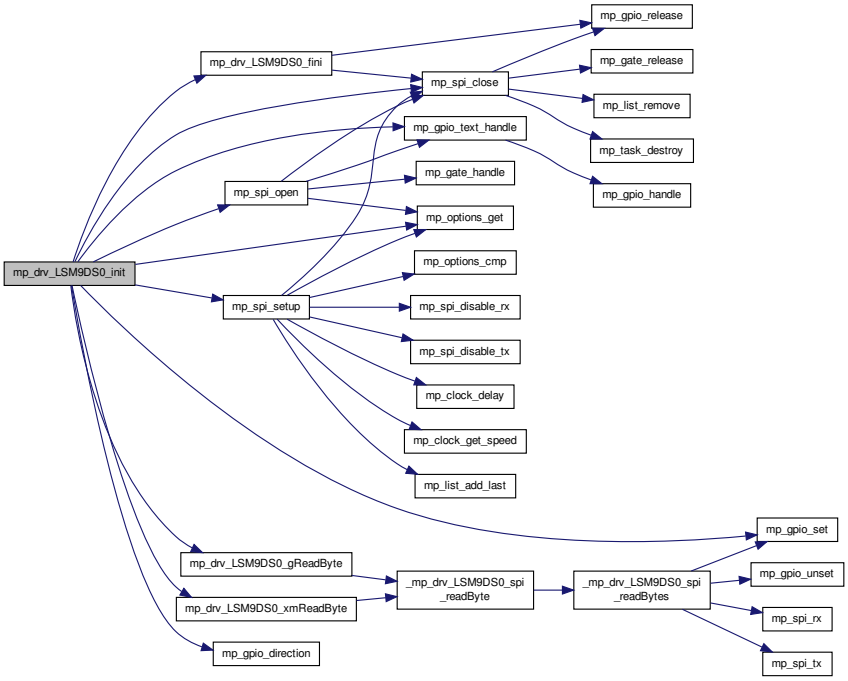
Here is the caller graph for this function:

2.8.2.2 mp_ret_t mp_drv_LSM9DS0_init (mp_kernel_t * *kernel*, mp_drv_LSM9DS0_t * *LSM9DS0*, mp_options_t * *options*, char * *who*)

Definition at line 62 of file LSM9DS0.c.

References mp_drv_LSM9DS0_s::csG, mp_drv_LSM9DS0_s::csXM, mp_drv_LSM9DS0_s::drdy, FALSE, mp_drv_LSM9DS0_s::int1, mp_drv_LSM9DS0_s::int2, mp_drv_LSM9DS0_s::intG, mp_drv_LSM9DS0_s::kernel, mp_drv_LSM9DS0_fini(), mp_drv_LSM9DS0_gReadByte(), mp_drv_LSM9DS0_xmReadByte(), mp_gpio_direction(), MP_GPIO_INPUT, MP_GPIO_OUTPUT, mp_gpio_set(), mp_gpio_text_handle(), mp_options_get(), mp_printk, mp_spi_close(), mp_spi_open(), mp_spi_setup(), NULL, mp_drv_LSM9DS0_s::spi, TRUE, WHO_AM_I_G, and WHO_AM_I_XM.

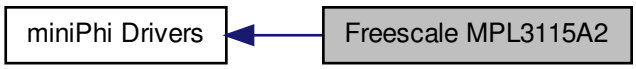
Here is the call graph for this function:



2.9 Freescale MPL3115A2

Freescale MPL3115A2 Altimeter / Barometer / Temperature.

Collaboration diagram for Freescale MPL3115A2:



Data Structures

- struct mp_drv_MPL3115A2_s

Macros

- #define MPL3115A2_ADDRESS 0x60

Typedefs

- typedef struct mp_drv_MPL3115A2_s mp_drv_MPL3115A2_t

Enumerations

- enum mp_drv_MPL3115A_OS_t {
  MPL3115A_6MS, MPL3115A_10MS, MPL3115A_18MS, MPL3115A_34MS,
  MPL3115A_66MS, MPL3115A_130MS, MPL3115A_258MS, MPL3115A_512MS }

Functions

- mp_ret_t mp_drv_MPL3115A2_init (mp_kernel_t *kernel, mp_drv_MPL3115A2_t *MPL3115A2, mp_options_t *options, char *who)
- void mp_drv_MPL3115A2_fini (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_sleep (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_wakeUp (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_reset (mp_drv_MPL3115A2_t *MPL3115A2)
- mp_ret_t mp_drv_MPL3115A2_acquisitionTimeStep (mp_drv_MPL3115A2_t *MPL3115A2, unsigned char st)
- void mp_drv_MPL3115A2_setModeBarometer (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_setModeAltimeter (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_setSeaLevel (mp_drv_MPL3115A2_t *MPL3115A2, int Pa)
- void mp_drv_MPL3115A2_OST (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_OSTimer (mp_drv_MPL3115A2_t *MPL3115A2, mp_drv_MPL3115A_OS_t timer)
- void mp_drv_MPL3115A2_enableTemperature (mp_drv_MPL3115A2_t *MPL3115A2)
- void mp_drv_MPL3115A2_disableTemperature (mp_drv_MPL3115A2_t *MPL3115A2)

2.9.1 Detailed Description

Freescale MPL3115A2 Altimeter / Barometer / Temperature.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

Informations :

- **INT1** is configured by design for DRDY interrupt.
- **DRDY** must be connected to drive the read of datas.

Breakouts available on :

- https://www.sparkfun.com/products/11084
- http://www.adafruit.com/product/1893
- http://www.artekit.eu/products/breakout-boards/ak-mpl3115a2-altimeter-pressure-sensc

Configuration for MPL3115A2 example :

- gate = USCI_B3 // msp430 based
- SDA = 10.1 / ext 1-17
- SCL = 10.2 / ext 1-16
- DRDY = 1.1 / ext 2-5 (to INT1)

Initializing the driver :

```
typedef struct olimex_msp430_s olimex_msp430_t;

struct olimex_msp430_s {
        mp_kernel_t kernel;

        mp_drv_MPL3115A2_t bat;
};

// [...]
{
        mp_options_t options[] = {
                { "gate", "USCI_B3" },
                { "sda", "p10.1" },
                { "clk", "p10.2" },
                { "drdy", "p1.1" },
                { NULL, NULL }
        };

        mp_drv_MPL3115A2_init(&olimex->kernel, &olimex->bat, options, "Freescale
       MPL3115A2");
}
```

2.9.2 Macro Definition Documentation

2.9.2.1 #define MPL3115A2_ADDRESS 0x60

Slave address

Definition at line 68 of file MPL3115A2.h.

Referenced by mp_drv_MPL3115A2_init().

2.9.3 Typedef Documentation

2.9.3.1 typedef struct mp_drv_MPL3115A2_s mp_drv_MPL3115A2_t

Definition at line 33 of file MPL3115A2.h.

2.9.4 Enumeration Type Documentation

2.9.4.1 enum mp_drv_MPL3115A_OS_t

Enumerator

MPL3115A_6MS

MPL3115A_10MS

MPL3115A_18MS

MPL3115A_34MS

MPL3115A_66MS

MPL3115A_130MS

MPL3115A_258MS

MPL3115A_512MS

Definition at line 56 of file MPL3115A2.h.

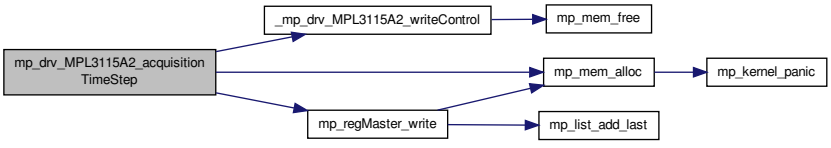
2.9.5 Function Documentation

2.9.5.1 mp_ret_t mp_drv_MPL3115A2_acquisitionTimeStep (mp_drv_MPL3115A2_t * *MPL3115A2,* unsigned char *st*)

Definition at line 278 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_writeControl(), FALSE, mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG2, mp_drv_MPL3115A2_s::regMaster, and TRUE.

Here is the call graph for this function:

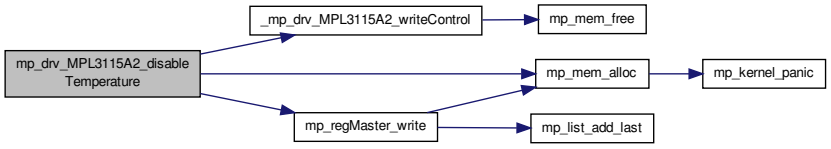


2.9.5.2 void mp_drv_MPL3115A2_disableTemperature (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 450 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG4, MPL3115A2_CTRL_REG5, mp_drv_MPL3115A2_s::regMaster, mp_drv_MPL3115A2_s::sensor, and mp_drv_MPL3115A2_s::temperature.

Here is the call graph for this function:



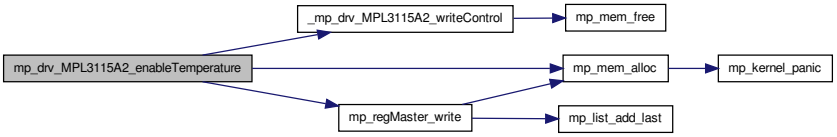
2.9.5.3 void mp_drv_MPL3115A2_enableTemperature (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 413 of file MPL3115A2.c.

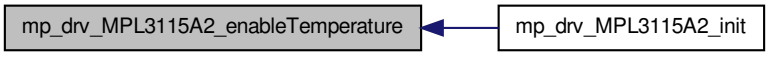
References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG4, MPL3115A2_CTRL_REG5, mp_drv_MPL3115A2_s::regMaster, mp_drv_MPL3115A2_s::sensor, and mp_drv_MPL3115A2_s::temperature.

Referenced by mp_drv_MPL3115A2_init().

Here is the call graph for this function:



Here is the caller graph for this function:



2.9.5.4 void mp_drv_MPL3115A2_fini (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 226 of file MPL3115A2.c.

References mp_printk.

Referenced by mp_drv_MPL3115A2_init().

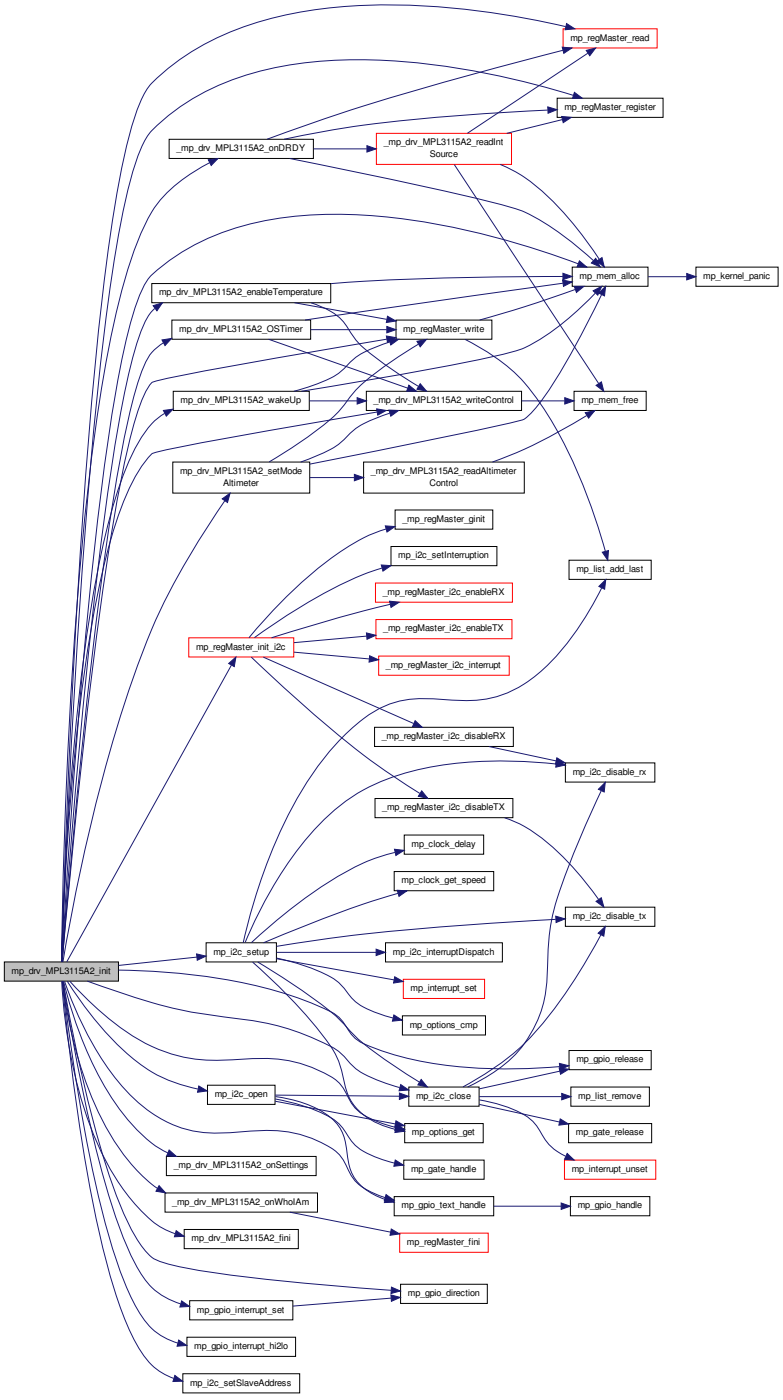
Here is the caller graph for this function:

2.9.5.5 mp_ret_t mp_drv_MPL3115A2_init (mp_kernel_t * *kernel,* mp_drv_MPL3115A2_t * *MPL3115A2,* mp_options_t * *options,* char * *who*)

Definition at line 96 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_onSettings(), _mp_drv_MPL3115A2_onWhoIAm(), _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::drdy, FALSE, mp_drv_MPL3115A2_s::i2c, mp_drv_MPL3115A2_s::kernel, mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_fini(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_wakeUp(), mp_gpio_direction(), MP_GPIO_INPUT, mp_gpio_interrupt_hi2lo(), mp_gpio_interrupt_set(), mp_gpio_release(), mp_gpio_text_handle(), mp_i2c_close(), mp_i2c_open(), mp_i2c_setSlaveAddress(), mp_i2c_setup(), mp_mem_alloc(), mp_options_get(), mp_printk, mp_regMaster_init_i2c(), mp_regMaster_read(), mp_regMaster_register(), mp_regMaster_write(), MPL3115A2_ADDRESS, MPL3115A2_CTRL_REG1, MPL3115A2_PT_DATA_CFG, MPL3115A2_WHO_AM_I, MPL3115A_512MS, NULL, mp_drv_MPL3115A2_s::regMaster, mp_drv_MPL3115A2_s::settings, TRUE, and mp_drv_MPL3115A2_s::whoIam.

Here is the call graph for this function:

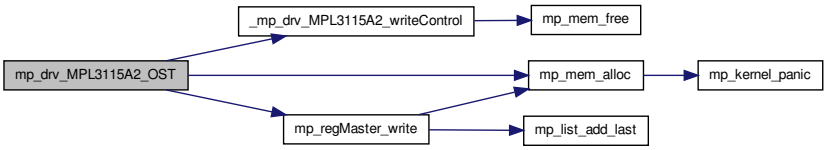


2.9.5.6 void mp_drv_MPL3115A2_OST (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 361 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::regMaster, and mp_drv_MPL3115A2_s::settings.

Here is the call graph for this function:



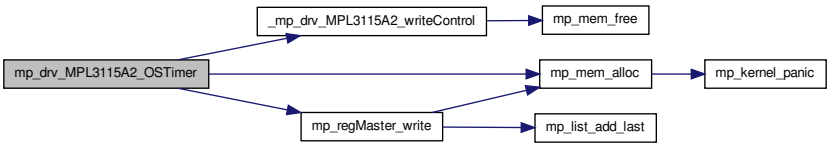
2.9.5.7 void mp_drv_MPL3115A2_OSTimer (mp_drv_MPL3115A2_t * *MPL3115A2,* mp_drv_MPL3115A_OS_t *timer*)

Definition at line 395 of file MPL3115A2.c.

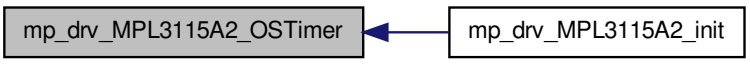
References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::regMaster, and mp_drv_MPL3115A2_s::settings.

Referenced by mp_drv_MPL3115A2_init().

Here is the call graph for this function:



Here is the caller graph for this function:

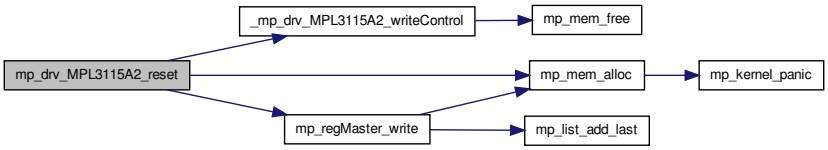


2.9.5.8 void mp_drv_MPL3115A2_reset (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 263 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::regMaster, and mp_drv_MPL3115A2_s::settings.

Here is the call graph for this function:



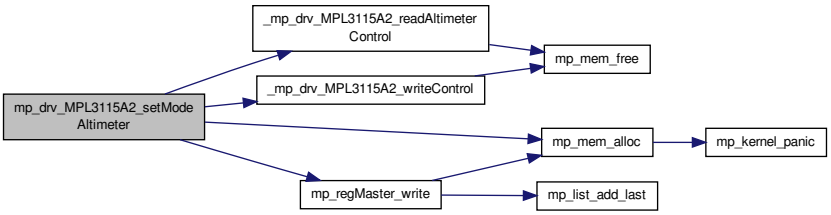
2.9.5.9 void mp_drv_MPL3115A2_setModeAltimeter (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 323 of file MPL3115A2.c.

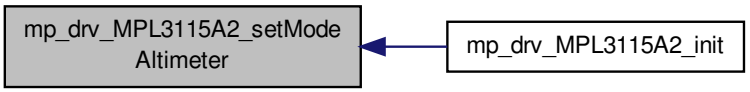
References _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::readerControl, mp_drv_MPL3115A2_s::regMaster, mp_drv_MPL3115A2_s::sensor, and mp_drv_MPL3115A2_s::settings.

Referenced by mp_drv_MPL3115A2_init().

Here is the call graph for this function:



Here is the caller graph for this function:

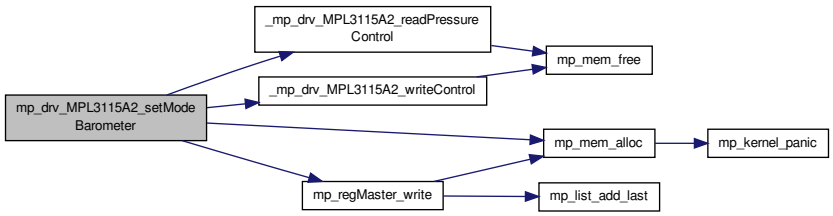


2.9.5.10 void mp_drv_MPL3115A2_setModeBarometer (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 299 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::readerControl, mp_drv_MPL3115A2_s::regMaster, mp_drv_MPL3115A2_s::sensor, and mp_drv_MPL3115A2_s::settings.

Here is the call graph for this function:



2.9.5.11 void mp_drv_MPL3115A2_setSeaLevel (mp_drv_MPL3115A2_t * *MPL3115A2,* int *Pa*)

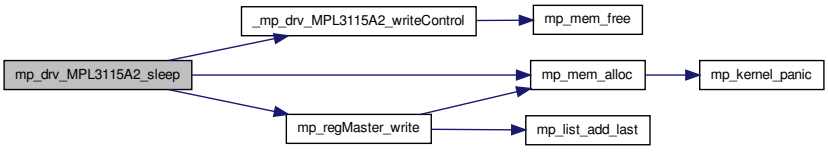
Definition at line 347 of file MPL3115A2.c.

2.9.5.12 void mp_drv_MPL3115A2_sleep (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 231 of file MPL3115A2.c.

References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::regMaster, and mp_drv_MPL3115A2_s::settings.

Here is the call graph for this function:



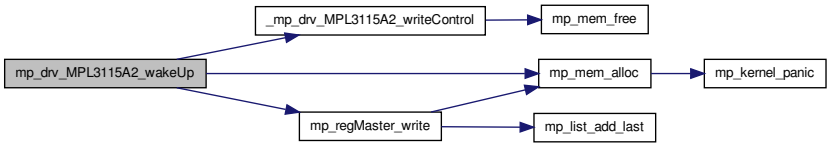
2.9.5.13 void mp_drv_MPL3115A2_wakeUp (mp_drv_MPL3115A2_t * *MPL3115A2*)

Definition at line 248 of file MPL3115A2.c.

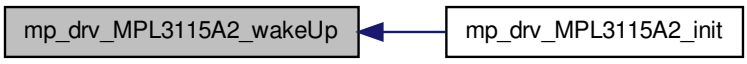
References _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_s::kernel, mp_mem_alloc(), mp_regMaster_write(), MPL3115A2_CTRL_REG1, mp_drv_MPL3115A2_s::regMaster, and mp_drv_MPL3115A2_s::settings.

Referenced by mp_drv_MPL3115A2_init().

Here is the call graph for this function:



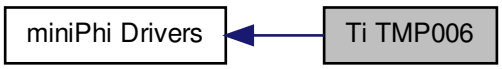
Here is the caller graph for this function:



2.10 Ti TMP006

Ti TMP006 Object temperature.

Collaboration diagram for Ti TMP006:



Data Structures

- struct mp_drv_TMP006_s

Typedefs

- typedef struct mp_drv_TMP006_s mp_drv_TMP006_t

Enumerations

- enum mp_drv_TMP006_sample_t {
  TMP006_CFG_1SAMPLE = 0x0000, TMP006_CFG_2SAMPLE = 0x0200, TMP006_CFG_4SAMPLE = 0x0400, TMP006_CFG_8SAMPLE = 0x0600,
  TMP006_CFG_16SAMPLE = 0x0800 }

Functions

- mp_sensor_t * mp_drv_TMP006_init (mp_kernel_t *kernel, mp_drv_TMP006_t *TMP006, mp_options_t *options, char *who)
- void mp_drv_TMP006_fini (mp_drv_TMP006_t *TMP006)
- void mp_drv_TMP006_sleep (mp_drv_TMP006_t *TMP006)

  Power down TMP006.
- void mp_drv_TMP006_wakeUp (mp_drv_TMP006_t *TMP006)

  Power up TMP006.
- void mp_drv_TMP006_sample (mp_drv_TMP006_t *TMP006, mp_drv_TMP006_sample_t sample)

  Change sample rate for TMP006.
- static void mp_drv_TMP006_setB0 (mp_drv_TMP006_t *TMP006, float value)

  Change B0 calculation register.
- static void mp_drv_TMP006_setB1 (mp_drv_TMP006_t *TMP006, float value)

  Change B1 calculation register.
- static void mp_drv_TMP006_setB2 (mp_drv_TMP006_t *TMP006, float value)

  Change B2 calculation register.
- static void mp_drv_TMP006_setS0 (mp_drv_TMP006_t *TMP006, float value)

  Change S0 calculation register.

TMP006 Constants

- #define TMP006_B0 -0.0000294
- #define TMP006_B1 -0.00000057
- #define TMP006_B2 0.00000000463
- #define TMP006_C2 13.4
- #define TMP006_TREF 298.15
- #define TMP006_A2 -0.00001678
- #define TMP006_A1 0.00175
- #define TMP006_S0 6.4

2.10.1 Detailed Description

Ti TMP006 Object temperature.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

Configuration for TMP006 example :

- gate = USCI_B3
- SDA = 10.1 / ext 1-17
- SCL = 10.2 / ext 1-16
- DRDY = 1.1 / ext 2-5

Initializing the driver :

```
typedef struct olimex_msp430_s olimex_msp430_t;

struct olimex_msp430_s {
        mp_kernel_t kernel;

        mp_drv_TMP006_t tmp006;

};

// [...]
{
        mp_options_t options[] = {
                { "gate", "USCI_B3" },
                { "sda", "p10.1" },
                { "clk", "p10.2" },
                { "drdy", "p1.1" },
                { NULL, NULL }
        };

        mp_drv_TMP006_init(&olimex->kernel, &olimex->tmp006, options, "Ti TMP006");
}
```

In order to understand the role of the FOV using TMP006 i recommand to watch the cool video from Adafruit/Ti on

- https://learn.adafruit.com/infrared-thermopile-sensor-breakout/using-the-thermopile-

2.10.2 Macro Definition Documentation

2.10.2.1 #define TMP006_A1 0.00175

Definition at line 141 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage().

2.10.2.2 #define TMP006_A2 -0.00001678

Definition at line 140 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage().

2.10.2.3 #define TMP006_B0 -0.0000294

Definition at line 135 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

2.10.2.4 #define TMP006_B1 -0.00000057

Definition at line 136 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

2.10.2.5 #define TMP006_B2 0.00000000463

Definition at line 137 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

2.10.2.6 #define TMP006_C2 13.4

Definition at line 138 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage().

2.10.2.7 #define TMP006_S0 6.4

Definition at line 142 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

2.10.2.8 #define TMP006_TREF 298.15

Definition at line 139 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage().

2.10.3 Typedef Documentation

2.10.3.1 typedef struct mp_drv_TMP006_s mp_drv_TMP006_t

Definition at line 33 of file TMP006.h.

2.10.4 Enumeration Type Documentation

2.10.4.1 enum mp_drv_TMP006_sample_t

Enumerator

TMP006_CFG_1SAMPLE
TMP006_CFG_2SAMPLE
TMP006_CFG_4SAMPLE
TMP006_CFG_8SAMPLE
TMP006_CFG_16SAMPLE

Definition at line 63 of file TMP006.h.

2.10.5 Function Documentation

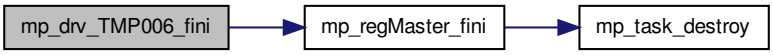
2.10.5.1 void mp_drv_TMP006_fini (mp_drv_TMP006_t * *TMP006*)

Definition at line 198 of file TMP006.c.

References mp_printk, mp_regMaster_fini(), and mp_drv_TMP006_s::regMaster.

Referenced by mp_drv_TMP006_init().

Here is the call graph for this function:



Here is the caller graph for this function:

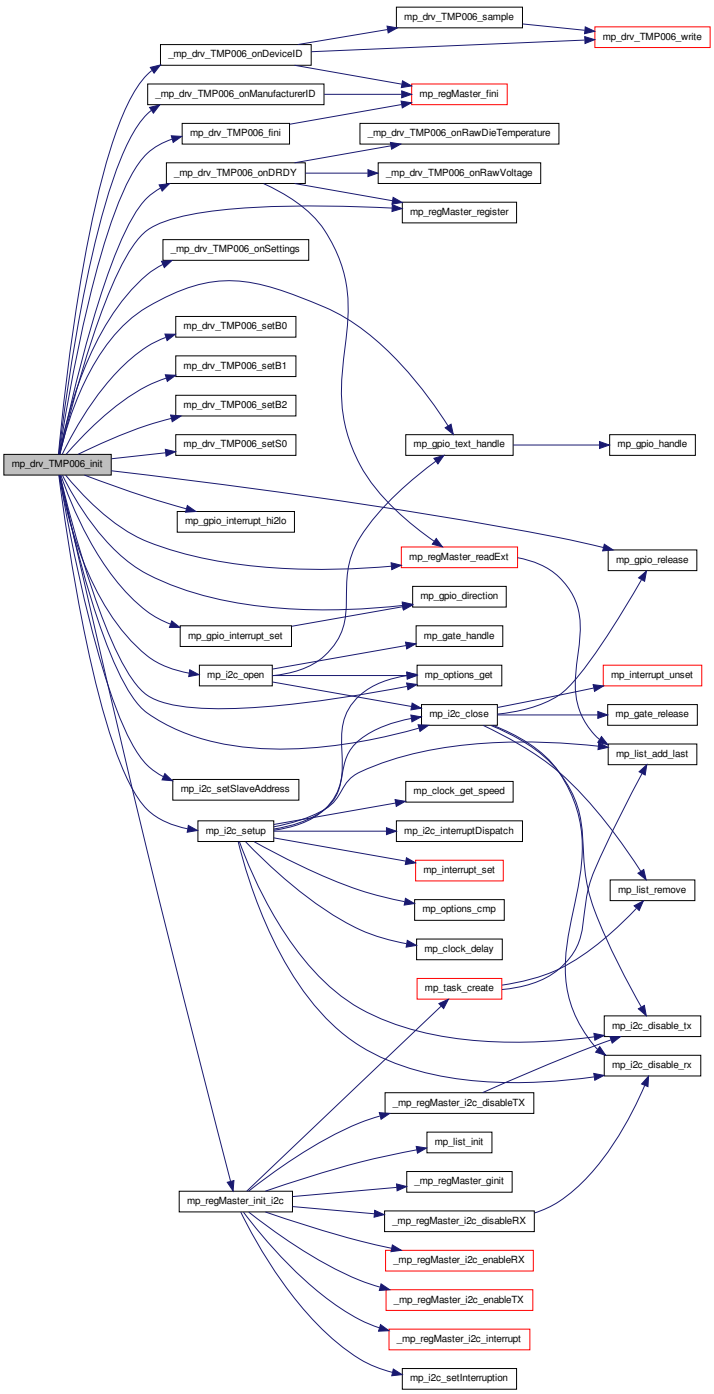
2.10.5.2 mp_sensor_t* mp_drv_TMP006_init (mp_kernel_t * *kernel,* mp_drv_TMP006_t * *TMP006,* mp_options_t * *options,* char * *who*)

Definition at line 89 of file TMP006.c.

References _mp_drv_TMP006_onDeviceID(), _mp_drv_TMP006_onDRDY(), _mp_drv_TMP006_onManufacturerID(), _mp_drv_TMP006_onSettings(), mp_drv_TMP006_s::deviceId, mp_drv_TMP006_s::drdy, FALSE, mp_drv_TMP006_s::i2c, mp_drv_TMP006_s::kernel, mp_drv_TMP006_s::manufacturerId, mp_drv_TMP006_fini(), mp_drv_TMP006_setB0(), mp_drv_TMP006_setB1(), mp_drv_TMP006_setB2(), mp_drv_TMP006_setS0(), mp_gpio_direction(), MP_GPIO_INPUT, mp_gpio_interrupt_hi2lo(), mp_gpio_interrupt_set(), mp_gpio_release(), mp_gpio_text_handle(), mp_i2c_close(), mp_i2c_open(), mp_i2c_setSlaveAddress(), mp_i2c_setup(), mp_options_get(),

mp_printk, mp_regMaster_init_i2c(), mp_regMaster_readExt(), mp_regMaster_register(), NULL, mp_drv_TMP006-_s::regMaster, mp_drv_TMP006_s::sensor, mp_drv_TMP006_s::settings, TMP006_B0, TMP006_B1, TMP006_B2, TMP006_REG_DEVICE_ID, TMP006_REG_MAN_ID, TMP006_REG_WRITE_REG, TMP006_S0, and TRUE.

Here is the call graph for this function:



2.10.5.3 void mp_drv_TMP006_sample (mp_drv_TMP006_t * *TMP006*, mp_drv_TMP006_sample_t *sample*)

Change sample rate for TMP006.

Parameters

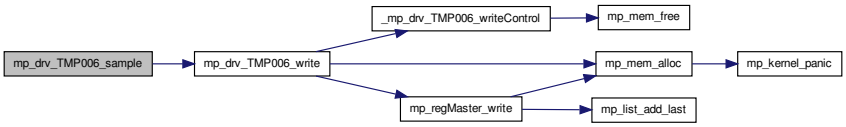
in	*TMP006*	context
in	*sample*	Sample rate possible value are : • TMP006_CFG_1SAMPLE : conversion rate 4 per second • TMP006_CFG_2SAMPLE : conversion rate 2 per second • TMP006_CFG_4SAMPLE : conversion rate 1 per second • TMP006_CFG_8SAMPLE : conversion rate 0.5 per second • TMP006_CFG_16SAMPLE : conversion rate 0.25 per second

Definition at line 245 of file TMP006.c.

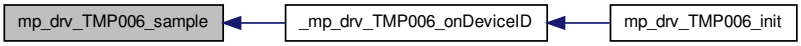
References mp_drv_TMP006_write(), mp_drv_TMP006_s::settings, and TMP006_REG_WRITE_REG.

Referenced by _mp_drv_TMP006_onDeviceID().

Here is the call graph for this function:



Here is the caller graph for this function:



2.10.5.4 static void mp_drv_TMP006_setB0 (mp_drv_TMP006_t * *TMP006*, float *value*) [inline], [static]

Change B0 calculation register.

Corrects for energy sources Environment dependent Calibrate in end-application environment Default value is set by TMP006_B0

Parameters

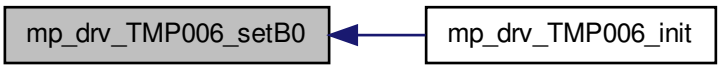
in	*TMP006*	context
in	*value*	new value

Definition at line 82 of file TMP006.h.

References mp_drv_TMP006_s::b0.

Referenced by mp_drv_TMP006_init().

Here is the caller graph for this function:



2.10.5.5 static void mp_drv_TMP006_setB1 (mp_drv_TMP006_t * *TMP006,* float *value*) [inline], [static]

Change B1 calculation register.

Corrects for energy sources Environment dependent Calibrate in end-application environment Default value is set by TMP006_B1

Parameters

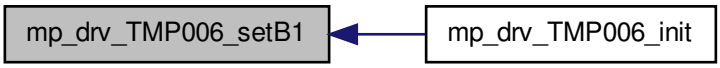
in	*TMP006*	context
in	*value*	new value

Definition at line 97 of file TMP006.h.

References mp_drv_TMP006_s::b1.

Referenced by mp_drv_TMP006_init().

Here is the caller graph for this function:



2.10.5.6 static void mp_drv_TMP006_setB2 (mp_drv_TMP006_t * *TMP006,* float *value*) [inline], [static]

Change B2 calculation register.

Corrects for energy sources Environment dependent Calibrate in end-application environment Default value is set by TMP006_B2

Parameters

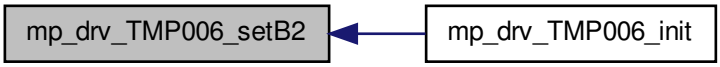
in	*TMP006*	context
in	*value*	new value

Definition at line 112 of file TMP006.h.

References mp_drv_TMP006_s::b2.

Referenced by mp_drv_TMP006_init().

Here is the caller graph for this function:



2.10.5.7 static void mp_drv_TMP006_setS0 (mp_drv_TMP006_t * *TMP006,* float *value*) [inline],[static]

Change S0 calculation register.

FOV and emissivity of object Application and object dependent Default values based on black body with epsilon = 0.95, and 110 FOV Default value is set by TMP006_S0

Parameters

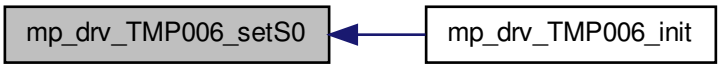
in	*TMP006*	context
in	*value*	new value

Definition at line 127 of file TMP006.h.

References mp_drv_TMP006_s::s0.

Referenced by mp_drv_TMP006_init().

Here is the caller graph for this function:



2.10.5.8 void mp_drv_TMP006_sleep (mp_drv_TMP006_t * *TMP006*)

Power down TMP006.

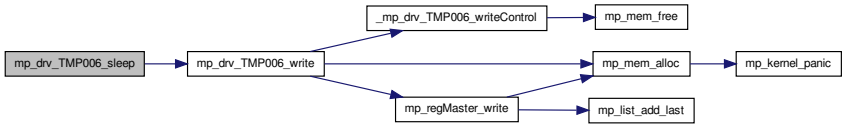
Parameters

in	*TMP006*	context

Definition at line 209 of file TMP006.c.

References mp_drv_TMP006_write(), mp_drv_TMP006_s::settings, TMP006_CFG_MODEON, and TMP006_REG_WRITE_REG.

Here is the call graph for this function:



2.10.5.9 void mp_drv_TMP006_wakeUp (mp_drv_TMP006_t * *TMP006*)

Power up TMP006.

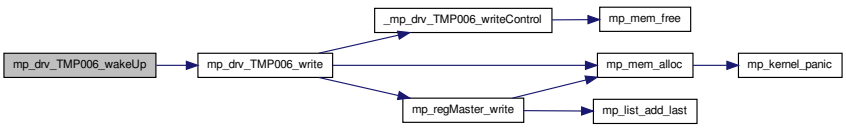
Parameters

in	*TMP006*	context

Definition at line 224 of file TMP006.c.

References mp_drv_TMP006_write(), mp_drv_TMP006_s::settings, TMP006_CFG_DRDYEN, TMP006_CFG_MODEON, and TMP006_REG_WRITE_REG.

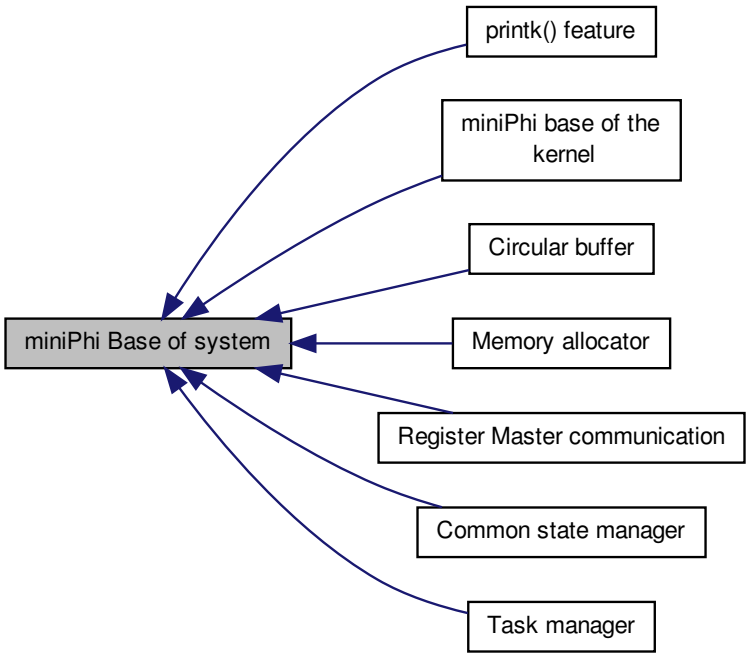
Here is the call graph for this function:



2.11 miniPhi Base of system

Implementation of features to build your firmware.

Collaboration diagram for miniPhi Base of system:



Modules

- Circular buffer

  Circular queueing system.
- miniPhi base of the kernel

  Provides logical base for an embedded kernel.
- Memory allocator

  Tiny memory allocation system.
- printk() feature

  Common kernel print.
- Register Master communication

  Circular register model for Master node.
- Common state manager

  States machine.
- Task manager

  Create and control kernel task.

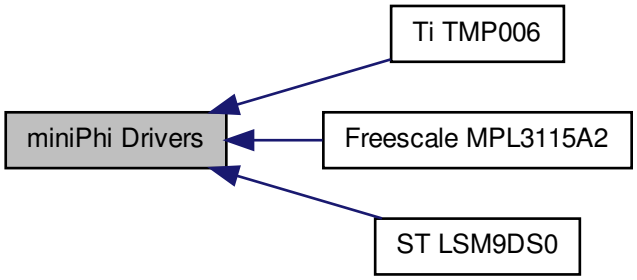
2.11.1 Detailed Description

Implementation of features to build your firmware.

2.12 miniPhi Drivers

Implementation of different chips on miniPhi.

Collaboration diagram for miniPhi Drivers:



Modules

- ST LSM9DS0

  ST LSM9DS0 3-axis Accelerometer / Gyroscope / Magneto.
- Freescale MPL3115A2

  Freescale MPL3115A2 Altimeter / Barometer / Temperature.
- Ti TMP006

  Ti TMP006 Object temperature.

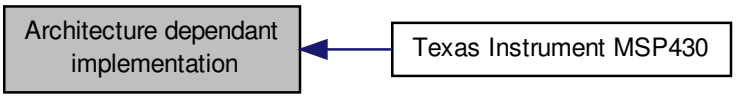
2.12.1 Detailed Description

Implementation of different chips on miniPhi.

2.13 Architecture dependant implementation

Differents processor architecture.

Collaboration diagram for Architecture dependant implementation:



Modules

- Texas Instrument MSP430

  Ti MSP430 implementation.

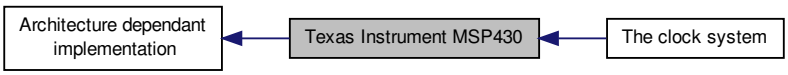
2.13.1 Detailed Description

Differents processor architecture.

2.14 Texas Instrument MSP430

Ti MSP430 implementation.

Collaboration diagram for Texas Instrument MSP430:



Modules

- The clock system

  The clock system for Ti MSP430 F5xx/F6xx.

2.14.1 Detailed Description

Ti MSP430 implementation.

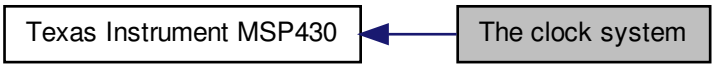
Author

2015 Michael Vergoz mv@verman.fr

2.15 The clock system

The clock system for Ti MSP430 F5xx/F6xx.

Collaboration diagram for The clock system:



Functions

- mp_ret_t mp_clock_init (mp_kernel_t ∗kernel)
- mp_ret_t mp_clock_fini (mp_kernel_t ∗kernel)
- mp_ret_t mp_clock_low_energy ()
- mp_ret_t mp_clock_high_energy ()
- unsigned long mp_clock_ticks ()
- void mp_clock_delay (int delay)
- unsigned long mp_clock_get_speed ()
- const char ∗ mp_clock_name (mp_clock_t clock)

2.15.1 Detailed Description

The clock system for Ti MSP430 F5xx/F6xx.

Version

1.0.0

Author

2015 Michael Vergoz mv@verman.fr

Date

03 Feb 2015

The clock system for Ti MSP430 F5xx/F6xx is based on Ti UCS (Unified Clock System).

The UCS module supports low system cost and ultralow power consumption. Using three internal clock signals, the user can select the best balance of performance and low power consumption. The UCS module can be configured to operate without any external components, with one or two external crystals, or with resonators, under full software control.

The UCS module includes up to five clock sources:

- XT1CLK: Low-frequency or high-frequency oscillator that can be used either with low-frequency 32768 Hz watch crystals, standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT1-CLK can be used as a clock reference into the FLL. Some devices only support the low frequency oscillator for XT1CLK. See the device-specific data sheet for supported functions.

- VLOCLK: Internal very low power, low frequency oscillator with 10 kHz typical frequency
- REFOCLK: Internal, trimmed, low-frequency oscillator with 32768 Hz typical frequency, with the ability to be used as a clock reference into the FLL
- DCOCLK: Internal digitally-controlled oscillator (DCO) that can be stabilized by the FLL
- XT2CLK: Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT2CLK can be used as a clock reference into the FLL.

Three clock signals are available from the UCS module:

- ACLK: Auxiliary clock. The ACLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. ACLK can be divided by 1, 2, 4, 8, 16, or 32. ACLK/n is ACLK divided by 1, 2, 4, 8, 16, or 32 and is available externally at a pin. ACLK is software selectable by individual peripheral modules.
- MCLK: Master clock. MCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. MCLK can be divided by 1, 2, 4, 8, 16, or 32. MCLK is used by the CPU and system.
- SMCLK: Subsystem master clock. SMCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. SMCLK can be divided by 1, 2, 4, 8, 16, or 32. SMCLK is software selectable by individual peripheral modules.

Depending the maximum CPU frequency miniPhi will choose the best way to drive clocks on the device.

- ACLK is driven by XT1 at anytime fixed to 32Khz
- MCLK could be driven by XT1 or XT2 depending if MP_CLOCK_XT2_DRIVE is set and the kernel is running in high power mode. At full speed the MCLK is driven by XT2 is possible.
- SCLK could be driven by XT1 or XT2

You can specify the frequency of each oscillator by setting MP_CLOCK_XT1_FREQ and MP_CLOCK_XT2_FREQ

2.15.2 Function Documentation

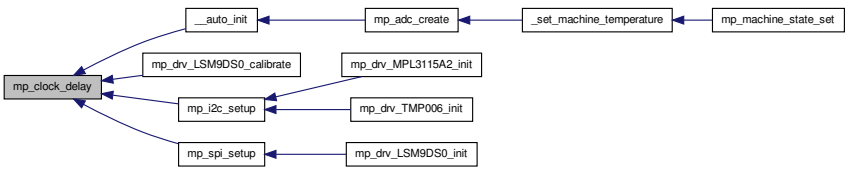
2.15.2.1 void mp_clock_delay (int *delay*)

Definition at line 178 of file clock.c.

References __ticks.

Referenced by __auto_init(), mp_drv_LSM9DS0_calibrate(), mp_i2c_setup(), and mp_spi_setup().

Here is the caller graph for this function:



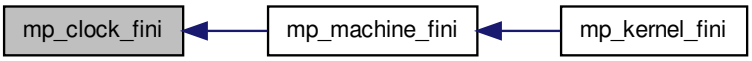
2.15.2.2 mp_ret_t mp_clock_fini (mp_kernel_t * *kernel*)

Definition at line 156 of file clock.c.

References TRUE.

Referenced by mp_machine_fini().

Here is the caller graph for this function:



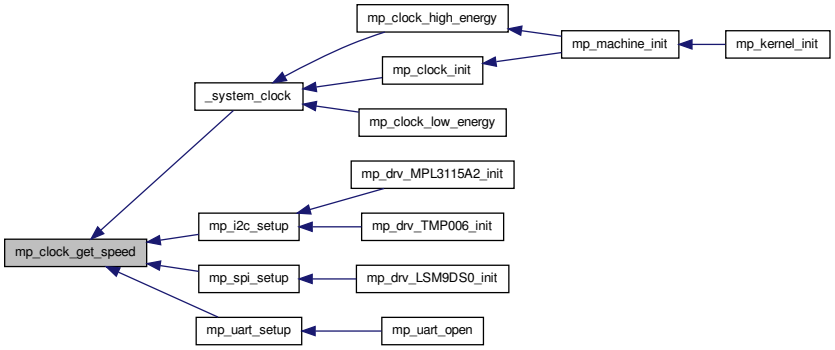
2.15.2.3 unsigned long mp_clock_get_speed ( )

Definition at line 183 of file clock.c.

References __frequency, mp_clock_freq_settings_s::DCO, and MHZ8_t.

Referenced by _system_clock(), mp_i2c_setup(), mp_spi_setup(), and mp_uart_setup().

Here is the caller graph for this function:



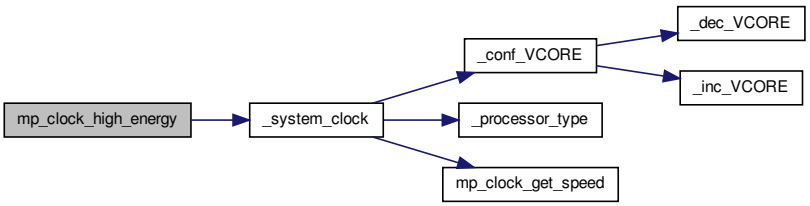
2.15.2.4 mp_ret_t mp_clock_high_energy ( )

Definition at line 167 of file clock.c.

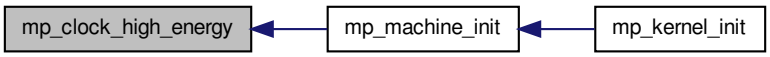
References _system_clock(), MP_CLOCK_HE_FREQ, and TRUE.

Referenced by mp_machine_init().

Here is the call graph for this function:



Here is the caller graph for this function:



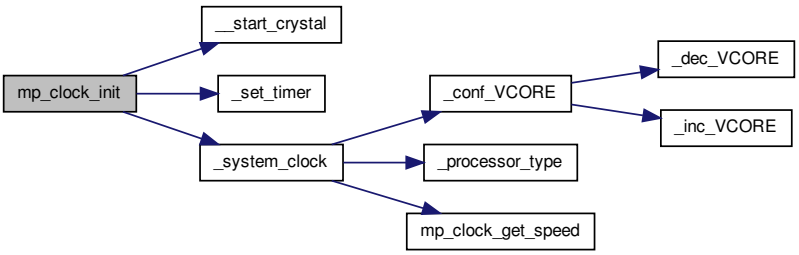
2.15.2.5 mp_ret_t mp_clock_init (mp_kernel_t * *kernel*)

Definition at line 143 of file clock.c.

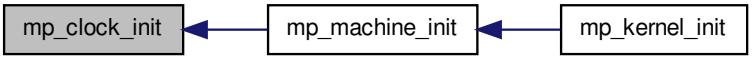
References __start_crystal(), _set_timer(), _system_clock(), MP_CLOCK_LE_FREQ, and TRUE.

Referenced by mp_machine_init().

Here is the call graph for this function:



Here is the caller graph for this function:

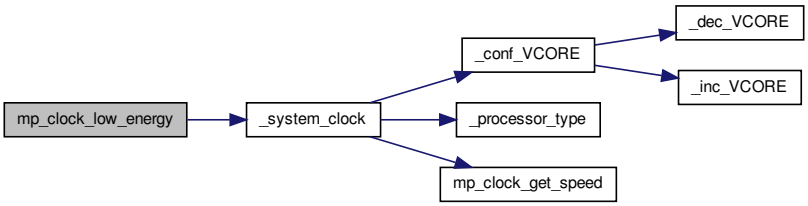


2.15.2.6 mp_ret_t mp_clock_low_energy ( )

Definition at line 161 of file clock.c.

References _system_clock(), MP_CLOCK_LE_FREQ, and TRUE.

Here is the call graph for this function:



2.15.2.7 const char* mp_clock_name (mp_clock_t *clock*)

Definition at line 189 of file clock.c.

References _mp_clock_freq_name.

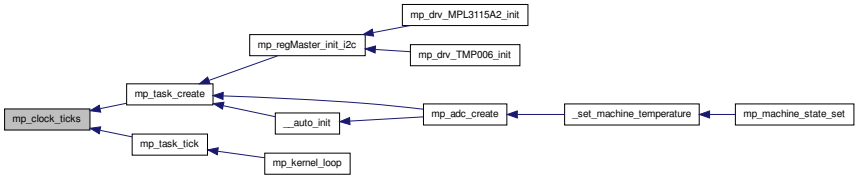
2.15.2.8 unsigned long mp_clock_ticks ( )

Definition at line 173 of file clock.c.

References __ticks.

Referenced by mp_task_create(), and mp_task_tick().

Here is the caller graph for this function:



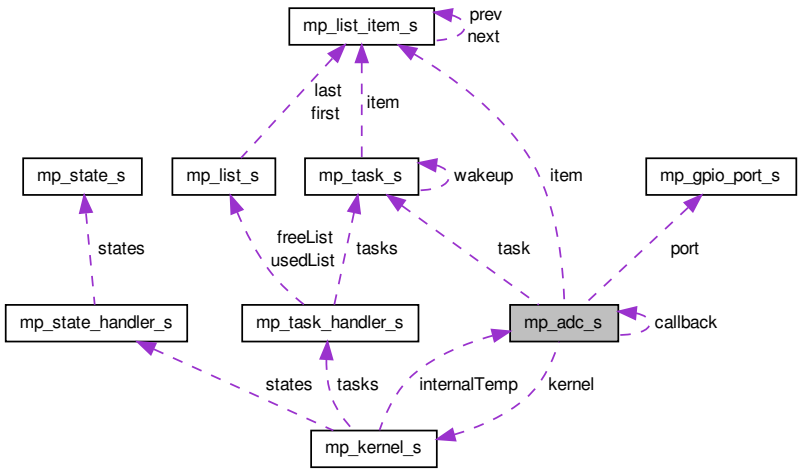
Chapter 3

Data Structure Documentation

3.1 mp_adc_s Struct Reference

`#include <adc.h>`

Collaboration diagram for mp_adc_s:



Data Fields

- mp_gpio_port_t * port
- mp_kernel_t * kernel
- char state
- char channelId
- int result
- mp_adc_callback_t callback
- void * user
- mp_task_t * task
- mp_list_item_t item

3.1.1 Detailed Description

Definition at line 28 of file adc.h.

3.1.2 Field Documentation

3.1.2.1 mp_adc_callback_t mp_adc_s::callback

Definition at line 37 of file adc.h.

Referenced by _set_machine_temperature(), and MP_TASK().

3.1.2.2 char mp_adc_s::channelId

Definition at line 34 of file adc.h.

Referenced by mp_adc_create(), mp_adc_remove(), and MP_TASK().

3.1.2.3 mp_list_item_t mp_adc_s::item

Definition at line 42 of file adc.h.

Referenced by mp_adc_create(), mp_adc_remove(), and MP_TASK().

3.1.2.4 mp_kernel_t* mp_adc_s::kernel

Definition at line 30 of file adc.h.

Referenced by mp_adc_create(), and mp_adc_remove().

3.1.2.5 mp_gpio_port_t* mp_adc_s::port

Definition at line 29 of file adc.h.

Referenced by mp_adc_create(), and mp_adc_remove().

3.1.2.6 int mp_adc_s::result

Definition at line 36 of file adc.h.

Referenced by switch().

3.1.2.7 char mp_adc_s::state

Definition at line 32 of file adc.h.

Referenced by mp_adc_create(), MP_TASK(), and switch().

3.1.2.8 mp_task_t* mp_adc_s::task

Definition at line 40 of file adc.h.

Referenced by mp_adc_create(), mp_adc_remove(), and switch().

3.1.2.9 void* mp_adc_s::user

Definition at line 38 of file adc.h.

Referenced by MP_TASK().

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

- include/msp430/adc.h

3.2 mp_circular_buffer_s Struct Reference

`#include <circular.h>`

Collaboration diagram for mp_circular_buffer_s:

Data Fields

- unsigned char data [MP_CIRCULAR_BUFFER_SIZE]
- unsigned short size
- unsigned short pos
- mp_circular_buffer_t * next

3.2.1 Detailed Description

Definition at line 39 of file circular.h.

3.2.2 Field Documentation

3.2.2.1 unsigned char mp_circular_buffer_s::data[MP_CIRCULAR_BUFFER_SIZE]

Definition at line 40 of file circular.h.

Referenced by mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

3.2.2.2 mp_circular_buffer_t* mp_circular_buffer_s::next

Definition at line 43 of file circular.h.

Referenced by mp_circular_fini(), mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

3.2.2.3 unsigned short mp_circular_buffer_s::pos

Definition at line 42 of file circular.h.

Referenced by mp_circular_txInterrupt(), and mp_circular_write().

3.2.2.4 unsigned short mp_circular_buffer_s::size

Definition at line 41 of file circular.h.

Referenced by mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

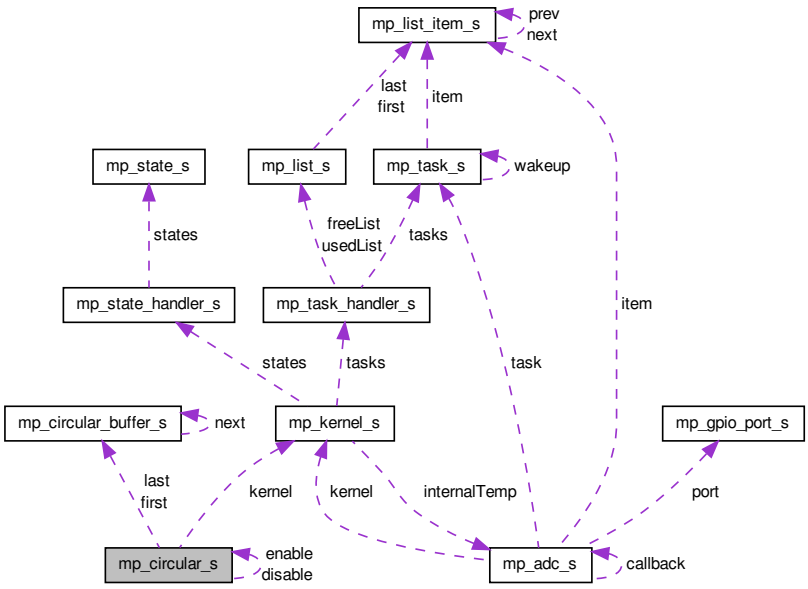
The documentation for this struct was generated from the following file:

- include/common/circular.h

3.3 mp_circular_s Struct Reference

`#include <circular.h>`

Collaboration diagram for mp_circular_s:



Data Fields

- mp_kernel_t * kernel
- int totalSize
- mp_circular_buffer_t * first
- mp_circular_buffer_t * last
- mp_circular_int_t enable
- mp_circular_int_t disable
- void * user

3.3.1 Detailed Description

Definition at line 46 of file circular.h.

3.3.2 Field Documentation

3.3.2.1 mp_circular_int_t mp_circular_s::disable

Definition at line 55 of file circular.h.

Referenced by mp_circular_init(), mp_circular_read(), mp_circular_txInterrupt(), and mp_circular_write().

3.3.2.2 mp_circular_int_t mp_circular_s::enable

Definition at line 54 of file circular.h.

Referenced by mp_circular_init(), mp_circular_read(), and mp_circular_write().

3.3.2.3 mp_circular_buffer_t* mp_circular_s::first

Definition at line 51 of file circular.h.

Referenced by mp_circular_fini(), mp_circular_init(), mp_circular_read(), mp_circular_txInterrupt(), and mp_circular_write().

3.3.2.4 mp_kernel_t* mp_circular_s::kernel

Definition at line 47 of file circular.h.

Referenced by mp_circular_fini(), mp_circular_init(), mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

3.3.2.5 mp_circular_buffer_t* mp_circular_s::last

Definition at line 52 of file circular.h.

Referenced by mp_circular_init(), mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

3.3.2.6 int mp_circular_s::totalSize

Definition at line 49 of file circular.h.

Referenced by mp_circular_read(), mp_circular_rxInterrupt(), mp_circular_txInterrupt(), and mp_circular_write().

3.3.2.7 void* mp_circular_s::user

Definition at line 57 of file circular.h.

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

- include/common/circular.h

3.4 mp_clock_freq_settings_s Struct Reference

`#include <clock.h>`

Data Fields

- unsigned char ID
- unsigned char DCORSEL
- unsigned char VCORE
- unsigned int DCO

3.4.1 Detailed Description

Definition at line 29 of file clock.h.

3.4.2 Field Documentation

3.4.2.1 unsigned int mp_clock_freq_settings_s::DCO

Definition at line 33 of file clock.h.

Referenced by _system_clock(), and mp_clock_get_speed().

3.4.2.2 unsigned char mp_clock_freq_settings_s::DCORSEL

Definition at line 31 of file clock.h.

Referenced by _system_clock().

3.4.2.3 unsigned char mp_clock_freq_settings_s::ID

Definition at line 30 of file clock.h.

3.4.2.4 unsigned char mp_clock_freq_settings_s::VCORE

Definition at line 32 of file clock.h.

Referenced by _system_clock().

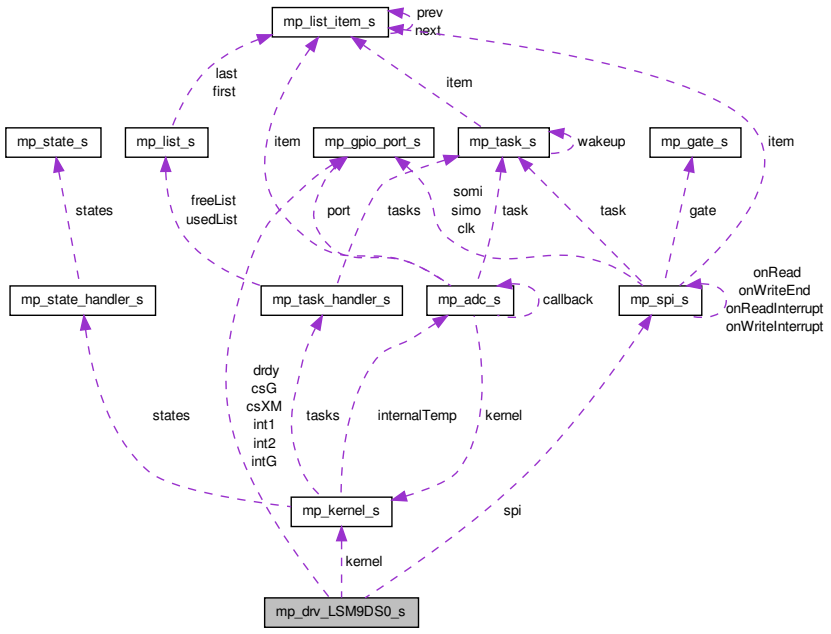
The documentation for this struct was generated from the following file:

- include/msp430/clock.h

3.5 mp_drv_LSM9DS0_s Struct Reference

`#include <LSM9DS0.h>`

Collaboration diagram for mp_drv_LSM9DS0_s:



Data Fields

- mp_kernel_t * kernel
- mp_drv_LSM9DS0_gyro_scale_t gyro_scale
- mp_drv_LSM9DS0_accel_scale_t accel_scale
- mp_drv_LSM9DS0_mag_scale_t mag_scale
- mp_gpio_port_t * csG
- mp_gpio_port_t * csXM
- mp_gpio_port_t * int1
- mp_gpio_port_t * int2
- mp_gpio_port_t * intG
- mp_gpio_port_t * drdy
- union {
  mp_spi_t spi
  };

- float gRes
- float aRes
- float mRes
- signed short gx
- signed short gy
- signed short gz
- signed short ax
- signed short ay
- signed short az
- signed short mx
- signed short my
- signed short mz
- signed short temperature

- float abias [3]
- float gbias [3]

3.5.1 Detailed Description

Definition at line 113 of file LSM9DS0.h.

3.5.2 Field Documentation

3.5.2.1 union { ... }

3.5.2.2 float mp_drv_LSM9DS0_s::abias[3]

Definition at line 139 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_calibrate().

3.5.2.3 mp_drv_LSM9DS0_accel_scale_t mp_drv_LSM9DS0_s::accel_scale

Definition at line 117 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcaRes(), and mp_drv_LSM9DS0_setAccelScale().

3.5.2.4 float mp_drv_LSM9DS0_s::aRes

Definition at line 134 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcaRes(), mp_drv_LSM9DS0_calcAccel(), and mp_drv_LSM9DS0_-calibrate().

3.5.2.5 signed short mp_drv_LSM9DS0_s::ax

Definition at line 136 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readAccel().

3.5.2.6 signed short mp_drv_LSM9DS0_s::ay

Definition at line 136 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readAccel().

3.5.2.7 signed short mp_drv_LSM9DS0_s::az

Definition at line 136 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readAccel().

3.5.2.8 mp_gpio_port_t* mp_drv_LSM9DS0_s::csG

Definition at line 120 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), mp_drv_LSM9DS0_gReadByte(), mp_drv_LSM9DS0_gReadBytes(), mp_drv_LSM9DS0_gWriteByte(), and mp_drv_LSM9DS0_init().

3.5.2.9 mp_gpio_port_t* mp_drv_LSM9DS0_s::csXM

Definition at line 121 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), mp_drv_LSM9DS0_init(), mp_drv_LSM9DS0_xmReadByte(), mp_drv_LSM9DS0_xmReadBytes(), and mp_drv_LSM9DS0_xmWriteByte().

3.5.2.10 mp_gpio_port_t* mp_drv_LSM9DS0_s::drdy

Definition at line 125 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), and mp_drv_LSM9DS0_init().

3.5.2.11 float mp_drv_LSM9DS0_s::gbias[3]

Definition at line 140 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_calibrate().

3.5.2.12 float mp_drv_LSM9DS0_s::gRes

Definition at line 134 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcgRes(), mp_drv_LSM9DS0_calcGyro(), and mp_drv_LSM9DS0_calibrate().

3.5.2.13 signed short mp_drv_LSM9DS0_s::gx

Definition at line 135 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readGyro().

3.5.2.14 signed short mp_drv_LSM9DS0_s::gy

Definition at line 135 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readGyro().

3.5.2.15 mp_drv_LSM9DS0_gyro_scale_t mp_drv_LSM9DS0_s::gyro_scale

Definition at line 116 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcgRes(), and mp_drv_LSM9DS0_setGyroScale().

3.5.2.16 signed short mp_drv_LSM9DS0_s::gz

Definition at line 135 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readGyro().

3.5.2.17 mp_gpio_port_t* mp_drv_LSM9DS0_s::int1

Definition at line 122 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), and mp_drv_LSM9DS0_init().

3.5.2.18 mp_gpio_port_t* mp_drv_LSM9DS0_s::int2

Definition at line 123 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), and mp_drv_LSM9DS0_init().

3.5.2.19 mp_gpio_port_t* mp_drv_LSM9DS0_s::intG

Definition at line 124 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_fini(), and mp_drv_LSM9DS0_init().

3.5.2.20 mp_kernel_t* mp_drv_LSM9DS0_s::kernel

Definition at line 114 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_init().

3.5.2.21 mp_drv_LSM9DS0_mag_scale_t mp_drv_LSM9DS0_s::mag_scale

Definition at line 118 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcmRes(), and mp_drv_LSM9DS0_setMagScale().

3.5.2.22 float mp_drv_LSM9DS0_s::mRes

Definition at line 134 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_calcmRes(), and mp_drv_LSM9DS0_calcMag().

3.5.2.23 signed short mp_drv_LSM9DS0_s::mx

Definition at line 137 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readMag().

3.5.2.24 signed short mp_drv_LSM9DS0_s::my

Definition at line 137 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readMag().

3.5.2.25 signed short mp_drv_LSM9DS0_s::mz

Definition at line 137 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readMag().

3.5.2.26 mp_spi_t mp_drv_LSM9DS0_s::spi

Definition at line 127 of file LSM9DS0.h.

Referenced by _mp_drv_LSM9DS0_spi_readBytes(), _mp_drv_LSM9DS0_spi_writeByte(), mp_drv_LSM9DS0_-fini(), and mp_drv_LSM9DS0_init().

3.5.2.27 signed short mp_drv_LSM9DS0_s::temperature

Definition at line 138 of file LSM9DS0.h.

Referenced by mp_drv_LSM9DS0_readTemp().

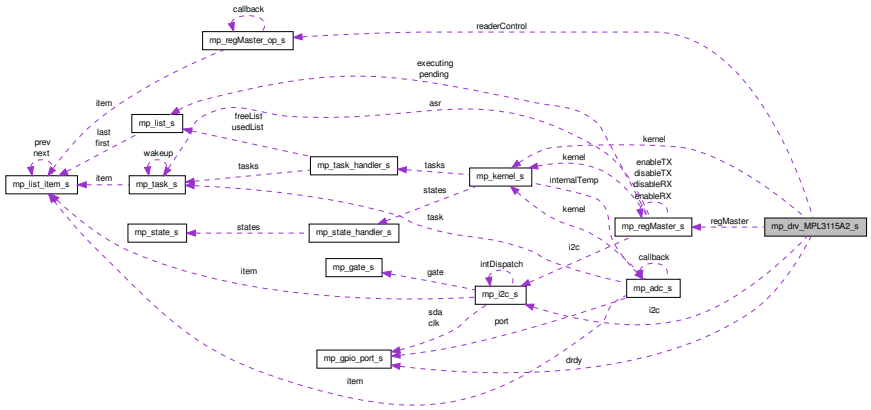
The documentation for this struct was generated from the following file:

- include/drivers/sensors/LSM9DS0.h

3.6 mp_drv_MPL3115A2_s Struct Reference

`#include <MPL3115A2.h>`

Collaboration diagram for mp_drv_MPL3115A2_s:



Data Fields

- mp_kernel_t ∗ kernel
- mp_i2c_t i2c
- mp_regMaster_t regMaster
- mp_gpio_port_t ∗ drdy
- mp_sensor_t ∗ sensor
- mp_sensor_t ∗ temperature
- unsigned char settings
- char whoIam
- mp_regMaster_cb_t readerControl

3.6.1 Detailed Description

Definition at line 35 of file MPL3115A2.h.

3.6.2 Field Documentation

3.6.2.1 mp_gpio_port_t∗ mp_drv_MPL3115A2_s::drdy

Definition at line 43 of file MPL3115A2.h.

Referenced by mp_drv_MPL3115A2_init().

3.6.2.2 mp_i2c_t mp_drv_MPL3115A2_s::i2c

Definition at line 39 of file MPL3115A2.h.

Referenced by mp_drv_MPL3115A2_init().

3.6.2.3 mp_kernel_t* mp_drv_MPL3115A2_s::kernel

kernel handler

Definition at line 37 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_readIntSource(), _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_MPL3115A2_readTemperature(), _mp_drv_MPL3115A2_writeControl(), mp_drv_MPL3115A2_acquisitionTimeStep(), mp_drv_MPL3115A2_disableTemperature(), mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_init(), mp_drv_MPL3115A2_OST(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_reset(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_setModeBarometer(), mp_drv_MPL3115A2_sleep(), and mp_drv_MPL3115A2_wakeUp().

3.6.2.4 mp_regMaster_cb_t mp_drv_MPL3115A2_s::readerControl

Definition at line 53 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_readIntSource(), mp_drv_MPL3115A2_setModeAltimeter(), and mp_drv_MPL3115A2_setModeBarometer().

3.6.2.5 mp_regMaster_t mp_drv_MPL3115A2_s::regMaster

Definition at line 41 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_onDRDY(), _mp_drv_MPL3115A2_onWhoIAm(), _mp_drv_MPL3115A2_readIntSource(), mp_drv_MPL3115A2_acquisitionTimeStep(), mp_drv_MPL3115A2_disableTemperature(), mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_init(), mp_drv_MPL3115A2_OST(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_reset(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_setModeBarometer(), mp_drv_MPL3115A2_sleep(), and mp_drv_MPL3115A2_wakeUp().

3.6.2.6 mp_sensor_t* mp_drv_MPL3115A2_s::sensor

Definition at line 45 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_readPressureControl(), mp_drv_MPL3115A2_disableTemperature(), mp_drv_MPL3115A2_enableTemperature(), mp_drv_MPL3115A2_setModeAltimeter(), and mp_drv_MPL3115A2_setModeBarometer().

3.6.2.7 unsigned char mp_drv_MPL3115A2_s::settings

Definition at line 49 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_onSettings(), mp_drv_MPL3115A2_init(), mp_drv_MPL3115A2_OST(), mp_drv_MPL3115A2_OSTimer(), mp_drv_MPL3115A2_reset(), mp_drv_MPL3115A2_setModeAltimeter(), mp_drv_MPL3115A2_setModeBarometer(), mp_drv_MPL3115A2_sleep(), and mp_drv_MPL3115A2_wakeUp().

3.6.2.8 mp_sensor_t* mp_drv_MPL3115A2_s::temperature

Definition at line 47 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_readTemperature(), mp_drv_MPL3115A2_disableTemperature(), and mp_drv_MPL3115A2_enableTemperature().

3.6.2.9 char mp_drv_MPL3115A2_s::wholam

Definition at line 51 of file MPL3115A2.h.

Referenced by _mp_drv_MPL3115A2_onWhoIAm(), and mp_drv_MPL3115A2_init().

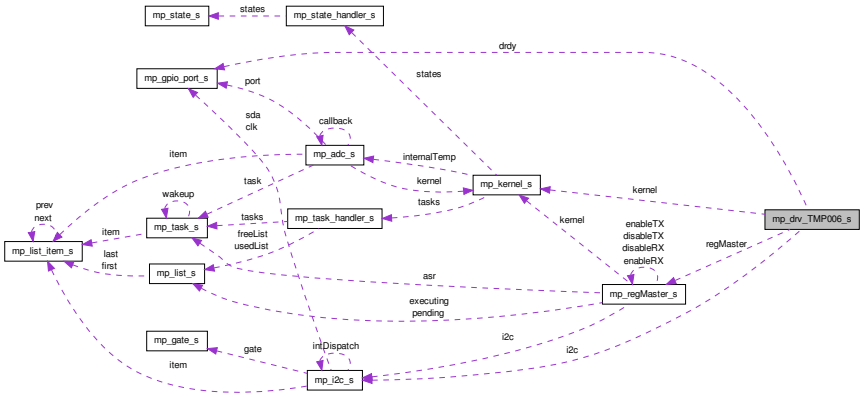
The documentation for this struct was generated from the following file:

- include/drivers/sensors/MPL3115A2.h

3.7 mp_drv_TMP006_s Struct Reference

`#include <TMP006.h>`

Collaboration diagram for mp_drv_TMP006_s:



Data Fields

- mp_kernel_t * kernel
- mp_i2c_t i2c
- mp_gpio_port_t * drdy
- mp_regMaster_t regMaster
- mp_sensor_t * sensor
- unsigned short deviceId
- unsigned short manufacturerId
- unsigned short settings
- unsigned short rawDieTemperature
- unsigned short rawVoltage
- float b0
- float b1
- float b2
- float s0

3.7.1 Detailed Description

Definition at line 35 of file TMP006.h.

3.7.2 Field Documentation

3.7.2.1 float mp_drv_TMP006_s::b0

Corrects for energy sources B0

Definition at line 51 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage(), and mp_drv_TMP006_setB0().

3.7.2.2 float mp_drv_TMP006_s::b1

Corrects for energy sources B1

Definition at line 54 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage(), and mp_drv_TMP006_setB1().

3.7.2.3 float mp_drv_TMP006_s::b2

Corrects for energy sources B2

Definition at line 57 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage(), and mp_drv_TMP006_setB2().

3.7.2.4 unsigned short mp_drv_TMP006_s::deviceId

Definition at line 44 of file TMP006.h.

Referenced by _mp_drv_TMP006_onDeviceID(), and mp_drv_TMP006_init().

3.7.2.5 mp_gpio_port_t* mp_drv_TMP006_s::drdy

Definition at line 38 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

3.7.2.6 mp_i2c_t mp_drv_TMP006_s::i2c

Definition at line 37 of file TMP006.h.

Referenced by mp_drv_TMP006_init().

3.7.2.7 mp_kernel_t* mp_drv_TMP006_s::kernel

Definition at line 36 of file TMP006.h.

Referenced by _mp_drv_TMP006_writeControl(), mp_drv_TMP006_init(), and mp_drv_TMP006_write().

3.7.2.8 unsigned short mp_drv_TMP006_s::manufacturerId

Definition at line 45 of file TMP006.h.

Referenced by _mp_drv_TMP006_onManufacturerID(), and mp_drv_TMP006_init().

3.7.2.9 unsigned short mp_drv_TMP006_s::rawDieTemperature

Definition at line 47 of file TMP006.h.

Referenced by _mp_drv_TMP006_onDRDY(), _mp_drv_TMP006_onRawDieTemperature(), and _mp_drv_TMP006_onRawVoltage().

3.7.2.10 unsigned short mp_drv_TMP006_s::rawVoltage

Definition at line 48 of file TMP006.h.

Referenced by _mp_drv_TMP006_onDRDY(), and _mp_drv_TMP006_onRawVoltage().

3.7.2.11 mp_regMaster_t mp_drv_TMP006_s::regMaster

Definition at line 40 of file TMP006.h.

Referenced by _mp_drv_TMP006_onDeviceID(), _mp_drv_TMP006_onDRDY(), _mp_drv_TMP006_onManufacturerID(), mp_drv_TMP006_fini(), mp_drv_TMP006_init(), and mp_drv_TMP006_write().

3.7.2.12 float mp_drv_TMP006_s::s0

FOV and emissivity of object

Definition at line 60 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage(), and mp_drv_TMP006_setS0().

3.7.2.13 mp_sensor_t* mp_drv_TMP006_s::sensor

Definition at line 42 of file TMP006.h.

Referenced by _mp_drv_TMP006_onRawVoltage(), and mp_drv_TMP006_init().

3.7.2.14 unsigned short mp_drv_TMP006_s::settings

Definition at line 46 of file TMP006.h.

Referenced by _mp_drv_TMP006_onSettings(), mp_drv_TMP006_init(), mp_drv_TMP006_sample(), mp_drv_TMP006_sleep(), and mp_drv_TMP006_wakeUp().

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

- include/drivers/sensors/TMP006.h

3.8 mp_gate_s Struct Reference

`#include <gate.h>`

Data Fields

- char * portDevice
- unsigned char gateFlags
- mp_bool_t isBusy
- char * byWho
- unsigned int _baseAddress

- unsigned int _ISRVector
- unsigned char _registersB

3.8.1 Detailed Description

Definition at line 6 of file gate.h.

3.8.2 Field Documentation

3.8.2.1 unsigned int mp_gate_s::_baseAddress

internal: register control

Definition at line 20 of file gate.h.

Referenced by mp_gate_init().

3.8.2.2 unsigned int mp_gate_s::_ISRVector

internal: ISR vector

Definition at line 23 of file gate.h.

Referenced by mp_gate_init(), mp_i2c_close(), mp_i2c_setup(), mp_uart_close(), and mp_uart_setup().

3.8.2.3 unsigned char mp_gate_s::_registersB

internal: register displacement

Definition at line 26 of file gate.h.

Referenced by mp_gate_init().

3.8.2.4 char* mp_gate_s::byWho

busy by who ?

Definition at line 17 of file gate.h.

Referenced by mp_gate_handle(), and mp_gate_release().

3.8.2.5 unsigned char mp_gate_s::gateFlags

gate user flags

Definition at line 11 of file gate.h.

3.8.2.6 mp_bool_t mp_gate_s::isBusy

is uart gate busy ?

Definition at line 14 of file gate.h.

Referenced by mp_gate_handle(), mp_gate_init(), and mp_gate_release().

3.8.2.7 char* mp_gate_s::portDevice

Gate name

Definition at line 8 of file gate.h.

Referenced by mp_gate_handle(), mp_gate_init(), mp_i2c_setup(), and mp_uart_setup().

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

- include/msp430/gate.h

3.9 mp_gpio_pair_s Struct Reference

`#include <gpio.h>`

Data Fields

- unsigned int port
- unsigned int pin

3.9.1 Detailed Description

Definition at line 33 of file gpio.h.

3.9.2 Field Documentation

3.9.2.1 unsigned int mp_gpio_pair_s::pin

Definition at line 35 of file gpio.h.

Referenced by mp_rtc_create().

3.9.2.2 unsigned int mp_gpio_pair_s::port

Definition at line 34 of file gpio.h.

Referenced by mp_rtc_create().

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

- include/msp430/gpio.h

3.10 mp_gpio_port_s Struct Reference

`#include <gpio.h>`

Data Fields

- unsigned int port
- unsigned int pin
- char * who
- mp_bool_t used

- char direction
- mp_interrupt_cb_t callback
- void * user
- char reverse
- unsigned int base
- unsigned char isr

3.10.1 Detailed Description

Definition at line 38 of file gpio.h.

3.10.2 Field Documentation

3.10.2.1 unsigned int mp_gpio_port_s::base

Definition at line 66 of file gpio.h.

Referenced by mp_gpio_init().

3.10.2.2 mp_interrupt_cb_t mp_gpio_port_s::callback

interrupt callback

Definition at line 57 of file gpio.h.

Referenced by _receive_port_interrupt(), mp_gpio_interrupt_set(), and mp_gpio_release().

3.10.2.3 char mp_gpio_port_s::direction

pin is interruptible

Definition at line 54 of file gpio.h.

Referenced by mp_gpio_direction(), mp_gpio_set(), mp_gpio_turn(), and mp_gpio_unset().

3.10.2.4 unsigned char mp_gpio_port_s::isr

Definition at line 69 of file gpio.h.

Referenced by mp_gpio_init(), and mp_gpio_interrupt_set().

3.10.2.5 unsigned int mp_gpio_port_s::pin

pin number

Definition at line 45 of file gpio.h.

Referenced by _receive_port_interrupt(), mp_adc_create(), mp_adc_remove(), mp_gpio_direction(), mp_gpio_init(), mp_gpio_interrupt_disable(), mp_gpio_interrupt_enable(), mp_gpio_interrupt_hi2lo(), mp_gpio_interrupt_hilo_switch(), mp_gpio_interrupt_lo2hi(), mp_gpio_interrupt_set(), mp_gpio_interrupt_unset(), mp_gpio_release(), mp_gpio_set(), mp_gpio_turn(), mp_gpio_unset(), mp_i2c_close(), mp_i2c_setup(), mp_spi_close(), mp_spi_setup(), mp_uart_close(), and mp_uart_open().

3.10.2.6 unsigned int mp_gpio_port_s::port

port number

Definition at line 42 of file gpio.h.

Referenced by mp_gpio_init().

3.10.2.7 char mp_gpio_port_s::reverse

Definition at line 63 of file gpio.h.

Referenced by mp_gpio_release(), mp_gpio_set(), and mp_gpio_unset().

3.10.2.8 mp_bool_t mp_gpio_port_s::used

pin used

Definition at line 51 of file gpio.h.

Referenced by mp_gpio_handle(), mp_gpio_init(), mp_gpio_release(), mp_gpio_set(), mp_gpio_turn(), and mp_gpio_unset().

3.10.2.9 void* mp_gpio_port_s::user

callback user pointer

Definition at line 60 of file gpio.h.

Referenced by _receive_port_interrupt(), and mp_gpio_interrupt_set().

3.10.2.10 char* mp_gpio_port_s::who

who is the handler

Definition at line 48 of file gpio.h.

Referenced by mp_gpio_handle(), and mp_gpio_release().

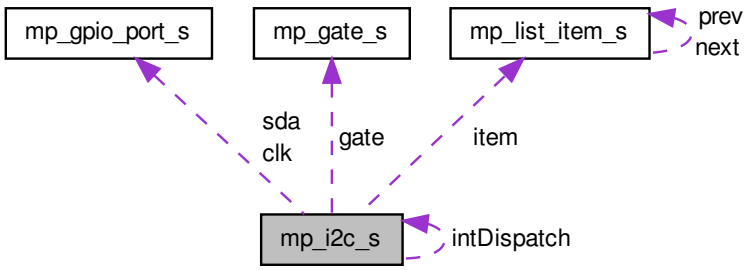
The documentation for this struct was generated from the following file:

- include/msp430/gpio.h

3.11 mp_i2c_s Struct Reference

`#include <i2c.h>`

Collaboration diagram for mp_i2c_s:



Data Fields

- mp_i2c_interrupt_t intDispatch
- mp_gate_t * gate
- mp_gpio_port_t * sda
- mp_gpio_port_t * clk
- mp_list_item_t item
- void * user

3.11.1 Detailed Description

Definition at line 38 of file i2c.h.

3.11.2 Field Documentation

3.11.2.1 mp_gpio_port_t* mp_i2c_s::clk

Definition at line 44 of file i2c.h.

Referenced by mp_i2c_close(), mp_i2c_open(), and mp_i2c_setup().

3.11.2.2 mp_gate_t* mp_i2c_s::gate

internal: gate

Definition at line 42 of file i2c.h.

Referenced by mp_i2c_clearFlags(), mp_i2c_close(), mp_i2c_disable_rx(), mp_i2c_disable_tx(), mp_i2c_enable_rx(), mp_i2c_enable_tx(), mp_i2c_interruptDispatch(), mp_i2c_mode(), mp_i2c_open(), mp_i2c_rx(), mp_i2c_setMyAddress(), mp_i2c_setSlaveAddress(), mp_i2c_setup(), mp_i2c_tx(), mp_i2c_txNACK(), mp_i2c_txStart(), mp_i2c_txStop(), mp_i2c_waitRX(), mp_i2c_waitStart(), mp_i2c_waitStop(), and mp_i2c_waitTX().

3.11.2.3 mp_i2c_interrupt_t mp_i2c_s::intDispatch

Definition at line 39 of file i2c.h.

Referenced by mp_i2c_interruptDispatch(), and mp_i2c_setInterruption().

3.11.2.4 mp_list_item_t mp_i2c_s::item

Definition at line 46 of file i2c.h.

Referenced by mp_i2c_close(), and mp_i2c_setup().

3.11.2.5 mp_gpio_port_t* mp_i2c_s::sda

Definition at line 43 of file i2c.h.

Referenced by mp_i2c_close(), mp_i2c_open(), and mp_i2c_setup().

3.11.2.6 void* mp_i2c_s::user

Definition at line 48 of file i2c.h.

Referenced by _mp_regMaster_i2c_interrupt(), and mp_regMaster_init_i2c().

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

- include/msp430/i2c.h

3.12 mp_interrupt_s Struct Reference

`#include <interrupt.h>`

Data Fields

- mp_interrupt_cb_t callback
- void * user
- char * who

3.12.1 Detailed Description

Definition at line 8 of file interrupt.h.

3.12.2 Field Documentation

3.12.2.1 mp_interrupt_cb_t mp_interrupt_s::callback

Definition at line 9 of file interrupt.h.

Referenced by mp_interrupt_init(), and mp_interrupt_unset().

3.12.2.2 void* mp_interrupt_s::user

Definition at line 11 of file interrupt.h.

Referenced by mp_interrupt_unset().

3.12.2.3 char* mp_interrupt_s::who

Definition at line 13 of file interrupt.h.

Referenced by mp_interrupt_unset().

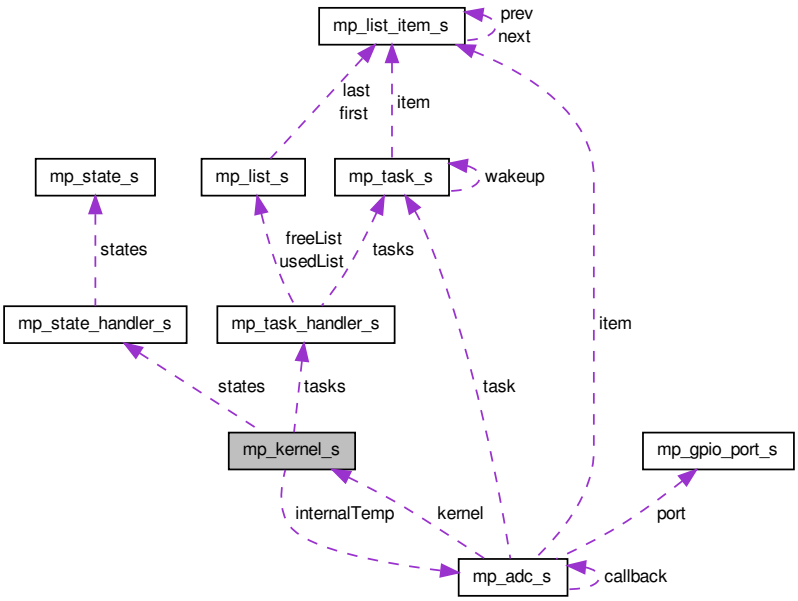
The documentation for this struct was generated from the following file:

- include/msp430/interrupt.h

3.13 mp_kernel_s Struct Reference

`#include <mp.h>`

Collaboration diagram for mp_kernel_s:



Data Fields

- char * mcuVendor
- char * mcuName
- char * version
- mp_state_handler_t states
- mp_task_handler_t tasks
- mp_sensor_handler_t sensors
- mp_kernel_onBoot_t onBoot
- void * onBootUser
- mp_adc_t internalTemp
- mp_sensor_t * sensorMCU

3.13.1 Detailed Description

Definition at line 125 of file mp.h.

3.13.2 Field Documentation

3.13.2.1 mp_adc_t mp_kernel_s::internalTemp

Definition at line 151 of file mp.h.

Referenced by _set_machine_temperature(), and _unset_machine_temperature().

3.13.2.2 char* mp_kernel_s::mcuName

Microchip name

Definition at line 130 of file mp.h.

Referenced by mp_machine_init().

3.13.2.3 char* mp_kernel_s::mcuVendor

Vendor name

Definition at line 127 of file mp.h.

Referenced by mp_machine_init().

3.13.2.4 mp_kernel_onBoot_t mp_kernel_s::onBoot

Kernel on boot state

Definition at line 145 of file mp.h.

Referenced by _mp_kernel_state_boot_tick(), and mp_kernel_init().

3.13.2.5 void* mp_kernel_s::onBootUser

User pointer for onBoot

Definition at line 148 of file mp.h.

Referenced by _mp_kernel_state_boot_tick(), and mp_kernel_init().

3.13.2.6 mp_sensor_t* mp_kernel_s::sensorMCU

Definition at line 152 of file mp.h.

Referenced by _set_machine_temperature(), and _unset_machine_temperature().

3.13.2.7 mp_sensor_handler_t mp_kernel_s::sensors

Sensors handler

Definition at line 142 of file mp.h.

3.13.2.8 mp_state_handler_t mp_kernel_s::states

Kernel states

Definition at line 136 of file mp.h.

Referenced by mp_kernel_fini(), mp_kernel_init(), mp_kernel_loop(), and mp_kernel_state().

3.13.2.9 mp_task_handler_t mp_kernel_s::tasks

Kernel tasks

Definition at line 139 of file mp.h.

Referenced by __auto_init(), mp_adc_create(), mp_kernel_fini(), mp_kernel_init(), mp_kernel_loop(), and mp_regMaster_init_i2c().

3.13.2.10 char* mp_kernel_s::version

Kernel version

Definition at line 133 of file mp.h.

Referenced by mp_kernel_init().

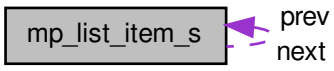
The documentation for this struct was generated from the following file:

- include/mp.h

3.14 mp_list_item_s Struct Reference

`#include <list.h>`

Collaboration diagram for mp_list_item_s:



Data Fields

- mp_list_item_t * next
- mp_list_item_t * prev
- void * user

3.14.1 Detailed Description

Definition at line 13 of file list.h.

3.14.2 Field Documentation

3.14.2.1 mp_list_item_t* mp_list_item_s::next

Definition at line 14 of file list.h.

Referenced by mp_list_add_first(), mp_list_add_last(), mp_list_remove(), MP_TASK(), and mp_task_flush().

3.14.2.2 mp_list_item_t* mp_list_item_s::prev

Definition at line 15 of file list.h.

Referenced by mp_list_add_first(), mp_list_add_last(), mp_list_remove(), and mp_task_tick().

3.14.2.3 void* mp_list_item_s::user

Definition at line 16 of file list.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_list_add_first(), mp_list_add_last(), mp_list_switch_first(), mp_list_switch_last(), MP_TASK(), mp_task_create(), mp_task_flush(), and mp_task_tick().

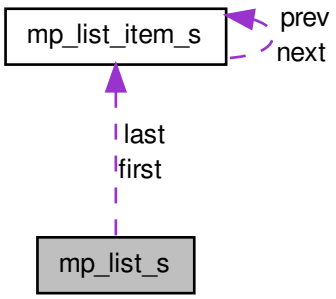
The documentation for this struct was generated from the following file:

- include/common/list.h

3.15 mp_list_s Struct Reference

`#include <list.h>`

Collaboration diagram for mp_list_s:



Data Fields

- mp_list_item_t * first
- mp_list_item_t * last

3.15.1 Detailed Description

Definition at line 7 of file list.h.

3.15.2 Field Documentation

3.15.2.1 mp_list_item_t* mp_list_s::first

Definition at line 8 of file list.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_list_add_first(), mp_list_add_last(), mp_list_init(), mp_list_remove(), MP_TASK(), mp_task_create(), mp_task_fini(), and mp_task_flush().

3.15.2.2 mp_list_item_t* mp_list_s::last

Definition at line 9 of file list.h.

Referenced by mp_list_add_first(), mp_list_add_last(), mp_list_init(), mp_list_remove(), mp_task_create(), and mp_task_tick().

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

- include/common/list.h

3.16 mp_mem_chunk_s Struct Reference

`#include <mem.h>`

Data Fields

- unsigned char data [MP_MEM_CHUNK]

3.16.1 Detailed Description

Definition at line 30 of file mem.h.

3.16.2 Field Documentation

3.16.2.1 unsigned char mp_mem_chunk_s::data[MP_MEM_CHUNK]

Definition at line 31 of file mem.h.

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

- include/common/mem.h

3.17 mp_options_s Struct Reference

`#include <mp.h>`

Data Fields

- char * key
- char * value

3.17.1 Detailed Description

Definition at line 70 of file mp.h.

3.17.2 Field Documentation

3.17.2.1 char* mp_options_s::key

Definition at line 71 of file mp.h.

Referenced by mp_options_get().

3.17.2.2 char* mp_options_s::value

Definition at line 72 of file mp.h.

Referenced by mp_options_get().

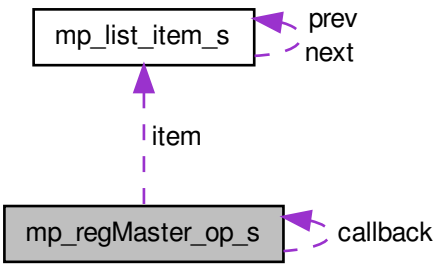
The documentation for this struct was generated from the following file:

- include/mp.h

3.18 mp_regMaster_op_s Struct Reference

`#include <regMaster.h>`

Collaboration diagram for mp_regMaster_op_s:



Data Fields

- char state
- unsigned char * reg
- int regSize
- int regPos
- unsigned char * wait
- int waitSize
- int waitPos
- mp_regMaster_cb_t callback
- void * user
- mp_bool_t swap
- mp_list_item_t item

3.18.1 Detailed Description

Definition at line 40 of file regMaster.h.

3.18.2 Field Documentation

3.18.2.1 mp_regMaster_cb_t mp_regMaster_op_s::callback

Definition at line 51 of file regMaster.h.

Referenced by mp_regMaster_readExt(), mp_regMaster_write(), and MP_TASK().

3.18.2.2 mp_list_item_t mp_regMaster_op_s::item

Definition at line 57 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), mp_regMaster_write(), and MP_TASK().

3.18.2.3 unsigned char* mp_regMaster_op_s::reg

Definition at line 43 of file regMaster.h.

Referenced by _mp_drv_MPL3115A2_writeControl(), _mp_drv_TMP006_writeControl(), _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), and mp_regMaster_write().

3.18.2.4 int mp_regMaster_op_s::regPos

Definition at line 45 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), and mp_regMaster_write().

3.18.2.5 int mp_regMaster_op_s::regSize

Definition at line 44 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), and mp_regMaster_write().

3.18.2.6 char mp_regMaster_op_s::state

Definition at line 41 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), mp_regMaster_write(), and MP_TASK().

3.18.2.7 mp_bool_t mp_regMaster_op_s::swap

Activate swap

Definition at line 55 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), and mp_regMaster_readExt().

3.18.2.8 void* mp_regMaster_op_s::user

Definition at line 52 of file regMaster.h.

Referenced by _mp_drv_MPL3115A2_onSettings(), _mp_drv_MPL3115A2_onWhoIAm(), _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_readIntSource(), _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_MPL3115A2_readTemperature(), _mp_drv_MPL3115A2_writeControl(), _mp_drv_TMP006_onDeviceID(), _mp_drv_TMP006_onManufacturerID(), _mp_drv_TMP006_onRawDieTemperature(), _mp_drv_TMP006_onRawVoltage(), _mp_drv_TMP006_onSettings(), _mp_drv_TMP006_writeControl(), mp_regMaster_readExt(), and mp_regMaster_write().

3.18.2.9 unsigned char* mp_regMaster_op_s::wait

Definition at line 47 of file regMaster.h.

Referenced by _mp_drv_MPL3115A2_readAltimeterControl(), _mp_drv_MPL3115A2_readIntSource(), _mp_drv_MPL3115A2_readPressureControl(), _mp_drv_MPL3115A2_readTemperature(), _mp_regMaster_i2c_interrupt(), and mp_regMaster_readExt().

3.18.2.10 int mp_regMaster_op_s::waitPos

Definition at line 49 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), and mp_regMaster_readExt().

3.18.2.11 int mp_regMaster_op_s::waitSize

Definition at line 48 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_readExt(), and MP_TASK().

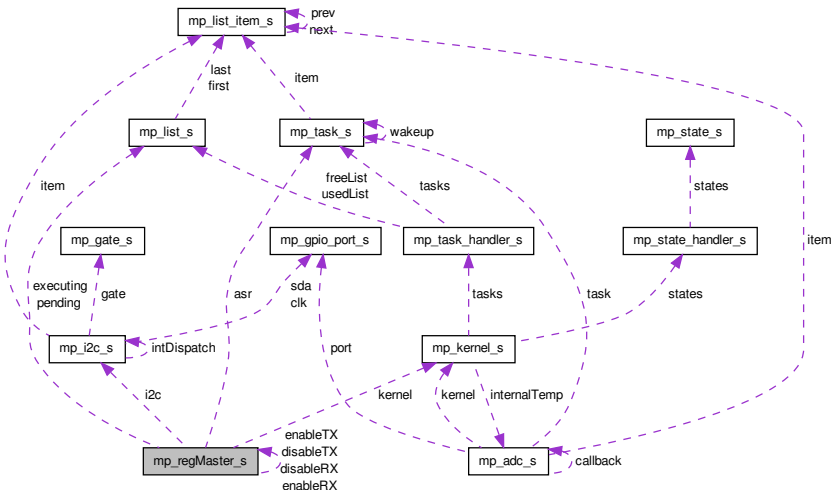
The documentation for this struct was generated from the following file:

- include/common/regMaster.h

3.19 mp_regMaster_s Struct Reference

`#include <regMaster.h>`

Collaboration diagram for mp_regMaster_s:



Data Fields

- mp_kernel_t * kernel
- mp_list_t executing
- mp_list_t pending
- mp_regMaster_int_t enableRX
- mp_regMaster_int_t disableRX
- mp_regMaster_int_t enableTX
- mp_regMaster_int_t disableTX
- mp_i2c_t * i2c
- void * user
- mp_task_t * asr

3.19.1 Detailed Description

Definition at line 60 of file regMaster.h.

3.19.2 Field Documentation

3.19.2.1 mp_task_t* mp_regMaster_s::asr

Definition at line 80 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_fini(), mp_regMaster_init_i2c(), mp_regMaster_readExt(), and mp_regMaster_write().

3.19.2.2 mp_regMaster_int_t mp_regMaster_s::disableRX

Definition at line 67 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_fini(), mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.3 mp_regMaster_int_t mp_regMaster_s::disableTX

Definition at line 70 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_fini(), mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.4 mp_regMaster_int_t mp_regMaster_s::enableRX

Definition at line 66 of file regMaster.h.

Referenced by mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.5 mp_regMaster_int_t mp_regMaster_s::enableTX

Definition at line 69 of file regMaster.h.

Referenced by mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.6 mp_list_t mp_regMaster_s::executing

Definition at line 63 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.7 mp_i2c_t* mp_regMaster_s::i2c

On bus error

Definition at line 75 of file regMaster.h.

Referenced by _mp_regMaster_i2c_disableRX(), _mp_regMaster_i2c_disableTX(), _mp_regMaster_i2c_enableRX(), _mp_regMaster_i2c_enableTX(), mp_regMaster_init_i2c(), and MP_TASK().

3.19.2.8 mp_kernel_t* mp_regMaster_s::kernel

Definition at line 61 of file regMaster.h.

Referenced by mp_regMaster_init_i2c(), mp_regMaster_readExt(), mp_regMaster_write(), and MP_TASK().

3.19.2.9 mp_list_t mp_regMaster_s::pending

Definition at line 64 of file regMaster.h.

Referenced by _mp_regMaster_i2c_interrupt(), mp_regMaster_init_i2c(), mp_regMaster_readExt(), mp_regMaster_write(), and MP_TASK().

3.19.2.10 void* mp_regMaster_s::user

Definition at line 78 of file regMaster.h.

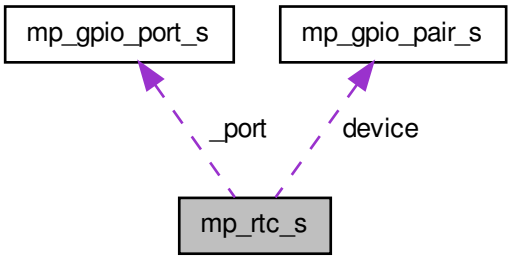
Referenced by mp_regMaster_init_i2c(), and MP_TASK().

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

- include/common/regMaster.h

3.20 mp_rtc_s Struct Reference

Collaboration diagram for mp_rtc_s:



Data Fields

- mp_gpio_pair_t device
- mp_gpio_direction_t direction

- mp_rtc_callback_t callback
- void * user
- mp_gpio_port_t * _port

3.20.1 Detailed Description

Definition at line 18 of file rtc.c.

3.20.2 Field Documentation

3.20.2.1 mp_gpio_port_t* mp_rtc_s::_port

gpio port

Definition at line 36 of file rtc.c.

Referenced by mp_rtc_create(), and mp_rtc_remove().

3.20.2.2 mp_rtc_callback_t mp_rtc_s::callback

callback on direction input

Definition at line 28 of file rtc.c.

3.20.2.3 mp_gpio_pair_t mp_rtc_s::device

GPIO pair define

Definition at line 22 of file rtc.c.

Referenced by mp_rtc_create().

3.20.2.4 mp_gpio_direction_t mp_rtc_s::direction

ping direction

Definition at line 25 of file rtc.c.

Referenced by mp_rtc_create().

3.20.2.5 void* mp_rtc_s::user

user pointer used at callback

Definition at line 31 of file rtc.c.

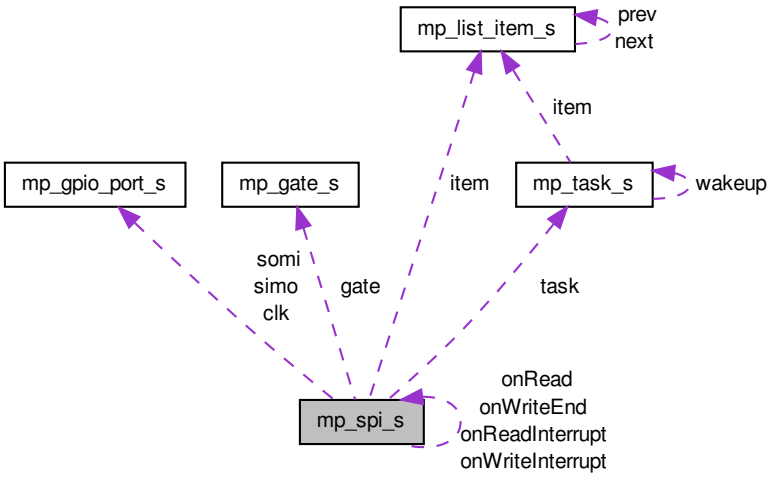
The documentation for this struct was generated from the following file:

- msp430/rtc.c

3.21 mp_spi_s Struct Reference

```
#include <spi.h>
```

Collaboration diagram for mp_spi_s:



Data Fields

- unsigned long frequency
- unsigned char tx_buffer [MP_SPI_TX_BUFFER]
- unsigned int tx_size
- unsigned int tx_pos
- unsigned int tx_reference
- unsigned char rx_buffer [MP_SPI_RX_BUFFER]
- unsigned int rx_size
- mp_spi_callback_t onRead
- mp_spi_callback_t onWriteEnd
- mp_spi_callback_t onReadInterrupt
- mp_spi_callback_t onWriteInterrupt
- void * user
- mp_gate_t * gate
- mp_gpio_port_t * simo
- mp_gpio_port_t * somi
- mp_gpio_port_t * clk
- mp_list_item_t item
- mp_task_t * task

3.21.1 Detailed Description

Definition at line 28 of file spi.h.

3.21.2 Field Documentation

3.21.2.1 mp_gpio_port_t* mp_spi_s::clk

Definition at line 54 of file spi.h.

Referenced by mp_spi_close(), mp_spi_open(), and mp_spi_setup().

3.21.2.2 unsigned long mp_spi_s::frequency

SPI frequency

Definition at line 31 of file spi.h.

3.21.2.3 mp_gate_t* mp_spi_s::gate

internal: gate

Definition at line 51 of file spi.h.

Referenced by mp_spi_close(), mp_spi_disable_rx(), mp_spi_disable_tx(), mp_spi_enable_rx(), mp_spi_enable_tx(), mp_spi_open(), mp_spi_rx(), mp_spi_setup(), and mp_spi_tx().

3.21.2.4 mp_list_item_t mp_spi_s::item

Definition at line 55 of file spi.h.

Referenced by mp_spi_close(), and mp_spi_setup().

3.21.2.5 mp_spi_callback_t mp_spi_s::onRead

Definition at line 44 of file spi.h.

3.21.2.6 mp_spi_callback_t mp_spi_s::onReadInterrupt

Definition at line 46 of file spi.h.

3.21.2.7 mp_spi_callback_t mp_spi_s::onWriteEnd

Definition at line 45 of file spi.h.

3.21.2.8 mp_spi_callback_t mp_spi_s::onWriteInterrupt

Definition at line 47 of file spi.h.

Referenced by mp_spi_write().

3.21.2.9 unsigned char mp_spi_s::rx_buffer[MP_SPI_RX_BUFFER]

Definition at line 41 of file spi.h.

3.21.2.10 unsigned int mp_spi_s::rx_size

Definition at line 42 of file spi.h.

3.21.2.11 mp_gpio_port_t* mp_spi_s::simo

Definition at line 52 of file spi.h.

Referenced by mp_spi_close(), mp_spi_open(), and mp_spi_setup().

3.21.2.12 mp_gpio_port_t* mp_spi_s::somi

Definition at line 53 of file spi.h.

Referenced by mp_spi_close(), mp_spi_open(), and mp_spi_setup().

3.21.2.13 mp_task_t* mp_spi_s::task

Definition at line 57 of file spi.h.

Referenced by mp_spi_close().

3.21.2.14 unsigned char mp_spi_s::tx_buffer[MP_SPI_TX_BUFFER]

Definition at line 36 of file spi.h.

Referenced by mp_spi_write().

3.21.2.15 unsigned int mp_spi_s::tx_pos

Definition at line 38 of file spi.h.

Referenced by mp_spi_write().

3.21.2.16 unsigned int mp_spi_s::tx_reference

Definition at line 39 of file spi.h.

Referenced by mp_spi_open().

3.21.2.17 unsigned int mp_spi_s::tx_size

Definition at line 37 of file spi.h.

Referenced by mp_spi_write().

3.21.2.18 void* mp_spi_s::user

Definition at line 48 of file spi.h.

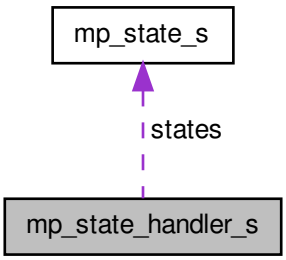
The documentation for this struct was generated from the following file:

- include/msp430/spi.h

3.22 mp_state_handler_s Struct Reference

`#include <state.h>`

Collaboration diagram for mp_state_handler_s:



Data Fields

- mp_state_t states [MP_STATE_MAX]
- char currentState
- char changeState

3.22.1 Detailed Description

Definition at line 24 of file state.h.

3.22.2 Field Documentation

3.22.2.1 char mp_state_handler_s::changeState

Definition at line 28 of file state.h.

Referenced by mp_state_switch(), and mp_state_tick().

3.22.2.2 char mp_state_handler_s::currentState

Definition at line 27 of file state.h.

Referenced by mp_state_tick().

3.22.2.3 mp_state_t mp_state_handler_s::states[MP_STATE_MAX]

Definition at line 25 of file state.h.

Referenced by mp_state_define(), mp_state_init(), and mp_state_tick().

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

- include/common/state.h

3.23 mp_state_s Struct Reference

`#include <state.h>`

Data Fields

- char number
- char * name
- char used
- void * user
- mp_state_callback_t set
- mp_state_callback_t unset
- mp_state_callback_t tick

3.23.1 Detailed Description

Definition at line 14 of file state.h.

3.23.2 Field Documentation

3.23.2.1 char* mp_state_s::name

Definition at line 16 of file state.h.

Referenced by mp_state_define().

3.23.2.2 char mp_state_s::number

Definition at line 15 of file state.h.

Referenced by mp_state_define().

3.23.2.3 mp_state_callback_t mp_state_s::set

Definition at line 19 of file state.h.

Referenced by mp_state_define(), and mp_state_tick().

3.23.2.4 mp_state_callback_t mp_state_s::tick

Definition at line 21 of file state.h.

Referenced by mp_state_define(), and mp_state_tick().

3.23.2.5 mp_state_callback_t mp_state_s::unset

Definition at line 20 of file state.h.

Referenced by mp_state_define(), and mp_state_tick().

3.23.2.6 char mp_state_s::used

Definition at line 17 of file state.h.

Referenced by mp_state_define().

3.23.2.7 void* mp_state_s::user

Definition at line 18 of file state.h.

Referenced by mp_state_define(), and mp_state_tick().

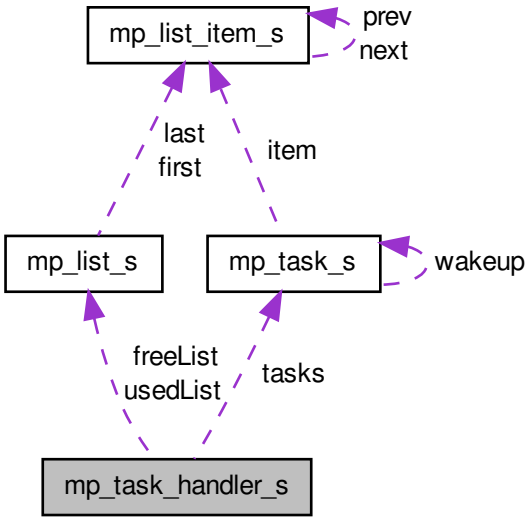
The documentation for this struct was generated from the following file:

- include/common/state.h

3.24 mp_task_handler_s Struct Reference

`#include <task.h>`

Collaboration diagram for mp_task_handler_s:



Data Fields

- mp_task_t tasks [MP_TASK_MAX]
- mp_list_t usedList
- unsigned int usedNumber
- mp_list_t freeList
- mp_task_signal_t signal

3.24.1 Detailed Description

Definition at line 71 of file task.h.

3.24.2 Field Documentation

3.24.2.1 mp_list_t mp_task_handler_s::freeList

free organized list

Definition at line 82 of file task.h.

Referenced by mp_task_create(), mp_task_init(), and mp_task_tick().

3.24.2.2 mp_task_signal_t mp_task_handler_s::signal

global signal

Definition at line 85 of file task.h.

Referenced by mp_task_fini(), mp_task_init(), and mp_task_tick().

3.24.2.3 mp_task_t mp_task_handler_s::tasks[MP_TASK_MAX]

inline tasks

Definition at line 73 of file task.h.

Referenced by mp_task_init().

3.24.2.4 mp_list_t mp_task_handler_s::usedList

used organized list

Definition at line 76 of file task.h.

Referenced by mp_task_create(), mp_task_fini(), mp_task_flush(), and mp_task_tick().

3.24.2.5 unsigned int mp_task_handler_s::usedNumber

number of items into used list

Definition at line 79 of file task.h.

Referenced by mp_task_create(), and mp_task_tick().

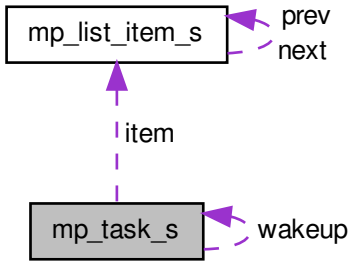
The documentation for this struct was generated from the following file:

- include/common/task.h

3.25 mp_task_s Struct Reference

`#include <task.h>`

Collaboration diagram for mp_task_s:



Data Fields

- unsigned char * name
- void * user
- unsigned long delay
- unsigned long check
- mp_task_wakeup_t wakeup
- mp_task_signal_t signal
- mp_list_item_t item

3.25.1 Detailed Description

Definition at line 48 of file task.h.

3.25.2 Field Documentation

3.25.2.1 unsigned long mp_task_s::check

last checking moment

Definition at line 59 of file task.h.

Referenced by mp_task_create(), and mp_task_tick().

3.25.2.2 unsigned long mp_task_s::delay

task scheduled delay

Definition at line 56 of file task.h.

Referenced by mp_task_create(), and mp_task_tick().

3.25.2.3 mp_list_item_t mp_task_s::item

list input

Definition at line 68 of file task.h.

Referenced by mp_task_create(), mp_task_flush(), mp_task_init(), and mp_task_tick().

3.25.2.4 unsigned char* mp_task_s::name

task name

Definition at line 50 of file task.h.

Referenced by mp_task_create().

3.25.2.5 mp_task_signal_t mp_task_s::signal

actual signal

Definition at line 65 of file task.h.

Referenced by __auto_init(), _mp_regMaster_i2c_interrupt(), mp_regMaster_init_i2c(), mp_regMaster_readExt(), mp_regMaster_write(), MP_TASK(), mp_task_create(), mp_task_destroy(), mp_task_flush(), mp_task_tick(), and switch().

3.25.2.6 void* mp_task_s::user

user pointer

Definition at line 53 of file task.h.

Referenced by mp_task_create().

3.25.2.7 mp_task_wakeup_t mp_task_s::wakeup

task wake up callback

Definition at line 62 of file task.h.

Referenced by mp_task_create(), and mp_task_tick().

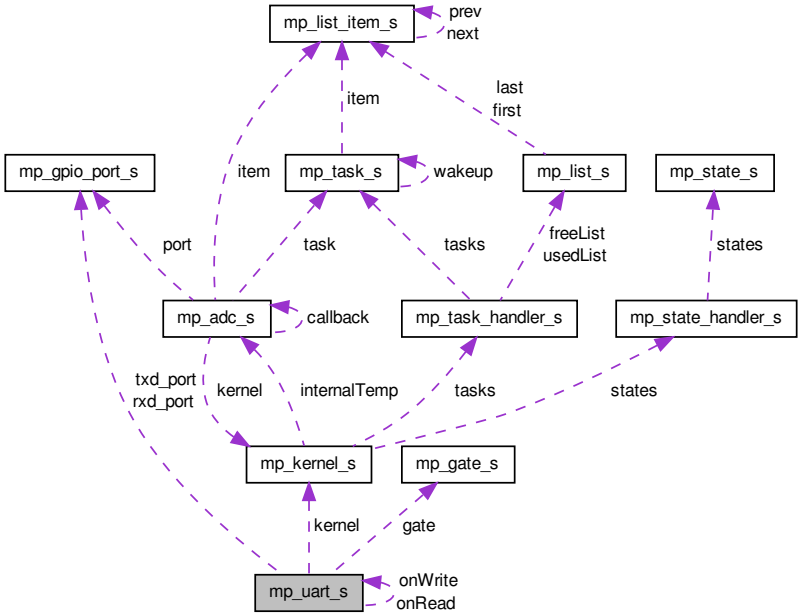
The documentation for this struct was generated from the following file:

- include/common/task.h

3.26 mp_uart_s Struct Reference

`#include <uart.h>`

Collaboration diagram for mp_uart_s:



Data Fields

- mp_kernel_t * kernel
- unsigned long baudRate
- mp_uart_on_t onWrite
- mp_uart_on_t onRead
- void * user
- mp_gate_t * gate
- mp_gpio_port_t * rxd_port
- mp_gpio_port_t * txd_port

3.26.1 Detailed Description

Definition at line 29 of file uart.h.

3.26.2 Field Documentation

3.26.2.1 unsigned long mp_uart_s::baudRate

Baud rate

Definition at line 34 of file uart.h.

Referenced by mp_uart_setup().

3.26.2.2 mp_gate_t* mp_uart_s::gate

internal: UART gate

Definition at line 41 of file uart.h.

Referenced by _mp_uart_interrupt(), mp_uart_close(), mp_uart_disable_rx_int(), mp_uart_disable_tx_int(), mp_uart_enable_rx_int(), mp_uart_enable_tx_int(), mp_uart_hasOverrun(), mp_uart_isBusy(), mp_uart_open(), mp_uart_rx(), mp_uart_setup(), and mp_uart_tx().

3.26.2.3 mp_kernel_t* mp_uart_s::kernel

Kernel handler

Definition at line 31 of file uart.h.

3.26.2.4 mp_uart_on_t mp_uart_s::onRead

Definition at line 37 of file uart.h.

Referenced by _mp_uart_interrupt().

3.26.2.5 mp_uart_on_t mp_uart_s::onWrite

Definition at line 36 of file uart.h.

Referenced by _mp_uart_interrupt().

3.26.2.6 mp_gpio_port_t* mp_uart_s::rxd_port

internal: rxd_port

Definition at line 44 of file uart.h.

Referenced by mp_uart_close(), and mp_uart_open().

3.26.2.7 mp_gpio_port_t* mp_uart_s::txd_port

internal: txd_port

Definition at line 47 of file uart.h.

Referenced by mp_uart_close(), and mp_uart_open().

3.26.2.8 void* mp_uart_s::user

Definition at line 38 of file uart.h.

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

- include/msp430/uart.h