

MS100 Firmware

0.3.9

Generated by Doxygen 1.8.7

Wed Mar 23 2016 11:11:25

Contents

Chapter 1

Sensurity PIDS Firmware

This project represents the universal source code for the Sensurity PIDS products.

Chapter 2

Todo List

Global `check_reset_condition` (void)

Alarm behaviour to be determined [modified and checked by PL on 9/10/15 and now is triggering error alarms as expected when a watchdog reset occurs]

Global `comms_action_entry` (void *p)

Disable this if vault is used

Global `comms_entry` (void *p)

It may be necessary to gracefully shut the peripherals and OS down, e.g. make sure SD card files aren't corrupted etc.

Global `comms_init` (void)

This is a waste of resources resulting from C++ port, during code beautification/optimisation fix it

Global `crc32_init` (uint32_t *crc)

Add the ability to dynamically create the CRC table or place it in ROM

Global `link_packet_alarm_encode` (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)

Create a global temperature variable

Global `LWIP_NETIF_TX_SINGLE_PBUF`

: TCP and IP-frag do not work with this, yet:

Global `main` (void)

Change the microwave link enable to a programmable feature

Group `OUI`

This is IOT's OUI, this must be changed...

Global `rfauth_dec` (uint8_t *key, uint8_t *buffer, uint32_t *timestamp, uint32_t *rolling)

Switch to 256-bit keys using VaultIC

Global `rfauth_enc` (uint8_t *key, uint32_t ctr, uint8_t *buffer, int32_t offset, uint32_t timestamp, uint32_t rolling)

Switch to 256-bit keys using VaultIC

Global `send_auth` (int32_t interface, bool *is_authenticated)

Global `tcpclient_restart_ipv4` (uint32_t ipv4, int32_t port)

Shutdown the thread, reconfigure the parameters and then create it again

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Config	??
IWDG	??
EEPROM_Emulation	??
STM32F4xx_StdPeriph_Driver	??
FLASH	??
FLASH_Private_Functions	??
FLASH Interface configuration functions	??
FLASH Memory Programming functions	??
Option Bytes Programming functions	??
Interrupts and flags management functions	??
FLASH_Exported_Constants	??
Flash_Latency	??
FLASH_Voltage_Range	??
FLASH_Sectors	??
Base address of the Flash sectors	??
Option_Bytes_Write_Protection	??
FLASH_Option_Bytes_Read_Protection	??
FLASH_Option_Bytes_IWatchdog	??
FLASH_Option_Bytes_nRST_STOP	??
FLASH_Option_Bytes_nRST_STDBY	??
FLASH_BOR_Reset_Level	??
FLASH_Interrupts	??
FLASH_Flags	??
FLASH_Program_Parallelism	??
FLASH_Keys	??
HAL_CONF	??
WEB_THREAD	??

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

_alarm_item_t	??
_alarms_t	??
_analog_alarms_t	??
_comms_hub_t	??
_default_vars_t	??
_float_uint32_u	??
_gps_info_t	??
_InitCfg	??
_int_frac_t	??
_link_auth_t	??
_link_packet_data_t	??
_link_rolling_t	??
_msg_info_t	??
_msg_ip_worker_t	??
_network_alarms_t	??
_nmea_utc_t	??
_osdp_ack_t	??
_osdp_alarm_zone_t	??
_osdp_busy_t	??
_osdp_cap_t	??
_osdp_halo_alarm_request_t	??
_osdp_halo_alarm_t	??
_osdp_halo_sync_t	??
_osdp_hub_messages_t	??
_osdp_istatr_t	??
_osdp_lstatr_t	??
_osdp_master_beacon_t	??
_osdp_nak_t	??
_osdp_pdcap_t	??
_osdp_rfauth_t	??
_osdp_sample_buffer_t	??
_packet_obj_t	??
_param_name_t	??
_relay_queue_t	??
_rsa_priv_t	??
_tuple_pdcap_t	??
CircularBuffer	??
creal32_T	??

creal64_T	??
creal_T	??
data_in	??
destructor	??
ElemType	??
hdl_wavelet	??
ip_settings_t	??
osdp_auth_rolling_t	??
osdp_ccrypt_t	??
osdp_comset_t	??
osdp_data	
osdp_ack	??
osdp_alarm_zone	??
osdp_auth_rolling	??
osdp_busy	??
osdp_cap	??
osdp_ccrypt	??
osdp_comset	??
osdp_halo_alarm	??
osdp_halo_alarm_request	??
osdp_halo_sync	??
osdp_hub_messages	??
osdp_id	??
osdp_istat	??
osdp_istatr	??
osdp_keyset	??
osdp_lstat	??
osdp_lstatr	??
osdp_master_beacon	??
osdp_ms100_mfg	??
osdp_ms100_mfgrep	??
osdp_nak	??
osdp_pdcap	??
osdp_pdid	??
osdp_poll	??
osdp_rmaci	??
osdp_scrypt	??
osdp_id_report_request	??
osdp_id_t	??
osdp_istat_t	??
osdp_keyset_t	??
osdp_lstat_t	??
osdp_ms100_mfg_t	??
osdp_ms100_mfgrep_t	??
osdp_pdid_t	??
osdp_poll_t	??
osdp_rmaci_t	??
osdp_scrypt_t	??
proxy_item_t	??
proxy_list_t	??
server_framework	??
server_framework_concrete	??
SHA256_CTX	??
statisticsC	??
ThreshStuff	??
wrapper_msg_base	??
wrapper_msg_pDIRpFILINFO	??
wrapper_msg_pDIRpTCHAR	??

wrapper_msg_pFIL	??
wrapper_msg_pFILpTCHARvBYTE	??
wrapper_msg_pFILpVOIDvUINTpUINT	??
wrapper_msg_pFILvDWORD	??
wrapper_msg_pTCHAR	??
wrapper_msg_pTCHARpDWORDppFATFS	??
wrapper_msg_pTCHARpFIL	??
wrapper_msg_pTCHARpFILINFO	??
wrapper_msg_pTCHARpTCHAR	??
wrapper_msg_pTCHARvBYTEvBYTE	??
wrapper_msg_pTCHARvINTpFILpTCHAR	??
wrapper_msg_pTCHARvUINT	??
wrapper_msg_vBYTE	??
wrapper_msg_vBYTEpDWORDpVOID	??
wrapper_msg_vBYTEpFATFS	??
wrapper_msg_vBYTEvBYTEvUINT	??
wrapper_msg_vTCHARpFIL	??

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

_alarm_item_t	??
_alarms_t	??
A struct definition used to provide flags for alarms	
_analog_alarms_t	??
_comms_hub_t	??
A struct used to store information for an OSDP communications interface	
_default_vars_t	??
_float_uint32_u	??
_gps_info_t	??
_InitCfg	??
_int_frac_t	??
A struct used to define an integer/fractional divider pair	
_link_auth_t	??
A struct used to store the id and alarm information for a specific device	
_link_packet_data_t	??
A struct used store the id and alarm information for a specific device	
_link_rolling_t	??
A struct used to store the rolling code data	
_msg_info_t	??
A struct used to define the position and time of the message origin	
_msg_ip_worker_t	??
_network_alarms_t	??
_nmea_utc_t	??
_osdp_ack_t	??
_osdp_alarm_zone_t	??
_osdp_busy_t	??
_osdp_cap_t	??
_osdp_halo_alarm_request_t	??
_osdp_halo_alarm_t	??
_osdp_halo_sync_t	??
_osdp_hub_messages_t	??
_osdp_istatr_t	??
_osdp_lstatr_t	??
_osdp_master_beacon_t	??
_osdp_nak_t	??
_osdp_pdcap_t	??
_osdp_rfauth_t	??
_osdp_sample_buffer_t	??

_packet_obj_t	??
_param_name_t	??
_relay_queue_t	??
_rsa_priv_t	??
_tuple_pdcap_t	??
CircularBuffer	??
creal32_T	??
creal64_T	??
creal_T	??
data_in	??
destructor	??
ElemType	??
hdl_wavelet	
A struct defining the parameters required to operate a single wavelet filter	??
ip_settings_t	
A structure defining IP settings	??
osdp_ack	??
osdp_alarm_zone	??
osdp_auth_rolling	??
osdp_auth_rolling_t	??
osdp_busy	??
osdp_cap	??
osdp_ccrypt	??
osdp_ccrypt_t	??
osdp_comset	??
osdp_comset_t	??
osdp_halo_alarm	??
osdp_halo_alarm_request	??
osdp_halo_sync	??
osdp_hub_messages	??
osdp_id	??
osdp_id_report_request	??
osdp_id_t	??
osdp_istat	??
osdp_istat_t	??
osdp_istatr	??
osdp_keyset	??
osdp_keyset_t	??
osdp_lstat	??
osdp_lstat_t	??
osdp_lstatr	??
osdp_master_beacon	??
osdp_ms100_mfg	??
osdp_ms100_mfg_t	??
osdp_ms100_mfgrep	??
osdp_ms100_mfgrep_t	??
osdp_nak	??
osdp_pdcap	??
osdp_pdid	??
osdp_pdid_t	??
osdp_poll	??
osdp_poll_t	??
osdp_rmaci	??
osdp_rmaci_t	??
osdp_scrypt	??
osdp_scrypt_t	??
proxy_item_t	??
proxy_list_t	??

server_framework	??
server_framework_concrete	??
SHA256_CTX	??
statisticsC	??
ThreshStuff	??
wrapper_msg_base	??
wrapper_msg_pDIRpFILINFO	??
wrapper_msg_pDIRpTCHAR	??
wrapper_msg_pFIL	??
wrapper_msg_pFILpTCHARvBYTE	??
wrapper_msg_pFILpVOIDvUINTpUINT	??
wrapper_msg_pFILvDWORD	??
wrapper_msg_pTCHAR	??
wrapper_msg_pTCHARpDWORDppFATFS	??
wrapper_msg_pTCHARpFIL	??
wrapper_msg_pTCHARpFILINFO	??
wrapper_msg_pTCHARpTCHAR	??
wrapper_msg_pTCHARvBYTEvBYTE	??
wrapper_msg_pTCHARvINTpFILpTCHAR	??
wrapper_msg_pTCHARvUINT	??
wrapper_msg_vBYTE	??
wrapper_msg_vBYTEpDWORDpVOID	??
wrapper_msg_vBYTEpFATFS	??
wrapper_msg_vBYTEvBYTEvUINT	??
wrapper_msg_vTCHARpFIL	??

Chapter 6

File Index

6.1 File List

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

chconf.h	??
crc32.c	??
crc32.h	
C functions for IEEE CRC-32 (reversed, non-inverted)	??
ext_interrupts.c	??
ext_interrupts.h	
External interrupts used by the MS100 system	??
ffconf.h	??
global.h	
A global include file for distribution of common definitions and variables	??
halconf.h	??
idle_thread.c	??
idle_thread.h	
A custom idle thread used to estimate CPU usage	??
main.c	
The main process and executable	??
maxconf.h	??
mcuconf.h	??
param_storage.c	??
param_storage.h	
Functions for storing the (encrypted) parameters in emulated EEPROM and an SD card file	??
prelnit.c	??
prelnit.h	
RF Initialization	??
status.c	??
status.h	
A thread used to operate the status LED on the rear of the board and the HW watchdog	??
stringutils.c	??
stringutils.h	
A thread used to provide an interface to the VaultIC device and perform the required cryptographic actions	??
write_eeprom.c	??
write_eeprom.h	??
comms/comms_action.c	??
comms/comms_action.h	
A thread dedicated to acting upon commands received via comms channels	??
comms/comms_hub.c	??

comms/comms_hub.h	
An interface to the various means of communicating with the MS100	??
comms/comms_packet.c	??
comms/comms_packet.h	
A packet processing worker thread for comms_hub	??
comms/comms_relay.c	??
comms/comms_relay.h	
A worker thread dedicated to relaying messages rather than processing them	??
comms/lwipopts.h	??
comms/lwippools.h	??
comms/packet_linked_list.c	??
comms/packet_linked_list.h	
A linked list resource used to share packet storage across the comms interface	??
comms/rs485_if.c	??
comms/rs485_if.h	
A C library for RS485 half-duplex communication	??
comms/tcpclient_thread.c	??
comms/tcpclient_thread.h	
A C library for a TCP client using lwIP. Currently this library supports IPv4 addresses only	??
comms/tcpserver.c	??
comms/tcpserver.h	??
crypto/aes.c	??
crypto/aes.h	
C functions for AES (LUT accelerated)	??
crypto/rng.c	??
crypto/rng.h	
C functions for random number generation	??
crypto/sha256.c	??
crypto/sha256.h	??
crypto/vaultic.c	??
crypto/vaultic.h	
A thread used to provide an interface to the VaultIC device and perform the required cryptographic actions	??
device/device_mac.c	??
device/device_mac.h	
Function to configure a unique MAC address from a Microsense OUI and the STM32 OUI	??
device/device_serial.c	??
device/device_serial.h	
Function to generate a unique 32-bit serial number based upon its STM32 UID	??
device/iwdg.c	
IWDG Driver code	??
device/iwdg.h	
IWDG Driver macros and structures	??
device/iwdg_ild.c	??
device/iwdg_ild.h	??
flash/eeeprom.c	??
flash/eeeprom.h	??
flash/external_eeeprom.c	??
flash/external_eeeprom.h	
Functions to enable the external EEPROM device to be used for storage	??
flash/stm32f4xx_flash.c	
This file provides firmware functions to manage the following functionalities of the FLASH peripheral:	??
flash/stm32f4xx_flash.h	
This file contains all the functions prototypes for the FLASH firmware library	??
gps/gps.c	??
gps/gps.h	
A thread used to provide a GPS location and manage the PPS	??

gps/gps_parser.c	??
gps/gps_parser.h	
A NMEA sentence parser with guarded access to decoded data using a mutex	??
link/link_alarm.c	??
link/link_alarm.h	
Encode/decode microwave link alarm packets for transmission between MS100 devices	??
link/link_alarm_list.c	??
link/link_alarm_list.h	
A doubly-linked list used to store link alarms	??
link/link_auth.c	??
link/link_auth.h	
Encode/decode microwave link authentication messages	??
link/link_control.c	??
link/link_control.h	
Microwave link control	??
link/link_if.c	??
link/link_if.h	
A C library for microwave link half-duplex communication	??
link/link_incoming.c	??
link/link_incoming.h	??
link/link_packet.c	??
link/link_packet.h	
Encode/decode microwave link alarm packets for transmission between MS100 devices	??
link/link_rolling.c	??
link/link_rolling.h	
Encode/decode microwave link rolling codes	??
sdcard/fatfsWrapper.c	??
sdcard/fatfsWrapper.h	
Wrapper around FatFS to localize all access over one single thread	??
sdcard/fatfsWrapperConfig.h	??
sdcard/sample_record.c	??
sdcard/sample_record.h	??
sdcard/sd_fw_download.c	??
sdcard/sd_fw_download.h	??
sdcard/sdcard.c	??
sdcard/sdcard.h	
A thread used to save buffered FW images as a background task	??
services/accelerometer.c	
Accelerometer driver code	??
services/accelerometer.h	
Accelerometer driver code	??
services/alarm.c	??
services/alarm.h	??
services/algorithm_thread.c	??
services/algorithm_thread.h	
FIFO adapted for use with microwave intruder detection algorithms	??
services/algorithms.c	??
services/algorithms.h	??
services/analog_input_driver.c	
Analog input sampling and alarm service, currently samples analog channels ADCIN0, ADCIN1 and ADCIN2. Alarm generated depending on how many samples are above the required threshold which is 2.3 volts at PSU	??
services/analog_input_driver.h	
Settings used in the analog input sampling algorithm	??
services/analog_input_driver_exp.c	??
services/analog_input_driver_exp.h	??

services/ dz_comms.c	Communicates between STM32 CPU and lower/upper DZ CPUs to enable: (i) LDZ/UDZ Sensitivity selection via control panel, (ii) LDZ/UDZ steady-state level available via control panel, (iii) Detection of whether LDZ/UDZ sensors are present in a Halo unit, (iv) Single version of D↔Z firmware (rather than separate versions for LDZ/UDZ), (v) Error alarm if DZ loses initialised settings or cannot be communicated with	??
services/ dz_comms.h	Contains constants for other Dead zone comms C files	??
services/ FIRcoeff_1.h		??
services/ FIRfunction.c		??
services/ led_driver.c		??
services/ led_driver.h	A control thread for the APDS-9700 optical sensor to provide tamper	??
services/ microwave.c		??
services/ microwave.h	Functions and timer callbacks used to detect alarms from sampled RF	??
services/ relays.c		??
services/ relays.h	Threads used to operate the output relays	??
services/ scanning_antenna.c		??
services/ scanning_antenna.h	Contains constants for other Scanning Antenna C files	??
services/ temp_sensor.c		??
services/ temp_sensor.h	API functions for reading and manipulating the board temperature	??
services/ tmwtypes.h		??
services/ wavelet.c		??
services/ wavelet.h		??
services/ wavelet_thread.c		??
services/ wavelet_thread.h		??
test/ func_link_authentication.cpp		??
test/ minunit.h		??
test/ unit_crc32.c	Unit testing for CRC32 functions	??
test/ unit_crypto.c		??
test/ unit_gps.c	Unit testing for the GPS parser	??
test/ unit_link_alarm.c	Unit testing for link alarm functions	??
test/ unit_link_alarm_list.c		??
test/ unit_link_auth.c	Unit testing for link authentication functions	??
test/ unit_link_packet.c	Unit testing for link packet functions	??
test/ unit_link_rolling.c		??
test/ unit_temperature.c		??
test/ unit_trace.c	Unit testing for the trace FIFO	??
trace/ trace.c		??
trace/ trace.h	A C library for recording trace strings in RAM and SD card storage	??
web/ web.c	HTTP server wrapper thread code	??
web/ web.h	HTTP server wrapper thread macros and structures	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ halo_alarms.h		??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ halo_proxy.c		??

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ halo_proxy.h	
A concrete implementation of osdp_data for ACK responses	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ack.cpp . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ack.h	
A concrete implementation of osdp_data for ACK responses	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ack_base.c .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ack_base.h	
A C implementation of ACK packet encode function	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_alarm_↵	
zone.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_alarm_zone.h	
A concrete implementation of osdp_data for ALARM_ZONE responses	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_alarm_↵	
zone_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_alarm_zone↵	
_base.h	
A C implementation of the proprietary Sensurity alarm zone packet	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_auth_↵	
rolling.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_auth_rolling.h	
A concrete implementation of osdp_data to send an MS100 rolling code	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_auth_↵	
rolling_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_auth_rolling↵	
_base.h	
Encode and decode functionality for MS100 rolling codes	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_busy.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_busy.h	
Encode and decode functionality for MS100 rolling codes	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_busy_↵	
base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_busy_base.h	
A C implementation of ACK packet encode function	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_cap.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_cap.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_cap_base.c .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_cap_base.h	
Encode and decode functionality for OSDP capability packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ccrypt.cpp .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ccrypt.h	
A concrete implementation of osdp_data to transmit SCS-CS cryptograms	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ccrypt_↵	
base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ccrypt_↵	
base.h	
The low-level encode/decode functions for osdp_CCRYPT functionality	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_comset.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_comset.h . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_comset_↵	
base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_comset_↵	
base.h	
Encode and decode functionality for OSDP COMSET packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_↵	
alarm.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_alarm.h	
A concrete implementation of osdp_data to transfer alarms	??

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_alarm_↵ _base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_alarm_↵ base.h A C implementation of the proprietary Halo alarm packet	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_sync_↵ cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_sync.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_sync_↵ _base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_halo_sync_↵ base.h A C implementation of the proprietary Halo sync packet	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_hub_↵ messages.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_hub_messages_↵ h A concrete implementation of osdp_data to send OSDP Hub commands and replies	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_hub_↵ messages_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_hub_↵ messages_base.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_id.cpp . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_id.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_id_base.c . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_id_base.h Encode and decode functionality for OSDP ID packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istat.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istat.h . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istat_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istat_base.h A C implementation of ISTAT packet encode function	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istatr.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istatr.h . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istatr_↵ base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_istatr_base.h A C implementation of ISTATR packet encode function	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_keyset.cpp .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_keyset.h A concrete implementation of osdp_data to configure the SCBK	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_keyset_↵ base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_keyset_↵ base.h The low-level encode/decode functions for osdp_KEYSET functionality	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstat.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstat.h . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstat_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstat_base.h A C implementation of LSTAT packet encode functionality	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstatr.cpp . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstatr.h . . .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstatr_↵ base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_lstatr_base.h A C implementation of LSTATR packet encode function	??

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_master_↵ beacon.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_master_↵ beacon.h A concrete implementation of osdp_data for MASTER_BEACON responses	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_master_↵ beacon_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_master_↵ beacon_base.h A C implementation of the proprietary master beacon packet	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfg.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_mfg.h A concrete implementation of osdp_data to send an MS100 Configuration command	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfg_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_mfg_↵ _base.h Encode and decode functionality for OSDP MFG packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfgrep.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfgrep.h A concrete implementation of osdp_data to set communication parameters	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfgrep_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_ms100_↵ mfgrep_base.h Encode and decode functionality for OSDP MFGREP packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_nak.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_nak.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_nak_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_nak_base.h Encode and decode functionality for OSDP NAK packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdcap.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdcap.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdcap_↵ base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdcap_↵ base.h Encode and decode functionality for OSDP PDCAP packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdid.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdid.h	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdid_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_pdid_base.h Encode and decode functionality for OSDP PDID packets	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_poll.cpp	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_poll.h A concrete implementation of osdp_data for ISTAT requests	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_poll_base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_poll_base.h A C implementation of POLL packet encode function	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rfauth_↵ base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rfauth_base.h A C implementation of the proprietary RF Authentication packet	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rmaci.cpp	??

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rmaci.h	
A concrete implementation of osdp_data to transmit SCS initial MAC	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rmaci_↵	
base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_rmaci_base.h	
The low-level encode/decode functions for osdp_RMAC_I functionality	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_scrypt.cpp .	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_scrypt.h	
A concrete implementation of osdp_data to transmit SCS-CS cryptograms	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_scrypt_↵	
base.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ osdp_scrypt_↵	
base.h	
The low-level encode/decode functions for osdp_SCRIPT functionality	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ rfauth.c	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ rfauth.h	
Encode and decode functionality for RF authentication	??
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/ server_framework.h	
An abstract base class used to distribute resources throughout the system	??

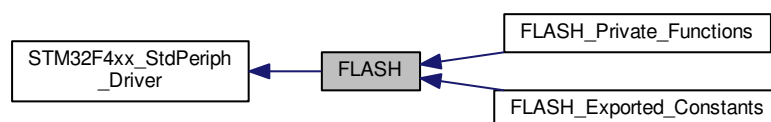
Chapter 7

Module Documentation

7.1 FLASH

FLASH driver modules.

Collaboration diagram for FLASH:



Modules

- [FLASH_Private_Functions](#)
- [FLASH_Exported_Constants](#)

Macros

- `#define SECTOR_MASK ((uint32_t)0xFFFFF07)`

Enumerations

- `enum FLASH_Status {
 FLASH_BUSY = 1, FLASH_ERROR_PGS, FLASH_ERROR_PGP, FLASH_ERROR_PGA,
 FLASH_ERROR_WRP, FLASH_ERROR_PROGRAM, FLASH_ERROR_OPERATION, FLASH_COMPLETE,
 FLASH_ERROR_ECC }`
 FLASH Status.

Functions

- `void FLASH\_SetLatency (uint32_t FLASH_Latency)`
 Sets the code latency value.

- void [FLASH_PrefetchBufferCmd](#) (FunctionalState NewState)
Enables or disables the Prefetch Buffer.
- void [FLASH_InstructionCacheCmd](#) (FunctionalState NewState)
Enables or disables the Instruction Cache feature.
- void [FLASH_DataCacheCmd](#) (FunctionalState NewState)
Enables or disables the Data Cache feature.
- void [FLASH_InstructionCacheReset](#) (void)
Resets the Instruction Cache.
- void [FLASH_DataCacheReset](#) (void)
Resets the Data Cache.
- uint32_t [FLASH_GetSector](#) (uint32_t Address)
Return the Flash sector bitmask of the address.
- uint32_t [FLASH_GetSectorNumber](#) (uint32_t Address)
Return the Flash sector Number of the address.
- void [FLASH_Unlock](#) (void)
Unlocks the FLASH control register access.
- void [FLASH_Lock](#) (void)
Locks the FLASH control register access.
- [FLASH_Status FLASH_EraseSector](#) (uint32_t FLASH_Sector, uint8_t VoltageRange)
Erases a specified FLASH Sector.
- [FLASH_Status FLASH_EraseAllSectors](#) (uint8_t VoltageRange)
Erases all FLASH Sectors.
- [FLASH_Status FLASH_ProgramDoubleWord](#) (uint32_t Address, uint64_t Data)
Programs a double word (64-bit) at a specified address.
- [FLASH_Status FLASH_ProgramWord](#) (uint32_t Address, uint32_t Data)
Programs a word (32-bit) at a specified address.
- [FLASH_Status FLASH_ProgramHalfWord](#) (uint32_t Address, uint16_t Data)
Programs a half word (16-bit) at a specified address.
- [FLASH_Status FLASH_ProgramByte](#) (uint32_t Address, uint8_t Data)
Programs a byte (8-bit) at a specified address.
- void [FLASH_OB_Unlock](#) (void)
Unlocks the FLASH Option Control Registers access.
- void [FLASH_OB_Lock](#) (void)
Locks the FLASH Option Control Registers access.
- void [FLASH_OB_WRPConfig](#) (uint32_t OB_WRP, FunctionalState NewState)
Enables or disables the write protection of the desired sectors.
- void [FLASH_OB_RDPCConfig](#) (uint8_t OB_RDP)
Sets the read protection level.
- void [FLASH_OB_UserConfig](#) (uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)
Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.
- void [FLASH_OB_BORConfig](#) (uint8_t OB_BOR)
Sets the BOR Level.
- [FLASH_Status FLASH_OB_Launch](#) (void)
Launch the option byte loading.
- uint8_t [FLASH_OB_GetUser](#) (void)
Returns the FLASH User Option Bytes values.
- uint16_t [FLASH_OB_GetWRP](#) (void)
Returns the FLASH Write Protection Option Bytes value.
- FlagStatus [FLASH_OB_GetRDP](#) (void)
Returns the FLASH Read Protection level.
- uint8_t [FLASH_OB_GetBOR](#) (void)

- Returns the FLASH BOR level.*
- void **FLASH_ITConfig** (uint32_t FLASH_IT, FunctionalState NewState)
Enables or disables the specified FLASH interrupts.
- FlagStatus **FLASH_GetFlagStatus** (uint32_t FLASH_FLAG)
Checks whether the specified FLASH flag is set or not.
- void **FLASH_ClearFlag** (uint32_t FLASH_FLAG)
Clears the FLASH's pending flags.
- **FLASH_Status FLASH_GetStatus** (void)
Returns the FLASH Status.
- **FLASH_Status FLASH_WaitForLastOperation** (void)
Waits for a FLASH operation to complete.

7.1.1 Detailed Description

FLASH driver modules.

7.1.2 Function Documentation

7.1.2.1 void FLASH_ClearFlag (uint32_t FLASH_FLAG)

Clears the FLASH's pending flags.

Parameters

<i>FLASH_FLAG</i>	specifies the FLASH flags to clear. This parameter can be any combination of the following values: • FLASH_FLAG_EOP: FLASH End of Operation flag • FLASH_FLAG_OPERR: FLASH operation Error flag • FLASH_FLAG_WRPERR: FLASH Write protected error flag • FLASH_FLAG_PGAERR: FLASH Programming Alignment error flag • FLASH_FLAG_PGPERR: FLASH Programming Parallelism error flag • FLASH_FLAG_PGSERR: FLASH Programming Sequence error flag
-------------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 966 of file stm32f4xx_flash.c.

7.1.2.2 void FLASH_DataCacheCmd (FunctionalState *NewState*)

Enables or disables the Data Cache feature.

Parameters

<i>NewState</i>	new state of the Data Cache. This parameter can be: ENABLE or DISABLE.
-----------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 226 of file stm32f4xx_flash.c.

7.1.2.3 void FLASH_DataCacheReset (void)

Resets the Data Cache.

Note

This function must be used only when the Data Cache is disabled.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 258 of file stm32f4xx_flash.c.

7.1.2.4 FLASH_Status FLASH_EraseAllSectors (uint8_t *VoltageRange*)

Erases all FLASH Sectors.

Parameters

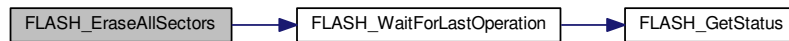
<i>VoltageRange</i>	The device voltage range which defines the erase parallelism. This parameter can be one of the following values: • <i>VoltageRange_1</i>: when the device voltage range is 1.8V to 2.1V, the operation will be done by byte (8-bit) • <i>VoltageRange_2</i>: when the device voltage range is 2.1V to 2.7V, the operation will be done by half word (16-bit) • <i>VoltageRange_3</i>: when the device voltage range is 2.7V to 3.6V, the operation will be done by word (32-bit) • <i>VoltageRange_4</i>: when the device voltage range is 2.7V to 3.6V + External Vpp, the operation will be done by double word (64-bit)
---------------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG← RAM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP← LETE.
--------------	---

Definition at line 408 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.5 FLASH_Status FLASH_EraseSector (uint32_t FLASH_Sector, uint8_t VoltageRange)

Erases a specified FLASH Sector.

Parameters

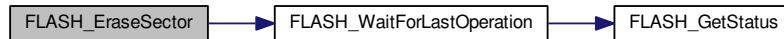
<i>FLASH_Sector</i>	The Sector number to be erased. This parameter can be a value between FLASH_Sector_0 and FLASH_Sector_11
<i>VoltageRange</i>	The device voltage range which defines the erase parallelism. This parameter can be one of the following values: • VoltageRange_1: when the device voltage range is 1.8V to 2.1V, the operation will be done by byte (8-bit) • VoltageRange_2: when the device voltage range is 2.1V to 2.7V, the operation will be done by half word (16-bit) • VoltageRange_3: when the device voltage range is 2.7V to 3.6V, the operation will be done by word (32-bit) • VoltageRange_4: when the device voltage range is 2.7V to 3.6V + External Vpp, the operation will be done by double word (64-bit)

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↵AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↵LETE.
--------------	---

Definition at line 343 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.6 FlagStatus FLASH_GetFlagStatus (uint32_t *FLASH_FLAG*)

Checks whether the specified FLASH flag is set or not.

Parameters

<i>FLASH_FLAG</i>	specifies the FLASH flag to check. This parameter can be one of the following values: • FLASH_FLAG_EOP: FLASH End of Operation flag • FLASH_FLAG_OPERR: FLASH operation Error flag • FLASH_FLAG_WRPERR: FLASH Write protected error flag • FLASH_FLAG_PGAERR: FLASH Programming Alignment error flag • FLASH_FLAG_PGPERR: FLASH Programming Parallelism error flag • FLASH_FLAG_PGSERR: FLASH Programming Sequence error flag • FLASH_FLAG_BSY: FLASH Busy flag
-------------------	---

Return values

<i>The</i>	new state of FLASH_FLAG (SET or RESET).
------------	---

Definition at line 936 of file stm32f4xx_flash.c.

7.1.2.7 uint32_t FLASH_GetSector (uint32_t Address)

Return the Flash sector bitmask of the address.

Parameters

<i>None.</i>	
--------------	--

Return values

<i>The</i>	Flash sector bitmask of the address
------------	-------------------------------------

Definition at line 1049 of file stm32f4xx_flash.c.

7.1.2.8 uint32_t FLASH_GetSectorNumber (uint32_t Address)

Return the Flash sector Number of the address.

Parameters

<i>None.</i>	
--------------	--

Return values

<i>The</i>	Flash sector Number of the address
------------	------------------------------------

Definition at line 1109 of file stm32f4xx_flash.c.

7.1.2.9 FLASH_Status FLASH_GetStatus (void)

Returns the FLASH Status.

Parameters

<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG← AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP← LETE.
--------------	--

Definition at line 981 of file stm32f4xx_flash.c.

7.1.2.10 void FLASH_InstructionCacheCmd (FunctionalState NewState)

Enables or disables the Instruction Cache feature.

Parameters

<i>NewState</i>	new state of the Instruction Cache. This parameter can be: ENABLE or DISABLE.
-----------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 205 of file stm32f4xx_flash.c.

7.1.2.11 void FLASH_InstructionCacheReset (void)

Resets the Instruction Cache.

Note

This function must be used only when the Instruction Cache is disabled.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 247 of file stm32f4xx_flash.c.

7.1.2.12 void FLASH_ITConfig (uint32_t FLASH_IT, FunctionalState NewState)

Enables or disables the specified FLASH interrupts.

Parameters

<i>FLASH_IT</i>	specifies the FLASH interrupt sources to be enabled or disabled. This parameter can be any combination of the following values: • FLASH_IT_ERR: FLASH Error Interrupt • FLASH_IT_EOP: FLASH end of operation Interrupt
-----------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 905 of file stm32f4xx_flash.c.

7.1.2.13 void FLASH_Lock (void)

Locks the FLASH control register access.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 317 of file stm32f4xx_flash.c.

7.1.2.14 void FLASH_OB_BORConfig (uint8_t *OB_BOR*)

Sets the BOR Level.

Parameters

<i>OB_BOR</i>	specifies the Option Bytes BOR Reset Level. This parameter can be one of the following values: • OB_BOR_LEVEL3: Supply voltage ranges from 2.7 to 3.6 V • OB_BOR_LEVEL2: Supply voltage ranges from 2.4 to 2.7 V • OB_BOR_LEVEL1: Supply voltage ranges from 2.1 to 2.4 V • OB_BOR_OFF: Supply voltage ranges from 1.62 to 2.1 V
---------------	---

Return values

<i>None</i>

Definition at line 791 of file stm32f4xx_flash.c.

7.1.2.15 uint8_t FLASH_OB_GetBOR (void)

Returns the FLASH BOR level.

Parameters

<i>None</i>

Return values

<i>The</i>	FLASH BOR level: • OB_BOR_LEVEL3: Supply voltage ranges from 2.7 to 3.6 V • OB_BOR_LEVEL2: Supply voltage ranges from 2.4 to 2.7 V • OB_BOR_LEVEL1: Supply voltage ranges from 2.1 to 2.4 V • OB_BOR_OFF : Supply voltage ranges from 1.62 to 2.1 V
------------	--

Definition at line 875 of file stm32f4xx_flash.c.

7.1.2.16 FlagStatus FLASH_OB_GetRDP (void)

Returns the FLASH Read Protection level.

Parameters

<i>None</i>

Return values

<i>FLASH</i>	ReadOut Protection Status: • SET, when OB_RDP_Level_1 or OB_RDP_Level_2 is set • RESET, when OB_RDP_Level_0 is set
--------------	---

Definition at line 851 of file stm32f4xx_flash.c.

7.1.2.17 `uint8_t FLASH_OB_GetUser ( void )`

Returns the FLASH User Option Bytes values.

Parameters

<i>None</i>	
-------------	--

Return values

<i>The</i>	FLASH User Option Bytes values: IWDG_SW(Bit0), RST_STOP(Bit1) and RS← T_STDBY(Bit2).
------------	---

Definition at line 827 of file stm32f4xx_flash.c.

7.1.2.18 `uint16_t FLASH_OB_GetWRP ( void )`

Returns the FLASH Write Protection Option Bytes value.

Parameters

<i>None</i>	
-------------	--

Return values

<i>The</i>	FLASH Write Protection Option Bytes value
------------	---

Definition at line 838 of file stm32f4xx_flash.c.

7.1.2.19 `FLASH_Status FLASH_OB_Launch ( void )`

Launch the option byte loading.

Parameters

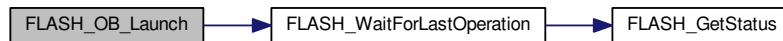
<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR← AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP← LETE.
--------------	---

Definition at line 808 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.20 void FLASH_OB_Lock (void)

Locks the FLASH Option Control Registers access.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 675 of file stm32f4xx_flash.c.

7.1.2.21 void FLASH_OB_RDPConfig (uint8_t OB_RDP)

Sets the read protection level.

Parameters

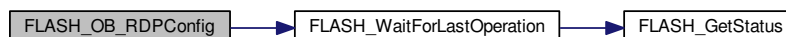
<i>OB_RDP</i>	specifies the read protection level. This parameter can be one of the following values: • OB_RDP_Level_0: No protection • OB_RDP_Level_1: Read protection of the memory • OB_RDP_Level_2: Full chip protection !!!Warning!!! When enabling OB_RDP level 2 it's no more possible to go back to level 1 or 0
---------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 726 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.22 void FLASH_OB_Unlock (void)

Unlocks the FLASH Option Control Registers access.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 660 of file stm32f4xx_flash.c.

7.1.2.23 void FLASH_OB_UserConfig (uint8_t *OB_IWDG*, uint8_t *OB_STOP*, uint8_t *OB_STDBY*)

Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.

Parameters

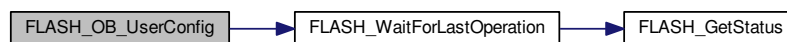
<i>OB_IWDG</i>	Selects the IWDG mode This parameter can be one of the following values: • <i>OB_IWDG_SW</i>: Software IWDG selected • <i>OB_IWDG_HW</i>: Hardware IWDG selected
<i>OB_STOP</i>	Reset event when entering STOP mode. This parameter can be one of the following values: • <i>OB_STOP_NoRST</i>: No reset generated when entering in STOP • <i>OB_STOP_RST</i>: Reset generated when entering in STOP
<i>OB_STDBY</i>	Reset event when entering Standby mode. This parameter can be one of the following values: • <i>OB_STDBY_NoRST</i>: No reset generated when entering in STANDBY • <i>OB_STDBY_RST</i>: Reset generated when entering in STANDBY

Return values

<i>None</i>	
-------------	--

Definition at line 758 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.24 void FLASH_OB_WRPConfig (uint32_t *OB_WRP*, FunctionalState *NewState*)

Enables or disables the write protection of the desired sectors.

Parameters

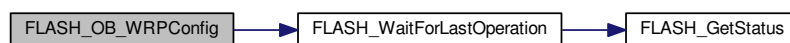
<i>OB_WRP</i>	specifies the sector(s) to be write protected or unprotected. This parameter can be one of the following values: • OB_WRP: A value between OB_WRP_Sector0 and OB_WRP_Sector11 • OB_WRP_Sector_All
<i>Newstate</i>	new state of the Write Protection. This parameter can be: ENABLE or DISABLE.

Return values

<i>None</i>

Definition at line 691 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.25 void FLASH_PrefetchBufferCmd (FunctionalState NewState)

Enables or disables the Prefetch Buffer.

Parameters

<i>NewState</i>	new state of the Prefetch Buffer. This parameter can be: ENABLE or DISABLE.
-----------------	---

Return values

<i>None</i>

Definition at line 183 of file stm32f4xx_flash.c.

7.1.2.26 FLASH_Status FLASH_ProgramByte (uint32_t Address, uint8_t Data)

Programs a byte (8-bit) at a specified address.

Note

This function can be used within all the device supply voltage ranges.

Parameters

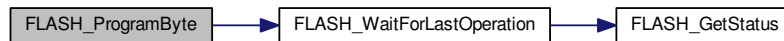
<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG← AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP← LETE.
--------------	--

Definition at line 575 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.27 FLASH_Status FLASH_ProgramDoubleWord (uint32_t Address, uint64_t Data)

Programs a double word (64-bit) at a specified address.

Note

This function must be used when the device voltage range is from 2.7V to 3.6V and an External Vpp is present.

Parameters

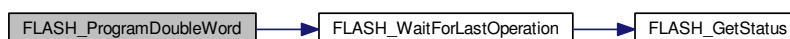
<i>Address</i>	specifies the address to be programmed.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG←AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP←LETE.
--------------	--

Definition at line 461 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.28 FLASH_Status FLASH_ProgramHalfWord (uint32_t Address, uint16_t Data)

Programs a half word (16-bit) at a specified address.

Note

This function must be used when the device voltage range is from 2.1V to 3.6V.

Parameters

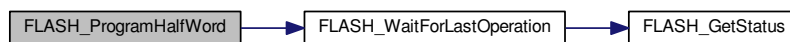
<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG←AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP←LETE.
--------------	--

Definition at line 537 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.29 FLASH_Status FLASH_ProgramWord (uint32_t Address, uint32_t Data)

Programs a word (32-bit) at a specified address.

Parameters

<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
----------------	--

Note

This function must be used when the device voltage range is from 2.7V to 3.6V.

Parameters

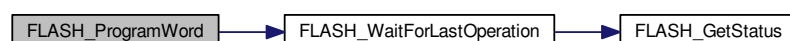
<i>Data</i>	specifies the data to be programmed.
-------------	--------------------------------------

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG←AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP←LETE.
--------------	--

Definition at line 499 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.1.2.30 void FLASH_SetLatency (uint32_t *FLASH_Latency*)

Sets the code latency value.

Parameters

<i>FLASH_Latency</i>	specifies the FLASH Latency value. This parameter can be one of the following values: - FLASH_Latency_0: FLASH Zero Latency cycle - FLASH_Latency_1: FLASH One Latency cycle - FLASH_Latency_2: FLASH Two Latency cycles - FLASH_Latency_3: FLASH Three Latency cycles - FLASH_Latency_4: FLASH Four Latency cycles - FLASH_Latency_5: FLASH Five Latency cycles - FLASH_Latency_6: FLASH Six Latency cycles - FLASH_Latency_7: FLASH Seven Latency cycles
----------------------	--

Return values

<i>None</i>

Definition at line 168 of file stm32f4xx_flash.c.

7.1.2.31 void FLASH_Unlock (void)

Unlocks the FLASH control register access.

Parameters

<i>None</i>

Return values

<i>None</i>

Definition at line 302 of file stm32f4xx_flash.c.

7.1.2.32 FLASH_Status FLASH_WaitForLastOperation (void)

Waits for a FLASH operation to complete.

Parameters

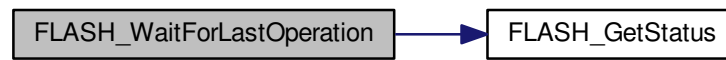
<i>None</i>

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG [↵] RAM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP [↵] LETE.
--------------	---

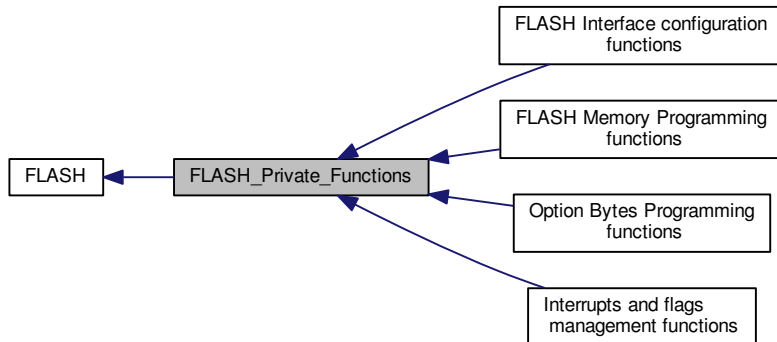
Definition at line 1026 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.2 FLASH_Private_Functions

Collaboration diagram for FLASH_Private_Functions:



Modules

- [FLASH Interface configuration functions](#)
FLASH Interface configuration functions.
- [FLASH Memory Programming functions](#)
FLASH Memory Programming functions.
- [Option Bytes Programming functions](#)
Option Bytes Programming functions.
- [Interrupts and flags management functions](#)
Interrupts and flags management functions.

7.2.1 Detailed Description

7.3 FLASH Interface configuration functions

FLASH Interface configuration functions.

Collaboration diagram for FLASH Interface configuration functions:



Functions

- void [FLASH_SetLatency](#) (uint32_t FLASH_Latency)
Sets the code latency value.
- void [FLASH_PrefetchBufferCmd](#) (FunctionalState NewState)
Enables or disables the Prefetch Buffer.
- void [FLASH_InstructionCacheCmd](#) (FunctionalState NewState)
Enables or disables the Instruction Cache feature.
- void [FLASH_DataCacheCmd](#) (FunctionalState NewState)
Enables or disables the Data Cache feature.
- void [FLASH_InstructionCacheReset](#) (void)
Resets the Instruction Cache.
- void [FLASH_DataCacheReset](#) (void)
Resets the Data Cache.

7.3.1 Detailed Description

FLASH Interface configuration functions.

```

=====
FLASH Interface configuration functions
=====
  
```

```

This group includes the following functions:
- void FLASH_SetLatency(uint32_t FLASH_Latency)
  To correctly read data from FLASH memory, the number of wait states (LATENCY)
  must be correctly programmed according to the frequency of the CPU clock
  (HCLK) and the supply voltage of the device.
  
```

Latency	HCLK clock frequency (MHz)			
	voltage range 2.7 V - 3.6 V	voltage range 2.4 V - 2.7 V	voltage range 2.1 V - 2.4 V	voltage range 1.8 V - 2.1 V
0WS(1CPU cycle)	0 < HCLK <= 30	0 < HCLK <= 24	0 < HCLK <= 18	0 < HCLK <= 16
1WS(2CPU cycle)	30 < HCLK <= 60	24 < HCLK <= 48	18 < HCLK <= 36	16 < HCLK <= 32
2WS(3CPU cycle)	60 < HCLK <= 90	48 < HCLK <= 72	36 < HCLK <= 54	32 < HCLK <= 48
3WS(4CPU cycle)	90 < HCLK <= 120	72 < HCLK <= 96	54 < HCLK <= 72	48 < HCLK <= 64
4WS(5CPU cycle)	120 < HCLK <= 150	96 < HCLK <= 120	72 < HCLK <= 90	64 < HCLK <= 80

5WS (6CPU cycle)	120< HCLK <= 168	120< HCLK <= 144	90 < HCLK <= 108	80 < HCLK <= 96	
-----	-----	-----	-----	-----	
6WS (7CPU cycle)	NA	144< HCLK <= 168	108 < HCLK <= 120	96 < HCLK <= 112	
-----	-----	-----	-----	-----	
7WS (8CPU cycle)	NA	NA	120 < HCLK <= 138	112 < HCLK <= 120	
*****	*****	*****	*****	*****	*****
	voltage range	voltage range	voltage range	voltage range	voltage range 2.7 V -
	2.7 V - 3.6 V	2.4 V - 2.7 V	2.1 V - 2.4 V	1.8 V - 2.1 V	with External Vpp = 9
-----	-----	-----	-----	-----	
Max Parallelism	x32	x16	x8	x4	
-----	-----	-----	-----	-----	
PSIZE[1:0]	10	01	00	11	
-----	-----	-----	-----	-----	

@note When VOS bit (in PWR_CR register) is reset to '0, the maximum value of HCLK is 144 MHz.
You can use PWR_MainRegulatorModeConfig() function to set or reset this bit.

- void FLASH_PrefetchBufferCmd(FunctionalState NewState)
- void FLASH_InstructionCacheCmd(FunctionalState NewState)
- void FLASH_DataCacheCmd(FunctionalState NewState)
- void FLASH_InstructionCacheReset(void)
- void FLASH_DataCacheReset(void)

The unlock sequence is not needed for these functions.

7.3.2 Function Documentation

7.3.2.1 void FLASH_DataCacheCmd (FunctionalState NewState)

Enables or disables the Data Cache feature.

Parameters

<i>NewState</i>	new state of the Data Cache. This parameter can be: ENABLE or DISABLE.
-----------------	--

Return values

<i>None</i>

Definition at line 226 of file stm32f4xx_flash.c.

7.3.2.2 void FLASH_DataCacheReset (void)

Resets the Data Cache.

Note

This function must be used only when the Data Cache is disabled.

Parameters

<i>None</i>

Return values

<i>None</i>

Definition at line 258 of file stm32f4xx_flash.c.

7.3.2.3 void FLASH_InstructionCacheCmd (FunctionalState NewState)

Enables or disables the Instruction Cache feature.

Parameters

<i>NewState</i>	new state of the Instruction Cache. This parameter can be: ENABLE or DISABLE.
-----------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 205 of file stm32f4xx_flash.c.

7.3.2.4 void FLASH_InstructionCacheReset (void)

Resets the Instruction Cache.

Note

This function must be used only when the Instruction Cache is disabled.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 247 of file stm32f4xx_flash.c.

7.3.2.5 void FLASH_PrefetchBufferCmd (FunctionalState *NewState*)

Enables or disables the Prefetch Buffer.

Parameters

<i>NewState</i>	new state of the Prefetch Buffer. This parameter can be: ENABLE or DISABLE.
-----------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 183 of file stm32f4xx_flash.c.

7.3.2.6 void FLASH_SetLatency (uint32_t *FLASH_Latency*)

Sets the code latency value.

Parameters

<i>FLASH_Latency</i>	specifies the FLASH Latency value. This parameter can be one of the following values: • FLASH_Latency_0: FLASH Zero Latency cycle • FLASH_Latency_1: FLASH One Latency cycle • FLASH_Latency_2: FLASH Two Latency cycles • FLASH_Latency_3: FLASH Three Latency cycles • FLASH_Latency_4: FLASH Four Latency cycles • FLASH_Latency_5: FLASH Five Latency cycles • FLASH_Latency_6: FLASH Six Latency cycles • FLASH_Latency_7: FLASH Seven Latency cycles
----------------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 168 of file stm32f4xx_flash.c.

7.4 FLASH Memory Programming functions

FLASH Memory Programming functions.

Collaboration diagram for FLASH Memory Programming functions:



Functions

- void [FLASH_Unlock](#) (void)
Unlocks the FLASH control register access.
- void [FLASH_Lock](#) (void)
Locks the FLASH control register access.
- [FLASH_Status FLASH_EraseSector](#) (uint32_t FLASH_Sector, uint8_t VoltageRange)
Erases a specified FLASH Sector.
- [FLASH_Status FLASH_EraseAllSectors](#) (uint8_t VoltageRange)
Erases all FLASH Sectors.
- [FLASH_Status FLASH_ProgramDoubleWord](#) (uint32_t Address, uint64_t Data)
Programs a double word (64-bit) at a specified address.
- [FLASH_Status FLASH_ProgramWord](#) (uint32_t Address, uint32_t Data)
Programs a word (32-bit) at a specified address.
- [FLASH_Status FLASH_ProgramHalfWord](#) (uint32_t Address, uint16_t Data)
Programs a half word (16-bit) at a specified address.
- [FLASH_Status FLASH_ProgramByte](#) (uint32_t Address, uint8_t Data)
Programs a byte (8-bit) at a specified address.

7.4.1 Detailed Description

FLASH Memory Programming functions.

```

=====
FLASH Memory Programming functions
=====
  
```

This group includes the following functions:

- void `FLASH_Unlock(void)`
- void `FLASH_Lock(void)`
- `FLASH_Status FLASH_EraseSector(uint32_t FLASH_Sector, uint8_t VoltageRange)`
- `FLASH_Status FLASH_EraseAllSectors(uint8_t VoltageRange)`
- `FLASH_Status FLASH_ProgramDoubleWord(uint32_t Address, uint64_t Data)`
- `FLASH_Status FLASH_ProgramWord(uint32_t Address, uint32_t Data)`
- `FLASH_Status FLASH_ProgramHalfWord(uint32_t Address, uint16_t Data)`
- `FLASH_Status FLASH_ProgramByte(uint32_t Address, uint8_t Data)`

Any operation of erase or program should follow these steps:

1. Call the `FLASH_Unlock()` function to enable the FLASH control register access
2. Call the desired function to erase sector(s) or program data
3. Call the `FLASH_Lock()` function to disable the FLASH control register access (recommended to protect the FLASH memory against possible unwanted operation)

7.4.2 Function Documentation

7.4.2.1 FLASH_Status FLASH_EraseAllSectors (uint8_t VoltageRange)

Erases all FLASH Sectors.

Parameters

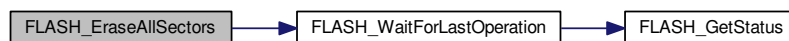
<i>VoltageRange</i>	The device voltage range which defines the erase parallelism. This parameter can be one of the following values: • VoltageRange_1: when the device voltage range is 1.8V to 2.1V, the operation will be done by byte (8-bit) • VoltageRange_2: when the device voltage range is 2.1V to 2.7V, the operation will be done by half word (16-bit) • VoltageRange_3: when the device voltage range is 2.7V to 3.6V, the operation will be done by word (32-bit) • VoltageRange_4: when the device voltage range is 2.7V to 3.6V + External Vpp, the operation will be done by double word (64-bit)
---------------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↵AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↵LETE.
--------------	--

Definition at line 408 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.2 FLASH_Status FLASH_EraseSector (uint32_t FLASH_Sector, uint8_t VoltageRange)

Erases a specified FLASH Sector.

Parameters

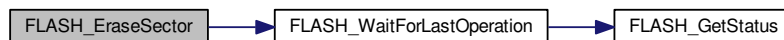
<i>FLASH_Sector</i>	The Sector number to be erased. This parameter can be a value between FLASH_Sector_0 and FLASH_Sector_11
<i>VoltageRange</i>	The device voltage range which defines the erase parallelism. This parameter can be one of the following values: - VoltageRange_1: when the device voltage range is 1.8V to 2.1V, the operation will be done by byte (8-bit) - VoltageRange_2: when the device voltage range is 2.1V to 2.7V, the operation will be done by half word (16-bit) - VoltageRange_3: when the device voltage range is 2.7V to 3.6V, the operation will be done by word (32-bit) - VoltageRange_4: when the device voltage range is 2.7V to 3.6V + External Vpp, the operation will be done by double word (64-bit)

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↵AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↵LETE.
--------------	---

Definition at line 343 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.3 void FLASH_Lock (void)

Locks the FLASH control register access.

Parameters

<i>None</i>

Return values

<i>None</i>

Definition at line 317 of file stm32f4xx_flash.c.

7.4.2.4 FLASH_Status FLASH_ProgramByte (uint32_t Address, uint8_t Data)

Programs a byte (8-bit) at a specified address.

Note

This function can be used within all the device supply voltage ranges.

Parameters

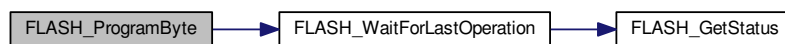
<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG←AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP←LETE.
--------------	--

Definition at line 575 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.5 FLASH_Status FLASH_ProgramDoubleWord (uint32_t Address, uint64_t Data)

Programs a double word (64-bit) at a specified address.

Note

This function must be used when the device voltage range is from 2.7V to 3.6V and an External Vpp is present.

Parameters

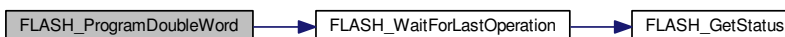
<i>Address</i>	specifies the address to be programmed.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG←AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP←LETE.
--------------	--

Definition at line 461 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.6 FLASH_Status FLASH_ProgramHalfWord (uint32_t Address, uint16_t Data)

Programs a half word (16-bit) at a specified address.

Note

This function must be used when the device voltage range is from 2.1V to 3.6V.

Parameters

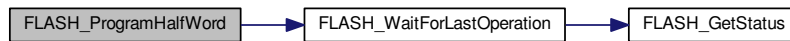
<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
<i>Data</i>	specifies the data to be programmed.

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↔AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↔LETE.
--------------	---

Definition at line 537 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.7 FLASH_Status FLASH_ProgramWord (uint32_t Address, uint32_t Data)

Programs a word (32-bit) at a specified address.

Parameters

<i>Address</i>	specifies the address to be programmed. This parameter can be any address in Program memory zone or in OTP zone.
----------------	--

Note

This function must be used when the device voltage range is from 2.7V to 3.6V.

Parameters

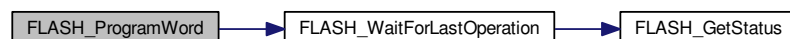
<i>Data</i>	specifies the data to be programmed.
-------------	--------------------------------------

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↔AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↔LETE.
--------------	---

Definition at line 499 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.4.2.8 void FLASH_Unlock (void)

Unlocks the FLASH control register access.

Parameters

<i>None</i>	
-------------	--

Return values

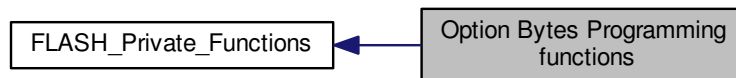
<i>None</i>	
-------------	--

Definition at line 302 of file stm32f4xx_flash.c.

7.5 Option Bytes Programming functions

Option Bytes Programming functions.

Collaboration diagram for Option Bytes Programming functions:



Functions

- void [FLASH_OB_Unlock](#) (void)
Unlocks the FLASH Option Control Registers access.
- void [FLASH_OB_Lock](#) (void)
Locks the FLASH Option Control Registers access.
- void [FLASH_OB_WRPConfig](#) (uint32_t OB_WRP, FunctionalState NewState)
Enables or disables the write protection of the desired sectors.
- void [FLASH_OB_RDPConfig](#) (uint8_t OB_RDP)
Sets the read protection level.
- void [FLASH_OB_UserConfig](#) (uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)
Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.
- void [FLASH_OB_BORConfig](#) (uint8_t OB_BOR)
Sets the BOR Level.
- [FLASH_Status](#) [FLASH_OB_Launch](#) (void)
Launch the option byte loading.
- uint8_t [FLASH_OB_GetUser](#) (void)
Returns the FLASH User Option Bytes values.
- uint16_t [FLASH_OB_GetWRP](#) (void)
Returns the FLASH Write Protection Option Bytes value.
- FlagStatus [FLASH_OB_GetRDP](#) (void)
Returns the FLASH Read Protection level.
- uint8_t [FLASH_OB_GetBOR](#) (void)
Returns the FLASH BOR level.

7.5.1 Detailed Description

Option Bytes Programming functions.

```

=====
                        Option Bytes Programming functions
=====

This group includes the following functions:
- void FLASH_OB_Unlock(void)
- void FLASH_OB_Lock(void)
- void FLASH_OB_WRPConfig(uint32_t OB_WRP, FunctionalState NewState)
- void FLASH_OB_RDPConfig(uint8_t OB_RDP)
- void FLASH_OB_UserConfig(uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)
  
```

```

- void FLASH_OB_BORConfig(uint8_t OB_BOR)
- FLASH_Status FLASH_ProgramOTP(uint32_t Address, uint32_t Data)
- FLASH_Status FLASH_OB_Launch(void)
- uint32_t FLASH_OB_GetUser(void)
- uint8_t FLASH_OB_GetWRP(void)
- uint8_t FLASH_OB_GetRDP(void)
- uint8_t FLASH_OB_GetBOR(void)

```

Any operation of erase or program should follow these steps:

1. Call the FLASH_OB_Unlock() function to enable the FLASH option control register access
 2. Call one or several functions to program the desired Option Bytes:
 - void FLASH_OB_WRPConfig(uint32_t OB_WRP, FunctionalState NewState) => to Enable/Disable the desired sector write protection
 - void FLASH_OB_RDPConfig(uint8_t OB_RDP) => to set the desired read Protection Level
 - void FLASH_OB_UserConfig(uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY) => to configure the user Option Bytes.
 - void FLASH_OB_BORConfig(uint8_t OB_BOR) => to set the BOR Level
 3. Once all needed Option Bytes to be programmed are correctly written, call the FLASH_OB_Launch() function to launch the Option Bytes programming process.
- @note When changing the IWDG mode from HW to SW or from SW to HW, a system reset is needed to make the change effective.
4. Call the FLASH_OB_Lock() function to disable the FLASH option control register access (recommended to protect the Option Bytes against possible unwanted operations)

7.5.2 Function Documentation

7.5.2.1 void FLASH_OB_BORConfig (uint8_t OB_BOR)

Sets the BOR Level.

Parameters

OB_BOR	specifies the Option Bytes BOR Reset Level. This parameter can be one of the following values: • OB_BOR_LEVEL3: Supply voltage ranges from 2.7 to 3.6 V • OB_BOR_LEVEL2: Supply voltage ranges from 2.4 to 2.7 V • OB_BOR_LEVEL1: Supply voltage ranges from 2.1 to 2.4 V • OB_BOR_OFF: Supply voltage ranges from 1.62 to 2.1 V
---------------	---

Return values

<i>None</i>

Definition at line 791 of file stm32f4xx_flash.c.

7.5.2.2 uint8_t FLASH_OB_GetBOR (void)

Returns the FLASH BOR level.

Parameters

<i>None</i>

Return values

<i>The</i>	FLASH BOR level: • OB_BOR_LEVEL3: Supply voltage ranges from 2.7 to 3.6 V • OB_BOR_LEVEL2: Supply voltage ranges from 2.4 to 2.7 V • OB_BOR_LEVEL1: Supply voltage ranges from 2.1 to 2.4 V • OB_BOR_OFF : Supply voltage ranges from 1.62 to 2.1 V
------------	--

Definition at line 875 of file stm32f4xx_flash.c.

7.5.2.3 FlagStatus FLASH_OB_GetRDP (void)

Returns the FLASH Read Protection level.

Parameters

<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	ReadOut Protection Status: • SET, when OB_RDP_Level_1 or OB_RDP_Level_2 is set • RESET, when OB_RDP_Level_0 is set
--------------	---

Definition at line 851 of file stm32f4xx_flash.c.

7.5.2.4 uint8_t FLASH_OB_GetUser (void)

Returns the FLASH User Option Bytes values.

Parameters

<i>None</i>	
-------------	--

Return values

<i>The</i>	FLASH User Option Bytes values: IWDG_SW(Bit0), RST_STOP(Bit1) and RS← T_STDBY(Bit2).
------------	---

Definition at line 827 of file stm32f4xx_flash.c.

7.5.2.5 uint16_t FLASH_OB_GetWRP (void)

Returns the FLASH Write Protection Option Bytes value.

Parameters

<i>None</i>	
-------------	--

Return values

<i>The</i>	FLASH Write Protection Option Bytes value
------------	---

Definition at line 838 of file stm32f4xx_flash.c.

7.5.2.6 FLASH_Status FLASH_OB_Launch (void)

Launch the option byte loading.

Parameters

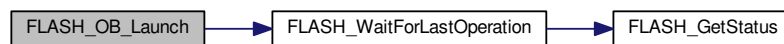
<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROGR↔AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP↔LETE.
--------------	---

Definition at line 808 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.5.2.7 void FLASH_OB_Lock (void)

Locks the FLASH Option Control Registers access.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

Definition at line 675 of file stm32f4xx_flash.c.

7.5.2.8 void FLASH_OB_RDPConfig (uint8_t OB_RDP)

Sets the read protection level.

Parameters

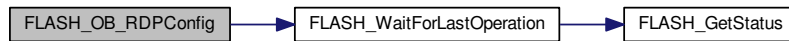
<i>OB_RDP</i>	specifies the read protection level. This parameter can be one of the following values: • OB_RDP_Level_0: No protection • OB_RDP_Level_1: Read protection of the memory • OB_RDP_Level_2: Full chip protection !!!Warning!!! When enabling OB_RDP level 2 it's no more possible to go back to level 1 or 0
---------------	--

Return values

None	
------	--

Definition at line 726 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.5.2.9 void FLASH_OB_Unlock (void)

Unlocks the FLASH Option Control Registers access.

Parameters

None	
------	--

Return values

None	
------	--

Definition at line 660 of file stm32f4xx_flash.c.

7.5.2.10 void FLASH_OB_UserConfig (uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)

Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.

Parameters

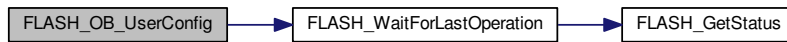
<i>OB_IWDG</i>	Selects the IWDG mode This parameter can be one of the following values: • OB_IWDG_SW: Software IWDG selected • OB_IWDG_HW: Hardware IWDG selected
<i>OB_STOP</i>	Reset event when entering STOP mode. This parameter can be one of the following values: • OB_STOP_NoRST: No reset generated when entering in STOP • OB_STOP_RST: Reset generated when entering in STOP
<i>OB_STDBY</i>	Reset event when entering Standby mode. This parameter can be one of the following values: • OB_STDBY_NoRST: No reset generated when entering in STANDBY • OB_STDBY_RST: Reset generated when entering in STANDBY

Return values

None

Definition at line 758 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.5.2.11 void FLASH_OB_WRPConfig (uint32_t OB_WRP, FunctionalState NewState)

Enables or disables the write protection of the desired sectors.

Parameters

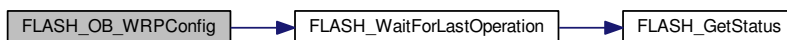
<i>OB_WRP</i>	specifies the sector(s) to be write protected or unprotected. This parameter can be one of the following values: • OB_WRP: A value between OB_WRP_Sector0 and OB_WRP_Sector11 • OB_WRP_Sector_All
<i>Newstate</i>	new state of the Write Protection. This parameter can be: ENABLE or DISABLE.

Return values

None

Definition at line 691 of file stm32f4xx_flash.c.

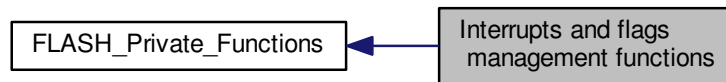
Here is the call graph for this function:



7.6 Interrupts and flags management functions

Interrupts and flags management functions.

Collaboration diagram for Interrupts and flags management functions:



Functions

- void [FLASH_ITConfig](#) (uint32_t FLASH_IT, FunctionalState NewState)
Enables or disables the specified FLASH interrupts.
- FlagStatus [FLASH_GetFlagStatus](#) (uint32_t FLASH_FLAG)
Checks whether the specified FLASH flag is set or not.
- void [FLASH_ClearFlag](#) (uint32_t FLASH_FLAG)
Clears the FLASH's pending flags.
- FLASH_Status [FLASH_GetStatus](#) (void)
Returns the FLASH Status.
- FLASH_Status [FLASH_WaitForLastOperation](#) (void)
Waits for a FLASH operation to complete.
- uint32_t [FLASH_GetSector](#) (uint32_t Address)
Return the Flash sector bitmask of the address.
- uint32_t [FLASH_GetSectorNumber](#) (uint32_t Address)
Return the Flash sector Number of the address.

7.6.1 Detailed Description

Interrupts and flags management functions.

```

=====
Interrupts and flags management functions
=====
  
```

7.6.2 Function Documentation

7.6.2.1 void FLASH_ClearFlag (uint32_t FLASH_FLAG)

Clears the FLASH's pending flags.

Parameters

<i>FLASH_FLAG</i>	specifies the FLASH flags to clear. This parameter can be any combination of the following values: • FLASH_FLAG_EOP: FLASH End of Operation flag • FLASH_FLAG_OPERR: FLASH operation Error flag • FLASH_FLAG_WRPERR: FLASH Write protected error flag • FLASH_FLAG_PGAERR: FLASH Programming Alignment error flag • FLASH_FLAG_PGPERR: FLASH Programming Parallelism error flag • FLASH_FLAG_PGSERR: FLASH Programming Sequence error flag
-------------------	--

Return values

<i>None</i>

Definition at line 966 of file stm32f4xx_flash.c.

7.6.2.2 FlagStatus FLASH_GetFlagStatus (uint32_t FLASH_FLAG)

Checks whether the specified FLASH flag is set or not.

Parameters

<i>FLASH_FLAG</i>	specifies the FLASH flag to check. This parameter can be one of the following values: • FLASH_FLAG_EOP: FLASH End of Operation flag • FLASH_FLAG_OPERR: FLASH operation Error flag • FLASH_FLAG_WRPERR: FLASH Write protected error flag • FLASH_FLAG_PGAERR: FLASH Programming Alignment error flag • FLASH_FLAG_PGPERR: FLASH Programming Parallelism error flag • FLASH_FLAG_PGSERR: FLASH Programming Sequence error flag • FLASH_FLAG_BSY: FLASH Busy flag
-------------------	--

Return values

<i>The</i>	new state of FLASH_FLAG (SET or RESET).
------------	---

Definition at line 936 of file stm32f4xx_flash.c.

7.6.2.3 uint32_t FLASH_GetSector (uint32_t Address)

Return the Flash sector bitmask of the address.

Parameters

<i>None.</i>	
--------------	--

Return values

<i>The</i>	Flash sector bitmask of the address
------------	-------------------------------------

Definition at line 1049 of file stm32f4xx_flash.c.

7.6.2.4 uint32_t FLASH_GetSectorNumber (uint32_t Address)

Return the Flash sector Number of the address.

Parameters

<i>None.</i>	
--------------	--

Return values

<i>The</i>	Flash sector Number of the address
------------	------------------------------------

Definition at line 1109 of file stm32f4xx_flash.c.

7.6.2.5 FLASH_Status FLASH_GetStatus (void)

Returns the FLASH Status.

Parameters

<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG← AM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP← LETE.
--------------	--

Definition at line 981 of file stm32f4xx_flash.c.

7.6.2.6 void FLASH_ITConfig (uint32_t FLASH_IT, FunctionalState NewState)

Enables or disables the specified FLASH interrupts.

Parameters

<i>FLASH_IT</i>	specifies the FLASH interrupt sources to be enabled or disabled. This parameter can be any combination of the following values: • FLASH_IT_ERR: FLASH Error Interrupt • FLASH_IT_EOP: FLASH end of operation Interrupt
-----------------	---

Return values

<i>None</i>	
-------------	--

Definition at line 905 of file stm32f4xx_flash.c.

7.6.2.7 FLASH_Status FLASH_WaitForLastOperation (void)

Waits for a FLASH operation to complete.

Parameters

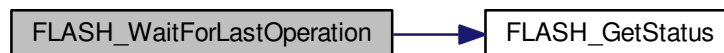
<i>None</i>	
-------------	--

Return values

<i>FLASH</i>	Status: The returned value can be: FLASH_BUSY, FLASH_ERROR_PROG [↵] RAM, FLASH_ERROR_WRP, FLASH_ERROR_OPERATION or FLASH_COMP [↵] LETE.
--------------	---

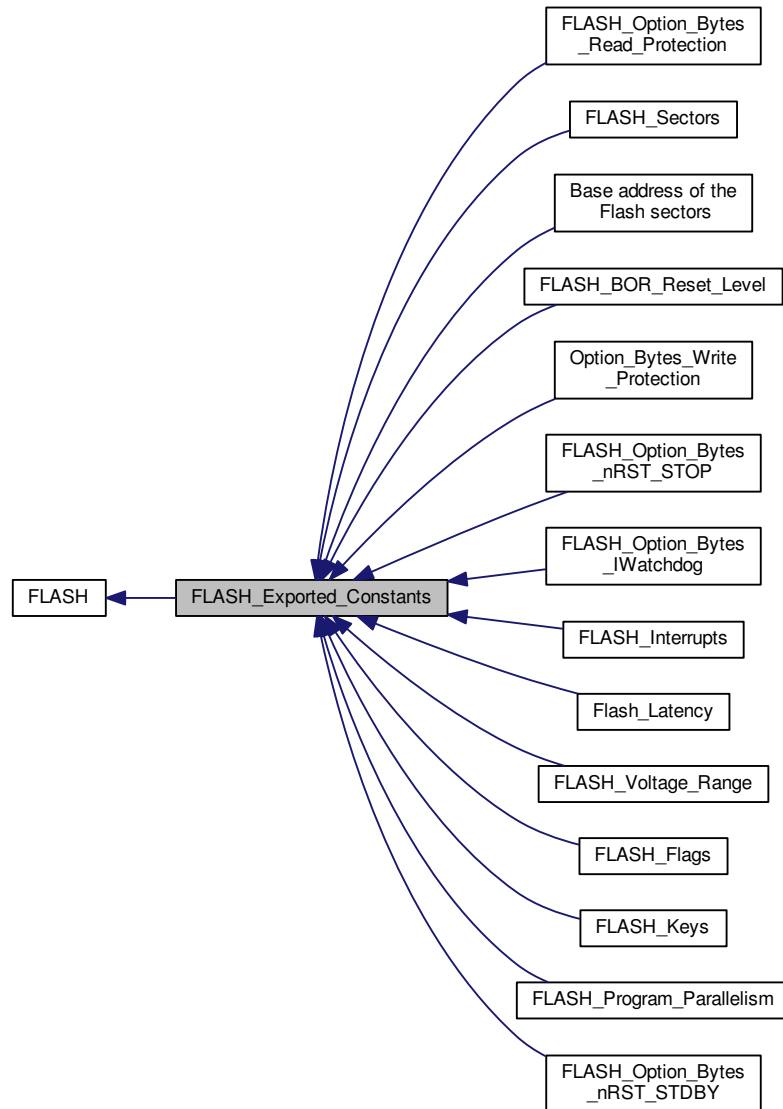
Definition at line 1026 of file stm32f4xx_flash.c.

Here is the call graph for this function:



7.7 FLASH_Exported_Constants

Collaboration diagram for FLASH_Exported_Constants:



Modules

- [Flash_Latency](#)
- [FLASH_Voltage_Range](#)
- [FLASH_Sectors](#)
- [Base address of the Flash sectors](#)
- [Option_Bytes_Write_Protection](#)
- [FLASH_Option_Bytes_Read_Protection](#)
- [FLASH_Option_Bytes_IWatchdog](#)
- [FLASH_Option_Bytes_nRST_STOP](#)
- [FLASH_Option_Bytes_nRST_STDBY](#)

- [FLASH_BOR_Reset_Level](#)
- [FLASH_Interrupts](#)
- [FLASH_Flags](#)
- [FLASH_Program_Parallelism](#)
- [FLASH_Keys](#)

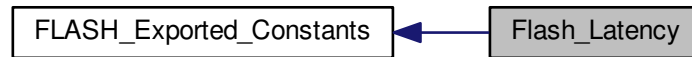
Macros

- `#define ACR\_BYTE0\_ADDRESS ((uint32_t)0x40023C00)`
ACR register byte 0 (Bits[8:0]) base address.
- `#define OPTCR\_BYTE0\_ADDRESS ((uint32_t)0x40023C14)`
OPTCR register byte 3 (Bits[24:16]) base address.
- `#define OPTCR_BYTE1_ADDRESS ((uint32_t)0x40023C15)`
- `#define OPTCR_BYTE2_ADDRESS ((uint32_t)0x40023C16)`

7.7.1 Detailed Description

7.8 Flash_Latency

Collaboration diagram for Flash_Latency:



Macros

- `#define FLASH_Latency_0 ((uint8_t)0x0000)`
- `#define FLASH_Latency_1 ((uint8_t)0x0001)`
- `#define FLASH_Latency_2 ((uint8_t)0x0002)`
- `#define FLASH_Latency_3 ((uint8_t)0x0003)`
- `#define FLASH_Latency_4 ((uint8_t)0x0004)`
- `#define FLASH_Latency_5 ((uint8_t)0x0005)`
- `#define FLASH_Latency_6 ((uint8_t)0x0006)`
- `#define FLASH_Latency_7 ((uint8_t)0x0007)`
- `#define IS_FLASH_LATENCY(LATENCY)`

7.8.1 Detailed Description

7.8.2 Macro Definition Documentation

7.8.2.1 `#define FLASH_Latency_0 ((uint8_t)0x0000)`

FLASH Zero Latency cycle

Definition at line 67 of file stm32f4xx_flash.h.

7.8.2.2 `#define FLASH_Latency_1 ((uint8_t)0x0001)`

FLASH One Latency cycle

Definition at line 68 of file stm32f4xx_flash.h.

7.8.2.3 `#define FLASH_Latency_2 ((uint8_t)0x0002)`

FLASH Two Latency cycles

Definition at line 69 of file stm32f4xx_flash.h.

7.8.2.4 `#define FLASH_Latency_3 ((uint8_t)0x0003)`

FLASH Three Latency cycles

Definition at line 70 of file stm32f4xx_flash.h.

7.8.2.5 #define FLASH_Latency_4 ((uint8_t)0x0004)

FLASH Four Latency cycles

Definition at line 71 of file stm32f4xx_flash.h.

7.8.2.6 #define FLASH_Latency_5 ((uint8_t)0x0005)

FLASH Five Latency cycles

Definition at line 72 of file stm32f4xx_flash.h.

7.8.2.7 #define FLASH_Latency_6 ((uint8_t)0x0006)

FLASH Six Latency cycles

Definition at line 73 of file stm32f4xx_flash.h.

7.8.2.8 #define FLASH_Latency_7 ((uint8_t)0x0007)

FLASH Seven Latency cycles

Definition at line 74 of file stm32f4xx_flash.h.

7.8.2.9 #define IS_FLASH_LATENCY(LATENCY)**Value:**

```

(( (LATENCY) == FLASH_Latency_0) || \
  ((LATENCY) == FLASH_Latency_1) || \
  ((LATENCY) == FLASH_Latency_2) || \
  ((LATENCY) == FLASH_Latency_3) || \
  ((LATENCY) == FLASH_Latency_4) || \
  ((LATENCY) == FLASH_Latency_5) || \
  ((LATENCY) == FLASH_Latency_6) || \
  ((LATENCY) == FLASH_Latency_7))

```

Definition at line 76 of file stm32f4xx_flash.h.

7.9 FLASH_Voltage_Range

Collaboration diagram for FLASH_Voltage_Range:



Macros

- #define VoltageRange_1 ((uint8_t)0x00)
- #define VoltageRange_2 ((uint8_t)0x01)
- #define VoltageRange_3 ((uint8_t)0x02)
- #define VoltageRange_4 ((uint8_t)0x03)
- #define IS_VOLTAGERANGE(RANGE)

7.9.1 Detailed Description

7.9.2 Macro Definition Documentation

7.9.2.1 #define IS_VOLTAGERANGE(RANGE)

Value:

```

(( (RANGE) == VoltageRange_1) || \
  ((RANGE) == VoltageRange_2) || \
  ((RANGE) == VoltageRange_3) || \
  ((RANGE) == VoltageRange_4))

```

Definition at line 96 of file stm32f4xx_flash.h.

7.9.2.2 #define VoltageRange_1 ((uint8_t)0x00)

Device operating range: 1.8V to 2.1V

Definition at line 91 of file stm32f4xx_flash.h.

7.9.2.3 #define VoltageRange_2 ((uint8_t)0x01)

Device operating range: 2.1V to 2.7V

Definition at line 92 of file stm32f4xx_flash.h.

7.9.2.4 #define VoltageRange_3 ((uint8_t)0x02)

Device operating range: 2.7V to 3.6V

Definition at line 93 of file stm32f4xx_flash.h.

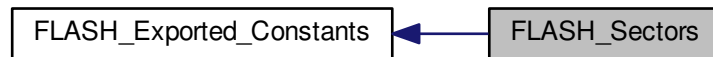
7.9.2.5 #define VoltageRange_4 ((uint8_t)0x03)

Device operating range: 2.7V to 3.6V + External Vpp

Definition at line 94 of file stm32f4xx_flash.h.

7.10 FLASH_Sectors

Collaboration diagram for FLASH_Sectors:



Macros

- `#define FLASH_Sector_0 ((uint16_t)0x0000)`
- `#define FLASH_Sector_1 ((uint16_t)0x0008)`
- `#define FLASH_Sector_2 ((uint16_t)0x0010)`
- `#define FLASH_Sector_3 ((uint16_t)0x0018)`
- `#define FLASH_Sector_4 ((uint16_t)0x0020)`
- `#define FLASH_Sector_5 ((uint16_t)0x0028)`
- `#define FLASH_Sector_6 ((uint16_t)0x0030)`
- `#define FLASH_Sector_7 ((uint16_t)0x0038)`
- `#define FLASH_Sector_8 ((uint16_t)0x0040)`
- `#define FLASH_Sector_9 ((uint16_t)0x0048)`
- `#define FLASH_Sector_10 ((uint16_t)0x0050)`
- `#define FLASH_Sector_11 ((uint16_t)0x0058)`
- `#define IS_FLASH_SECTOR(SECTOR)`
- `#define IS_FLASH_ADDRESS(ADDRESS)`

7.10.1 Detailed Description

7.10.2 Macro Definition Documentation

7.10.2.1 `#define FLASH_Sector_0 ((uint16_t)0x0000)`

Sector Number 0

Definition at line 107 of file stm32f4xx_flash.h.

7.10.2.2 `#define FLASH_Sector_1 ((uint16_t)0x0008)`

Sector Number 1

Definition at line 108 of file stm32f4xx_flash.h.

7.10.2.3 `#define FLASH_Sector_10 ((uint16_t)0x0050)`

Sector Number 10

Definition at line 117 of file stm32f4xx_flash.h.

7.10.2.4 #define FLASH_Sector_11 ((uint16_t)0x0058)

Sector Number 11

Definition at line 118 of file stm32f4xx_flash.h.

7.10.2.5 #define FLASH_Sector_2 ((uint16_t)0x0010)

Sector Number 2

Definition at line 109 of file stm32f4xx_flash.h.

7.10.2.6 #define FLASH_Sector_3 ((uint16_t)0x0018)

Sector Number 3

Definition at line 110 of file stm32f4xx_flash.h.

7.10.2.7 #define FLASH_Sector_4 ((uint16_t)0x0020)

Sector Number 4

Definition at line 111 of file stm32f4xx_flash.h.

7.10.2.8 #define FLASH_Sector_5 ((uint16_t)0x0028)

Sector Number 5

Definition at line 112 of file stm32f4xx_flash.h.

7.10.2.9 #define FLASH_Sector_6 ((uint16_t)0x0030)

Sector Number 6

Definition at line 113 of file stm32f4xx_flash.h.

7.10.2.10 #define FLASH_Sector_7 ((uint16_t)0x0038)

Sector Number 7

Definition at line 114 of file stm32f4xx_flash.h.

7.10.2.11 #define FLASH_Sector_8 ((uint16_t)0x0040)

Sector Number 8

Definition at line 115 of file stm32f4xx_flash.h.

7.10.2.12 #define FLASH_Sector_9 ((uint16_t)0x0048)

Sector Number 9

Definition at line 116 of file stm32f4xx_flash.h.

7.10.2.13 #define IS_FLASH_ADDRESS(ADDRESS)

Value:

```
(( (ADDRESS) >= 0x08000000) && ((ADDRESS) < 0x080FFFFF)) ||\
(( (ADDRESS) >= 0x1FFF7800) && ((ADDRESS) < 0x1FFF7A0F))
```

Definition at line 125 of file stm32f4xx_flash.h.

7.10.2.14 #define IS_FLASH_SECTOR(SECTOR)

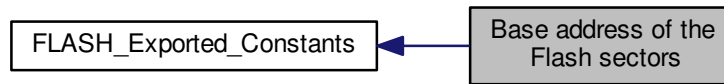
Value:

```
(( (SECTOR) == FLASH_Sector_0) || ((SECTOR) == FLASH_Sector_1) ||\
FLASH_Sector_3) ||\
FLASH_Sector_5) ||\
FLASH_Sector_7) ||\
FLASH_Sector_9) ||\
FLASH_Sector_11)) ((SECTOR) == FLASH_Sector_2) || ((SECTOR) ==
FLASH_Sector_4) || ((SECTOR) ==
FLASH_Sector_6) || ((SECTOR) ==
FLASH_Sector_8) || ((SECTOR) ==
FLASH_Sector_10) || ((SECTOR) ==
```

Definition at line 119 of file stm32f4xx_flash.h.

7.11 Base address of the Flash sectors

Collaboration diagram for Base address of the Flash sectors:



Macros

- **#define ADDR_FLASH_SECTOR_0** ((uint32_t)0x08000000) /* Base @ of Sector 0, 16 Kbyte */
- **#define ADDR_FLASH_SECTOR_1** ((uint32_t)0x08004000) /* Base @ of Sector 1, 16 Kbyte */
- **#define ADDR_FLASH_SECTOR_2** ((uint32_t)0x08008000) /* Base @ of Sector 2, 16 Kbyte */
- **#define ADDR_FLASH_SECTOR_3** ((uint32_t)0x0800C000) /* Base @ of Sector 3, 16 Kbyte */
- **#define ADDR_FLASH_SECTOR_4** ((uint32_t)0x08010000) /* Base @ of Sector 4, 64 Kbyte */
- **#define ADDR_FLASH_SECTOR_5** ((uint32_t)0x08020000) /* Base @ of Sector 5, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_6** ((uint32_t)0x08040000) /* Base @ of Sector 6, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_7** ((uint32_t)0x08060000) /* Base @ of Sector 7, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_8** ((uint32_t)0x08080000) /* Base @ of Sector 8, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_9** ((uint32_t)0x080A0000) /* Base @ of Sector 9, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_10** ((uint32_t)0x080C0000) /* Base @ of Sector 10, 128 Kbyte */
- **#define ADDR_FLASH_SECTOR_11** ((uint32_t)0x080E0000) /* Base @ of Sector 11, 128 Kbyte */

7.11.1 Detailed Description

7.12 Option_Bytes_Write_Protection

Collaboration diagram for Option_Bytes_Write_Protection:



Macros

- `#define OB_WRP_Sector_0 ((uint32_t)0x00000001)`
- `#define OB_WRP_Sector_1 ((uint32_t)0x00000002)`
- `#define OB_WRP_Sector_2 ((uint32_t)0x00000004)`
- `#define OB_WRP_Sector_3 ((uint32_t)0x00000008)`
- `#define OB_WRP_Sector_4 ((uint32_t)0x00000010)`
- `#define OB_WRP_Sector_5 ((uint32_t)0x00000020)`
- `#define OB_WRP_Sector_6 ((uint32_t)0x00000040)`
- `#define OB_WRP_Sector_7 ((uint32_t)0x00000080)`
- `#define OB_WRP_Sector_8 ((uint32_t)0x00000100)`
- `#define OB_WRP_Sector_9 ((uint32_t)0x00000200)`
- `#define OB_WRP_Sector_10 ((uint32_t)0x00000400)`
- `#define OB_WRP_Sector_11 ((uint32_t)0x00000800)`
- `#define OB_WRP_Sector_All ((uint32_t)0x00000FFF)`
- `#define IS_OB_WRP(SECTOR) (((SECTOR) & (uint32_t)0xFFFFF000) == 0x00000000) && ((SECTOR) != 0x00000000)`

7.12.1 Detailed Description

7.12.2 Macro Definition Documentation

7.12.2.1 `#define OB_WRP_Sector_0 ((uint32_t)0x00000001)`

Write protection of Sector0

Definition at line 153 of file stm32f4xx_flash.h.

7.12.2.2 `#define OB_WRP_Sector_1 ((uint32_t)0x00000002)`

Write protection of Sector1

Definition at line 154 of file stm32f4xx_flash.h.

7.12.2.3 `#define OB_WRP_Sector_10 ((uint32_t)0x00000400)`

Write protection of Sector10

Definition at line 163 of file stm32f4xx_flash.h.

7.12.2.4 `#define OB_WRP_Sector_11 ((uint32_t)0x00000800)`

Write protection of Sector11

Definition at line 164 of file stm32f4xx_flash.h.

7.12.2.5 `#define OB_WRP_Sector_2 ((uint32_t)0x00000004)`

Write protection of Sector2

Definition at line 155 of file stm32f4xx_flash.h.

7.12.2.6 `#define OB_WRP_Sector_3 ((uint32_t)0x00000008)`

Write protection of Sector3

Definition at line 156 of file stm32f4xx_flash.h.

7.12.2.7 `#define OB_WRP_Sector_4 ((uint32_t)0x00000010)`

Write protection of Sector4

Definition at line 157 of file stm32f4xx_flash.h.

7.12.2.8 `#define OB_WRP_Sector_5 ((uint32_t)0x00000020)`

Write protection of Sector5

Definition at line 158 of file stm32f4xx_flash.h.

7.12.2.9 `#define OB_WRP_Sector_6 ((uint32_t)0x00000040)`

Write protection of Sector6

Definition at line 159 of file stm32f4xx_flash.h.

7.12.2.10 `#define OB_WRP_Sector_7 ((uint32_t)0x00000080)`

Write protection of Sector7

Definition at line 160 of file stm32f4xx_flash.h.

7.12.2.11 `#define OB_WRP_Sector_8 ((uint32_t)0x00000100)`

Write protection of Sector8

Definition at line 161 of file stm32f4xx_flash.h.

7.12.2.12 `#define OB_WRP_Sector_9 ((uint32_t)0x00000200)`

Write protection of Sector9

Definition at line 162 of file stm32f4xx_flash.h.

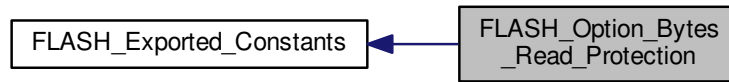
7.12.2.13 `#define OB_WRP_Sector_All ((uint32_t)0x00000FFF)`

Write protection of all Sectors

Definition at line 165 of file stm32f4xx_flash.h.

7.13 FLASH_Option_Bytes_Read_Protection

Collaboration diagram for FLASH_Option_Bytes_Read_Protection:



Macros

- #define **OB_RDP_Level_0** ((uint8_t)0xAA)
- #define **OB_RDP_Level_1** ((uint8_t)0x55)
- #define **IS_OB_RDP**(LEVEL)

7.13.1 Detailed Description

7.13.2 Macro Definition Documentation

7.13.2.1 #define IS_OB_RDP(LEVEL)

Value:

```

(((LEVEL) == OB_RDP_Level_0) || \
  ((LEVEL) == OB_RDP_Level_1)) /* || \
  ((LEVEL) == OB_RDP_Level_2)) */

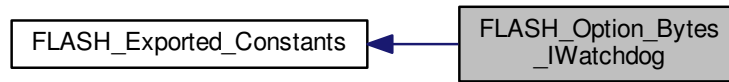
```

< Warning: When enabling read protection level 2 it's no more possible to go back to level 1 or 0

Definition at line 179 of file stm32f4xx_flash.h.

7.14 FLASH_Option_Bytes_IWatchdog

Collaboration diagram for FLASH_Option_Bytes_IWatchdog:



Macros

- `#define OB_IWDG_SW ((uint8_t)0x20)`
- `#define OB_IWDG_HW ((uint8_t)0x00)`
- `#define IS_OB_IWDG_SOURCE(SOURCE) (((SOURCE) == OB_IWDG_SW) || ((SOURCE) == OB_IWDG_HW))`

7.14.1 Detailed Description

7.14.2 Macro Definition Documentation

7.14.2.1 `#define OB_IWDG_HW ((uint8_t)0x00)`

Hardware IWDG selected

Definition at line 190 of file stm32f4xx_flash.h.

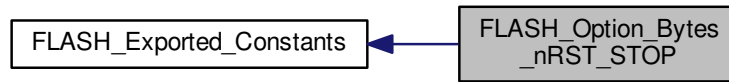
7.14.2.2 `#define OB_IWDG_SW ((uint8_t)0x20)`

Software IWDG selected

Definition at line 189 of file stm32f4xx_flash.h.

7.15 FLASH_Option_Bytes_nRST_STOP

Collaboration diagram for FLASH_Option_Bytes_nRST_STOP:



Macros

- `#define OB_STOP_NoRST ((uint8_t)0x40)`
- `#define OB_STOP_RST ((uint8_t)0x00)`
- `#define IS_OB_STOP_SOURCE(SOURCE) (((SOURCE) == OB_STOP_NoRST) || ((SOURCE) == OB_STOP_RST))`

7.15.1 Detailed Description

7.15.2 Macro Definition Documentation

7.15.2.1 `#define OB_STOP_NoRST ((uint8_t)0x40)`

No reset generated when entering in STOP

Definition at line 199 of file stm32f4xx_flash.h.

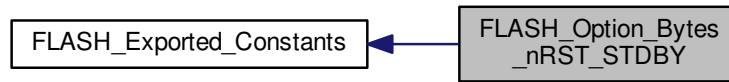
7.15.2.2 `#define OB_STOP_RST ((uint8_t)0x00)`

Reset generated when entering in STOP

Definition at line 200 of file stm32f4xx_flash.h.

7.16 FLASH_Option_Bytes_nRST_STDBY

Collaboration diagram for FLASH_Option_Bytes_nRST_STDBY:



Macros

- `#define OB_STDBY_NoRST ((uint8_t)0x80)`
- `#define OB_STDBY_RST ((uint8_t)0x00)`
- `#define IS_OB_STDBY_SOURCE(SOURCE) (((SOURCE) == OB_STDBY_NoRST) || ((SOURCE) == OB_STDBY_RST))`

7.16.1 Detailed Description

7.16.2 Macro Definition Documentation

7.16.2.1 `#define OB_STDBY_NoRST ((uint8_t)0x80)`

No reset generated when entering in STANDBY

Definition at line 210 of file stm32f4xx_flash.h.

7.16.2.2 `#define OB_STDBY_RST ((uint8_t)0x00)`

Reset generated when entering in STANDBY

Definition at line 211 of file stm32f4xx_flash.h.

7.17 FLASH_BOR_Reset_Level

Collaboration diagram for FLASH_BOR_Reset_Level:



Macros

- `#define OB_BOR_LEVEL3 ((uint8_t)0x00)`
- `#define OB_BOR_LEVEL2 ((uint8_t)0x04)`
- `#define OB_BOR_LEVEL1 ((uint8_t)0x08)`
- `#define OB_BOR_OFF ((uint8_t)0x0C)`
- `#define IS_OB_BOR(LEVEL)`

7.17.1 Detailed Description

7.17.2 Macro Definition Documentation

7.17.2.1 `#define IS_OB_BOR( LEVEL )`

Value:

```
(( (LEVEL) == OB_BOR_LEVEL1) || ((LEVEL) == OB_BOR_LEVEL2) || \
  ((LEVEL) == OB_BOR_LEVEL3) || ((LEVEL) == OB_BOR_OFF))
```

Definition at line 224 of file stm32f4xx_flash.h.

7.17.2.2 `#define OB_BOR_LEVEL1 ((uint8_t)0x08)`

Supply voltage ranges from 2.10 to 2.40 V

Definition at line 222 of file stm32f4xx_flash.h.

7.17.2.3 `#define OB_BOR_LEVEL2 ((uint8_t)0x04)`

Supply voltage ranges from 2.40 to 2.70 V

Definition at line 221 of file stm32f4xx_flash.h.

7.17.2.4 `#define OB_BOR_LEVEL3 ((uint8_t)0x00)`

Supply voltage ranges from 2.70 to 3.60 V

Definition at line 220 of file stm32f4xx_flash.h.

7.17.2.5 `#define OB_BOR_OFF ((uint8_t)0x0C)`

Supply voltage ranges from 1.62 to 2.10 V

Definition at line 223 of file stm32f4xx_flash.h.

7.18 FLASH_Interrupts

Collaboration diagram for FLASH_Interrupts:



Macros

- `#define FLASH_IT_EOP ((uint32_t)0x01000000)`
- `#define FLASH_IT_ERR ((uint32_t)0x02000000)`
- `#define IS_FLASH_IT(IT) (((IT) & (uint32_t)0xFCFFFFFF) == 0x00000000) && ((IT) != 0x00000000)`

7.18.1 Detailed Description

7.18.2 Macro Definition Documentation

7.18.2.1 `#define FLASH_IT_EOP ((uint32_t)0x01000000)`

End of FLASH Operation Interrupt source

Definition at line 233 of file stm32f4xx_flash.h.

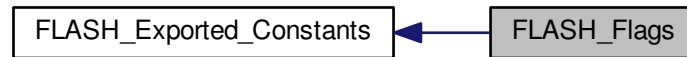
7.18.2.2 `#define FLASH_IT_ERR ((uint32_t)0x02000000)`

Error Interrupt source

Definition at line 234 of file stm32f4xx_flash.h.

7.19 FLASH_Flags

Collaboration diagram for FLASH_Flags:



Macros

- `#define FLASH_FLAG_EOP ((uint32_t)0x00000001)`
- `#define FLASH_FLAG_OPERR ((uint32_t)0x00000002)`
- `#define FLASH_FLAG_WRPERR ((uint32_t)0x00000010)`
- `#define FLASH_FLAG_PGAERR ((uint32_t)0x00000020)`
- `#define FLASH_FLAG_PGPERR ((uint32_t)0x00000040)`
- `#define FLASH_FLAG_PGSERR ((uint32_t)0x00000080)`
- `#define FLASH_FLAG_BSY ((uint32_t)0x00010000)`
- `#define IS_FLASH_CLEAR_FLAG(FLAG) (((FLAG) & (uint32_t)0xFFFFF0C) == 0x00000000) && ((FLAG) != 0x00000000)`
- `#define IS_FLASH_GET_FLAG(FLAG)`

7.19.1 Detailed Description

7.19.2 Macro Definition Documentation

7.19.2.1 `#define FLASH_FLAG_BSY ((uint32_t)0x00010000)`

FLASH Busy flag

Definition at line 249 of file stm32f4xx_flash.h.

7.19.2.2 `#define FLASH_FLAG_EOP ((uint32_t)0x00000001)`

FLASH End of Operation flag

Definition at line 243 of file stm32f4xx_flash.h.

7.19.2.3 `#define FLASH_FLAG_OPERR ((uint32_t)0x00000002)`

FLASH operation Error flag

Definition at line 244 of file stm32f4xx_flash.h.

7.19.2.4 `#define FLASH_FLAG_PGAERR ((uint32_t)0x00000020)`

FLASH Programming Alignment error flag

Definition at line 246 of file stm32f4xx_flash.h.

7.19.2.5 #define FLASH_FLAG_PGPERR ((uint32_t)0x00000040)

FLASH Programming Parallelism error flag

Definition at line 247 of file stm32f4xx_flash.h.

7.19.2.6 #define FLASH_FLAG_PGSERR ((uint32_t)0x00000080)

FLASH Programming Sequence error flag

Definition at line 248 of file stm32f4xx_flash.h.

7.19.2.7 #define FLASH_FLAG_WRPERR ((uint32_t)0x00000010)

FLASH Write protected error flag

Definition at line 245 of file stm32f4xx_flash.h.

7.19.2.8 #define IS_FLASH_GET_FLAG(FLAG)

Value:

```
((FLAG) == FLASH_FLAG_EOP) || ((FLAG) == FLASH_FLAG_OPERR) || \
                                     ((FLAG) == FLASH_FLAG_WRPERR) || ((FLAG) ==
FLASH_FLAG_PGAERR) || \
                                     ((FLAG) == FLASH_FLAG_PGPERR) || ((FLAG) ==
FLASH_FLAG_PGSERR) || \
                                     ((FLAG) == FLASH_FLAG_BSY))
```

Definition at line 251 of file stm32f4xx_flash.h.

7.20 FLASH_Program_Parallelism

Collaboration diagram for FLASH_Program_Parallelism:



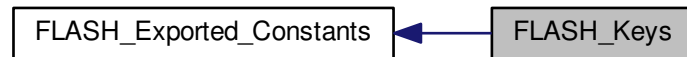
Macros

- `#define FLASH_PSIZE_BYTE ((uint32_t)0x00000000)`
- `#define FLASH_PSIZE_HALF_WORD ((uint32_t)0x00000100)`
- `#define FLASH_PSIZE_WORD ((uint32_t)0x00000200)`
- `#define FLASH_PSIZE_DOUBLE_WORD ((uint32_t)0x00000300)`
- `#define CR_PSIZE_MASK ((uint32_t)0xFFFFCFF)`

7.20.1 Detailed Description

7.21 FLASH_Keys

Collaboration diagram for FLASH_Keys:



Macros

- `#define RDP_KEY ((uint16_t)0x00A5)`
- `#define FLASH_KEY1 ((uint32_t)0x45670123)`
- `#define FLASH_KEY2 ((uint32_t)0xCDEF89AB)`
- `#define FLASH_OPT_KEY1 ((uint32_t)0x08192A3B)`
- `#define FLASH_OPT_KEY2 ((uint32_t)0x4C5D6E7F)`

7.21.1 Detailed Description

7.22 Config

Kernel parameters and options

- #define `CH_FREQUENCY` 1000
Vector table initialization address.
- #define `CH_TIME_QUANTUM` 20
Round robin interval.
- #define `CH_MEMCORE_SIZE` 0
Managed RAM size.
- #define `CH_NO_IDLE_THREAD` FALSE
Idle thread automatic spawn suppression.

Performance options

- #define `CH_OPTIMIZE_SPEED` TRUE
OS optimization.

Subsystem options

- #define `CH_USE_REGISTRY` TRUE
Threads registry APIs.
- #define `CH_USE_WAITEXIT` TRUE
Threads synchronization APIs.
- #define `CH_USE_SEMAPHORES` TRUE
Semaphores APIs.
- #define `CH_USE_SEMAPHORES_PRIORITY` FALSE
Semaphores queuing mode.
- #define `CH_USE_SEMSW` TRUE
Atomic semaphore API.
- #define `CH_USE_MUTEXES` TRUE
Mutexes APIs.
- #define `CH_USE_CONDVARS` TRUE
Conditional Variables APIs.
- #define `CH_USE_CONDVARS_TIMEOUT` TRUE
Conditional Variables APIs with timeout.
- #define `CH_USE_EVENTS` TRUE
Events Flags APIs.
- #define `CH_USE_EVENTS_TIMEOUT` TRUE
Events Flags APIs with timeout.
- #define `CH_USE_MESSAGES` TRUE
Synchronous Messages APIs.
- #define `CH_USE_MESSAGES_PRIORITY` FALSE
Synchronous Messages queuing mode.
- #define `CH_USE_MAILBOXES` TRUE
Mailboxes APIs.
- #define `CH_USE_QUEUES` TRUE
I/O Queues APIs.
- #define `CH_USE_MEMCORE` TRUE
Core Memory Manager APIs.

- #define `CH_USE_HEAP` TRUE
Heap Allocator APIs.
- #define `CH_USE_MALLOC_HEAP` FALSE
C-runtime allocator.
- #define `CH_USE_MEMPOOLS` TRUE
Memory Pools Allocator APIs.
- #define `CH_USE_DYNAMIC` TRUE
Dynamic Threads APIs.

Debug options

- #define `CH_DBG_SYSTEM_STATE_CHECK` TRUE
Debug option, system state check.
- #define `CH_DBG_ENABLE_CHECKS` FALSE
Debug option, parameters checks.
- #define `CH_DBG_ENABLE_ASSERTS` TRUE
Debug option, consistency checks.
- #define `CH_DBG_ENABLE_TRACE` TRUE
Debug option, trace buffer.
- #define `CH_DBG_ENABLE_STACK_CHECK` TRUE
Debug option, stack checks.
- #define `CH_DBG_FILL_THREADS` FALSE
Debug option, stacks initialization.
- #define `CH_DBG_THREADS_PROFILING` TRUE
Debug option, threads profiling.

Kernel hooks

- #define `THREAD_EXT_FIELDS` /* Add threads custom fields here.*/
Threads descriptor structure extension.
- #define `THREAD_EXT_INIT_HOOK(tp)`
Threads initialization hook.
- #define `THREAD_EXT_EXIT_HOOK(tp)`
Threads finalization hook.
- #define `THREAD_CONTEXT_SWITCH_HOOK(ntp, otp)`
Context switch hook.
- #define `IDLE_LOOP_HOOK()`
Idle Loop hook.
- #define `SYSTEM_TICK_EVENT_HOOK()`
System tick event hook.
- #define `SYSTEM_HALT_HOOK()`
System halt hook.

7.22.1 Detailed Description

Kernel related settings and hooks.

7.22.2 Macro Definition Documentation

7.22.2.1 `#define CH_DBG_ENABLE_ASSERTS TRUE`

Debug option, consistency checks.

If enabled then all the assertions in the kernel code are activated. This includes consistency checks inside the kernel, runtime anomalies and port-defined checks.

Note

The default is `FALSE`.

Definition at line 410 of file `chconf.h`.

7.22.2.2 `#define CH_DBG_ENABLE_CHECKS FALSE`

Debug option, parameters checks.

If enabled then the checks on the API functions input parameters are activated.

Note

The default is `FALSE`.

Definition at line 398 of file `chconf.h`.

7.22.2.3 `#define CH_DBG_ENABLE_STACK_CHECK TRUE`

Debug option, stack checks.

If enabled then a runtime stack check is performed.

Note

The default is `FALSE`.

The stack check is performed in a architecture/port dependent way. It may not be implemented on some ports.

The default failure mode is to halt the system with the global `panic_msg` variable set to `NULL`.

Definition at line 435 of file `chconf.h`.

7.22.2.4 `#define CH_DBG_ENABLE_TRACE TRUE`

Debug option, trace buffer.

If enabled then the context switch circular trace buffer is activated.

Note

The default is `FALSE`.

Definition at line 421 of file `chconf.h`.

7.22.2.5 `#define CH_DBG_FILL_THREADS FALSE`

Debug option, stacks initialization.

If enabled then the threads working area is filled with a byte value when a thread is created. This can be useful for the runtime measurement of the used stack.

Note

The default is `FALSE`.

Definition at line 447 of file `chconf.h`.

7.22.2.6 #define CH_DBG_SYSTEM_STATE_CHECK TRUE

Debug option, system state check.

If enabled the correct call protocol for system APIs is checked at runtime.

Note

The default is `FALSE`.

Definition at line 387 of file `chconf.h`.

7.22.2.7 #define CH_DBG_THREADS_PROFILING TRUE

Debug option, threads profiling.

If enabled then a field is added to the `Thread` structure that counts the system ticks occurred while executing the thread.

Note

The default is `TRUE`.

This debug option is defaulted to `TRUE` because it is required by some test cases into the test suite.

Definition at line 460 of file `chconf.h`.

7.22.2.8 #define CH_FREQUENCY 1000

Vector table initialization address.

Define the start address of the application relative to ROM base address. System tick frequency.

Frequency of the system timer that drives the system ticks. This setting also defines the system tick time unit.

Definition at line 71 of file `chconf.h`.

7.22.2.9 #define CH_MEMCORE_SIZE 0

Managed RAM size.

Size of the RAM area to be managed by the OS. If set to zero then the whole available RAM is used. The core memory is made available to the heap allocator and/or can be used directly through the simplified core memory allocator.

Note

In order to let the OS manage the whole RAM the linker script must provide the **heap_base** and **heap_end** symbols.

Requires `CH_USE_MEMCORE`.

Definition at line 101 of file `chconf.h`.

7.22.2.10 `#define CH_NO_IDLE_THREAD FALSE`

Idle thread automatic spawn suppression.

When this option is activated the function `chSysInit()` does not spawn the idle thread automatically. The application has then the responsibility to do one of the following:

- Spawn a custom idle thread at priority `IDLEPRIO`.
- Change the `main()` thread priority to `IDLEPRIO` then enter an endless loop. In this scenario the `main()` thread acts as the idle thread.

Note

Unless an idle thread is spawned the `main()` thread must not enter a sleep state.

Definition at line 118 of file `chconf.h`.

7.22.2.11 `#define CH_OPTIMIZE_SPEED TRUE`

OS optimization.

If enabled then time efficient rather than space efficient code is used when two possible implementations exist.

Note

This is not related to the compiler optimization options.
The default is `TRUE`.

Definition at line 139 of file `chconf.h`.

7.22.2.12 `#define CH_TIME_QUANTUM 20`

Round robin interval.

This constant is the number of system ticks allowed for the threads before preemption occurs. Setting this value to zero disables the preemption for threads with equal priority and the round robin becomes cooperative. Note that higher priority threads can still preempt, the kernel is always preemptive.

Note

Disabling the round robin preemption makes the kernel more compact and generally faster.

Definition at line 86 of file `chconf.h`.

7.22.2.13 `#define CH_USE_CONDVARS TRUE`

Conditional Variables APIs.

If enabled then the conditional variables APIs are included in the kernel.

Note

The default is `TRUE`.
Requires `CH_USE_MUTEXES`.

Definition at line 225 of file `chconf.h`.

7.22.2.14 #define CH_USE_CONDVARs_TIMEOUT TRUE

Conditional Variables APIs with timeout.

If enabled then the conditional variables APIs with timeout specification are included in the kernel.

Note

The default is `TRUE`.

Requires `CH_USE_CONDVARs`.

Definition at line 237 of file `chconf.h`.

7.22.2.15 #define CH_USE_DYNAMIC TRUE

Dynamic Threads APIs.

If enabled then the dynamic threads creation APIs are included in the kernel.

Note

The default is `TRUE`.

Requires `CH_USE_WAITEXIT`.

Requires `CH_USE_HEAP` and/or `CH_USE_MEMPOOLS`.

Definition at line 367 of file `chconf.h`.

7.22.2.16 #define CH_USE_EVENTS TRUE

Events Flags APIs.

If enabled then the event flags APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 247 of file `chconf.h`.

7.22.2.17 #define CH_USE_EVENTS_TIMEOUT TRUE

Events Flags APIs with timeout.

If enabled then the events APIs with timeout specification are included in the kernel.

Note

The default is `TRUE`.

Requires `CH_USE_EVENTS`.

Definition at line 259 of file `chconf.h`.

7.22.2.18 #define CH_USE_HEAP TRUE

Heap Allocator APIs.

If enabled then the memory heap allocator APIs are included in the kernel.

Note

The default is `TRUE`.
Requires `CH_USE_MEMCORE` and either `CH_USE_MUTEXES` or `CH_USE_SEMAPHORES`.
Mutexes are recommended.

Definition at line 329 of file `chconf.h`.

7.22.2.19 #define CH_USE_MAILBOXES TRUE

Mailboxes APIs.

If enabled then the asynchronous messages (mailboxes) APIs are included in the kernel.

Note

The default is `TRUE`.
Requires `CH_USE_SEMAPHORES`.

Definition at line 294 of file `chconf.h`.

7.22.2.20 #define CH_USE_MALLOC_HEAP FALSE

C-runtime allocator.

If enabled the the heap allocator APIs just wrap the C-runtime `malloc()` and `free()` functions.

Note

The default is `FALSE`.
Requires `CH_USE_HEAP`.
The C-runtime may or may not require `CH_USE_MEMCORE`, see the appropriate documentation.

Definition at line 343 of file `chconf.h`.

7.22.2.21 #define CH_USE_MEMCORE TRUE

Core Memory Manager APIs.

If enabled then the core memory manager APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 315 of file `chconf.h`.

7.22.2.22 #define CH_USE_MEMPOOLS TRUE

Memory Pools Allocator APIs.

If enabled then the memory pools allocator APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 354 of file `chconf.h`.

7.22.2.23 #define CH_USE_MESSAGES TRUE

Synchronous Messages APIs.

If enabled then the synchronous messages APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 270 of file `chconf.h`.

7.22.2.24 #define CH_USE_MESSAGES_PRIORITY FALSE

Synchronous Messages queuing mode.

If enabled then messages are served by priority rather than in FIFO order.

Note

The default is `FALSE`. Enable this if you have special requirements.
Requires `CH_USE_MESSAGES`.

Definition at line 282 of file `chconf.h`.

7.22.2.25 #define CH_USE_MUTEXES TRUE

Mutexes APIs.

If enabled then the mutexes APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 213 of file `chconf.h`.

7.22.2.26 #define CH_USE_QUEUES TRUE

I/O Queues APIs.

If enabled then the I/O queues APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 304 of file `chconf.h`.

7.22.2.27 #define CH_USE_REGISTRY TRUE

Threads registry APIs.

If enabled then the registry APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 158 of file `chconf.h`.

7.22.2.28 #define CH_USE_SEMAPHORES TRUE

Semaphores APIs.

If enabled then the Semaphores APIs are included in the kernel.

Note

The default is `TRUE`.

Definition at line 179 of file `chconf.h`.

7.22.2.29 #define CH_USE_SEMAPHORES_PRIORITY FALSE

Semaphores queuing mode.

If enabled then the threads are enqueued on semaphores by priority rather than in FIFO order.

Note

The default is `FALSE`. Enable this if you have special requirements.

Requires `CH_USE_SEMAPHORES`.

Definition at line 191 of file `chconf.h`.

7.22.2.30 #define CH_USE_SEMSW TRUE

Atomic semaphore API.

If enabled then the semaphores the `chSemSignalWait()` API is included in the kernel.

Note

The default is `TRUE`.

Requires `CH_USE_SEMAPHORES`.

Definition at line 203 of file `chconf.h`.

7.22.2.31 #define CH_USE_WAITEXIT TRUE

Threads synchronization APIs.

If enabled then the `chThdWait()` function is included in the kernel.

Note

The default is `TRUE`.

Definition at line 169 of file `chconf.h`.

7.22.2.32 #define IDLE_LOOP_HOOK()**Value:**

```
{
    /* Idle loop code here.*/
}
```

Idle Loop hook.

This hook is continuously invoked by the idle thread loop.

Definition at line 523 of file `chconf.h`.

7.22.2.33 #define SYSTEM_HALT_HOOK()

Value:

```
{
    /* System halt code here.*/
}
```

System halt hook.

This hook is invoked in case to a system halting error before the system is halted.

Definition at line 545 of file chconf.h.

7.22.2.34 #define SYSTEM_TICK_EVENT_HOOK()

Value:

```
{
    /* System tick event code here.*/
}
```

System tick event hook.

This hook is invoked in the system tick handler immediately after processing the virtual timers queue.

Definition at line 534 of file chconf.h.

7.22.2.35 #define THREAD_CONTEXT_SWITCH_HOOK(ntp, otp)

Value:

```
{
    /* System halt code here.*/
}
```

Context switch hook.

This hook is invoked just before switching between threads.

Definition at line 513 of file chconf.h.

7.22.2.36 #define THREAD_EXT_EXIT_HOOK(tp)

Value:

```
{
    /* Add threads finalization code here.*/
}
```

Threads finalization hook.

User finalization code added to the `chThdExit()` API.

Note

It is inserted into lock zone.

It is also invoked when the threads simply return in order to terminate.

Definition at line 503 of file chconf.h.

7.22.2.37 `#define THREAD_EXT_FIELDS /* Add threads custom fields here.*/`

Threads descriptor structure extension.

User fields added to the end of the `Thread` structure.

Definition at line 477 of file `chconf.h`.

7.22.2.38 `#define THREAD_EXT_INIT_HOOK( tp )`

Value:

```
{  
    /* Add threads initialization code here.*/  
}
```

Threads initialization hook.

User initialization code added to the `chThdInit()` API.

Note

It is invoked from within `chThdInit()` and implicitly from all the threads creation APIs.

Definition at line 489 of file `chconf.h`.

7.23 IWDG

7.24 EEPROM_Emulation

Functions

- uint16_t [EE_Init](#) (void)
Restore the pages to a known good state in case of page's status corruption after a power loss.
- uint16_t [EE_ReadVariable](#) (uint16_t VirtAddress, uint16_t *Data)
Returns the last stored variable data, if found, which correspond to the passed virtual address.
- uint16_t [EE_WriteVariable](#) (uint16_t VirtAddress, uint16_t Data)
Writes/upadtes variable data in EEPROM.

Variables

- uint16_t **DataVar** = 0
- const uint16_t [VirtAddVarTab](#) [NB_OF_VAR]
This table is used to store the virtual address of the persistent parameters.

7.24.1 Detailed Description

7.24.2 Function Documentation

7.24.2.1 uint16_t EE_Init (void)

Restore the pages to a known good state in case of page's status corruption after a power loss.

Parameters

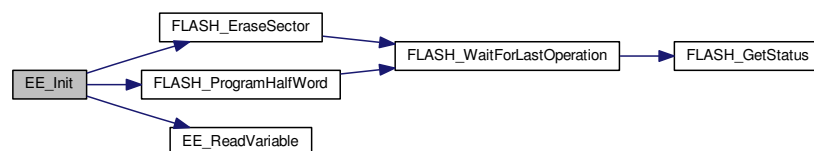
<i>None.</i>	
--------------	--

Return values

-	Flash error code: on write Flash error • FLASH_COMPLETE: on success
---	--

Definition at line 56 of file eeprom.c.

Here is the call graph for this function:



7.24.2.2 uint16_t EE_ReadVariable (uint16_t VirtAddress, uint16_t * Data)

Returns the last stored variable data, if found, which correspond to the passed virtual address.

Parameters

<i>VirtAddress</i>	Variable virtual address
<i>Data</i>	Global variable contains the read variable value

Return values

<i>Success</i>	or error status: • 0: if variable was found • 1: if the variable was not found • NO_VALID_PAGE: if no valid page was found.
----------------	---

Definition at line 271 of file eeprom.c.

7.24.2.3 uint16_t EE_WriteVariable (uint16_t *VirtAddress*, uint16_t *Data*)

Writes/upadtes variable data in EEPROM.

Parameters

<i>VirtAddress</i>	Variable virtual address
<i>Data</i>	16 bit data to be written

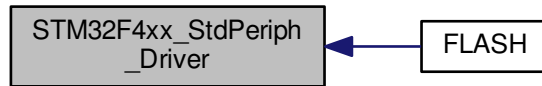
Return values

<i>Success</i>	or error status: • FLASH_COMPLETE: on success • PAGE_FULL: if valid page is full • NO_VALID_PAGE: if no valid page was found • Flash error code: on write Flash error
----------------	---

Definition at line 330 of file eeprom.c.

7.25 STM32F4xx_StdPeriph_Driver

Collaboration diagram for STM32F4xx_StdPeriph_Driver:



Modules

- [FLASH](#)

FLASH driver modules.

7.25.1 Detailed Description

7.26 HAL_CONF

Macros

- #define HAL_USE_TM TRUE
Enables the TM subsystem.
- #define HAL_USE_PAL TRUE
Enables the PAL subsystem.
- #define HAL_USE_ADC TRUE
Enables the ADC subsystem.
- #define HAL_USE_CAN FALSE
Enables the CAN subsystem.
- #define HAL_USE_EXT TRUE
Enables the EXT subsystem.
- #define HAL_USE_GPT FALSE
Enables the GPT subsystem.
- #define HAL_USE_I2C TRUE
Enables the I2C subsystem.
- #define HAL_USE_ICU FALSE
Enables the ICU subsystem.
- #define HAL_USE_MAC FALSE
Enables the MAC subsystem.
- #define HAL_USE_MMC_SPI TRUE
Enables the MMC_SPI subsystem.
- #define HAL_USE_PWM TRUE
Enables the PWM subsystem.
- #define HAL_USE_RTC TRUE
Enables the RTC subsystem.
- #define HAL_USE_SDC FALSE
Enables the SDC subsystem.
- #define HAL_USE_SERIAL TRUE
Enables the SERIAL subsystem.
- #define HAL_USE_SERIAL_USB FALSE
Enables the SERIAL over USB subsystem.
- #define HAL_USE_SPI TRUE
Enables the SPI subsystem.
- #define HAL_USE_UART TRUE
Enables the UART subsystem.
- #define HAL_USE_USB FALSE
Enables the USB subsystem.
- #define HAL_USE_IWDG TRUE
Enables the Independent Watchdog.
- #define ADC_USE_WAIT TRUE
Enables synchronous APIs.
- #define ADC_USE_MUTUAL_EXCLUSION TRUE
Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.
- #define CAN_USE_SLEEP_MODE TRUE
Sleep mode related APIs inclusion switch.
- #define I2C_USE_MUTUAL_EXCLUSION TRUE
Enables the mutual exclusion APIs on the I2C bus.
- #define MAC_USE_ZERO_COPY FALSE

- Enables an event sources for incoming packets.*

 - `#define MAC_USE_EVENTS TRUE`
- Enables an event sources for incoming packets.*

 - `#define MMC_NICE_WAITING TRUE`
- Delays insertions.*

 - `#define SDC_INIT_RETRY 100`
- Number of initialization attempts before rejecting the card.*

 - `#define SDC_MMC_SUPPORT FALSE`
- Include support for MMC cards.*

 - `#define SDC_NICE_WAITING TRUE`
- Delays insertions.*

 - `#define SERIAL_DEFAULT_BITRATE 38400`
- Default bit rate.*

 - `#define SERIAL_BUFFERS_SIZE 64`
- Serial buffers size.*

 - `#define SPI_USE_WAIT TRUE`
- Enables synchronous APIs.*

 - `#define SPI_USE_MUTUAL_EXCLUSION TRUE`
- Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.*

7.26.1 Detailed Description

7.26.2 Macro Definition Documentation

7.26.2.1 `#define ADC_USE_MUTUAL_EXCLUSION TRUE`

Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

Definition at line 187 of file `halconf.h`.

7.26.2.2 `#define ADC_USE_WAIT TRUE`

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

Definition at line 179 of file `halconf.h`.

7.26.2.3 `#define MMC_NICE_WAITING TRUE`

Delays insertions.

If enabled this options inserts delays into the MMC waiting routines releasing some extra CPU time for the threads with lower priority, this may slow down the driver a bit however. This option is recommended also if the SPI driver does not use a DMA channel and heavily loads the CPU.

Definition at line 243 of file `halconf.h`.

7.26.2.4 #define SDC_INIT_RETRY 100

Number of initialization attempts before rejecting the card.

Note

Attempts are performed at 10mS intervals.

Definition at line 255 of file halconf.h.

7.26.2.5 #define SDC_MMC_SUPPORT FALSE

Include support for MMC cards.

Note

MMC support is not yet implemented so this option must be kept at FALSE.

Definition at line 264 of file halconf.h.

7.26.2.6 #define SDC_NICE_WAITING TRUE

Delays insertions.

If enabled this options inserts delays into the MMC waiting routines releasing some extra CPU time for the threads with lower priority, this may slow down the driver a bit however.

Definition at line 274 of file halconf.h.

7.26.2.7 #define SERIAL_BUFFERS_SIZE 64

Serial buffers size.

Configuration parameter, you can change the depth of the queue buffers depending on the requirements of your application.

Note

The default is 64 bytes for both the transmission and receive buffers.

Definition at line 298 of file halconf.h.

7.26.2.8 #define SERIAL_DEFAULT_BITRATE 38400

Default bit rate.

Configuration parameter, this is the baud rate selected for the default configuration.

Definition at line 287 of file halconf.h.

7.26.2.9 #define SPI_USE_MUTUAL_EXCLUSION TRUE

Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

Definition at line 318 of file halconf.h.

7.26.2.10 `#define SPI_USE_WAIT TRUE`

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

Definition at line 310 of file halconf.h.

7.27 WEB_THREAD

Macros

- `#define WEB_THREAD_STACK_SIZE 1024`
- `#define WEB_THREAD_PORT 80`
- `#define WEB_THREAD_PRIORITY (LOWPRIO + 2)`

Functions

- **WORKING_AREA** (wa_http_server, WEB_THREAD_STACK_SIZE)
- `msg_t http_server (void *p)`

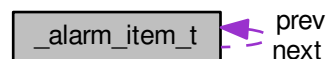
7.27.1 Detailed Description

Chapter 8

Data Structure Documentation

8.1 `_alarm_item_t` Struct Reference

Collaboration diagram for `_alarm_item_t`:



Data Fields

- `uint8_t id`
- `uint8_t code`
- `uint16_t t_off`
- `uint8_t serviced`
- `struct _alarm_item_t * next`
- `struct _alarm_item_t * prev`

8.1.1 Detailed Description

Definition at line 50 of file `link_alarm_list.h`.

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

- [link/link_alarm_list.h](#)

8.2 `_alarms_t` Struct Reference

A struct definition used to provide flags for alarms.

```
#include <global.h>
```

Data Fields

- `int32_t power`
- `int32_t tamper`
- `int32_t microwave`
- `int32_t microwave_auth`
- `int32_t upper_deadzone`
- `int32_t lower_deadzone`
- `int32_t environmental`
- `int32_t error`
- `int32_t aux_alarm [4]`

8.2.1 Detailed Description

A struct definition used to provide flags for alarms.

Definition at line 176 of file `global.h`.

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

- [global.h](#)

8.3 `_analog_alarms_t` Struct Reference

Data Fields

- `uint8_t adc_high_count`
- `uint8_t adc_low_count`
- `uint8_t adc_alarm_clear`
- `uint16_t alarm_threshold`
- `uint8_t alarm_clear_flag`
- volatile `analog_ch_state current_state`

8.3.1 Detailed Description

A struct definition to hold ADC sampling data for an analog pin

Definition at line 29 of file `analog_input_driver.c`.

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

- `services/analog_input_driver.c`
- `services/analog_input_driver_exp.c`

8.4 `_comms_hub_t` Struct Reference

A struct used to store information for an OSDP communications interface.

```
#include <comms_packet.h>
```

Collaboration diagram for _comms_hub_t:



Data Fields

- Mutex [mtx](#)
A mutex used to guard this struct.
- [comms_hub_iface_e](#) [iface_enable](#)
- [osdp_header_t](#) [osdp_header](#)
A struct used to store the metadata associated with a received packet.
- [relay_queue_t](#) * [write](#)
- [osdp_ack_t](#) [osdp_ack_msg](#)
- [osdp_busy_t](#) [osdp_busy_msg](#)
- [osdp_cap_t](#) [osdp_cap_msg](#)
- [osdp_ccrypt_t](#) [osdp_ccrypt_msg](#)
- [osdp_comset_t](#) [osdp_comset_msg](#)
- [osdp_id_t](#) [osdp_id_msg](#)

- [osdp_istat_t](#) [osdp_istat_msg](#)
- [osdp_istatr_t](#) [osdp_istatr_msg](#)
- [osdp_keyset_t](#) [osdp_keyset_msg](#)
- [osdp_lstat_t](#) [osdp_lstat_msg](#)
- [osdp_lstatr_t](#) [osdp_lstatr_msg](#)
- [osdp_ms100_mfg_t](#) [osdp_mfg_msg](#)
- [osdp_ms100_mfgrep_t](#) [osdp_mfgrep_msg](#)
- [osdp_nak_t](#) [osdp_nak_msg](#)
- [osdp_pdcap_t](#) [osdp_pdcap_msg](#)
- [osdp_pdid_t](#) [osdp_pdid_msg](#)
- [osdp_poll_t](#) [osdp_poll_msg](#)
- [osdp_scrypt_t](#) [osdp_scrypt_msg](#)
- [osdp_rmaci_t](#) [osdp_rmaci_msg](#)
- [osdp_master_beacon_t](#) [osdp_master_beacon_msg](#)
- [osdp_halo_sync_t](#) [osdp_halo_sync_msg](#)
- [osdp_halo_alarm_t](#) [osdp_halo_alarm_msg](#)
- [osdp_rfauth_t](#) [osdp_rfauth_msg](#)
- [osdp_hub_messages_t](#) [osdp_hub_msg](#)

8.4.1 Detailed Description

A struct used to store information for an OSDP communications interface.

Definition at line 77 of file `comms_packet.h`.

8.4.2 Field Documentation

8.4.2.1 `comms_hub_iface_e` `iface_enable`

An enumerated type used to indicate which communications interface(s) are enabled

Definition at line 84 of file `comms_packet.h`.

8.4.2.2 `osdp_ack_t` `osdp_ack_msg`

The packet type parsers

Definition at line 91 of file `comms_packet.h`.

8.4.2.3 `relay_queue_t*` `write`

A function pointer used to select the interface on which transmissions are enabled using the `relay_queue_t` interface

Definition at line 119 of file `comms_packet.h`.

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

- `comms/comms_packet.h`

8.5 `_default_vars_t` Struct Reference

Data Fields

- `uint8_t` `compliance`
- `uint8_t` `number`

8.5.1 Detailed Description

Definition at line 20 of file osdp_pdcap_base.h.

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdcap_base.h

8.6 _float_uint32_u Union Reference

Data Fields

- float **f**
- uint32_t **u**

8.6.1 Detailed Description

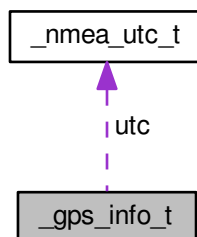
Definition at line 10 of file osdp_ms100_mfgrep_base.c.

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep_base.c

8.7 _gps_info_t Struct Reference

Collaboration diagram for _gps_info_t:



Data Fields

- [nmea_utc_t](#) **utc**
- float **latitude**
- float **longitude**
- float **altitude**
- uint32_t **num_sentences**
- uint32_t **signal_quality**
- uint32_t **satellites_in_use**

8.7.1 Detailed Description

Definition at line 33 of file `gps_parser.h`.

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

- [gps/gps_parser.h](#)

8.8 _InitCfg Struct Reference

Public Member Functions

- `uint8_t osdp_addr __attribute__((aligned(4)))`
OSDP Address.
- `uint32_t osdp_baud __attribute__((aligned(4)))`
OSDP RS485 baud rate.
- `uint8_t operating_mode __attribute__((aligned(4)))`
Max2828 operating mode - Tx or Rx.
- `uint8_t channel __attribute__((aligned(4)))`
The MAX2828 channel.
- `uint16_t int_freq_channel __attribute__((aligned(4)))`
The MAX2828 Integer Divider Ratio Register value.
- `uint32_t diff_freq_channel __attribute__((aligned(4)))`
The MAX2828 Fractional Divider Ratio register value.
- `uint16_t range __attribute__((aligned(4)))`
The microwave range value.
- `uint8_t sensitivity __attribute__((aligned(4)))`
The microwave sensitivity value.
- `uint8_t deadzone __attribute__((aligned(4)))`
Upper Deadzone enable.
- `uint16_t relays __attribute__((aligned(4)))`
Relay 1 & 2 duration and trigger action.
- `uint8_t alarm_class __attribute__((aligned(4)))`
Alarm Classes.
- `uint8_t adc __attribute__((aligned(4)))`
ADC.
- `uint32_t ipv4_address __attribute__((aligned(4)))`
IPv4 Static IP address.
- `uint32_t ipv4_netmask __attribute__((aligned(4)))`
IPv4 netmask.
- `uint32_t ipv4_gateway __attribute__((aligned(4)))`
IPv4 gateway address.
- `uint32_t ipv4_server_address __attribute__((aligned(4)))`
IPv4 Static server IP address.
- `uint16_t server_port __attribute__((aligned(4)))`
Server TCP/IP port.
- `uint8_t ipv4_dhcp_autoip __attribute__((aligned(4)))`
Server TCP/IP port.
- `uint32_t firmware_version __attribute__((aligned(4)))`
Firmware version.
- `uint32_t alarm_timeout __attribute__((aligned(4)))`

- `uint8_t osdp_scs_enable` [__attribute__](#) ((aligned(4)))
Temporary alarm disable timeout.
- `uint16_t osdp_scs_duration` [__attribute__](#) ((aligned(4)))
A flag indicating if security is enabled or otherwise.
- `uint32_t osdp_scs_duration_secs` [__attribute__](#) ((aligned(4)))
The maximum permitted duration of a secure OSDP session.
- `uint8_t trace_level` [__attribute__](#) ((aligned(4)))
The level of trace to be generated.
- `uint32_t record_index` [__attribute__](#) ((aligned(4)))
The CRC of the configuration parameters.
- `uint16_t relay_1_triggers` [__attribute__](#) ((aligned(4)))
Relay 1 trigger selection.
- `uint16_t relay_2_triggers` [__attribute__](#) ((aligned(4)))
Relay 2 trigger selection.
- `uint16_t sensurity_product_id` [__attribute__](#) ((aligned(4)))
The Sensurity Product ID.
- `uint8_t microwave_algorithm` [__attribute__](#) ((aligned(4)))
The selected microwave intruder algorithm.
- `uint8_t deadzone_sensitivity` [__attribute__](#) ((aligned(4)))
The two deadzone sensitivities.
- `uint8_t paired_mode` [__attribute__](#) ((aligned(4)))
A flag indicating if paired device security is enabled.
- `uint32_t paired_serial` [__attribute__](#) ((aligned(4)))
The serial number of the paired device.
- `uint32_t paired_key[4]` [__attribute__](#) ((aligned(4)))
The serial number of the paired device.
- `uint32_t crc32` [__attribute__](#) ((aligned(4)))
The CRC of the configuration parameters.

8.8.1 Detailed Description

Definition at line 300 of file `global.h`.

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

- [global.h](#)

8.9 _int_frac_t Struct Reference

A struct used to define an integer/fractional divider pair.

Data Fields

- `uint16_t int_val`
- `uint32_t frac_val`

8.9.1 Detailed Description

A struct used to define an integer/fractional divider pair.

Definition at line 25 of file preInit.c.

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

- [preInit.c](#)

8.10 _link_auth_t Struct Reference

A struct used to store the id and alarm information for a specific device.

```
#include <link_auth.h>
```

Data Fields

- uint8_t **id**
- uint8_t **auth** [[MAX_LINK_AUTH_SIZE](#)]
- int32_t **auth_size**
- int32_t **auth_size_dec**
- uint32_t **ready**

8.10.1 Detailed Description

A struct used to store the id and alarm information for a specific device.

Definition at line 38 of file link_auth.h.

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

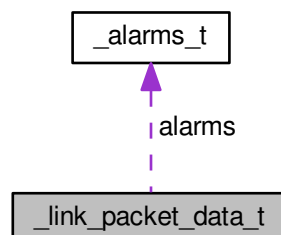
- [link/link_auth.h](#)

8.11 _link_packet_data_t Struct Reference

A struct used store the id and alarm information for a specific device.

```
#include <link_packet.h>
```

Collaboration diagram for _link_packet_data_t:



Data Fields

- `uint8_t id`
- [alarms_t](#) `alarms`
- `uint64_t epoch_usec`

8.11.1 Detailed Description

A struct used store the id and alarm information for a specific device.

Definition at line 47 of file `link_packet.h`.

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

- [link/link_packet.h](#)

8.12 `_link_rolling_t` Struct Reference

A struct used to store the rolling code data.

```
#include <link_rolling.h>
```

Data Fields

- `uint32_t offset`
- `uint32_t random`
- `uint32_t code`
- `int32_t(* enc_callback)(uint8_t *, int32_t *)`
- `int32_t(* dec_callback)(uint8_t *, int32_t)`

8.12.1 Detailed Description

A struct used to store the rolling code data.

Definition at line 38 of file `link_rolling.h`.

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

- [link/link_rolling.h](#)

8.13 `_msg_info_t` Struct Reference

A struct used to define the position and time of the message origin.

```
#include <trace.h>
```

Data Fields

- [trace_level_e](#) `level`
- `uint32_t timestamp`
- `char file` [MAX_FILE_LENGTH]
- `char function` [MAX_FUNCTION_LENGTH]
- `int32_t line`
- `char msg` [MAX_MSG_LENGTH]

8.13.1 Detailed Description

A struct used to define the position and time of the message origin.

Definition at line 56 of file trace.h.

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

- trace/[trace.h](#)

8.14 `_msg_ip_worker_t` Struct Reference

Data Fields

- int **client_fd**
- int **running**
- int **closed**
- Mutex **mtx**

8.14.1 Detailed Description

This struct is used to store the client socket, a flag to indicate the progress of the thread and a mutex controlling access to the struct contents

Definition at line 52 of file tcpserver.c.

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

- comms/[tcpserver.c](#)

8.15 `_network_alarms_t` Struct Reference

Data Fields

- uint32_t [devices](#) [4]
Fast determination of activated alarms.
- uint32_t [alarms](#) [NUM_NETWORK_ALARM_TYPES][4]
All alarms by type and address (exc. paired)

8.15.1 Detailed Description

Definition at line 49 of file halo_alarms.h.

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/halo_alarms.h

8.16 `_nmea_utc_t` Struct Reference

Data Fields

- int32_t **hour**

- int32_t **min**
- int32_t **sec**
- int32_t **year**
- int32_t **month**
- int32_t **day**

8.16.1 Detailed Description

Definition at line 21 of file `gps_parser.h`.

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

- [gps/gps_parser.h](#)

8.17 _osdp_ack_t Struct Reference

Data Fields

- osdp_header_t * **header**

8.17.1 Detailed Description

Definition at line 18 of file `osdp_ack_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ack_base.h](#)

8.18 _osdp_alarm_zone_t Struct Reference

Data Fields

- uint8_t **num_zones**
- uint8_t **zone**
- uint8_t **type**
- uint32_t **reserved**
- osdp_header_t * **header**

8.18.1 Detailed Description

Definition at line 18 of file `osdp_alarm_zone_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_alarm_zone_base.h](#)

8.19 _osdp_busy_t Struct Reference

Data Fields

- osdp_header_t * **header**

8.19.1 Detailed Description

Definition at line 18 of file `osdp_busy_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_busy_base.h](#)

8.20 `_osdp_cap_t` Struct Reference

Data Fields

- `uint8_t` **code**
- `osdp_header_t *` **header**
- `SecurityCallback` **callback_enc**
- `SecurityCallback` **callback_dec**

8.20.1 Detailed Description

Definition at line 23 of file `osdp_cap_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_cap_base.h](#)

8.21 `_osdp_halo_alarm_request_t` Struct Reference

Data Fields

- `osdp_header_t *` **header**

8.21.1 Detailed Description

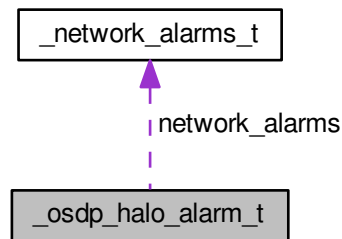
Definition at line 37 of file `osdp_halo_alarm_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm_base.h](#)

8.22 _osdp_halo_alarm_t Struct Reference

Collaboration diagram for _osdp_halo_alarm_t:



Data Fields

- `int32_t` **max_size**
- `int32_t` **start_addr**
- `int32_t` **end_addr**
- `uint32_t` **serial**
- `network_alarms_t` * **network_alarms**
- `osdp_header_t` * **header**

8.22.1 Detailed Description

Definition at line 24 of file `osdp_halo_alarm_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm_base.h](#)

8.23 _osdp_halo_sync_t Struct Reference

Data Fields

- `uint8_t` **type**
- `uint8_t` **payload** [16]
- `int32_t` **payload_length**
- `osdp_header_t` * **header**

8.23.1 Detailed Description

Definition at line 18 of file `osdp_halo_sync_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_sync_base.h](#)

8.24 `_osdp_hub_messages_t` Struct Reference

Data Fields

- `osdp_hub_e` **hub_msg**
- `bool` **enable**
- `uint8_t` **max_tcp**
- `uint8_t` **ip_addr** [16]
- `bool` **ipv4**
- `uint32_t` **port**
- `bool` **success**
- `uint8_t` **username** [17]
- `uint8_t` **publickey** [321]
- `uint8_t` **salt** [65]
- `uint8_t` **cryptogram** [65]
- `osdp_header_t` * **header**

8.24.1 Detailed Description

Definition at line 34 of file `osdp_hub_messages_base.h`.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_hub_messages_base.h`

8.25 `_osdp_istatr_t` Struct Reference

Data Fields

- `int32_t` **num_status**
- `uint8_t` **status** [32]
- `osdp_header_t` * **header**
- `SecurityCallback` **callback_enc**
- `SecurityCallback` **callback_dec**

8.25.1 Detailed Description

Definition at line 18 of file `osdp_istatr_base.h`.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp\_istatr\_base.h`

8.26 `_osdp_istatr_t` Struct Reference

Data Fields

- `uint8_t` **tamper**
- `uint8_t` **power**
- `osdp_header_t` * **header**
- `SecurityCallback` **callback_enc**
- `SecurityCallback` **callback_dec**

8.26.1 Detailed Description

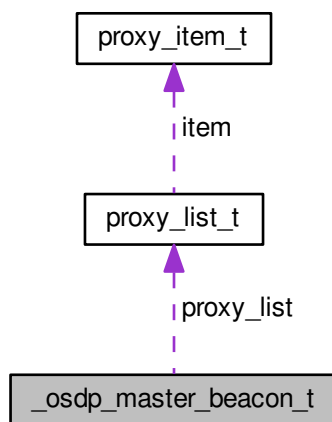
Definition at line 18 of file osdp_lstatr_base.h.

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[osdp_lstatr_base.h](#)

8.27 _osdp_master_beacon_t Struct Reference

Collaboration diagram for _osdp_master_beacon_t:



Data Fields

- `uint32_t` **serial**
- `uint8_t` **addr**
- `uint8_t` **default_sync_period**
- `uint8_t` **default_sync_count**
- `uint8_t` **capabilities**
- `proxy\_list\_t *` **proxy_list**
- `osdp_header_t *` **header**

8.27.1 Detailed Description

Definition at line 19 of file osdp_master_beacon_base.h.

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[osdp_master_beacon_base.h](#)

8.28 _osdp_nak_t Struct Reference

Data Fields

- uint8_t **code**
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.28.1 Detailed Description

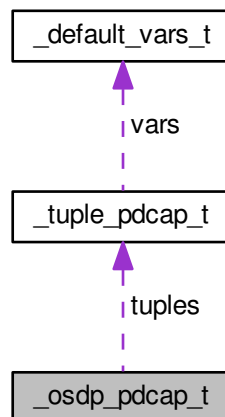
Definition at line 44 of file `osdp_nak_base.h`.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_nak_base.h`

8.29 _osdp_pdcap_t Struct Reference

Collaboration diagram for `_osdp_pdcap_t`:



Data Fields

- int32_t **num_tuples**
- [tuple_pdcap_t](#) **tuples** [MAX_PDCAP_TUPLES]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.29.1 Detailed Description

Definition at line 37 of file `osdp_pdcap_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_pdcap_base.h](#)

8.30 `_osdp_rfauth_t` Struct Reference

Data Fields

- `uint8_t salt` [16]
- `uint8_t hmac` [4]
- `osdp_header_t * header`

8.30.1 Detailed Description

Definition at line 18 of file `osdp_rfauth_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_rfauth_base.h](#)

8.31 `_osdp_sample_buffer_t` Struct Reference

Data Fields

- `int32_t wr_ptr`
- `int32_t rd_ptr`
- `int32_t depth`
- `int32_t max_depth`
- `uint16_t * samples`

8.31.1 Detailed Description

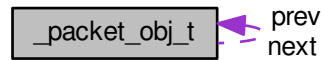
Definition at line 65 of file `osdp_ms100_mfgrep_base.h`.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep_base.h](#)

8.32 `_packet_obj_t` Struct Reference

Collaboration diagram for `_packet_obj_t`:



Data Fields

- `uint8_t` **buffer** [[COMMS_HUB_MAX_PACKET_SIZE](#)]
- `int` **length**
- `int` **interface**
- `struct _packet_obj_t *` **next**
- `struct _packet_obj_t *` **prev**

8.32.1 Detailed Description

Definition at line 25 of file `packet_linked_list.h`.

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

- [comms/packet_linked_list.h](#)

8.33 `_param_name_t` Struct Reference

```
#include <param_storage.h>
```

Data Fields

- `char` **name** [24]
- `uint8_t` **len**
- `uint8_t` **num**
- `uint8_t` **index**
- `void *` **data**

8.33.1 Detailed Description

A struct used to store the various naming and indexing values for a parameter

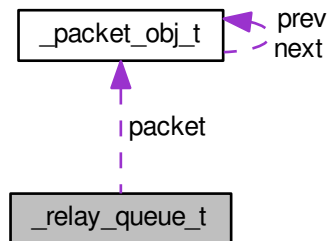
Definition at line 36 of file `param_storage.h`.

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

- [param_storage.h](#)

8.34 _relay_queue_t Struct Reference

Collaboration diagram for _relay_queue_t:



Data Fields

- int **depth**
- [packet_obj_t](#) * **packet**

8.34.1 Detailed Description

Definition at line 36 of file `packet_linked_list.h`.

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

- [comms/packet_linked_list.h](#)

8.35 _rsa_priv_t Struct Reference

Data Fields

- byte * **n**
- int32_t **n_length**
- byte * **e**
- int32_t **e_length**
- byte * **d**
- int32_t **d_length**

8.35.1 Detailed Description

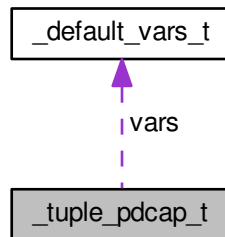
Definition at line 38 of file `func_link_authentication.cpp`.

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

- [test/func_link_authentication.cpp](#)

8.36 _tuple_pdcap_t Struct Reference

Collaboration diagram for _tuple_pdcap_t:



Data Fields

- `uint8_t function_code`

8.36.1 Detailed Description

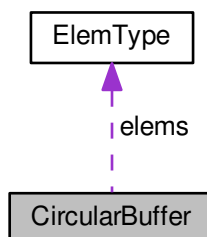
Definition at line 27 of file `osdp_pdcap_base.h`.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp\_pdcap\_base.h`

8.37 CircularBuffer Struct Reference

Collaboration diagram for CircularBuffer:



Data Fields

- int **size**
- int **start**
- int **end**
- int **xRead**
- [ElemType](#) **elems** [318]

8.37.1 Detailed Description

Definition at line 18 of file FIRcoeff_1.h.

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

- services/FIRcoeff_1.h

8.38 creal32_T Struct Reference

Data Fields

- real32_T **re**
- real32_T **im**

8.38.1 Detailed Description

Definition at line 564 of file tmwtypes.h.

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

- services/tmwtypes.h

8.39 creal64_T Struct Reference

Data Fields

- real64_T **re**
- real64_T **im**

8.39.1 Detailed Description

Definition at line 573 of file tmwtypes.h.

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

- services/tmwtypes.h

8.40 creal_T Struct Reference

Data Fields

- real_T **re**
- real_T **im**

8.40.1 Detailed Description

Definition at line 582 of file tmwtypes.h.

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

- services/tmwtypes.h

8.41 data_in Struct Reference

Data Fields

- float * **in**

8.41.1 Detailed Description

Definition at line 15 of file alarm.h.

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

- services/alarm.h

8.42 destructor Class Reference

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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[server_framework.h](#)

8.43 ElemType Struct Reference

Data Fields

- float **value**

8.43.1 Detailed Description

Definition at line 15 of file FIRcoeff_1.h.

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

- services/FIRcoeff_1.h

8.44 hdl_wavelet Struct Reference

A struct defining the parameters required to operate a single wavelet filter.

```
#include <wavelet.h>
```

Data Fields

- int32_t **n**
Number of block samples.
- float **memory** [5]
Memory associated with past edge samples.
- int32_t **mean**
The mean of the last input samples.

8.44.1 Detailed Description

A struct defining the parameters required to operate a single wavelet filter.

Definition at line 9 of file wavelet.h.

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

- services/wavelet.h

8.45 ip_settings_t Struct Reference

A structure defining IP settings.

```
#include <server_framework.h>
```

Data Fields

- common_utils::mutex **guard**
- bool **ipv6**
- uint8_t **addr** [16]
- uint8_t **mask** [16]
- uint8_t **gateway** [16]

8.45.1 Detailed Description

A structure defining IP settings.

Definition at line 15 of file server_framework.h.

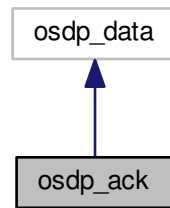
The documentation for this struct was generated from the following file:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[server_framework.h](#)

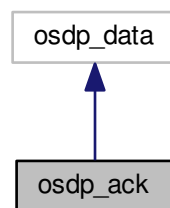
8.46 osdp_ack Class Reference

```
#include <osdp_ack.h>
```

Inheritance diagram for osdp_ack:



Collaboration diagram for osdp_ack:



Public Member Functions

- virtual `~osdp_ack()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.46.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide ACK responses.

Definition at line 16 of file `osdp_ack.h`.

8.46.2 Member Function Documentation

8.46.2.1 int32_t decode (uint8_t * *buffer*, int32_t *len*)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file osdp_ack.cpp.

8.46.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_ack.cpp.

Here is the call graph for this function:



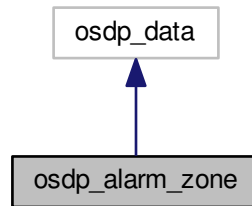
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/[osdp_ack.h](#)
- /home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_ack.cpp

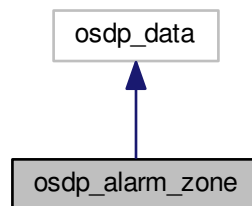
8.47 osdp_alarm_zone Class Reference

```
#include <osdp_alarm_zone.h>
```

Inheritance diagram for `osdp_alarm_zone`:



Collaboration diagram for `osdp_alarm_zone`:



Public Member Functions

- virtual `~osdp_alarm_zone()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_num_zones` (uint8_t num_zones)
- uint8_t `get_num_zones` ()
- void `set_zone` (uint8_t zone)
- uint8_t `get_zone` ()
- void `set_type` (uint8_t type)
- uint8_t `get_type` ()

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.47.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide ALARM_ZONE packets.

Definition at line 16 of file `osdp_alarm_zone.h`.

8.47.2 Member Function Documentation

8.47.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

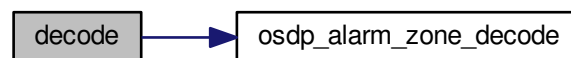
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_alarm_zone.cpp`.

Here is the call graph for this function:



8.47.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

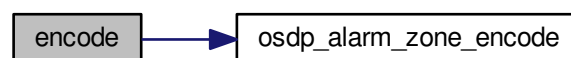
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file `osdp_alarm_zone.cpp`.

Here is the call graph for this function:



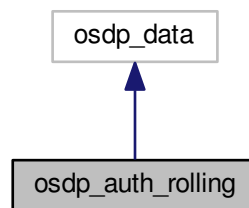
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[osdp_alarm_zone.h](#)
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_alarm_zone.cpp

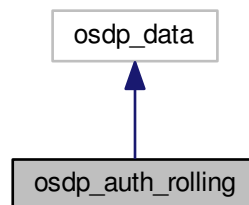
8.48 osdp_auth_rolling Class Reference

```
#include <osdp_auth_rolling.h>
```

Inheritance diagram for osdp_auth_rolling:



Collaboration diagram for osdp_auth_rolling:



Public Member Functions

- virtual [~osdp_auth_rolling](#) ()
Destructor.
- void [encode](#) (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [decode](#) (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static common_utils::osdp_data * [create](#) (osdp_header_t *hdr)

A factory method to create an osdp_data object.

8.48.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide an MS100 Configuration command.

Definition at line 15 of file osdp_auth_rolling.h.

8.48.2 Member Function Documentation

8.48.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

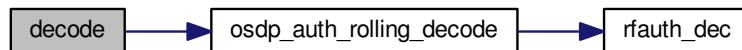
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 34 of file osdp_auth_rolling.cpp.

Here is the call graph for this function:



8.48.2.2 void encode (uint8_t * buffer, int32_t * len)

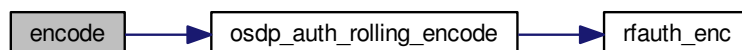
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 25 of file osdp_auth_rolling.cpp.

Here is the call graph for this function:



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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling.cpp](#)

8.49 osdp_auth_rolling_t Struct Reference

Data Fields

- uint32_t **timestamp**
- uint32_t **rolling**
- uint32_t **ctr**
- uint8_t * **key**
- osdp_header_t * **header**

8.49.1 Detailed Description

Definition at line 19 of file `osdp_auth_rolling_base.h`.

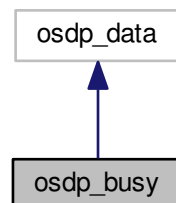
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling_base.h](#)

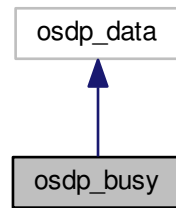
8.50 osdp_busy Class Reference

```
#include <osdp_busy.h>
```

Inheritance diagram for `osdp_busy`:



Collaboration diagram for osdp_busy:



Public Member Functions

- virtual `~osdp_busy()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.50.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide BUSY responses.

Definition at line 16 of file `osdp_busy.h`.

8.50.2 Member Function Documentation

8.50.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_busy.cpp`.

8.50.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_busy.cpp.

Here is the call graph for this function:



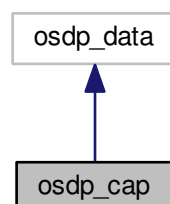
The documentation for this class was generated from the following files:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_busy.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_busy.cpp](#)

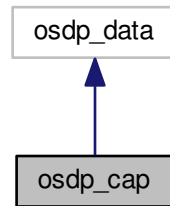
8.51 osdp_cap Class Reference

```
#include <osdp_cap.h>
```

Inheritance diagram for osdp_cap:



Collaboration diagram for osdp_cap:



Public Member Functions

- virtual `~osdp_cap ()`
Destructor.
- void `encode (uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- void `set_request_code (uint8_t code)`
Set the request code.
- uint8_t `get_request_code ()`

Static Public Member Functions

- static common_utils::osdp_data * `create (osdp_header_t *hdr)`
A factory method to create an osdp_data object.

8.51.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide CAP requests.

Definition at line 16 of file `osdp_cap.h`.

8.51.2 Member Function Documentation

8.51.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
---------------	---

<i>len</i>	The length of the input packet
------------	--------------------------------

Returns

The number of decoded bytes

Definition at line 29 of file osdp_cap.cpp.

Here is the call graph for this function:

**8.51.2.2 void encode (uint8_t * buffer, int32_t * len)**

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_cap.cpp.

Here is the call graph for this function:



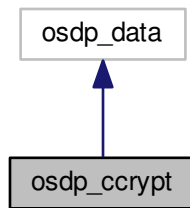
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_cap.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_cap.cpp

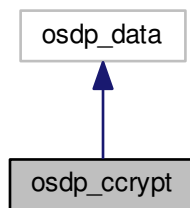
8.52 osdp_ccrypt Class Reference

```
#include <osdp_ccrypt.h>
```

Inheritance diagram for osdp_ccrypt:



Collaboration diagram for osdp_ccrypt:



Public Member Functions

- virtual `~osdp_ccrypt()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- `osdp_ccrypt_t * get_ccrypt()`
Setter and getter for the parameters.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.52.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to transmit random challenge data.

Definition at line 16 of file `osdp_ccrypt.h`.

8.52.2 Member Function Documentation

8.52.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 36 of file `osdp_ccrypt.cpp`.

Here is the call graph for this function:



8.52.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 24 of file `osdp_ccrypt.cpp`.

Here is the call graph for this function:



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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ccrypt.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ccrypt.cpp](#)

8.53 osdp_ccrypt_t Struct Reference

Data Fields

- bool **is_reply**
- uint8_t **cuid** [8]
- uint8_t **random** [8]
- uint8_t **cryptogram** [16]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.53.1 Detailed Description

Definition at line 20 of file [osdp_ccrypt_base.h](#).

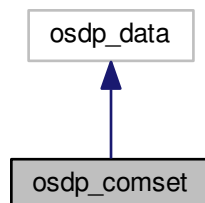
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ccrypt_base.h](#)

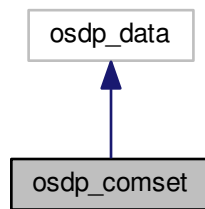
8.54 osdp_comset Class Reference

```
#include <osdp_comset.h>
```

Inheritance diagram for `osdp_comset`:



Collaboration diagram for osdp_comset:



Public Member Functions

- virtual `~osdp_comset()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_comset` (bool comset)
Setter and getter for the parameters.
- void `set_addr` (uint8_t addr)
- void `set_baud` (uint32_t baud)
- bool `get_comset` ()
- uint8_t `get_addr` ()
- uint32_t `get_baud` ()

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.54.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide PD Communication Configuration commands.

Definition at line 16 of file `osdp_comset.h`.

8.54.2 Member Function Documentation

8.54.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

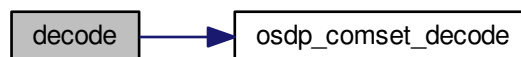
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 33 of file osdp_comset.cpp.

Here is the call graph for this function:

8.54.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

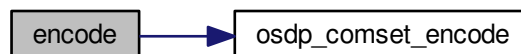
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 24 of file osdp_comset.cpp.

Here is the call graph for this function:



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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_comset.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_comset.cpp

8.55 osdp_comset_t Struct Reference

Data Fields

- bool **comset**

- uint8_t **addr**
- uint32_t **baud**
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.55.1 Detailed Description

Definition at line 20 of file osdp_comset_base.h.

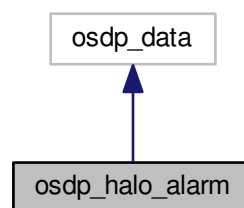
The documentation for this struct was generated from the following file:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[osdp_comset_base.h](#)

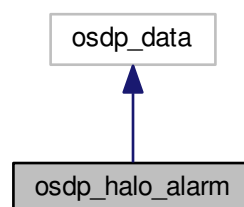
8.56 osdp_halo_alarm Class Reference

```
#include <osdp_halo_alarm.h>
```

Inheritance diagram for osdp_halo_alarm:



Collaboration diagram for osdp_halo_alarm:



Public Member Functions

- virtual [~osdp_halo_alarm](#) ()

Destructor.

- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_alarms` (network_alarms_t *alarm)
Setter and getter for the parameters.
- network_alarms_t * `get_alarms` ()

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.56.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide Halo alarm information.

Definition at line 16 of file osdp_halo_alarm.h.

8.56.2 Member Function Documentation

8.56.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

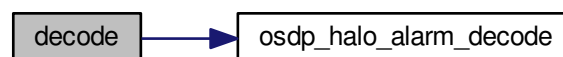
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 33 of file osdp_halo_alarm.cpp.

Here is the call graph for this function:



8.56.2.2 void encode (uint8_t * buffer, int32_t * len)

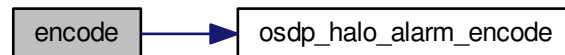
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 24 of file `osdp_halo_alarm.cpp`.

Here is the call graph for this function:



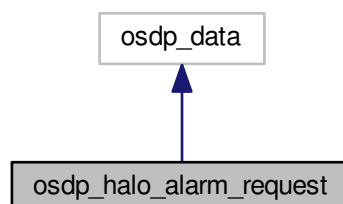
The documentation for this class was generated from the following files:

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp\_halo\_alarm.h`
- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm.cpp`

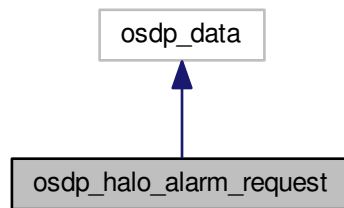
8.57 osdp_halo_alarm_request Class Reference

```
#include <osdp_halo_alarm.h>
```

Inheritance diagram for `osdp_halo_alarm_request`:



Collaboration diagram for `osdp_halo_alarm_request`:



Public Member Functions

- virtual `~osdp_halo_alarm_request()`
Destructor.
- void `encode` (`uint8_t *buffer`, `int32_t *len`)
Translate the data structure to a packet for transmission.
- `int32_t decode` (`uint8_t *buffer`, `int32_t len`)
Translate a received packet into a data structure.

Static Public Member Functions

- static `common_utils::osdp_data * create` (`osdp_header_t *hdr`)
A factory method to create a snap_status object.

8.57.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to request Halo alarms.

Definition at line 59 of file `osdp_halo_alarm.h`.

8.57.2 Member Function Documentation

8.57.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

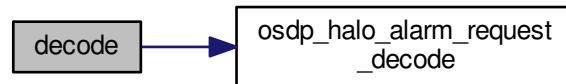
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 69 of file osdp_halo_alarm.cpp.

Here is the call graph for this function:

**8.57.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)**

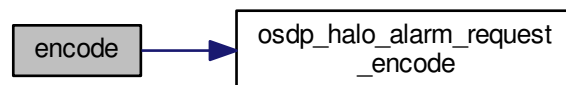
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 60 of file osdp_halo_alarm.cpp.

Here is the call graph for this function:

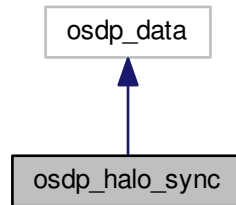


The documentation for this class was generated from the following files:

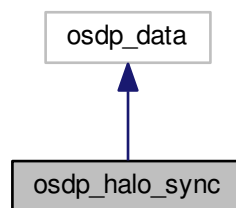
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm.cpp](#)

8.58 osdp_halo_sync Class Reference

Inheritance diagram for osdp_halo_sync:



Collaboration diagram for osdp_halo_sync:



Public Member Functions

- virtual `~osdp_halo_sync()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_type` (uint8_t type)
- uint8_t `get_type` ()
- void `set_payload` (uint8_t *payload, int32_t length)
- uint8_t * `get_payload` ()
- int32_t `get_payload_length` ()

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.58.1 Detailed Description

Definition at line 16 of file osdp_halo_sync.h.

8.58.2 Member Function Documentation

8.58.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

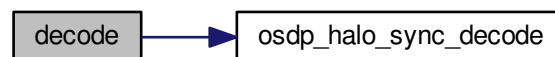
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file osdp_halo_sync.cpp.

Here is the call graph for this function:



8.58.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

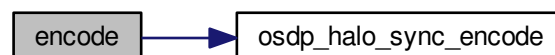
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_halo_sync.cpp.

Here is the call graph for this function:



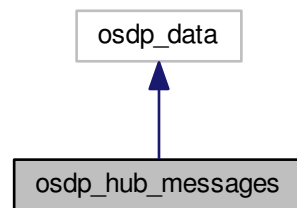
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_sync.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_sync.cpp

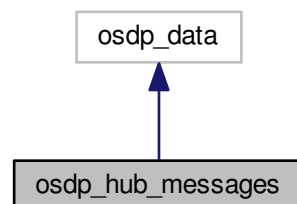
8.59 osdp_hub_messages Class Reference

```
#include <osdp_hub_messages.h>
```

Inheritance diagram for osdp_hub_messages:



Collaboration diagram for osdp_hub_messages:



Public Member Functions

- virtual `~osdp_hub_messages` ()
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_msg_type` (osdp_hub_e type)
Setter and getter for the parameters.
- osdp_hub_e `get_msg_type` ()
- void `set_enable` (bool enable)

- bool **get_enable** ()
- void **set_max_tcp** (uint8_t max_tcp)
- uint8_t **get_max_tcp** ()
- void **set_ip_addr** (bool ipv4, uint8_t *addr)
- void **get_ip_addr** (bool *ipv4, uint8_t *addr)
- void **set_tcp_port** (uint32_t socket)
- uint32_t **get_tcp_port** ()
- void **set_success** (bool success)
- bool **get_success** ()
- void **set_username** (std::string username)
- std::string **get_username** ()
- void **set_publickey** (std::string publickey)
- std::string **get_publickey** ()
- void **set_salt** (std::string salt)
- std::string **get_salt** ()
- void **set_cryptogram** (std::string cryptogram)
- std::string **get_cryptogram** ()

Static Public Member Functions

- static common_utils::osdp_data * **create** (osdp_header_t *hdr)
A factory method to create an [osdp_hub_messages](#) object.

8.59.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide OSDP Hub control and monitoring.

Definition at line 15 of file osdp_hub_messages.h.

8.59.2 Member Function Documentation

8.59.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 33 of file osdp_hub_messages.cpp.

8.59.2.2 void encode (uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

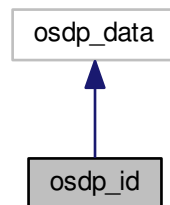
Definition at line 25 of file osdp_hub_messages.cpp.

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

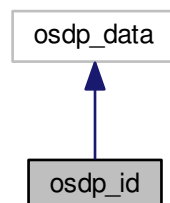
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_hub_messages.h](#)↔
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_hub_messages.cpp](#)↔

8.60 osdp_id Class Reference

Inheritance diagram for osdp_id:



Collaboration diagram for osdp_id:



Public Member Functions

- virtual [~osdp_id](#) ()
Destructor.
- void [encode](#) (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.

- `int32_t decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- `void set_extended (bool extended)`
Setter and getter for the parameters.
- `bool get_extended ()`

Static Public Member Functions

- `static common_utils::osdp_data * create (osdp_header_t *hdr)`
A factory method to create a snap_status object.

8.60.1 Detailed Description

Definition at line 16 of file `osdp_id.h`.

8.60.2 Member Function Documentation

8.60.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 32 of file `osdp_id.cpp`.

Here is the call graph for this function:



8.60.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

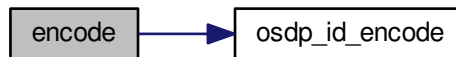
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 23 of file `osdp_id.cpp`.

Here is the call graph for this function:



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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_id.h`
- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_id.cpp`

8.61 `osdp_id_report_request` Class Reference

```
#include <osdp_id.h>
```

8.61.1 Detailed Description

A derived class of `snap_data`, this is used to store definable packets that can be encoded and decoded to provide device status information.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_id.h`

8.62 `osdp_id_t` Struct Reference

Data Fields

- `bool` **extended**
- `osdp_header_t *` **header**
- `SecurityCallback` **callback_enc**
- `SecurityCallback` **callback_dec**

8.62.1 Detailed Description

Definition at line 14 of file `osdp_id_base.h`.

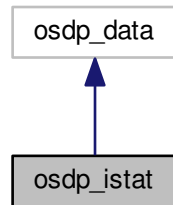
The documentation for this struct was generated from the following file:

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp\_id\_base.h`

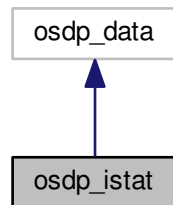
8.63 osdp_istat Class Reference

```
#include <osdp_istat.h>
```

Inheritance diagram for osdp_istat:



Collaboration diagram for osdp_istat:



Public Member Functions

- virtual `~osdp_istat()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.63.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide ISTAT requests.

Definition at line 16 of file `osdp_istat.h`.

8.63.2 Member Function Documentation

8.63.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_istat.cpp`.

8.63.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file `osdp_istat.cpp`.

Here is the call graph for this function:



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

- `/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_istat.h`
- `/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_istat.cpp`

8.64 `osdp_istat_t` Struct Reference

Data Fields

- `osdp_header_t * header`

8.64.1 Detailed Description

Definition at line 18 of file osdp_istat_base.h.

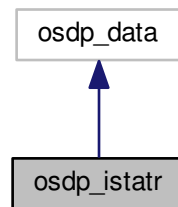
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istat_base.h](#)

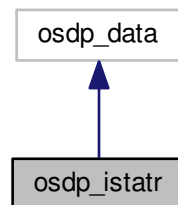
8.65 osdp_istatr Class Reference

```
#include <osdp_istatr.h>
```

Inheritance diagram for osdp_istatr:



Collaboration diagram for osdp_istatr:



Public Member Functions

- virtual [~osdp_istatr](#) ()
Destructor.
- void [encode](#) (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [decode](#) (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void [set_num_status](#) (int32_t num)

- `int32_t get_num_status ()`
- `void set_microwave (uint8_t alarm)`
- `uint8_t get_microwave ()`
- `void set_upper_deadzone (uint8_t alarm)`
- `uint8_t get_upper_deadzone ()`
- `void set_lower_deadzone (uint8_t alarm)`
- `uint8_t get_lower_deadzone ()`
- `void set_environmental (uint8_t alarm)`
- `uint8_t get_environmental ()`
- `void set_error (uint8_t alarm)`
- `uint8_t get_error ()`
- `void set_adc (int32_t index, uint8_t adc)`
- `uint8_t get_adc (int32_t index)`
- `void set_microwave_auth (uint8_t alarm)`
- `uint8_t get_microwave_auth ()`
- `void set_paired_tamper (uint8_t alarm)`
- `uint8_t get_paired_tamper ()`
- `void set_paired_power (uint8_t alarm)`
- `uint8_t get_paired_power ()`
- `void set_paired_upper_deadzone (uint8_t alarm)`
- `uint8_t get_paired_upper_deadzone ()`
- `void set_paired_lower_deadzone (uint8_t alarm)`
- `uint8_t get_paired_lower_deadzone ()`
- `void set_paired_environmental (uint8_t alarm)`
- `uint8_t get_paired_environmental ()`
- `void set_paired_error (uint8_t alarm)`
- `uint8_t get_paired_error ()`
- `void set_paired_adc (int32_t index, uint8_t adc)`
- `uint8_t get_paired_adc (int32_t index)`
- `void set_chain_alarm (uint8_t alarm)`
- `uint8_t get_chain_alarm ()`

Static Public Member Functions

- `static common_utils::osdp_data * create (osdp_header_t *hdr)`
A factory method to create an osdp_data object.

8.65.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide ISTATR responses.

Definition at line 16 of file `osdp_istatr.h`.

8.65.2 Member Function Documentation

8.65.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file osdp_istatr.cpp.

Here is the call graph for this function:

8.65.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_istatr.cpp.

Here is the call graph for this function:



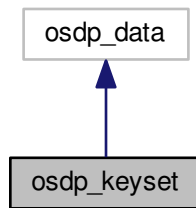
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istatr.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istatr.cpp

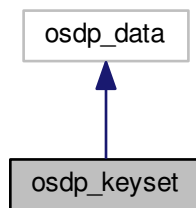
8.66 osdp_keyset Class Reference

```
#include <osdp_keyset.h>
```

Inheritance diagram for osdp_keyset:



Collaboration diagram for osdp_keyset:



Public Member Functions

- virtual `~osdp_keyset()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- `osdp_keyset_t * get_keyset()`
Setter and getter for the parameters.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.66.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to transmit a SCBK.

Definition at line 16 of file `osdp_keyset.h`.

8.66.2 Member Function Documentation

8.66.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

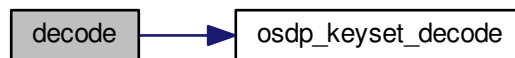
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 33 of file `osdp_keyset.cpp`.

Here is the call graph for this function:



8.66.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 24 of file `osdp_keyset.cpp`.

Here is the call graph for this function:



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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_keyset.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_keyset.cpp](#)

8.67 osdp_keyset_t Struct Reference

Data Fields

- uint8_t **type**
- uint8_t **length**
- uint8_t **key** [16]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.67.1 Detailed Description

Definition at line 19 of file `osdp_keyset_base.h`.

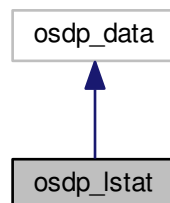
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_keyset_base.h](#)

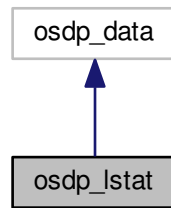
8.68 osdp_lstat Class Reference

```
#include <osdp_lstat.h>
```

Inheritance diagram for `osdp_lstat`:



Collaboration diagram for osdp_lstat:



Public Member Functions

- virtual `~osdp_lstat()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.68.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide LSTAT requests.

Definition at line 16 of file `osdp_lstat.h`.

8.68.2 Member Function Documentation

8.68.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_lstat.cpp`.

8.68.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

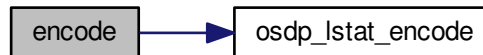
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_lstat.cpp.

Here is the call graph for this function:



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

- `/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_lstat.h`
- `/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_lstat.cpp`

8.69 osdp_lstat_t Struct Reference

Data Fields

- `osdp_header_t * header`

8.69.1 Detailed Description

Definition at line 18 of file `osdp_lstat_base.h`.

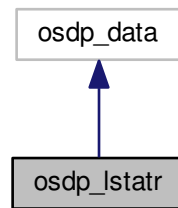
The documentation for this struct was generated from the following file:

- `/home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp\_lstat\_base.h`

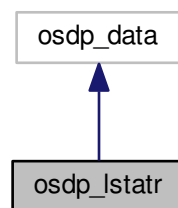
8.70 osdp_lstatr Class Reference

```
#include <osdp_lstatr.h>
```

Inheritance diagram for osdp_!statr:



Collaboration diagram for osdp_!statr:



Public Member Functions

- virtual `~osdp_!statr ()`
Destructor.
- void `encode (uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- void `set_power (uint8_t power)`
- uint8_t `get_power ()`
- void `set_tamper (uint8_t tamper)`
- uint8_t `get_tamper ()`

Static Public Member Functions

- static `common_utils::osdp_data * create (osdp_header_t *hdr)`
A factory method to create an osdp_data object.

8.70.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide LSTATR responses.

Definition at line 16 of file `osdp_Istatr.h`.

8.70.2 Member Function Documentation

8.70.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_Istatr.cpp`.

Here is the call graph for this function:



8.70.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file `osdp_Istatr.cpp`.

Here is the call graph for this function:



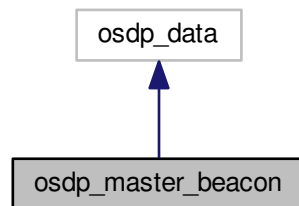
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istatr.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istatr.cpp

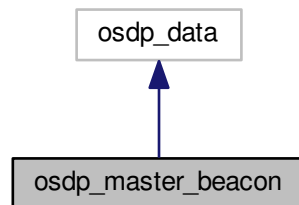
8.71 osdp_master_beacon Class Reference

```
#include <osdp_halo_sync.h>
```

Inheritance diagram for osdp_master_beacon:



Collaboration diagram for osdp_master_beacon:



Public Member Functions

- virtual `~osdp_master_beacon` ()
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_serial` (uint32_t serial)
- uint32_t `get_serial` ()
- void `set_addr` (uint8_t addr)
- uint8_t `get_addr` ()

- void **set_period** (uint8_t period)
- uint8_t **get_period** ()
- void **set_count** (uint8_t count)
- uint8_t **get_count** ()
- void **set_capabilities** (uint8_t cap)
- uint8_t **get_capabilities** ()

Static Public Member Functions

- static common_utils::osdp_data * **create** (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.71.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide HALO_SYNC packets.

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide LSTATR responses.

Definition at line 16 of file osdp_master_beacon.h.

8.71.2 Member Function Documentation

8.71.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

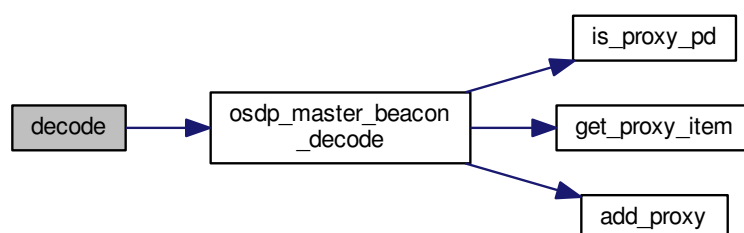
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file osdp_master_beacon.cpp.

Here is the call graph for this function:



8.71.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

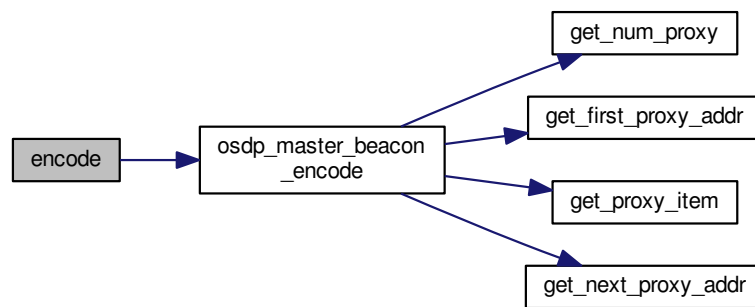
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file `osdp_master_beacon.cpp`.

Here is the call graph for this function:



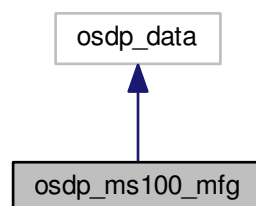
The documentation for this class was generated from the following files:

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_master_beacon.h`↔
- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_master_beacon.cpp`↔

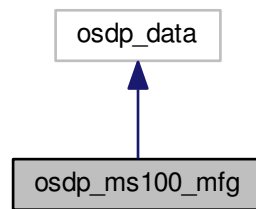
8.72 osdp_ms100_mfg Class Reference

```
#include <osdp_ms100_mfg.h>
```

Inheritance diagram for `osdp_ms100_mfg`:



Collaboration diagram for `osdp_ms100_mfg`:



Public Member Functions

- virtual `~osdp_ms100_mfg()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_mfg_type` (osdp_mfg_e type)
Setter and getter for the parameters.
- osdp_mfg_e `get_mfg_type` ()
- void `set_debug_enable` (uint8_t enable)
- uint8_t `get_debug_enable` ()
- void `set_vendor` (uint8_t *vendor)
- uint8_t * `get_vendor` ()
- void `set_operating_mode` (uint8_t operating_mode)
- void `set_channel` (uint8_t channel)
- void `set_range` (uint32_t range)
- void `set_sensitivity` (uint32_t sensitivity)
- void `set_dz_upper` (bool enable)
- void `set_dz_upper_sensitivity` (uint8_t sensitivity)
- void `set_dz_lower` (bool enable)
- void `set_dz_lower_sensitivity` (uint8_t sensitivity)
- bool `get_dz_upper` ()
- uint8_t `get_dz_upper_sensitivity` ()
- bool `get_dz_lower` ()
- uint8_t `get_dz_lower_sensitivity` ()
- void `set_relay_1_mode` (uint8_t relay_1_mode)
- void `set_relay_2_mode` (uint8_t relay_2_mode)
- void `set_relay_1_dur` (uint8_t relay_1_dur)
- void `set_relay_2_dur` (uint8_t relay_2_dur)
- void `set_alarm_class` (bool *classification)
- void `set_adc` (bool *enable)
- uint8_t `get_operating_mode` ()
- uint8_t `get_channel` ()
- uint32_t `get_range` ()

- uint32_t **get_sensitivity** ()
- uint8_t **get_relay_1_mode** ()
- uint8_t **get_relay_2_mode** ()
- uint8_t **get_relay_1_dur** ()
- uint8_t **get_relay_2_dur** ()
- bool * **get_alarm_class** ()
- bool * **get_adc** ()
- void **set_ipv4_address** (uint8_t *addr)
- void **set_ipv4_netmask** (uint8_t *addr)
- void **set_ipv4_gateway** (uint8_t *addr)
- void **set_ipv4_server_address** (uint8_t *addr)
- void **set_server_port** (uint32_t port)
- uint8_t * **get_ipv4_address** ()
- uint8_t * **get_ipv4_netmask** ()
- uint8_t * **get_ipv4_gateway** ()
- uint8_t * **get_ipv4_server_address** ()
- uint32_t **get_server_port** ()
- void **set_ipv4_dhcp_autoip** (uint8_t enable)
- uint8_t **get_ipv4_dhcp_autoip** ()
- void **set_enable_alarms** (bool enable)
- void **set_alarm_disable_timeout** (uint8_t timeout)
- bool **get_alarm_enable** ()
- uint8_t **get_alarm_disable_timeout** ()
- void **set_fw_update_buffer** (uint8_t *buffer, int32_t len, int32_t index)
- int32_t **get_fw_update_buffer** (uint8_t *buffer, int32_t *index)
- void **set_osdp_scs_enable** (bool enable)
- bool **get_osdp_scs_enable** ()
- void **set_osdp_scs_duration** (uint16_t duration)
- uint16_t **get_osdp_scs_duration** ()
- void **set_pki** (uint8_t type, uint8_t *data, int32_t length)
- void **get_pki** (uint8_t *type, uint8_t *data, int32_t *length)
- void **set_relay_triggers** (uint16_t relay_1, uint16_t relay_2)
- uint16_t **get_relay_1_triggers** ()
- uint16_t **get_relay_2_triggers** ()
- void **set_debug_alarms** (uint16_t device_alarms)
- uint16_t **get_debug_alarms** ()
- void **set_algorithm** (uint8_t algorithm)
- uint8_t **get_algorithm** ()
- void **set_paired_mode** (uint8_t mode)
- uint8_t **get_paired_mode** ()
- void **set_paired_serial** (uint32_t serial)
- uint32_t **get_paired_serial** ()
- void **set_paired_key** (uint32_t *key)
- uint32_t * **get_paired_key** ()

Static Public Member Functions

- static common_utils::osdp_data * **create** (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.72.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide an MS100 Configuration command.

Definition at line 15 of file osdp_ms100_mfg.h.

8.72.2 Member Function Documentation

8.72.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

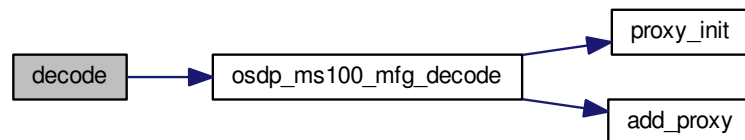
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 57 of file `osdp_ms100_mfg.cpp`.

Here is the call graph for this function:



8.72.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

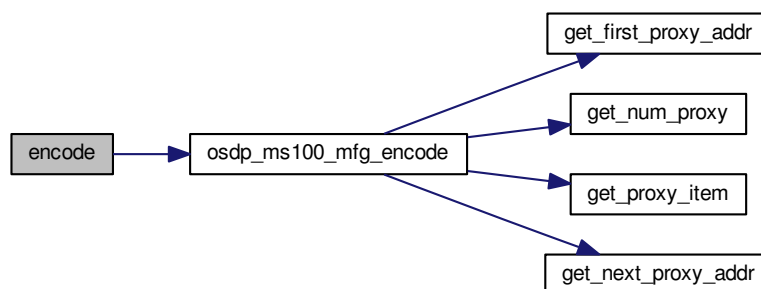
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 48 of file `osdp_ms100_mfg.cpp`.

Here is the call graph for this function:

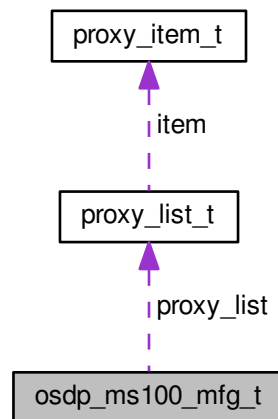


The documentation for this class was generated from the following files:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfg.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfg.cpp](#)

8.73 osdp_ms100_mfg_t Struct Reference

Collaboration diagram for osdp_ms100_mfg_t:



Data Fields

- `uint8_t vendor` [3]
- `osdp_mfg_e type`
- `uint8_t operating_mode`
- `uint8_t channel`
- `uint32_t range`
- `uint32_t sensitivity`
- `bool dz_upper`
- `uint8_t dz_upper_sensitivity`
- `bool dz_lower`
- `uint8_t dz_lower_sensitivity`
- `uint8_t relay_1_mode`
- `uint8_t relay_2_mode`
- `uint8_t relay_1_dur`
- `uint8_t relay_2_dur`
- `uint16_t relay_1_triggers`
- `uint16_t relay_2_triggers`
- `bool alarm_class` [6]
- `bool adc` [4]
- `uint8_t debug_msg_enable`
- `uint8_t ipv4_address` [4]
- `uint8_t ipv4_netmask` [4]

- uint8_t **ipv4_gateway** [4]
- uint8_t **ipv4_server_address** [4]
- uint32_t **server_port**
- bool **enable_alarms**
- uint8_t **alarm_disable_timeout**
- uint8_t * **pbuffer**
- int32_t **pbuffer_len**
- uint16_t **block_index**
- bool **osdp_scs_enable**
- uint16_t **osdp_scs_duration**
- uint8_t **algorithm**
- uint8_t **pki_type**
- uint8_t * **pki_data**
- int32_t **pki_length**
- uint8_t **paired_mode**
- uint32_t **paired_serial**
- uint32_t **paired_key** [4]
- uint8_t **scannable_beam_mode**
- uint8_t **ipv4_dhcp_autoip**
- [proxy_list_t](#) * **proxy_list**
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**
- uint16_t **debug_device_alarms**

8.73.1 Detailed Description

Definition at line 62 of file `osdp_ms100_mfg_base.h`.

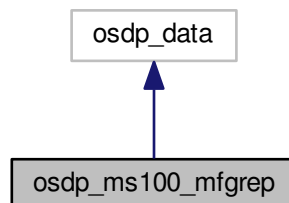
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfg_base.h](#)

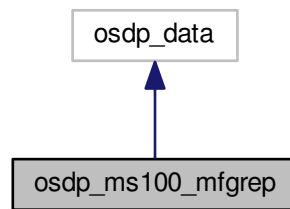
8.74 osdp_ms100_mfgrep Class Reference

```
#include <osdp_ms100_mfgrep.h>
```

Inheritance diagram for `osdp_ms100_mfgrep`:



Collaboration diagram for osdp_ms100_mfgrep:



Public Member Functions

- virtual `~osdp_ms100_mfgrep ()`
Destructor.
- void `encode (uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- void `set_mfgrep_type (osdp_mfgrep_e type)`
Setter and getter for the parameters.
- osdp_mfgrep_e `get_mfgrep_type ()`
- void `set_debug_enable (uint8_t enable)`
- uint8_t `get_debug_enable ()`
- void `set_debug_msg (char *buffer, int32_t n)`
- char * `get_debug_msg ()`
- void `set_debug_msg_level (uint8_t level)`
- int32_t `get_debug_msg_level ()`
- void `set_debug_msg_file (char *buffer, int32_t n)`
- char * `get_debug_msg_file ()`
- void `set_debug_msg_function (char *buffer, int32_t n)`
- char * `get_debug_msg_function ()`
- void `set_debug_msg_timestamp (uint32_t timestamp)`
- uint32_t `get_debug_msg_timestamp ()`
- void `set_debug_msg_line (int32_t line)`
- int32_t `get_debug_msg_line ()`
- void `set_vendor (uint8_t *vendor)`
- uint8_t * `get_vendor ()`
- void `set_operating_mode (uint8_t operating_mode)`
- void `set_channel (uint8_t channel)`
- void `set_range (uint32_t range)`
- void `set_sensitivity (uint32_t sensitivity)`
- void `set_relay_1_mode (uint8_t relay_1_mode)`
- void `set_relay_2_mode (uint8_t relay_2_mode)`
- void `set_relay_1_dur (uint8_t relay_1_dur)`
- void `set_relay_2_dur (uint8_t relay_2_dur)`
- void `set_alarm_class (bool *classification)`

- void **set_adc** (bool *enable)
- uint8_t **get_operating_mode** ()
- uint8_t **get_channel** ()
- uint32_t **get_range** ()
- uint32_t **get_sensitivity** ()
- void **set_dz_upper** (bool enable)
- void **set_dz_upper_sensitivity** (uint8_t sensitivity)
- void **set_dz_lower** (bool enable)
- void **set_dz_lower_sensitivity** (uint8_t sensitivity)
- bool **get_dz_upper_fitted** ()
- bool **get_dz_upper** ()
- uint8_t **get_dz_upper_sensitivity** ()
- bool **get_dz_lower_fitted** ()
- bool **get_dz_lower** ()
- uint8_t **get_dz_lower_sensitivity** ()
- uint8_t **get_relay_1_mode** ()
- uint8_t **get_relay_2_mode** ()
- uint8_t **get_relay_1_dur** ()
- uint8_t **get_relay_2_dur** ()
- bool * **get_alarm_class** ()
- bool * **get_adc** ()
- void **set_ipv4_address** (uint8_t *addr)
- void **set_ipv4_netmask** (uint8_t *addr)
- void **set_ipv4_gateway** (uint8_t *addr)
- void **set_ipv4_server_address** (uint8_t *addr)
- void **set_server_port** (uint32_t port)
- uint8_t * **get_ipv4_address** ()
- uint8_t * **get_ipv4_netmask** ()
- uint8_t * **get_ipv4_gateway** ()
- uint8_t * **get_ipv4_server_address** ()
- uint32_t **get_server_port** ()
- void **set_ipv4_dhcp_autoip** (uint8_t enable)
- uint8_t **get_ipv4_dhcp_autoip** ()
- void **set_enable_alarms** (bool enable)
- bool **get_alarm_enable** ()
- void **set_alarm_disable_timeout** (uint8_t timeout)
- uint8_t **get_alarm_disable_timeout** ()
- void **get_fw_update_wait** (uint8_t *wait)
- void **set_uptime** (uint32_t value)
- void **set_mem_core_free** (uint32_t value)
- void **set_mem_heap_fragments** (uint32_t value)
- void **set_mem_heap_free_total** (uint32_t value)
- void **set_num_threads** (uint32_t value)
- void **set_fs_mounted** (bool value)
- void **set_state** (bool value)
- void **set_thresh_high** (uint32_t value)
- void **set_thresh_low** (uint32_t value)
- void **set_filtered_signal** (uint32_t value)
- void **set_dir_path_rssi** (uint32_t value)
- void **set_threshold** (uint32_t value)
- void **set_thresh_ok** (bool value)
- void **set_tx_active** (bool value)
- uint32_t **get_uptime** ()
- uint32_t **get_mem_core_free** ()
- uint32_t **get_mem_heap_fragments** ()

- uint32_t **get_mem_heap_free_total** ()
- uint32_t **get_num_threads** ()
- bool **get_fs_mounted** ()
- uint32_t **get_state** ()
- uint32_t **get_thresh_high** ()
- uint32_t **get_thresh_low** ()
- uint32_t **get_filtered_signal** ()
- uint32_t **get_dir_path_rssi** ()
- uint32_t **get_threshold** ()
- bool **get_thresh_ok** ()
- bool **get_tx_active** ()
- void **set_samples** (osdp_sample_buffer_t *buffer)
- osdp_sample_buffer_t * **get_samples** ()
- bool **is_more_samples** ()
- void **set_osdp_scs_enable** (bool enable)
- bool **get_osdp_scs_enable** ()
- void **set_osdp_scs_duration** (uint16_t duration)
- uint16_t **get_osdp_scs_duration** ()
- void **set_gps_fix** (bool fix)
- void **set_date** (int32_t day, int32_t month, int32_t year)
- void **set_time** (int32_t hour, int32_t min, int32_t sec)
- void **set_location** (float latitude, float longitude, float altitude)
- bool **get_gps_fix** ()
- void **get_date** (int32_t *day, int32_t *month, int32_t *year)
- void **get_time** (int32_t *hour, int32_t *min, int32_t *sec)
- void **get_location** (float *latitude, float *longitude, float *altitude)
- void **set_pki** (uint8_t type, uint8_t *data, int32_t length)
- void **get_pki** (uint8_t *type, uint8_t *data, int32_t *length)
- void **set_relay_triggers** (uint16_t relay_1, uint16_t relay_2)
- uint16_t **get_relay_1_triggers** ()
- uint16_t **get_relay_2_triggers** ()
- void **set_debug_alarms** (uint16_t device_alarms)
- uint16_t **get_debug_alarms** ()
- void **set_algorithm** (uint8_t algorithm)
- uint8_t **get_algorithm** ()
- void **set_paired_mode** (uint8_t mode)
- uint8_t **get_paired_mode** ()
- void **set_paired_serial** (uint32_t serial)
- uint32_t **get_paired_serial** ()
- void **set_paired_key** (uint32_t *key)
- uint32_t * **get_paired_key** ()
- void **set_vigil_relay_fitted** (uint8_t fitted)
- uint8_t **get_vigil_relay_fitted** ()
- void **set_diagnostic** (uint32_t diagnostic)
- uint32_t **get_diagnostic** ()
- void **set_proxy_list** (proxy_list_t *proxy_list)
- proxy_list_t * **get_proxy_list** ()

Static Public Member Functions

- static common_utils::osdp_data * **create** (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.74.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide MS100 data requests.

Definition at line 16 of file `osdp_ms100_mfgrep.h`.

8.74.2 Member Function Documentation

8.74.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

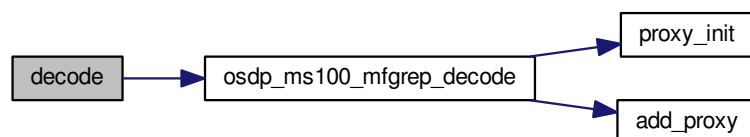
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 60 of file `osdp_ms100_mfgrep.cpp`.

Here is the call graph for this function:



8.74.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

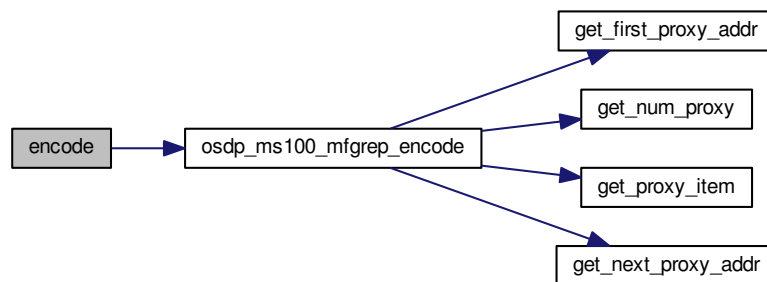
Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 51 of file `osdp_ms100_mfgrep.cpp`.

Here is the call graph for this function:

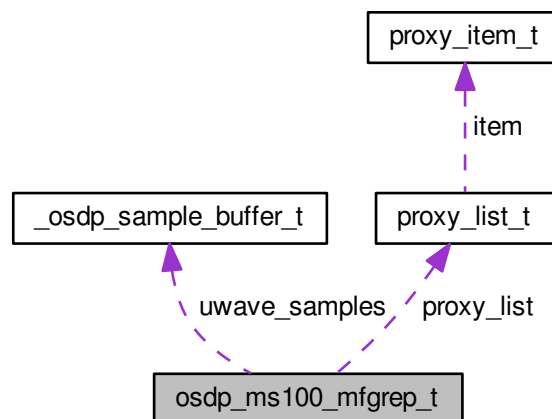


The documentation for this class was generated from the following files:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep.cpp](#)

8.75 osdp_ms100_mfgrep_t Struct Reference

Collaboration diagram for osdp_ms100_mfgrep_t:



Data Fields

- `uint8_t` **vendor** [3]
- `osdp_mfgrep_e` **type**
- `uint8_t` **operating_mode**

- `uint8_t channel`
- `uint32_t range`
- `uint32_t sensitivity`
- `bool dz_upper_fitted`
- `bool dz_upper`
- `uint8_t dz_upper_sensitivity`
- `bool dz_lower_fitted`
- `bool dz_lower`
- `uint8_t dz_lower_sensitivity`
- `uint8_t relay_1_mode`
- `uint8_t relay_2_mode`
- `uint8_t relay_1_dur`
- `uint8_t relay_2_dur`
- `uint16_t relay_1_triggers`
- `uint16_t relay_2_triggers`
- `bool alarm_class` [6]
- `bool adc` [4]
- `uint8_t debug_msg_enable`
- `uint8_t debug_msg_level`
- `uint32_t debug_msg_timestamp`
- `uint8_t debug_msg_file` [32]
- `uint8_t debug_msg_function` [32]
- `uint16_t debug_msg_line`
- `uint8_t debug_msg` [MAX_DEBUG_PAYLOAD]
- `int32_t debug_msg_length`
- `uint8_t ipv4_address` [4]
- `uint8_t ipv4_netmask` [4]
- `uint8_t ipv4_gateway` [4]
- `uint8_t ipv4_server_address` [4]
- `uint32_t server_port`
- `bool enable_alarms`
- `uint8_t alarm_disable_timeout`
- `uint32_t uptime`
- `uint32_t mem_core_free`
- `uint32_t mem_heap_fragments`
- `uint32_t mem_heap_free_total`
- `uint32_t num_threads`
- `bool fs_mounted`
- `uint16_t block_index`
- `uint8_t block_wait`
- `uint32_t state`
- `uint32_t thresh_high`
- `uint32_t thresh_low`
- `uint32_t filtered_signal`
- `uint32_t threshold`
- `bool thresh_ok`
- `bool tx_active`
- `uint32_t dir_path_rssi`
- `osdp_sample_buffer_t * uwave_samples`
- `bool more_uwave_samples`
- `bool osdp_scs_enable`
- `uint16_t osdp_scs_duration`
- `bool gps_fix`
- `float gps_latitude`
- `float gps_longitude`

- float **gps_altitude**
- uint8_t **gps_date** [4]
- uint8_t **gps_time** [3]
- uint8_t **algorithm**
- uint8_t **pki_type**
- uint8_t * **pki_data**
- int32_t **pki_length**
- uint16_t **debug_device_alarms**
- uint8_t **paired_mode**
- uint32_t **paired_serial**
- uint32_t **paired_key** [4]
- uint8_t **vigil_relay_fitted**
- uint32_t **diagnostic**
- uint8_t **scannable_beam_mode**
- uint8_t **ipv4_dhcp_autoip**
- uint8_t **uwave_intruder_type**
- uint16_t **uwave_intruder_zone**
- uint8_t **uwave_intruder_direction**
- uint8_t **uwave_intruder_speed**
- [proxy_list_t](#) * **proxy_list**
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.75.1 Detailed Description

Definition at line 74 of file `osdp_ms100_mfgrep_base.h`.

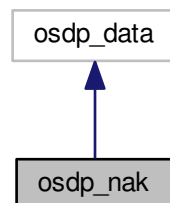
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep_base.h](#)

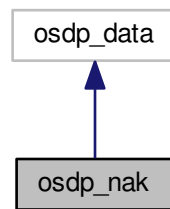
8.76 osdp_nak Class Reference

```
#include <osdp_nak.h>
```

Inheritance diagram for `osdp_nak`:



Collaboration diagram for osdp_nak:



Public Member Functions

- virtual `~osdp_nak()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- `osdp_nak_t * get_data()`
Return a pointer to the variables associated with NAK packets.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.76.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide NAK responses.

Definition at line 16 of file `osdp_nak.h`.

8.76.2 Member Function Documentation

8.76.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file osdp_nak.cpp.

Here is the call graph for this function:

**8.76.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)**

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file osdp_nak.cpp.

Here is the call graph for this function:



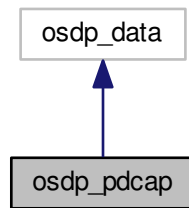
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_nak.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_nak.cpp

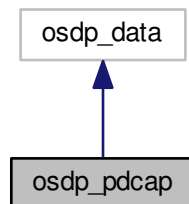
8.77 osdp_pdcap Class Reference

```
#include <osdp_pdcap.h>
```

Inheritance diagram for osdp_pdcap:



Collaboration diagram for osdp_pdcap:



Public Member Functions

- virtual `~osdp_pdcap ()`
Destructor.
- void `encode (uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- `osdp_pdcap_t * get_data ()`
Return a pointer to the data structure.

Static Public Member Functions

- static `common_utils::osdp_data * create (osdp_header_t *hdr)`
A factory method to create a snap_status object.

8.77.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide a Device Capabilities Report.

Definition at line 16 of file `osdp_pdcap.h`.

8.77.2 Member Function Documentation

8.77.2.1 int32_t decode (uint8_t * *buffer*, int32_t *len*)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 56 of file osdp_pdcap.cpp.

Here is the call graph for this function:



8.77.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 47 of file osdp_pdcap.cpp.

Here is the call graph for this function:



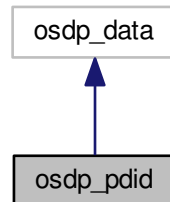
The documentation for this class was generated from the following files:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdcap.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdcap.cpp

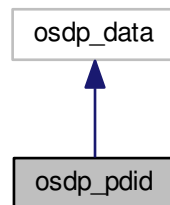
8.78 osdp_pdid Class Reference

```
#include <osdp_pdid.h>
```

Inheritance diagram for osdp_pdid:



Collaboration diagram for osdp_pdid:



Public Member Functions

- virtual `~osdp_pdid ()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- void `set_oui` (uint8_t *oui)
Setter and getter for the parameters.
- uint8_t * `get_oui` ()
- void `set_model_number` (uint8_t model)
- uint8_t `get_model_number` ()
- void `set_version` (uint8_t version)
- uint8_t `get_version` ()
- void `set_serial` (uint32_t serial)

- uint32_t **get_serial** ()
- void **set_firmware** (uint8_t *firmware)
- uint8_t * **get_firmware** ()
- osdp_pdid_t * **get_pdid** ()

Static Public Member Functions

- static common_utils::osdp_data * **create** (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.78.1 Detailed Description

A derived class of osdp_data, this is used to store definable packets that can be encoded and decoded to provide a Device Identification Report.

Definition at line 16 of file osdp_pdid.h.

8.78.2 Member Function Documentation

8.78.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 35 of file osdp_pdid.cpp.

Here is the call graph for this function:



8.78.2.2 void encode (uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 26 of file osdp_pdid.cpp.

Here is the call graph for this function:



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

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdid.h
- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdid.cpp

8.79 osdp_pdid_t Struct Reference

Data Fields

- uint8_t **oui** [3]
- uint8_t **model_number**
- uint8_t **version**
- uint32_t **serial**
- uint8_t **firmware** [3]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.79.1 Detailed Description

Definition at line 19 of file osdp_pdid_base.h.

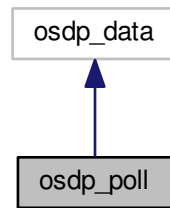
The documentation for this struct was generated from the following file:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/[osdp_pdid_base.h](#)

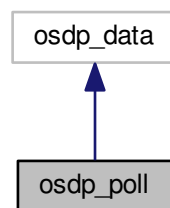
8.80 osdp_poll Class Reference

```
#include <osdp_poll.h>
```

Inheritance diagram for osdp_poll:



Collaboration diagram for osdp_poll:



Public Member Functions

- virtual `~osdp_poll()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

Static Public Member Functions

- static `common_utils::osdp_data * create` (osdp_header_t *hdr)
A factory method to create an osdp_data object.

8.80.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to provide poll requests.

Definition at line 16 of file `osdp_poll.h`.

8.80.2 Member Function Documentation

8.80.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 29 of file `osdp_poll.cpp`.

8.80.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 20 of file `osdp_poll.cpp`.

Here is the call graph for this function:



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

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_poll.h`
- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_poll.cpp`

8.81 `osdp_poll_t` Struct Reference

Data Fields

- `osdp_header_t * header`

8.81.1 Detailed Description

Definition at line 18 of file `osdp_poll_base.h`.

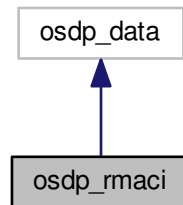
The documentation for this struct was generated from the following file:

- `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_poll_base.h`

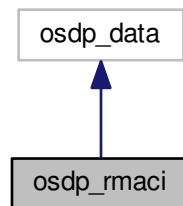
8.82 osdp_rmaci Class Reference

```
#include <osdp_rmaci.h>
```

Inheritance diagram for osdp_rmaci:



Collaboration diagram for osdp_rmaci:



Public Member Functions

- virtual `~osdp_rmaci()`
Destructor.
- void `encode` (uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t `decode` (uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- `osdp_rmaci_t` * `get_rmaci()`
Setter and getter for the parameters.

Static Public Member Functions

- static common_utils::osdp_data * `create` (osdp_header_t *hdr)
A factory method to create a snap_status object.

8.82.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to transmit random challenge data.

Definition at line 16 of file `osdp_rmaci.h`.

8.82.2 Member Function Documentation

8.82.2.1 `int32_t decode ( uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 31 of file `osdp_rmaci.cpp`.

Here is the call graph for this function:



8.82.2.2 `void encode ( uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 22 of file `osdp_rmaci.cpp`.

Here is the call graph for this function:



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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_rmaci.h](#)
- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_rmaci.cpp](#)

8.83 osdp_rmaci_t Struct Reference

Data Fields

- uint8_t **mac** [16]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.83.1 Detailed Description

Definition at line 19 of file `osdp_rmaci_base.h`.

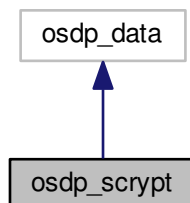
The documentation for this struct was generated from the following file:

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_rmaci_base.h](#)

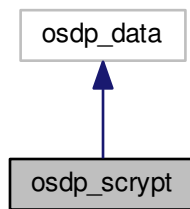
8.84 osdp_scrypt Class Reference

```
#include <osdp_scrypt.h>
```

Inheritance diagram for `osdp_scrypt`:



Collaboration diagram for osdp_scrypt:



Public Member Functions

- virtual `~osdp_scrypt ()`
Destructor.
- void `encode (uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `decode (uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- `osdp_scrypt_t * get_scrypt ()`
Setter and getter for the parameters.

Static Public Member Functions

- static `common_utils::osdp_data * create (osdp_header_t *hdr)`
A factory method to create a snap_status object.

8.84.1 Detailed Description

A derived class of `osdp_data`, this is used to store definable packets that can be encoded and decoded to transmit random challenge data.

Definition at line 16 of file `osdp_scrypt.h`.

8.84.2 Member Function Documentation

8.84.2.1 int32_t decode (uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
---------------	---

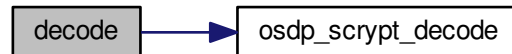
<i>len</i>	The length of the input packet
------------	--------------------------------

Returns

The number of decoded bytes

Definition at line 31 of file osdp_scrypt.cpp.

Here is the call graph for this function:

**8.84.2.2 void encode (uint8_t * *buffer*, int32_t * *len*)**

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 22 of file osdp_scrypt.cpp.

Here is the call graph for this function:



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

- /home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/[osdp_scrypt.h](#)
- /home/neil/Projects/temp/velox-project/proprietary/rtoS/apps/chibios/utilities/msp/src/osdp_scrypt.cpp

8.85 osdp_scrypt_t Struct Reference

Data Fields

- uint8_t **cryptogram** [16]
- osdp_header_t * **header**
- SecurityCallback **callback_enc**
- SecurityCallback **callback_dec**

8.85.1 Detailed Description

Definition at line 19 of file osdp_scrypt_base.h.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_scrypt_base.h](#)

8.86 proxy_item_t Struct Reference

Data Fields

- uint32_t **serial**
- uint8_t **interface**
- proxy_status_e **status**

8.86.1 Detailed Description

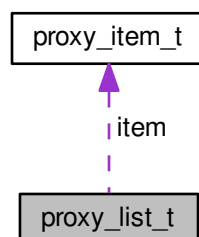
Definition at line 23 of file halo_proxy.h.

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

- [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/halo_proxy.h](#)

8.87 proxy_list_t Struct Reference

Collaboration diagram for proxy_list_t:



Data Fields

- [proxy_item_t](#) **item** [127]
- uint8_t **num_items**
- uint32_t **map** [4]

8.87.1 Detailed Description

Definition at line 30 of file halo_proxy.h.

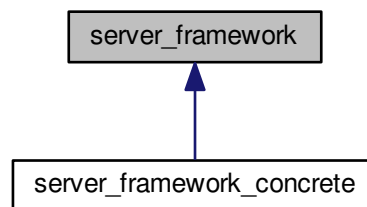
The documentation for this struct was generated from the following file:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/halo_proxy.h

8.88 server_framework Class Reference

```
#include <server_framework.h>
```

Inheritance diagram for server_framework:



Public Member Functions

- virtual bool **add_packet** (common_utils::CreateOsdpData create_func)=0
- virtual common_utils::osdp_data * **get_packet** (int32_t id)=0
- virtual common_utils::osdp_packets * **get_ctrl** ()=0
- virtual void **set_uptime** (uint32_t uptime)=0
- virtual uint32_t **get_uptime** ()=0
- virtual void **set_ip_settings** (bool ipv6, uint8_t *addr, uint8_t *mask, uint8_t *gateway)=0
- virtual ip_settings_t * **get_ip_settings** ()=0
- virtual void **set_serial** (uint32_t serial)=0
- virtual uint32_t **get_serial** ()=0
- virtual void **set_osdp_addr** (uint8_t addr)=0
- virtual uint8_t **get_osdp_addr** ()=0
- virtual void **set_osdp_baud** (uint32_t baud)=0
- virtual uint32_t **get_osdp_baud** ()=0

8.88.1 Detailed Description

The [server_framework](#) class provides a pure abstract interface class used to distribute settings throughout the system

Definition at line 28 of file server_framework.h.

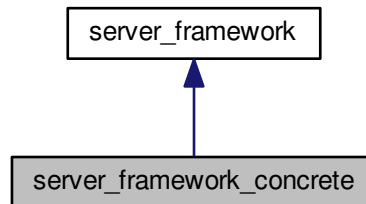
The documentation for this class was generated from the following file:

- /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/server_framework.h

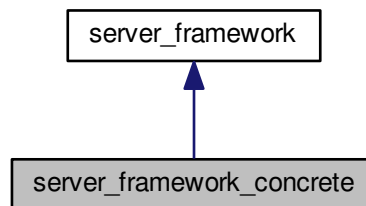
8.89 server_framework_concrete Class Reference

```
#include <server_framework.h>
```

Inheritance diagram for server_framework_concrete:



Collaboration diagram for server_framework_concrete:



Public Member Functions

- [server_framework_concrete](#) ()
Class constructor.
- bool [add_packet](#) (common_utils::CreateOsdpData create_func)
A method used to register a packet type within the system.
- common_utils::osdp_data * [get_packet](#) (int32_t id)
Retrieve a packet processing object.
- common_utils::osdp_packets * [get_ctrl](#) ()
- void [set_uptime](#) (uint32_t uptime)
- uint32_t [get_uptime](#) ()
- void [set_ip_settings](#) (bool ipv6, uint8_t *addr, uint8_t *mask, uint8_t *gateway)
Configure the IP settings of this device.
- [ip_settings_t](#) * [get_ip_settings](#) ()
Return a pointer to the IP settings.
- void [set_serial](#) (uint32_t serial)
Set the serial number to be used by this PD.

- uint32_t [get_serial](#) ()
Get the serial number to be used by this PD.
- void [set_osdp_addr](#) (uint8_t addr)
Set the address to be used by this PD over OSDP.
- uint8_t [get_osdp_addr](#) ()
Return the assigned OSDP address.
- void [set_osdp_baud](#) (uint32_t baud)
- uint32_t [get_osdp_baud](#) ()

8.89.1 Detailed Description

A concrete implementation of the [server_framework](#) class that implements the interface

Definition at line 49 of file server_framework.h.

8.89.2 Member Function Documentation

8.89.2.1 bool add_packet (common_utils::CreateOsdpData *create_func*) [inline], [virtual]

A method used to register a packet type within the system.

Parameters

<i>create_func</i>	A function pointer to a factory method used to create a packet processing object
--------------------	--

Returns

Returns true upon success, false on failure

Implements [server_framework](#).

Definition at line 61 of file server_framework.h.

8.89.2.2 ip_settings_t* get_ip_settings () [inline], [virtual]

Return a pointer to the IP settings.

Returns

A pointer to an [ip_settings_t](#) struct maintained by the framework

Implements [server_framework](#).

Definition at line 97 of file server_framework.h.

8.89.2.3 common_utils::osdp_data* get_packet (int32_t *id*) [inline], [virtual]

Retrieve a packet processing object.

Parameters

<i>id</i>	The identification number of the packet type
-----------	--

Returns

Returns an object if it exists, 0 otherwise

Implements [server_framework](#).

Definition at line 74 of file server_framework.h.

8.89.2.4 `void set_ip_settings ( bool ipv6, uint8_t * addr, uint8_t * mask, uint8_t * gateway )` `[inline], [virtual]`

Configure the IP settings of this device.

Parameters

<i>ip</i>	A pointer to a structure containing the required IP settings
-----------	--

Implements [server_framework](#).

Definition at line 85 of file `server_framework.h`.

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

- `/home/neil/Projects/temp/velox-project/proprietary/rtdos/apps/chibios/utilities/msp/src/server\_framework.h`

8.90 SHA256_CTX Struct Reference

Data Fields

- BYTE **data** [64]
- WORD **datalen**
- WORD **bitlen**
- WORD **state** [8]

8.90.1 Detailed Description

Definition at line 22 of file `sha256.h`.

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

- `crypto/sha256.h`

8.91 statisticsC Struct Reference

Data Fields

- float **mean**
- float **max**
- float **min**

8.91.1 Detailed Description

Definition at line 7 of file `alarm.h`.

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

- `services/alarm.h`

8.92 ThreshStuff Struct Reference

Data Fields

- float **thresh_low**
- float **thresh_high**

8.92.1 Detailed Description

Definition at line 20 of file alarm.h.

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

- services/alarm.h

8.93 wrapper_msg_base Struct Reference

8.93.1 Detailed Description

Definition at line 142 of file fatfsWrapper.c.

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

- sdcard/[fatfsWrapper.c](#)

8.94 wrapper_msg_pDIRpFILINFO Struct Reference

Data Fields

- _wrapper_msg_base DIR * **dirp**
- FILINFO * **filinfo**

8.94.1 Detailed Description

Definition at line 191 of file fatfsWrapper.c.

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

- sdcard/[fatfsWrapper.c](#)

8.95 wrapper_msg_pDIRpTCHAR Struct Reference

Data Fields

- _wrapper_msg_base DIR * **dirp**
- TCHAR * **string**

8.95.1 Detailed Description

Definition at line 184 of file fatfsWrapper.c.

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

- sdcard/[fatfsWrapper.c](#)

8.96 wrapper_msg_pFIL Struct Reference

Data Fields

- `_wrapper_msg_base FIL * filep`

8.96.1 Detailed Description

Definition at line 178 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.97 wrapper_msg_pFILpTCHARvBYTE Struct Reference

Data Fields

- `_wrapper_msg_base FIL * filep`
- `TCHAR * string`
- `BYTE byte`

8.97.1 Detailed Description

Definition at line 154 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.98 wrapper_msg_pFILpVOIDvUINTpUINT Struct Reference

Data Fields

- `_wrapper_msg_base FIL * filep`
- `void * voidp`
- `UINT uint`
- `UINT * uintp`

8.98.1 Detailed Description

Definition at line 162 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.99 wrapper_msg_pFILvDWORD Struct Reference

Data Fields

- `_wrapper_msg_base FIL * filep`

- DWORD **dword**

8.99.1 Detailed Description

Definition at line 171 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.100 wrapper_msg_pTCHAR Struct Reference

Data Fields

- _wrapper_msg_base TCHAR * **string**

8.100.1 Detailed Description

Definition at line 215 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.101 wrapper_msg_pTCHARpDWORDppFATFS Struct Reference

Data Fields

- _wrapper_msg_base TCHAR * **string**
- DWORD * **dwordp**
- FATFS ** **fatfspp**

8.101.1 Detailed Description

Definition at line 207 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.102 wrapper_msg_pTCHARpFIL Struct Reference

Data Fields

- _wrapper_msg_base TCHAR * **string**
- FIL * **filep**

8.102.1 Detailed Description

Definition at line 273 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.103 wrapper_msg_pTCHARpFILINFO Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR * **string**
- `FILINFO` * **filinfop**

8.103.1 Detailed Description

Definition at line 199 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.104 wrapper_msg_pTCHARpTCHAR Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR * **string1**
- TCHAR * **string2**

8.104.1 Detailed Description

Definition at line 230 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.105 wrapper_msg_pTCHARvBYTEvBYTE Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR * **string**
- BYTE **byte1**
- BYTE **byte2**

8.105.1 Detailed Description

Definition at line 222 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

8.106 wrapper_msg_pTCHARvINTpFILpTCHAR Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR * **string**
- `int` **n**
- `FIL` * **filep**
- TCHAR * **string2**

8.106.1 Detailed Description

Definition at line 280 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.107 wrapper_msg_pTCHARvUINT Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR * **string**
- `UINT` **uint**

8.107.1 Detailed Description

Definition at line 243 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.108 wrapper_msg_vBYTE Struct Reference

Data Fields

- `_wrapper_msg_base` BYTE **byte**

8.108.1 Detailed Description

Definition at line 237 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.109 wrapper_msg_vBYTEpDWORDpVOID Struct Reference

Data Fields

- `_wrapper_msg_base` BYTE **byte**

- DWORD * **dwordp**
- void * **voidp**

8.109.1 Detailed Description

Definition at line 258 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.110 wrapper_msg_vBYTEpFATFS Struct Reference

Data Fields

- `_wrapper_msg_base` BYTE **byte**
- FATFS * **fatfsp**

8.110.1 Detailed Description

Definition at line 147 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.111 wrapper_msg_vBYTEvBYTEvUINT Struct Reference

Data Fields

- `_wrapper_msg_base` BYTE **byte1**
- BYTE **byte2**
- UINT **uint**

8.111.1 Detailed Description

Definition at line 250 of file `fatfsWrapper.c`.

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

- `sdcard/fatfsWrapper.c`

8.112 wrapper_msg_vTCHARpFIL Struct Reference

Data Fields

- `_wrapper_msg_base` TCHAR **tchar**
- FIL * **filep**

8.112.1 Detailed Description

Definition at line 266 of file fatfsWrapper.c.

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

- [sdcard/fatfsWrapper.c](#)

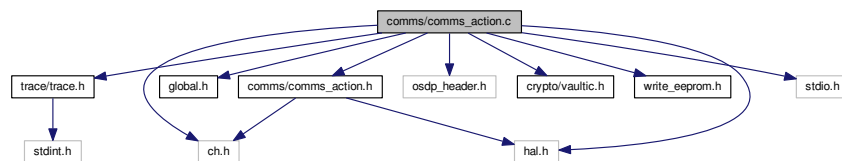
Chapter 9

File Documentation

9.1 comms/comms_action.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "global.h"
#include "trace/trace.h"
#include "osdp_header.h"
#include "crypto/vaultic.h"
#include "write_eeprom.h"
#include "comms/comms_action.h"
#include <stdio.h>
```

Include dependency graph for comms_action.c:



Functions

- **WORKING_AREA** (comms_action_thread, [COMMS_ACTION_THREAD_SIZE](#))
- `msg_t` [comms_action_entry](#) (void *p)

Worker thread to perform tasks related to the vault.

Variables

- volatile bool `g_scbk_set`
- volatile bool `g_osdp_addr_set`
- volatile bool `g_ms100_cfg_set`
- volatile bool `g_ms100_algorithm_set`
- volatile bool `g_ms100_trace_set`
- volatile bool `g_ms100_ipv4_set`
- volatile bool `g_ms100_security_set`
- volatile bool `g_ms100_record_index_set`

- volatile bool `g_ms100_relay_triggers_set`
- volatile bool `g_ms100_paired_set`

9.1.1 Detailed Description

Author

neil

Definition in file [comms_action.c](#).

9.1.2 Function Documentation

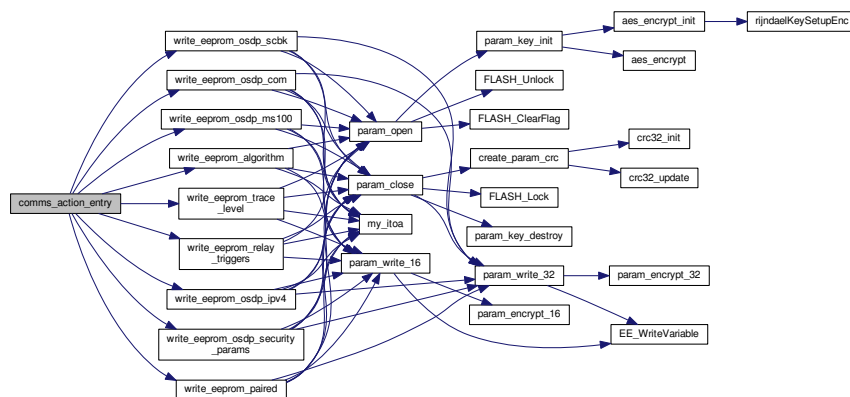
9.1.2.1 `msg_t comms_action_entry ( void * p )`

Worker thread to perform tasks related to the vault.

Todo Disable this if vault is used

Definition at line 54 of file [comms_action.c](#).

Here is the call graph for this function:

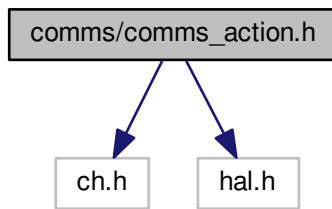


9.2 comms/comms_action.h File Reference

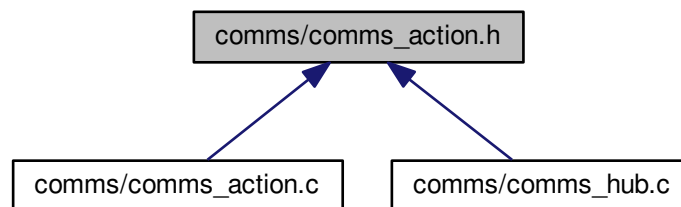
A thread dedicated to acting upon commands received via comms channels.

```
#include "ch.h"
#include "hal.h"
```

Include dependency graph for comms_action.h:



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



Macros

- `#define COMMS_ACTION_THREAD_SIZE 1024`
The size of the comms_action thread.

Functions

- `WORKING_AREA` (comms_action_thread, `COMMS_ACTION_THREAD_SIZE`)
Specify external linkage for a Communications Hub thread.
- `msg_t comms_action_entry` (void *p)
Worker thread to perform tasks related to the vault.

9.2.1 Detailed Description

A thread dedicated to acting upon commands received via comms channels.

Author

neil

Definition in file `comms_action.h`.

9.2.2 Function Documentation

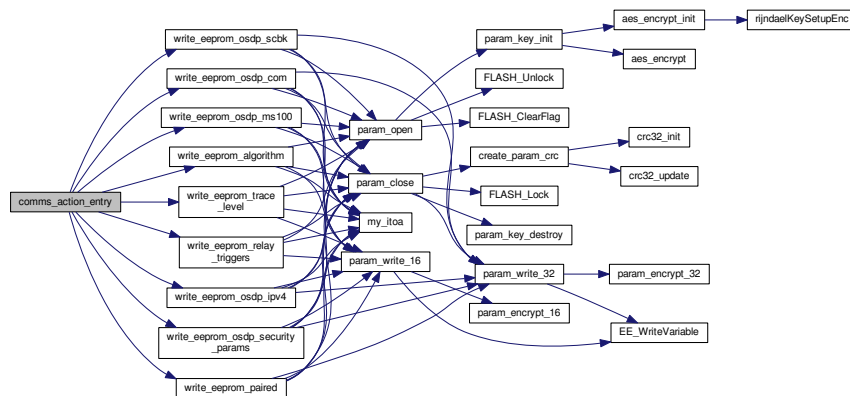
9.2.2.1 msg_t comms_action_entry (void * p)

Worker thread to perform tasks related to the vault.

Todo Disable this if vault is used

Definition at line 54 of file comms_action.c.

Here is the call graph for this function:



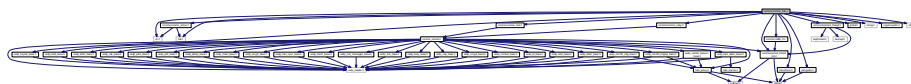
9.3 comms/comms_hub.c File Reference

```

#include "ch.h"
#include "hal.h"
#include "comms/comms_hub.h"
#include "comms/comms_action.h"
#include "comms/comms_relay.h"
#include "comms/packet_linked_list.h"
#include "osdp_header.h"
#include "global.h"
#include <string.h>
#include "comms/rs485_if.h"
#include "comms/tcpclient_thread.h"
#include "trace/trace.h"
#include "crypto/vaultic.h"
#include "stringutils.h"
#include <stdio.h>

```

Include dependency graph for comms_hub.c:



Functions

- [WORKING_AREA](#) (comms_hub_thread, COMMS_THREAD_SIZE)

- *A working area for the thread.*
- void [comms_init](#) (void)
- *A function used to buffer debug messages for transmission.*
- msg_t [comms_entry](#) (void *p)
- *Thread to parse incoming data and instigate an appropriate response.*

Variables

- [comms_hub_t g_hub](#) [NUM_COMMS_INTERFACES]
- *A global struct used to store all communications interface parameters.*
- char [g_encode_buffer](#) [COMMS_HUB_MAX_PACKET_SIZE]
- *The encode buffer.*
- bool **rs485_update**
- bool **soft_reset**
- uint8_t [g_fw_buffer](#) [256]
- *A buffer used to store downloaded firmware bytes.*
- VirtualTimer [vt_busy](#)
- *A timer used to measure how long it is taking to respond to a message.*
- EventSource **evt_packet_done**
- int32_t [g_len](#)
- TimeMeasurement [g_tm](#)
- TimeMeasurement [g_tm_write](#)
- Thread * [tp_comms_hub](#)
- *A pointer to the communications hub thread.*
- Mailbox [mb_packets](#)
- msg_t [wa_packets](#) [COMMS_MB_SIZE]
- Mailbox [mb_response](#)
- msg_t [wa_response](#) [COMMS_MB_SIZE]
- Mailbox [mb_relay_comms](#)
- msg_t [wa_relay_comms](#) [COMMS_RELAY_MB_SIZE]
- Mailbox [mb_comms](#)
- msg_t [wa_comms](#) [COMMS_MB_SIZE]

9.3.1 Detailed Description

Author

neil

Definition in file [comms_hub.c](#).

9.3.2 Function Documentation

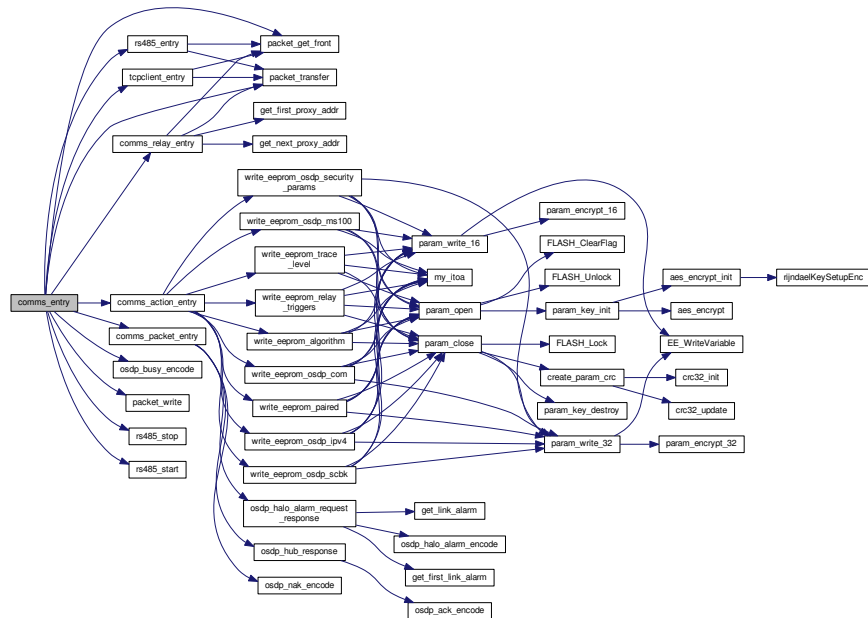
9.3.2.1 msg_t comms_entry (void * p)

Thread to parse incoming data and instigate an appropriate response.

Todo It may be necessary to gracefully shut the peripherals and OS down, e.g. make sure SD card files aren't corrupted etc.

Definition at line 318 of file comms_hub.c.

Here is the call graph for this function:



9.3.2.2 void comms_init (void)

A function used to buffer debug messages for transmission.

Parameters

<i>message</i>	The ASCII message to be buffered
<i>length</i>	The length of the debug message

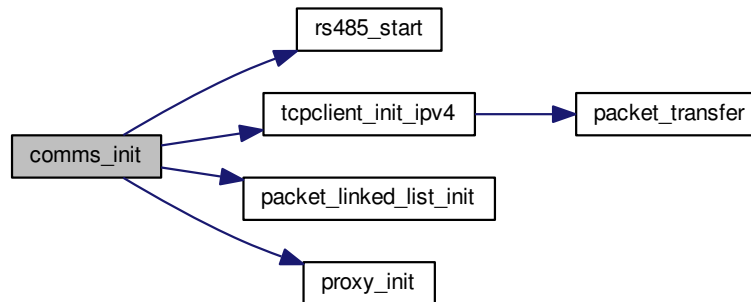
Returns

Returns true if successfully buffered, false if failed (i.e. not enough space) Reset all variables and FSM's to their initial state

Todo This is a waste of resources resulting from C++ port, during code beautification/optimisation fix it

Definition at line 149 of file comms_hub.c.

Here is the call graph for this function:



9.3.3 Variable Documentation

9.3.3.1 Mailbox `mb_comms`

A working area for the mailbox

Definition at line 100 of file `comms_hub.c`.

9.3.3.2 Mailbox `mb_packets`

A working area for the packet processing mailbox command

Definition at line 42 of file `comms_packet.c`.

9.3.3.3 Mailbox `mb_relay_comms`

A working area for the relay mailbox

Definition at line 31 of file `comms_relay.c`.

9.3.3.4 Mailbox `mb_response`

A working area for the packet processing mailbox response

Definition at line 48 of file `comms_packet.c`.

9.3.3.5 Thread* `tp_comms_hub`

A pointer to the communications hub thread.

a POINTER TO THE COMMS_HUB THREAD

Definition at line 117 of file `comms_hub.c`.

9.4 comms/comms_hub.h File Reference

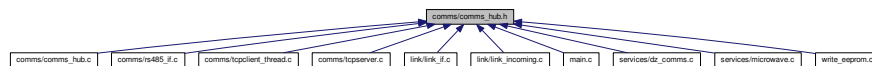
An interface to the various means of communicating with the MS100.

```
#include "comms_packet.h"
```

Include dependency graph for `comms_hub.h`:



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



Macros

- `#define COMMS_THREAD_SIZE 1024`
The size of the comms_hub thread.
- `#define OSDP_RECIPIENT 0`
- `#define OSDP_RELAY 1`

Functions

- `WORKING_AREA` (`comms_hub_thread`, `COMMS_THREAD_SIZE`)
Specify external linkage for a Communications Hub thread.
- `void comms_init (void)`
A function used to buffer debug messages for transmission.
- `msg_t comms_entry (void *p)`
Thread to parse incoming data and instigate an appropriate response.

9.4.1 Detailed Description

An interface to the various means of communicating with the MS100.

Author

neil

Definition in file `comms_hub.h`.

9.4.2 Function Documentation

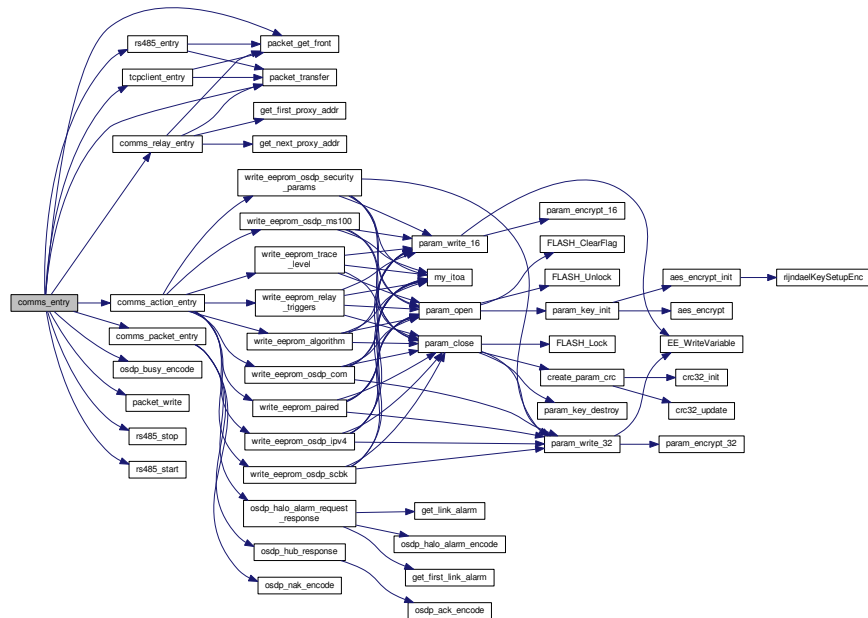
9.4.2.1 `msg_t comms_entry ( void * p )`

Thread to parse incoming data and instigate an appropriate response.

Todo It may be necessary to gracefully shut the peripherals and OS down, e.g. make sure SD card files aren't corrupted etc.

Definition at line 318 of file comms_hub.c.

Here is the call graph for this function:



9.4.2.2 void comms_init (void)

A function used to buffer debug messages for transmission.

Parameters

<i>message</i>	The ASCII message to be buffered
<i>length</i>	The length of the debug message

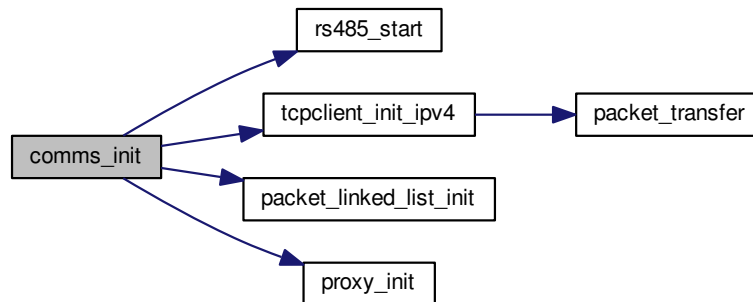
Returns

Returns true if successfully buffered, false if failed (i.e. not enough space) Reset all variables and FSM's to their initial state

Todo This is a waste of resources resulting from C++ port, during code beautification/optimisation fix it

Definition at line 149 of file comms hub.c.

Here is the call graph for this function:



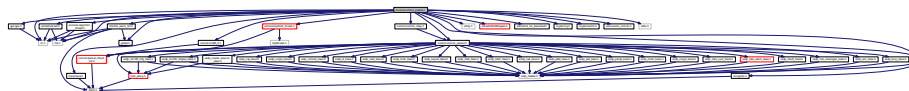
9.5 comms/comms_packet.c File Reference

```

#include "ch.h"
#include "hal.h"
#include "comms/comms_packet.h"
#include "comms/comms_relay.h"
#include "link/link_alarm_list.h"
#include "osdp_header.h"
#include "global.h"
#include <string.h>
#include "comms/rs485_if.h"
#include "comms/tcpclient_thread.h"
#include "sdcard/fatfsWrapper.h"
#include "sdcard/sd_fw_download.h"
#include "sdcard/sdcard.h"
#include "trace/trace.h"
#include "services/algorithm_thread.h"
#include "crypto/rng.h"
#include "crypto/vaultic.h"
#include "gps/gps.h"
#include "stringutils.h"
#include "services/dz_comms.h"
#include <stdio.h>

```

Include dependency graph for `comms_packet.c`:



Macros

- `#define NUM_ALARM_TIMEOUTS 23`
- `#define ALARM_TIMEOUT_DELAY 5`

Functions

- [WORKING_AREA](#) (comms_packet_thread, COMMS_PACKET_THREAD_SIZE)
A working area for the packet processing thread.
- void [osdp_halo_alarm_request_response](#) (int32_t hub)
Response to a non-standard osdp_HALO_ALARM_REQUEST.
- void [osdp_hub_response](#) (int32_t hub)
Response to an osdp_HUB.
- void [alarm_timeout_callback](#) (void *p)
- msg_t [comms_packet_entry](#) (void *p)
Thread to parse incoming data and instigate an appropriate response.

Variables

- [comms_hub_t g_hub](#) [NUM_COMMS_INTERFACES]
A global struct used to store all communications interface parameters.
- [network_alarms_t g_network_alarms](#)
- char [g_encode_buffer](#) [COMMS_HUB_MAX_PACKET_SIZE]
The encode buffer.
- uint8_t [g_fw_buffer](#) [256]
A buffer used to store downloaded firmware bytes.
- [proxy_list_t g_proxy_list](#)
A list of connected remote devices.
- EventSource [sdfw_status](#)
- struct EventListener [el_sdfw](#)
Event listener for SDC card firmware download events.
- VirtualTimer [vt_busy](#)
A timer used to measure how long it is taking to respond to a message.
- volatile bool [g_scbk_set](#) = false
- volatile bool [g_osdp_addr_set](#) = false
- volatile bool [g_ms100_cfg_set](#) = false
- volatile bool [g_ms100_algorithm_set](#) = false
- volatile bool [g_ms100_trace_set](#) = false
- volatile bool [g_ms100_ipv4_set](#) = false
- volatile bool [g_ms100_security_set](#) = false
- volatile bool [g_ms100_relay_triggers_set](#) = false
- volatile bool [g_ms100_paired_set](#) = false
- bool [rs485_update](#) = false
- bool [soft_reset](#) = false
- EventSource [evt_packet_done](#)
- int32_t [g_len](#)
- volatile uint8_t [g_osdp_alarm_timeout](#) [NUM_ALARM_TIMEOUTS] = { 0 }
- VirtualTimer [vt](#)
- Mailbox [mb_packets](#)
- msg_t [wa_packets](#) [COMMS_MB_SIZE]
- Mailbox [mb_response](#)
- msg_t [wa_response](#) [COMMS_MB_SIZE]

9.5.1 Detailed Description

Author

neil

Definition in file [comms_packet.c](#).

9.5.2 Function Documentation

9.5.2.1 void osdp_hub_response (int32_t hub)

Response to an osdp_HUB.

osdp_check_security(hub)

Definition at line 903 of file comms_packet.c.

Here is the call graph for this function:



9.5.3 Variable Documentation

9.5.3.1 Mailbox mb_packets

A working area for the packet processing mailbox command

Definition at line 42 of file comms_packet.c.

9.5.3.2 Mailbox mb_response

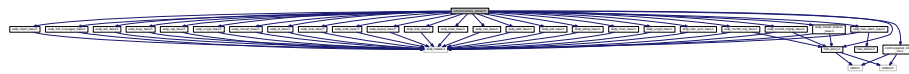
A working area for the packet processing mailbox response

Definition at line 48 of file comms_packet.c.

9.6 comms/comms_packet.h File Reference

A packet processing worker thread for comms_hub.

```
#include "osdp_ack_base.h"
#include "osdp_busy_base.h"
#include "osdp_cap_base.h"
#include "osdp_ccrypt_base.h"
#include "osdp_comset_base.h"
#include "osdp_id_base.h"
#include "osdp_istat_base.h"
#include "osdp_istatr_base.h"
#include "osdp_keyset_base.h"
#include "osdp_lstat_base.h"
#include "osdp_lstatr_base.h"
#include "osdp_ms100_mfg_base.h"
#include "osdp_ms100_mfgrep_base.h"
#include "osdp_nak_base.h"
#include "osdp_pdid_base.h"
#include "osdp_poll_base.h"
#include "osdp_pdcap_base.h"
#include "osdp_rmaci_base.h"
#include "osdp_sscript_base.h"
#include "osdp_master_beacon_base.h"
#include "osdp_halo_sync_base.h"
#include "osdp_halo_alarm_base.h"
#include "osdp_rfauth_base.h"
#include "osdp_hub_messages_base.h"
#include "halo_proxy.h"
#include "comms/packet_linked_list.h"
Include dependency graph for comms_packet.h:
```



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



Data Structures

- [struct `\_comms\_hub\_t`](#)

A struct used to store information for an OSDP communications interface.

Macros

- `#define` **EVT_COMMS_HUB_RECONFIGURE** 0
 - `#define` **EVT_COMMS_HUB_CONFIG_MODE** 1
 - `#define` **EVT_COMMS_HUB_CONFIG_MODE_EXIT** 2
 - `#define` [CKS_CRC_SUPPORT](#) 1
- Enable support of CRC rather than just the simple checksum.*
- `#define` **COMMS_HUB_MAX_PACKET_SIZE** 512
 - `#define` **COMMS_MB_SIZE** 5
 - `#define` **COMMS_PACKET_THREAD_SIZE** 2048

- `#define` [BUSY_TIMEOUT](#) 20
The response timeout before we must respond with an `osdp_BUSY`.
- `#define` [NUM_COMMS_INTERFACES](#) 3
An external linkage definition of the Communications Hub for sharing.

Typedefs

- `typedef struct` [_comms_hub_t](#) [comms_hub_t](#)
A typedef of the [_comms_hub_t](#) struct used for sharing.

Enumerations

- `enum` [comms_hub_iface_e](#) { [HUB_ENABLE_RS485](#) = 1, [HUB_ENABLE_TCP](#) = 2, [HUB_ENABLE_RF](#) = 4, [HUB_ENABLE_TCP_B2B](#) = 8 }
- Bitfield enumeration for the Communications Hub interface enable switches.*

Functions

- [WORKING_AREA](#) ([comms_packet_thread](#), [COMMS_PACKET_THREAD_SIZE](#))
Specify external linkage for the packet processing thread.
- `msg_t` [comms_packet_entry](#) (`void *p`)
Thread to parse incoming data and instigate an appropriate response.

Variables

- Thread * [tp_comms_hub](#)
a POINTER TO THE COMMS_HUB THREAD
- Mailbox [mb_comms](#)
- `msg_t` [wa_comms](#) [[COMMS_MB_SIZE](#)]
- `comms_hub_t` [g_hub](#) [[NUM_COMMS_INTERFACES](#)]
A global struct used to store all communications interface parameters.
- `proxy_list_t` [g_proxy_list](#)
A list of connected remote devices.

9.6.1 Detailed Description

A packet processing worker thread for `comms_hub`.

Author

neil

Definition in file [comms_packet.h](#).

9.6.2 Variable Documentation

9.6.2.1 Mailbox `mb_comms`

A working area for the mailbox

Definition at line 100 of file `comms_hub.c`.

9.6.2.2 Thread* tp_comms_hub

a POINTER TO THE COMMS_HUB THREAD

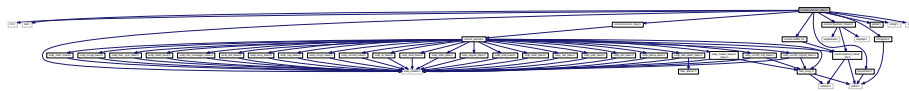
a POINTER TO THE COMMS_HUB THREAD

Definition at line 117 of file comms_hub.c.

9.7 comms/comms_relay.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "comms/comms_relay.h"
#include "osdp_header.h"
#include "global.h"
#include <string.h>
#include "comms/rs485_if.h"
#include "comms/tcpclient_thread.h"
#include "trace/trace.h"
#include "stringutils.h"
#include <stdio.h>
```

Include dependency graph for comms_relay.c:



Functions

- [WORKING_AREA](#) (comms_relay_thread, [COMMS_RELAY_THREAD_SIZE](#))
A working area for the thread.
- void **comms_relay_init** (void)
- msg_t [comms_relay_entry](#) (void *p)
Thread to parse incoming data and instigate an appropriate response.

Variables

- [relay_queue_t](#) [g_microwave_queue](#)
- Mailbox [mb_relay_comms](#)
- msg_t [wa_relay_comms](#) [[COMMS_RELAY_MB_SIZE](#)]

9.7.1 Detailed Description

Author

neil

Definition in file [comms_relay.c](#).

9.7.2 Variable Documentation

9.7.2.1 Mailbox mb_relay_comms

A working area for the relay mailbox

Definition at line 31 of file comms_relay.c.

9.8 comms/comms_relay.h File Reference

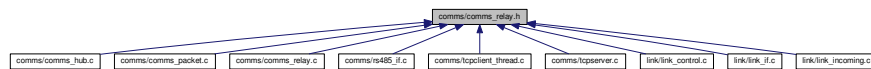
A worker thread dedicated to relaying messages rather than processing them.

```
#include "comms_packet.h"
```

Include dependency graph for comms_relay.h:



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



Macros

- `#define COMMS_RELAY_THREAD_SIZE 1024`
The size of the comms_relay thread.
- `#define COMMS_RELAY_MB_SIZE 5`
The size of the relay mailbox buffer.
- `#define HUB_IFACE_RS485 0`
- `#define HUB_IFACE_ETHERNET 1`
- `#define HUB_IFACE_MICROWAVE 2`

Functions

- `WORKING_AREA` (comms_relay_thread, COMMS_RELAY_THREAD_SIZE)
Specify external linkage for the relay thread.
- `bool comms_relay_write` (uint8_t *data, int32_t length)
Thread to parse incoming data and instigate an appropriate response.
- `msg_t comms_relay_entry` (void *p)
Thread to parse incoming data and instigate an appropriate response.

Variables

- Mailbox `mb_relay_comms`
- `msg_t wa_relay_comms` [COMMS_RELAY_MB_SIZE]

9.8.1 Detailed Description

A worker thread dedicated to relaying messages rather than processing them.

Author

neil

Definition in file [comms_relay.h](#).

9.8.2 Variable Documentation

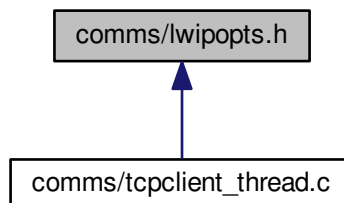
9.8.2.1 Mailbox mb_relay_comms

A working area for the relay mailbox

Definition at line 31 of file [comms_relay.c](#).

9.9 comms/lwipopts.h File Reference

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



Macros

- #define [SYS_LIGHTWEIGHT_PROT](#) 0
- #define [NO_SYS](#) 0
- #define [NO_SYS_NO_TIMERS](#) 0
- #define [MEMCPY](#)(dst, src, len) memcpy(dst,src,len)
- #define [SMEMCPY](#)(dst, src, len) memcpy(dst,src,len)
- #define [MEM_LIBC_MALLOC](#) 0
- #define [MEMP_MEM_MALLOC](#) 0
- #define [MEM_ALIGNMENT](#) 4
- #define [MEM_SIZE](#) 1600
- #define [MEMP_SEPARATE_POOLS](#) 0
- #define [MEMP_OVERFLOW_CHECK](#) 0
- #define [MEMP_SANITY_CHECK](#) 0
- #define [MEM_USE_POOLS](#) 1
- #define [MEM_USE_POOLS_TRY_BIGGER_POOL](#) 0
- #define [MEMP_USE_CUSTOM_POOLS](#) 1

- #define LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT 0
- #define MEMP_NUM_PBUF 16
- #define MEMP_NUM_RAW_PCB 4
- #define MEMP_NUM_UDP_PCB 4
- #define MEMP_NUM_TCP_PCB 5
- #define MEMP_NUM_TCP_PCB_LISTEN 8
- #define MEMP_NUM_TCP_SEG 16
- #define MEMP_NUM_REASSDATA 5
- #define MEMP_NUM_FRAG_PBUF 15
- #define MEMP_NUM_ARP_QUEUE 30
- #define MEMP_NUM_IGMP_GROUP 8
- #define MEMP_NUM_SYS_TIMEOUT (LWIP_TCP + IP_REASSEMBLY + LWIP_ARP + (2*LWIP_DHCP) + LWIP_AUTOIP + LWIP_IGMP + LWIP_DNS + PPP_SUPPORT)
- #define MEMP_NUM_NETBUF 2
- #define MEMP_NUM_NETCONN 4
- #define MEMP_NUM_TCPIP_MSG_API 8
- #define MEMP_NUM_TCPIP_MSG_INPKT 8
- #define MEMP_NUM_SNMP_NODE 50
- #define MEMP_NUM_SNMP_ROOTNODE 30
- #define MEMP_NUM_SNMP_VARBIND 2
- #define MEMP_NUM_SNMP_VALUE 3
- #define MEMP_NUM_NETDB 1
- #define MEMP_NUM_LOCALHOSTLIST 1
- #define MEMP_NUM_PPPOE_INTERFACES 1
- #define PBUF_POOL_SIZE 16
- #define LWIP_ARP 1
- #define ARP_TABLE_SIZE 10
- #define ARP_QUEUEING 0
- #define ETHARP_TRUST_IP_MAC 0
- #define ETHARP_SUPPORT_VLAN 0
- #define LWIP_ETHERNET (LWIP_ARP || PPPOE_SUPPORT)
- #define ETH_PAD_SIZE 0
- #define ETHARP_SUPPORT_STATIC_ENTRIES 0
- #define IP_FORWARD 0
- #define IP_OPTIONS_ALLOWED 1
- #define IP_REASSEMBLY 1
- #define IP_FRAG 1
- #define IP_REASS_MAXAGE 3
- #define IP_REASS_MAX_PBUFS 10
- #define IP_FRAG_USES_STATIC_BUF 0
- #define IP_DEFAULT_TTL 255
- #define IP_SOF_BROADCAST 0
- #define IP_SOF_BROADCAST_RECV 0
- #define IP_FORWARD_ALLOW_TX_ON_RX_NETIF 0
- #define LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS 0
- #define LWIP_ICMP 1
- #define ICMP_TTL (IP_DEFAULT_TTL)
- #define LWIP_BROADCAST_PING 0
- #define LWIP_MULTICAST_PING 0
- #define LWIP_RAW 1
- #define RAW_TTL (IP_DEFAULT_TTL)
- #define LWIP_DHCP 1
- #define DHCP_DOES_ARP_CHECK ((LWIP_DHCP) && (LWIP_ARP))
- #define LWIP_AUTOIP 1
- #define LWIP_DHCP_AUTOIP_COOP 1

- #define LWIP_DHCP_AUTOIP_COOP_TRIES 9
- #define LWIP_AUTOIP_CREATE_SEED_ADDR(netif) 0x0001FEA9
- #define LWIP_SNMP 0
- #define SNMP_CONCURRENT_REQUESTS 1
- #define SNMP_TRAP_DESTINATIONS 1
- #define SNMP_PRIVATE_MIB 0
- #define SNMP_SAFE_REQUESTS 1
- #define SNMP_MAX_OCTET_STRING_LEN 127
- #define SNMP_MAX_TREE_DEPTH 15
- #define SNMP_MAX_VALUE_SIZE LWIP_MAX((SNMP_MAX_OCTET_STRING_LEN)+1, sizeof(s32_t)*(SNMP_MAX_TREE_DEPTH))
- #define LWIP_IGMP 0
- #define LWIP_DNS 0
- #define DNS_TABLE_SIZE 4
- #define DNS_MAX_NAME_LENGTH 256
- #define DNS_MAX_SERVERS 2
- #define DNS_DOES_NAME_CHECK 1
- #define DNS_MSG_SIZE 512
- #define DNS_LOCAL_HOSTLIST 0
- #define DNS_LOCAL_HOSTLIST_IS_DYNAMIC 0
- #define LWIP_UDP 1
- #define LWIP_UDPLITE 0
- #define UDP_TTL (IP_DEFAULT_TTL)
- #define LWIP_NETBUF_RECVINFO 0
- #define LWIP_TCP 1
- #define TCP_TTL (IP_DEFAULT_TTL)
- #define TCP_WND (4 * TCP_MSS)
- #define TCP_MAXRTX 12
- #define TCP_SYNMAXRTX 6
- #define TCP_QUEUE_OOSEQ (LWIP_TCP)
- #define TCP_MSS 536
- #define TCP_CALCULATE_EFF_SEND_MSS 1
- #define TCP_SND_BUF (2 * TCP_MSS)
- #define TCP_SND_QUEUELEN ((4 * (TCP_SND_BUF) + (TCP_MSS - 1))/(TCP_MSS))
- #define TCP_SNDLOWAT LWIP_MIN(LWIP_MAX(((TCP_SND_BUF)/2), (2 * TCP_MSS) + 1), (TCP_SND_BUF) - 1)
- #define TCP_SNDQUEUELOWAT LWIP_MAX(((TCP_SND_QUEUELEN)/2), 5)
- #define TCP_OOSEQ_MAX_BYTES 0
- #define TCP_OOSEQ_MAX_PBUFS 0
- #define TCP_LISTEN_BACKLOG 0
- #define TCP_DEFAULT_LISTEN_BACKLOG 0xff
- #define TCP_OVERSIZE TCP_MSS
- #define LWIP_TCP_TIMESTAMPS 0
- #define TCP_WND_UPDATE_THRESHOLD (TCP_WND / 4)
- #define LWIP_EVENT_API 0
- #define LWIP_CALLBACK_API 1
- #define PBUF_LINK_HLEN (14 + ETH_PAD_SIZE)
- #define PBUF_POOL_BUFSIZE LWIP_MEM_ALIGN_SIZE(TCP_MSS+40+PBUF_LINK_HLEN)
- #define LWIP_NETIF_HOSTNAME 0
- #define LWIP_NETIF_API 1
- #define LWIP_NETIF_STATUS_CALLBACK 0
- #define LWIP_NETIF_LINK_CALLBACK 0
- #define LWIP_NETIF_REMOVE_CALLBACK 0
- #define LWIP_NETIF_HWADDRHINT 0
- #define LWIP_NETIF_LOOPBACK 0

- #define LWIP_LOOPBACK_MAX_PBUFS 0
- #define LWIP_NETIF_LOOPBACK_MULTITHREADING (!NO_SYS)
- #define LWIP_NETIF_TX_SINGLE_PBUF 0
- #define LWIP_HAVE_LOOPIF 0
- #define LWIP_HAVE_SLIPIF 0
- #define TCPIP_THREAD_NAME "tcpip_thread"
- #define TCPIP_THREAD_STACKSIZE 1024
- #define TCPIP_THREAD_PRIO (LOWPRIO + 1)
- #define TCPIP_MBOX_SIZE MEMP_NUM_PBUF
- #define SLIPIF_THREAD_NAME "slipif_loop"
- #define SLIPIF_THREAD_STACKSIZE 1024
- #define SLIPIF_THREAD_PRIO (LOWPRIO + 1)
- #define PPP_THREAD_NAME "pppInputThread"
- #define PPP_THREAD_STACKSIZE 1024
- #define PPP_THREAD_PRIO (LOWPRIO + 1)
- #define DEFAULT_THREAD_NAME "lwip"
- #define DEFAULT_THREAD_STACKSIZE 1024
- #define DEFAULT_THREAD_PRIO (LOWPRIO + 1)
- #define DEFAULT_RAW_RECVMBOX_SIZE 4
- #define DEFAULT_UDP_RECVMBOX_SIZE 4
- #define DEFAULT_TCP_RECVMBOX_SIZE 40
- #define DEFAULT_ACCEPTMBOX_SIZE 4
- #define LWIP_TCPIP_CORE_LOCKING 0
- #define LWIP_TCPIP_CORE_LOCKING_INPUT 0
- #define LWIP_NETCONN 1
- #define LWIP_TCPIP_TIMEOUT 1
- #define LWIP_SOCKET 1
- #define LWIP_COMPAT_SOCKETS 1
- #define LWIP_POSIX_SOCKETS_IO_NAMES 1
- #define LWIP_TCP_KEEPALIVE 1
- #define LWIP_SO_SNDTIMEO 1
- #define LWIP_SO_RCVTIMEO 1
- #define LWIP_SO_RCVBUF 0
- #define RECV_BUFSIZE_DEFAULT INT_MAX
- #define SO_REUSE 0
- #define SO_REUSE_RXTOALL 0
- #define LWIP_STATS 1
- #define LWIP_STATS_DISPLAY 0
- #define LINK_STATS 1
- #define ETHARP_STATS (LWIP_ARP)
- #define IP_STATS 1
- #define IPFRAG_STATS (IP_REASSEMBLY || IP_FRAG)
- #define ICMP_STATS 1
- #define IGMP_STATS (LWIP_IGMP)
- #define UDP_STATS (LWIP_UDP)
- #define TCP_STATS (LWIP_TCP)
- #define MEM_STATS ((MEM_LIBC_MALLOC == 0) && (MEM_USE_POOLS == 0))
- #define MEMP_STATS (MEMP_MEM_MALLOC == 0)
- #define SYS_STATS (NO_SYS == 0)
- #define PPP_SUPPORT 0
- #define PPPOE_SUPPORT 0
- #define PPPOS_SUPPORT PPP_SUPPORT
- #define CHECKSUM_GEN_IP 1
- #define CHECKSUM_GEN_UDP 1
- #define CHECKSUM_GEN_TCP 1

- `#define CHECKSUM_GEN_ICMP 1`
- `#define CHECKSUM_CHECK_IP 1`
- `#define CHECKSUM_CHECK_UDP 1`
- `#define CHECKSUM_CHECK_TCP 1`
- `#define LWIP_CHECKSUM_ON_COPY 0`
- `#define LWIP_DBG_MIN_LEVEL LWIP_DBG_LEVEL_ALL`
- `#define LWIP_DBG_TYPES_ON LWIP_DBG_ON`
- `#define ETHARP_DEBUG LWIP_DBG_OFF`
- `#define NETIF_DEBUG LWIP_DBG_OFF`
- `#define PBUF_DEBUG LWIP_DBG_OFF`
- `#define API_LIB_DEBUG LWIP_DBG_OFF`
- `#define API_MSG_DEBUG LWIP_DBG_OFF`
- `#define SOCKETS_DEBUG LWIP_DBG_OFF`
- `#define ICMP_DEBUG LWIP_DBG_OFF`
- `#define IGMP_DEBUG LWIP_DBG_OFF`
- `#define INET_DEBUG LWIP_DBG_OFF`
- `#define IP_DEBUG LWIP_DBG_OFF`
- `#define IP_REASS_DEBUG LWIP_DBG_OFF`
- `#define RAW_DEBUG LWIP_DBG_OFF`
- `#define MEM_DEBUG LWIP_DBG_OFF`
- `#define MEMP_DEBUG LWIP_DBG_OFF`
- `#define SYS_DEBUG LWIP_DBG_OFF`
- `#define TIMERS_DEBUG LWIP_DBG_OFF`
- `#define TCP_DEBUG LWIP_DBG_OFF`
- `#define TCP_INPUT_DEBUG LWIP_DBG_OFF`
- `#define TCP_FR_DEBUG LWIP_DBG_OFF`
- `#define TCP_RTO_DEBUG LWIP_DBG_OFF`
- `#define TCP_CWND_DEBUG LWIP_DBG_OFF`
- `#define TCP_WND_DEBUG LWIP_DBG_OFF`
- `#define TCP_OUTPUT_DEBUG LWIP_DBG_OFF`
- `#define TCP_RST_DEBUG LWIP_DBG_OFF`
- `#define TCP_QLEN_DEBUG LWIP_DBG_OFF`
- `#define UDP_DEBUG LWIP_DBG_OFF`
- `#define TCPIP_DEBUG LWIP_DBG_OFF`
- `#define PPP_DEBUG LWIP_DBG_OFF`
- `#define SLIP_DEBUG LWIP_DBG_OFF`
- `#define DHCP_DEBUG LWIP_DBG_OFF`
- `#define AUTOIP_DEBUG LWIP_DBG_OFF`
- `#define SNMP_MSG_DEBUG LWIP_DBG_OFF`
- `#define SNMP_MIB_DEBUG LWIP_DBG_OFF`
- `#define DNS_DEBUG LWIP_DBG_OFF`

9.9.1 Detailed Description

LwIP Options Configuration

Definition in file [lwipopts.h](#).

9.9.2 Macro Definition Documentation

9.9.2.1 `#define API_LIB_DEBUG LWIP_DBG_OFF`

API_LIB_DEBUG: Enable debugging in api_lib.c.

Definition at line 1921 of file lwipopts.h.

9.9.2.2 `#define API_MSG_DEBUG LWIP_DBG_OFF`

API_MSG_DEBUG: Enable debugging in api_msg.c.

Definition at line 1928 of file lwipopts.h.

9.9.2.3 `#define ARP_QUEUEING 0`

ARP_QUEUEING==1: Multiple outgoing packets are queued during hardware address resolution. By default, only the most recent packet is queued per IP address. This is sufficient for most protocols and mainly reduces TCP connection startup time. Set this to 1 if you know your application sends more than one packet in a row to an IP address that is not in the ARP cache.

Definition at line 438 of file lwipopts.h.

9.9.2.4 `#define ARP_TABLE_SIZE 10`

ARP_TABLE_SIZE: Number of active MAC-IP address pairs cached.

Definition at line 427 of file lwipopts.h.

9.9.2.5 `#define AUTOIP_DEBUG LWIP_DBG_OFF`

AUTOIP_DEBUG: Enable debugging in autoip.c.

Definition at line 2111 of file lwipopts.h.

9.9.2.6 `#define CHECKSUM_CHECK_IP 1`

CHECKSUM_CHECK_IP==1: Check checksums in software for incoming IP packets.

Definition at line 1821 of file lwipopts.h.

9.9.2.7 `#define CHECKSUM_CHECK_TCP 1`

CHECKSUM_CHECK_TCP==1: Check checksums in software for incoming TCP packets.

Definition at line 1835 of file lwipopts.h.

9.9.2.8 `#define CHECKSUM_CHECK_UDP 1`

CHECKSUM_CHECK_UDP==1: Check checksums in software for incoming UDP packets.

Definition at line 1828 of file lwipopts.h.

9.9.2.9 `#define CHECKSUM_GEN_ICMP 1`

CHECKSUM_GEN_ICMP==1: Generate checksums in software for outgoing ICMP packets.

Definition at line 1814 of file lwipopts.h.

9.9.2.10 `#define CHECKSUM_GEN_IP 1`

CHECKSUM_GEN_IP==1: Generate checksums in software for outgoing IP packets.

Definition at line 1793 of file lwipopts.h.

9.9.2.11 #define CHECKSUM_GEN_TCP 1

CHECKSUM_GEN_TCP==1: Generate checksums in software for outgoing TCP packets.

Definition at line 1807 of file lwipopts.h.

9.9.2.12 #define CHECKSUM_GEN_UDP 1

CHECKSUM_GEN_UDP==1: Generate checksums in software for outgoing UDP packets.

Definition at line 1800 of file lwipopts.h.

9.9.2.13 #define DEFAULT_ACCEPTMBOX_SIZE 4

DEFAULT_ACCEPTMBOX_SIZE: The mailbox size for the incoming connections. The queue size value itself is platform-dependent, but is passed to sys_mbox_new() when the acceptmbox is created.

Definition at line 1381 of file lwipopts.h.

9.9.2.14 #define DEFAULT_RAW_RECVMBOX_SIZE 4

DEFAULT_RAW_RECVMBOX_SIZE: The mailbox size for the incoming packets on a NETCONN_RAW. The queue size value itself is platform-dependent, but is passed to sys_mbox_new() when the recvmbox is created.

Definition at line 1354 of file lwipopts.h.

9.9.2.15 #define DEFAULT_TCP_RECVMBOX_SIZE 40

DEFAULT_TCP_RECVMBOX_SIZE: The mailbox size for the incoming packets on a NETCONN_TCP. The queue size value itself is platform-dependent, but is passed to sys_mbox_new() when the recvmbox is created.

Definition at line 1372 of file lwipopts.h.

9.9.2.16 #define DEFAULT_THREAD_NAME "lwIP"

DEFAULT_THREAD_NAME: The name assigned to any other lwIP thread.

Definition at line 1327 of file lwipopts.h.

9.9.2.17 #define DEFAULT_THREAD_PRIO (LOWPRIO + 1)

DEFAULT_THREAD_PRIO: The priority assigned to any other lwIP thread. The priority value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1345 of file lwipopts.h.

9.9.2.18 #define DEFAULT_THREAD_STACKSIZE 1024

DEFAULT_THREAD_STACKSIZE: The stack size used by any other lwIP thread. The stack size value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1336 of file lwipopts.h.

9.9.2.19 `#define DEFAULT_UDP_RECVMBOX_SIZE 4`

`DEFAULT_UDP_RECVMBOX_SIZE`: The mailbox size for the incoming packets on a `NETCONN_UDP`. The queue size value itself is platform-dependent, but is passed to `sys_mbox_new()` when the recvmbox is created.

Definition at line 1363 of file `lwipopts.h`.

9.9.2.20 `#define DHCP_DEBUG LWIP_DBG_OFF`

`DHCP_DEBUG`: Enable debugging in `dhcp.c`.

Definition at line 2104 of file `lwipopts.h`.

9.9.2.21 `#define DHCP_DOES_ARP_CHECK ((LWIP_DHCP) && (LWIP_ARP))`

`DHCP_DOES_ARP_CHECK==1`: Do an ARP check on the offered address.

Definition at line 683 of file `lwipopts.h`.

9.9.2.22 `#define DNS_DEBUG LWIP_DBG_OFF`

`DNS_DEBUG`: Enable debugging for DNS.

Definition at line 2132 of file `lwipopts.h`.

9.9.2.23 `#define DNS_DOES_NAME_CHECK 1`

DNS do a name checking between the query and the response.

Definition at line 840 of file `lwipopts.h`.

9.9.2.24 `#define DNS_LOCAL_HOSTLIST 0`

`DNS_LOCAL_HOSTLIST`: Implements a local host-to-address list. If enabled, you have to define `#define DNS_LOCAL_HOSTLIST_INIT {{"host1", 0x123}, {"host2", 0x234}}` (an array of structs name/address, where address is an `u32_t` in network byte order).

Instead, you can also use an external function: `#define DNS_LOOKUP_LOCAL_EXTERN(x) extern u32_t my_lookup_function(const char *name)` that returns the IP address or `INADDR_NONE` if not found.

Definition at line 859 of file `lwipopts.h`.

9.9.2.25 `#define DNS_LOCAL_HOSTLIST_IS_DYNAMIC 0`

If this is turned on, the local host-list can be dynamically changed at runtime.

Definition at line 865 of file `lwipopts.h`.

9.9.2.26 `#define DNS_MAX_NAME_LENGTH 256`

DNS maximum host name length supported in the name table.

Definition at line 830 of file `lwipopts.h`.

9.9.2.27 #define DNS_MAX_SERVERS 2

The maximum of DNS servers

Definition at line 835 of file lwipopts.h.

9.9.2.28 #define DNS_MSG_SIZE 512

DNS message max. size. Default value is RFC compliant.

Definition at line 845 of file lwipopts.h.

9.9.2.29 #define DNS_TABLE_SIZE 4

DNS maximum number of entries to maintain locally.

Definition at line 825 of file lwipopts.h.

9.9.2.30 #define ETH_PAD_SIZE 0

ETH_PAD_SIZE: number of bytes added before the ethernet header to ensure alignment of payload after that header. Since the header is 14 bytes long, without this padding e.g. addresses in the IP header will not be aligned on a 32-bit boundary, so setting this to 2 can speed up 32-bit-platforms.

Definition at line 480 of file lwipopts.h.

9.9.2.31 #define ETHARP_DEBUG LWIP_DBG_OFF

ETHARP_DEBUG: Enable debugging in etharp.c.

Definition at line 1900 of file lwipopts.h.

9.9.2.32 #define ETHARP_STATS (LWIP_ARP)

ETHARP_STATS==1: Enable etharp stats.

Definition at line 1535 of file lwipopts.h.

9.9.2.33 #define ETHARP_SUPPORT_STATIC_ENTRIES 0

ETHARP_SUPPORT_STATIC_ENTRIES==1: enable code to support static ARP table entries (using etharp_add↵_static_entry/etharp_remove_static_entry).

Definition at line 487 of file lwipopts.h.

9.9.2.34 #define ETHARP_SUPPORT_VLAN 0

ETHARP_SUPPORT_VLAN==1: support receiving ethernet packets with VLAN header. Additionally, you can define ETHARP_VLAN_CHECK to an u16_t VLAN ID to check. If ETHARP_VLAN_CHECK is defined, only VLA↵N-traffic for this VLAN is accepted. If ETHARP_VLAN_CHECK is not defined, all traffic is accepted. Alternatively, define a function/define ETHARP_VLAN_CHECK_FN(eth_hdr, vlan) that returns 1 to accept a packet or 0 to drop a packet.

Definition at line 464 of file lwipopts.h.

9.9.2.35 #define ETHARP_TRUST_IP_MAC 0

ETHARP_TRUST_IP_MAC==1: Incoming IP packets cause the ARP table to be updated with the source MAC and IP addresses supplied in the packet. You may want to disable this if you do not trust LAN peers to have the correct addresses, or as a limited approach to attempt to handle spoofing. If disabled, lwIP will need to make a new ARP request if the peer is not already in the ARP table, adding a little latency. The peer *is* in the ARP table if it requested our address before. Also notice that this slows down input processing of every IP packet!

Definition at line 452 of file lwipopts.h.

9.9.2.36 #define ICMP_DEBUG LWIP_DBG_OFF

ICMP_DEBUG: Enable debugging in icmp.c.

Definition at line 1942 of file lwipopts.h.

9.9.2.37 #define ICMP_STATS 1

ICMP_STATS==1: Enable ICMP stats.

Definition at line 1557 of file lwipopts.h.

9.9.2.38 #define ICMP_TTL (IP_DEFAULT_TTL)

ICMP_TTL: Default value for Time-To-Live used by ICMP packets.

Definition at line 631 of file lwipopts.h.

9.9.2.39 #define IGMP_DEBUG LWIP_DBG_OFF

IGMP_DEBUG: Enable debugging in igmp.c.

Definition at line 1949 of file lwipopts.h.

9.9.2.40 #define IGMP_STATS (LWIP_IGMP)

IGMP_STATS==1: Enable IGMP stats.

Definition at line 1564 of file lwipopts.h.

9.9.2.41 #define INET_DEBUG LWIP_DBG_OFF

INET_DEBUG: Enable debugging in inet.c.

Definition at line 1956 of file lwipopts.h.

9.9.2.42 #define IP_DEBUG LWIP_DBG_OFF

IP_DEBUG: Enable debugging for IP.

Definition at line 1963 of file lwipopts.h.

9.9.2.43 #define IP_DEFAULT_TTL 255

IP_FRAG_MAX_MTU: Assumed max MTU on any interface for IP frag buffer (requires IP_FRAG_USES_STATISTICS C_BUF==1) IP_DEFAULT_TTL: Default value for Time-To-Live used by transport layers.

Definition at line 574 of file lwipopts.h.

9.9.2.44 `#define IP_FORWARD 0`

`IP_FORWARD==1`: Enables the ability to forward IP packets across network interfaces. If you are going to run lwIP on a device with only one network interface, define this to 0.

Definition at line 502 of file lwipopts.h.

9.9.2.45 `#define IP_FORWARD_ALLOW_TX_ON_RX_NETIF 0`

`IP_FORWARD_ALLOW_TX_ON_RX_NETIF==1`: allow `ip_forward()` to send packets back out on the netif where it was received. This should only be used for wireless networks. ATTENTION: When this is 1, make sure your netif driver correctly marks incoming link-layer-broadcast/multicast packets as such using the corresponding pbuf flags!

Definition at line 602 of file lwipopts.h.

9.9.2.46 `#define IP_FRAG 1`

`IP_FRAG==1`: Fragment outgoing IP packets if their size exceeds MTU. Note that this option does not affect incoming packet sizes, which can be controlled via `IP_REASSEMBLY`.

Definition at line 529 of file lwipopts.h.

9.9.2.47 `#define IP_FRAG_USES_STATIC_BUF 0`

`IP_FRAG_USES_STATIC_BUF==1`: Use a static MTU-sized buffer for IP fragmentation. Otherwise pbufs are allocated and reference the original packet data to be fragmented (or with `LWIP_NETIF_TX_SINGLE_PBUF==1`, new `PBUF_RAM` pbufs are used for fragments). ATTENTION: `IP_FRAG_USES_STATIC_BUF==1` may not be used for DMA-enabled MACs!

Definition at line 559 of file lwipopts.h.

9.9.2.48 `#define IP_OPTIONS_ALLOWED 1`

`IP_OPTIONS_ALLOWED`: Defines the behavior for IP options. `IP_OPTIONS_ALLOWED==0`: All packets with IP options are dropped. `IP_OPTIONS_ALLOWED==1`: IP options are allowed (but not parsed).

Definition at line 511 of file lwipopts.h.

9.9.2.49 `#define IP_REASS_DEBUG LWIP_DBG_OFF`

`IP_REASS_DEBUG`: Enable debugging in `ip_frag.c` for both frag & reass.

Definition at line 1970 of file lwipopts.h.

9.9.2.50 `#define IP_REASS_MAX_PBUFS 10`

`IP_REASS_MAX_PBUFS`: Total maximum amount of pbufs waiting to be reassembled. Since the received pbufs are enqueued, be sure to configure `PBUF_POOL_SIZE > IP_REASS_MAX_PBUFS` so that the stack is still able to receive packets even if the maximum amount of fragments is enqueued for reassembly!

Definition at line 548 of file lwipopts.h.

9.9.2.51 #define IP_REASS_MAXAGE 3

IP_REASS_MAXAGE: Maximum time (in multiples of IP_TMR_INTERVAL - so seconds, normally) a fragmented IP packet waits for all fragments to arrive. If not all fragments arrived in this time, the whole packet is discarded.

Definition at line 538 of file lwipopts.h.

9.9.2.52 #define IP_REASSEMBLY 1

IP_REASSEMBLY==1: Reassemble incoming fragmented IP packets. Note that this option does not affect outgoing packet sizes, which can be controlled via IP_FRAG.

Definition at line 520 of file lwipopts.h.

9.9.2.53 #define IP_SOF_BROADCAST 0

IP_SOF_BROADCAST=1: Use the SOF_BROADCAST field to enable broadcast filter per pcb on udp and raw send operations. To enable broadcast filter on recv operations, you also have to set IP_SOF_BROADCAST_RECV=1.

Definition at line 583 of file lwipopts.h.

9.9.2.54 #define IP_SOF_BROADCAST_RECV 0

IP_SOF_BROADCAST_RECV (requires IP_SOF_BROADCAST=1) enable the broadcast filter on recv operations.

Definition at line 591 of file lwipopts.h.

9.9.2.55 #define IP_STATS 1

IP_STATS==1: Enable IP stats.

Definition at line 1542 of file lwipopts.h.

9.9.2.56 #define IPFRAG_STATS (IP_REASSEMBLY || IP_FRAG)

IPFRAG_STATS==1: Enable IP fragmentation stats. Default is on if using either frag or reass.

Definition at line 1550 of file lwipopts.h.

9.9.2.57 #define LINK_STATS 1

LINK_STATS==1: Enable link stats.

Definition at line 1528 of file lwipopts.h.

9.9.2.58 #define LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT 0

Set this to 1 if you want to free PBUF_RAM pbufs (or call mem_free()) from interrupt context (or another context that doesn't allow waiting for a semaphore). If set to 1, mem_malloc will be protected by a semaphore and SYS_ARCH_PROTECT, while mem_free will only use SYS_ARCH_PROTECT. mem_malloc SYS_ARCH_UNPROTECTs with each loop so that mem_free can run.

ATTENTION: As you can see from the above description, this leads to dis-/ enabling interrupts often, which can be slow! Also, on low memory, mem_malloc can need longer.

If you don't want that, at least for NO_SYS=0, you can still use the following functions to enqueue a deallocation call which then runs in the tcpip_thread context:

- `pbuf_free_callback(p);`
- `mem_free_callback(m);`

Definition at line 206 of file `lwipopts.h`.

9.9.2.59 `#define LWIP_ARP 1`

`LWIP_ARP==1`: Enable ARP functionality.

Definition at line 420 of file `lwipopts.h`.

9.9.2.60 `#define LWIP_AUTOIP 1`

`LWIP_AUTOIP==1`: Enable AUTOIP module.

Definition at line 695 of file `lwipopts.h`.

9.9.2.61 `#define LWIP_AUTOIP_CREATE_SEED_ADDR( netif ) 0x0001FEA9`

`LWIP_AUTOIP_CREATE_SEED_ADDR`: Macro that generates the initial IP address to be tried by AUTOIP.

Definition at line 722 of file `lwipopts.h`.

9.9.2.62 `#define LWIP_BROADCAST_PING 0`

`LWIP_BROADCAST_PING==1`: respond to broadcast pings (default is unicast only)

Definition at line 638 of file `lwipopts.h`.

9.9.2.63 `#define LWIP_CHECKSUM_ON_COPY 0`

`LWIP_CHECKSUM_ON_COPY==1`: Calculate checksum when copying data from application buffers to pbufs.

Definition at line 1843 of file `lwipopts.h`.

9.9.2.64 `#define LWIP_COMPAT_SOCKETS 1`

`LWIP_COMPAT_SOCKETS==1`: Enable BSD-style sockets functions names. (only used if you use `sockets.c`)

Definition at line 1436 of file `lwipopts.h`.

9.9.2.65 `#define LWIP_DBG_MIN_LEVEL LWIP_DBG_LEVEL_ALL`

`LWIP_HOOK_IP4_INPUT(pbuf, input_netif)`:

- called from `ip_input()` (IPv4)
- `pbuf`: received struct pbuf passed to `ip_input()`
- `input_netif`: struct `netif` on which the packet has been received Return values:
- 0: Hook has not consumed the packet, packet is processed as normal
- != 0: Hook has consumed the packet. If the hook consumed the packet, 'pbuf' is in the responsibility of the hook (i.e. free it when done). `LWIP_HOOK_IP4_ROUTE(dest)`:
- called from `ip_route()` (IPv4)

- **dest:** destination IPv4 address Returns the destination netif or NULL if no destination netif is found. In that case, `ip_route()` continues as normal. **LWIP_DBG_MIN_LEVEL:** After masking, the value of the debug is compared against this value. If it is smaller, then debugging messages are written.

Definition at line 1885 of file `lwipopts.h`.

9.9.2.66 `#define LWIP_DBG_TYPES_ON LWIP_DBG_ON`

LWIP_DBG_TYPES_ON: A mask that can be used to globally enable/disable debug messages of certain types.

Definition at line 1893 of file `lwipopts.h`.

9.9.2.67 `#define LWIP_DHCP 1`

LWIP_DHCP==1: Enable DHCP module.

Definition at line 676 of file `lwipopts.h`.

9.9.2.68 `#define LWIP_DHCP_AUTOIP_COOP 1`

LWIP_DHCP_AUTOIP_COOP==1: Allow DHCP and AUTOIP to be both enabled on the same interface at the same time.

Definition at line 703 of file `lwipopts.h`.

9.9.2.69 `#define LWIP_DHCP_AUTOIP_COOP_TRIES 9`

LWIP_DHCP_AUTOIP_COOP_TRIES: Set to the number of DHCP DISCOVER probes that should be sent before falling back on AUTOIP. This can be set as low as 1 to get an AutoIP address very quickly, but you should be prepared to handle a changing IP address when DHCP overrides AutoIP.

Definition at line 714 of file `lwipopts.h`.

9.9.2.70 `#define LWIP_DNS 0`

LWIP_DNS==1: Turn on DNS module. UDP must be available for DNS transport.

Definition at line 820 of file `lwipopts.h`.

9.9.2.71 `#define LWIP_ETHERNET (LWIP_ARP || PPPOE_SUPPORT)`

LWIP_ETHERNET==1: enable ethernet support for PPPoE even though ARP might be disabled

Definition at line 471 of file `lwipopts.h`.

9.9.2.72 `#define LWIP_EVENT_API 0`

LWIP_EVENT_API and **LWIP_CALLBACK_API:** Only one of these should be set to 1. **LWIP_EVENT_API==1:** The user defines `lwip_tcp_event()` to receive all events (accept, sent, etc) that happen in the system. **LWIP_CALLBACK_API==1:** The PCB callback function is called directly for the event. This is the default.

Definition at line 1081 of file `lwipopts.h`.

9.9.2.73 #define LWIP_HAVE_LOOPIF 0

LWIP_HAVE_LOOPIF==1: Support loop interface (127.0.0.1) and loopif.c

Definition at line 1219 of file lwipopts.h.

9.9.2.74 #define LWIP_HAVE_SLIPIF 0

LWIP_HAVE_SLIPIF==1: Support slip interface and slipif.c

Definition at line 1231 of file lwipopts.h.

9.9.2.75 #define LWIP_ICMP 1

LWIP_ICMP==1: Enable ICMP module inside the IP stack. Be careful, disable that make your product non-compliant to RFC1122

Definition at line 624 of file lwipopts.h.

9.9.2.76 #define LWIP_IGMP 0

LWIP_IGMP==1: Turn on IGMP module.

Definition at line 807 of file lwipopts.h.

9.9.2.77 #define LWIP_LOOPBACK_MAX_PBUFS 0

LWIP_LOOPBACK_MAX_PBUFS: Maximum number of pbufs on queue for loopback sending for each netif (0 = disabled)

Definition at line 1177 of file lwipopts.h.

9.9.2.78 #define LWIP_MULTICAST_PING 0

LWIP_MULTICAST_PING==1: respond to multicast pings (default is unicast only)

Definition at line 645 of file lwipopts.h.

9.9.2.79 #define LWIP_NETBUF_RECVINFO 0

LWIP_NETBUF_RECVINFO==1: append destination addr and port to every netbuf.

Definition at line 898 of file lwipopts.h.

9.9.2.80 #define LWIP_NETCONN 1

LWIP_NETCONN==1: Enable Netconn API (require to use api_lib.c)

Definition at line 1409 of file lwipopts.h.

9.9.2.81 #define LWIP_NETIF_API 1

LWIP_NETIF_API==1: Support netif api (in netifapi.c)

Definition at line 1126 of file lwipopts.h.

9.9.2.82 #define LWIP_NETIF_HOSTNAME 0

LWIP_NETIF_HOSTNAME==1: use DHCP_OPTION_HOSTNAME with netif's hostname field.

Definition at line 1119 of file lwipopts.h.

9.9.2.83 #define LWIP_NETIF_HWADDRHINT 0

LWIP_NETIF_HWADDRHINT==1: Cache link-layer-address hints (e.g. table indices) in struct netif. TCP and UDP can make use of this to prevent scanning the ARP table for every sent packet. While this is faster for big ARP tables or many concurrent connections, it might be counterproductive if you have a tiny ARP table or if there never are concurrent connections.

Definition at line 1161 of file lwipopts.h.

9.9.2.84 #define LWIP_NETIF_LINK_CALLBACK 0

LWIP_NETIF_LINK_CALLBACK==1: Support a callback function from an interface whenever the link changes (i.e., link down)

Definition at line 1142 of file lwipopts.h.

9.9.2.85 #define LWIP_NETIF_LOOPBACK 0

LWIP_NETIF_LOOPBACK==1: Support sending packets with a destination IP address equal to the netif IP address, looping them back up the stack.

Definition at line 1169 of file lwipopts.h.

9.9.2.86 #define LWIP_NETIF_LOOPBACK_MULTITHREADING (!NO_SYS)

LWIP_NETIF_LOOPBACK_MULTITHREADING: Indicates whether threading is enabled in the system, as netifs must change how they behave depending on this setting for the LWIP_NETIF_LOOPBACK option to work. Setting this is needed to avoid reentering non-reentrant functions like tcp_input(). LWIP_NETIF_LOOPBACK_MULTITHREADING==1: Indicates that the user is using a multithreaded environment like tcpip.c. In this case, netif->input() is called directly. LWIP_NETIF_LOOPBACK_MULTITHREADING==0: Indicates a polling (or NO_SYS) setup. The packets are put on a list and netif_poll() must be called in the main application loop.

Definition at line 1194 of file lwipopts.h.

9.9.2.87 #define LWIP_NETIF_REMOVE_CALLBACK 0

LWIP_NETIF_REMOVE_CALLBACK==1: Support a callback function that is called when a netif has been removed

Definition at line 1150 of file lwipopts.h.

9.9.2.88 #define LWIP_NETIF_STATUS_CALLBACK 0

LWIP_NETIF_STATUS_CALLBACK==1: Support a callback function whenever an interface changes its up/down status (i.e., due to DHCP IP acquisition)

Definition at line 1134 of file lwipopts.h.

9.9.2.89 #define LWIP_NETIF_TX_SINGLE_PBUF 0

LWIP_NETIF_TX_SINGLE_PBUF: if this is set to 1, lwIP tries to put all data to be sent into one single pbuf. This is for compatibility with DMA-enabled MACs that do not support scatter-gather. Beware that this might involve

CPU-memcpy before transmitting that would not be needed without this flag! Use this only if you need to!

Todo : TCP and IP-frag do not work with this, yet:

Definition at line 1207 of file lwipopts.h.

9.9.2.90 `#define LWIP_POSIX_SOCKETS_IO_NAMES 1`

LWIP_POSIX_SOCKETS_IO_NAMES==1: Enable POSIX-style sockets functions names. Disable this option if you use a POSIX operating system that uses the same names (read, write & close). (only used if you use sockets.c)

Definition at line 1445 of file lwipopts.h.

9.9.2.91 `#define LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS 0`

LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS==1: randomize the local port for the first local TCP/UDP pcb (default==0). This can prevent creating predictable port numbers after booting a device.

Definition at line 611 of file lwipopts.h.

9.9.2.92 `#define LWIP_RAW 1`

LWIP_RAW==1: Enable application layer to hook into the IP layer itself.

Definition at line 657 of file lwipopts.h.

9.9.2.93 `#define LWIP_SNMP 0`

LWIP_SNMP==1: Turn on SNMP module. UDP must be available for SNMP transport.

Definition at line 735 of file lwipopts.h.

9.9.2.94 `#define LWIP_SO_RCVBUF 0`

LWIP_SO_RCVBUF==1: Enable SO_RCVBUF processing.

Definition at line 1477 of file lwipopts.h.

9.9.2.95 `#define LWIP_SO_RCVTIMEO 1`

LWIP_SO_RCVTIMEO==1: Enable receive timeout for sockets/netconns and SO_RCVTIMEO processing.

Definition at line 1470 of file lwipopts.h.

9.9.2.96 `#define LWIP_SO_SNDTIMEO 1`

LWIP_SO_SNDTIMEO==1: Enable send timeout for sockets/netconns and SO_SNDTIMEO processing.

Definition at line 1462 of file lwipopts.h.

9.9.2.97 `#define LWIP_SOCKET 1`

LWIP_SOCKET==1: Enable Socket API (require to use sockets.c)

Definition at line 1428 of file lwipopts.h.

9.9.2.98 #define LWIP_STATS 1

LWIP_STATS==1: Enable statistics collection in lwip_stats.

Definition at line 1512 of file lwipopts.h.

9.9.2.99 #define LWIP_STATS_DISPLAY 0

LWIP_STATS_DISPLAY==1: Compile in the statistics output functions.

Definition at line 1521 of file lwipopts.h.

9.9.2.100 #define LWIP_TCP 1

LWIP_TCP==1: Turn on TCP.

Definition at line 910 of file lwipopts.h.

9.9.2.101 #define LWIP_TCP_KEEPALIVE 1

LWIP_TCP_KEEPALIVE==1: Enable TCP_KEEPIIDLE, TCP_KEEPIIDLE and TCP_KEEPCNT options processing. Note that TCP_KEEPIIDLE and TCP_KEEPIIDLE have to be set in seconds. (does not require sockets.c, and will affect tcp.c)

Definition at line 1454 of file lwipopts.h.

9.9.2.102 #define LWIP_TCP_TIMESTAMPS 0

LWIP_TCP_TIMESTAMPS==1: support the TCP timestamp option.

Definition at line 1062 of file lwipopts.h.

9.9.2.103 #define LWIP_TCPIP_CORE_LOCKING 0

LWIP_TCPIP_CORE_LOCKING: (EXPERIMENTAL!) Don't use it if you're not an active lwIP project member

Definition at line 1394 of file lwipopts.h.

9.9.2.104 #define LWIP_TCPIP_CORE_LOCKING_INPUT 0

LWIP_TCPIP_CORE_LOCKING_INPUT: (EXPERIMENTAL!) Don't use it if you're not an active lwIP project member

Definition at line 1402 of file lwipopts.h.

9.9.2.105 #define LWIP_TCPIP_TIMEOUT 1

LWIP_TCPIP_TIMEOUT==1: Enable tcpip_timeout/tcpip_untimeout to create timers running in tcpip_thread from another thread.

Definition at line 1416 of file lwipopts.h.

9.9.2.106 #define LWIP_UDP 1

LWIP_UDP==1: Turn on UDP.

Definition at line 877 of file lwipopts.h.

9.9.2.107 #define LWIP_UDPLITE 0

LWIP_UDPLITE==1: Turn on UDP-Lite. (Requires LWIP_UDP)

Definition at line 884 of file lwipopts.h.

9.9.2.108 #define MEM_ALIGNMENT 4

MEM_ALIGNMENT: should be set to the alignment of the CPU 4 byte alignment -> #define MEM_ALIGNMENT 4
2 byte alignment -> #define MEM_ALIGNMENT 2

Definition at line 118 of file lwipopts.h.

9.9.2.109 #define MEM_DEBUG LWIP_DBG_OFF

MEM_DEBUG: Enable debugging in mem.c.

Definition at line 1984 of file lwipopts.h.

9.9.2.110 #define MEM_LIBC_MALLOC 0

MEM_LIBC_MALLOC==1: Use malloc/free/realloc provided by your C-library instead of the lwip internal allocator.
Can save code size if you already use it.

Definition at line 100 of file lwipopts.h.

9.9.2.111 #define MEM_SIZE 1600

MEM_SIZE: the size of the heap memory. If the application will send a lot of data that needs to be copied, this should be set high.

Definition at line 126 of file lwipopts.h.

9.9.2.112 #define MEM_STATS ((MEM_LIBC_MALLOC == 0) && (MEM_USE_POOLS == 0))

MEM_STATS==1: Enable mem.c stats.

Definition at line 1587 of file lwipopts.h.

9.9.2.113 #define MEM_USE_POOLS 1

MEM_USE_POOLS==1: Use an alternative to malloc() by allocating from a set of memory pools of various sizes. When mem_malloc is called, an element of the smallest pool that can provide the length needed is returned. To use this, MEM_USE_CUSTOM_POOLS also has to be enabled.

Definition at line 166 of file lwipopts.h.

9.9.2.114 #define MEM_USE_POOLS_TRY_BIGGER_POOL 0

MEM_USE_POOLS_TRY_BIGGER_POOL==1: if one malloc-pool is empty, try the next bigger pool - WARNING: THIS MIGHT WASTE MEMORY but it can make a system more reliable.

Definition at line 174 of file lwipopts.h.

9.9.2.115 #define MEMCPY(*dst*, *src*, *len*) memcpy(dst,src,len)

MEMCPY: override this if you have a faster implementation at hand than the one included in your C library

Definition at line 78 of file lwipopts.h.

9.9.2.116 #define MEMP_DEBUG LWIP_DBG_OFF

MEMP_DEBUG: Enable debugging in memp.c.

Definition at line 1991 of file lwipopts.h.

9.9.2.117 #define MEMP_MEM_MALLOC 0

MEMP_MEM_MALLOC==1: Use mem_malloc/mem_free instead of the lwip pool allocator. Especially useful with MEM_LIBC_MALLOC but handle with care regarding execution speed and usage from interrupts!

Definition at line 109 of file lwipopts.h.

9.9.2.118 #define MEMP_NUM_ARP_QUEUE 30

MEMP_NUM_ARP_QUEUE: the number of simultaneously queued outgoing packets (pbufs) that are waiting for an ARP request (to resolve their destination address) to finish. (requires the ARP_QUEUEING option)

Definition at line 290 of file lwipopts.h.

9.9.2.119 #define MEMP_NUM_FRAG_PBUF 15

MEMP_NUM_FRAG_PBUF: the number of IP fragments simultaneously sent (fragments, not whole packets!). This is only used with IP_FRAG_USES_STATIC_BUF==0 and LWIP_NETIF_TX_SINGLE_PBUF==0 and only has to be > 1 with DMA-enabled MACs where the packet is not yet sent when netif->output returns.

Definition at line 280 of file lwipopts.h.

9.9.2.120 #define MEMP_NUM_IGMP_GROUP 8

MEMP_NUM_IGMP_GROUP: The number of multicast groups whose network interfaces can be members at the same time (one per netif - allsystems group -, plus one per netif membership). (requires the LWIP_IGMP option)

Definition at line 300 of file lwipopts.h.

9.9.2.121 #define MEMP_NUM_LOCALHOSTLIST 1

MEMP_NUM_LOCALHOSTLIST: the number of host entries in the local host list if DNS_LOCAL_HOSTLIST_IS_DYNAMIC==1.

Definition at line 393 of file lwipopts.h.

9.9.2.122 #define MEMP_NUM_NETBUF 2

MEMP_NUM_NETBUF: the number of struct netbufs. (only needed if you use the sequential API, like api_lib.c)

Definition at line 318 of file lwipopts.h.

9.9.2.123 #define MEMP_NUM_NETCONN 4

MEMP_NUM_NETCONN: the number of struct netconns. (only needed if you use the sequential API, like api_lib.c)
Definition at line 326 of file lwipopts.h.

9.9.2.124 #define MEMP_NUM_NETDB 1

MEMP_NUM_NETDB: the number of concurrently running lwip_addrinfo() calls (before freeing the corresponding memory using lwip_freeaddrinfo()).
Definition at line 385 of file lwipopts.h.

9.9.2.125 #define MEMP_NUM_PBUF 16

MEMP_NUM_PBUF: the number of memp struct pbufs (used for PBUF_ROM and PBUF_REF). If the application sends a lot of data out of ROM (or other static memory), this should be set high.
Definition at line 220 of file lwipopts.h.

9.9.2.126 #define MEMP_NUM_PPPOE_INTERFACES 1

MEMP_NUM_PPPOE_INTERFACES: the number of concurrently active PPPoE interfaces (only used with PPP↔OE_SUPPORT==1)
Definition at line 401 of file lwipopts.h.

9.9.2.127 #define MEMP_NUM_RAW_PCB 4

MEMP_NUM_RAW_PCB: Number of raw connection PCBs (requires the LWIP_RAW option)
Definition at line 228 of file lwipopts.h.

9.9.2.128 #define MEMP_NUM_REASSDATA 5

MEMP_NUM_REASSDATA: the number of IP packets simultaneously queued for reassembly (whole packets, not fragments!)
Definition at line 269 of file lwipopts.h.

9.9.2.129 #define MEMP_NUM_SNMP_NODE 50

MEMP_NUM_SNMP_NODE: the number of leafs in the SNMP tree.
Definition at line 351 of file lwipopts.h.

9.9.2.130 #define MEMP_NUM_SNMP_ROOTNODE 30

MEMP_NUM_SNMP_ROOTNODE: the number of branches in the SNMP tree. Every branch has one leaf (MEM↔P_NUM_SNMP_NODE) at least!
Definition at line 359 of file lwipopts.h.

9.9.2.131 #define MEMP_NUM_SNMP_VALUE 3

MEMP_NUM_SNMP_VALUE: the number of OID or values concurrently used (does not have to be changed normally) - 3 of these are used per request (1 for the value read and 2 for OIDs - input and output)

Definition at line 377 of file lwipopts.h.

9.9.2.132 #define MEMP_NUM_SNMP_VARBIND 2

MEMP_NUM_SNMP_VARBIND: the number of concurrent requests (does not have to be changed normally) - 2 of these are used per request (1 for input, 1 for output)

Definition at line 368 of file lwipopts.h.

9.9.2.133 #define MEMP_NUM_SYS_TIMEOUT (LWIP_TCP + IP_REASSEMBLY + LWIP_ARP + (2*LWIP_DHCP) + LWIP_AUTOIP + LWIP_IGMP + LWIP_DNS + PPP_SUPPORT)

MEMP_NUM_SYS_TIMEOUT: the number of simultaneously active timeouts. (requires NO_SYS==0) The default number of timeouts is calculated here for all enabled modules. The formula expects settings to be either '0' or '1'.

Definition at line 310 of file lwipopts.h.

9.9.2.134 #define MEMP_NUM_TCP_PCB 5

MEMP_NUM_TCP_PCB: the number of simulatenously active TCP connections. (requires the LWIP_TCP option)

Definition at line 245 of file lwipopts.h.

9.9.2.135 #define MEMP_NUM_TCP_PCB_LISTEN 8

MEMP_NUM_TCP_PCB_LISTEN: the number of listening TCP connections. (requires the LWIP_TCP option)

Definition at line 253 of file lwipopts.h.

9.9.2.136 #define MEMP_NUM_TCP_SEG 16

MEMP_NUM_TCP_SEG: the number of simultaneously queued TCP segments. (requires the LWIP_TCP option)

Definition at line 261 of file lwipopts.h.

9.9.2.137 #define MEMP_NUM_TCPIP_MSG_API 8

MEMP_NUM_TCPIP_MSG_API: the number of struct tcpip_msg, which are used for callback/timeout API communication. (only needed if you use tcpip.c)

Definition at line 335 of file lwipopts.h.

9.9.2.138 #define MEMP_NUM_TCPIP_MSG_INPKT 8

MEMP_NUM_TCPIP_MSG_INPKT: the number of struct tcpip_msg, which are used for incoming packets. (only needed if you use tcpip.c)

Definition at line 344 of file lwipopts.h.

9.9.2.139 #define MEMP_NUM_UDP_PCB 4

MEMP_NUM_UDP_PCB: the number of UDP protocol control blocks. One per active UDP "connection". (requires the LWIP_UDP option)

Definition at line 237 of file lwipopts.h.

9.9.2.140 #define MEMP_OVERFLOW_CHECK 0

MEMP_OVERFLOW_CHECK: memp overflow protection reserves a configurable amount of bytes before and after each memp element in every pool and fills it with a prominent default value. MEMP_OVERFLOW_CHECK == 0 no checking MEMP_OVERFLOW_CHECK == 1 checks each element when it is freed MEMP_OVERFLOW_CHECK >= 2 checks each element in every pool every time memp_malloc() or memp_free() is called (useful but slow!)

Definition at line 148 of file lwipopts.h.

9.9.2.141 #define MEMP_SANITY_CHECK 0

MEMP_SANITY_CHECK==1: run a sanity check after each memp_free() to make sure that there are no cycles in the linked lists.

Definition at line 156 of file lwipopts.h.

9.9.2.142 #define MEMP_SEPARATE_POOLS 0

MEMP_SEPARATE_POOLS: if defined to 1, each pool is placed in its own array. This can be used to individually change the location of each pool. Default is one big array for all pools

Definition at line 135 of file lwipopts.h.

9.9.2.143 #define MEMP_STATS (MEMP_MEM_MALLOC == 0)

MEMP_STATS==1: Enable memp.c pool stats.

Definition at line 1594 of file lwipopts.h.

9.9.2.144 #define MEMP_USE_CUSTOM_POOLS 1

MEMP_USE_CUSTOM_POOLS==1: whether to include a user file [lwippools.h](#) that defines additional pools beyond the "standard" ones required by lwIP. If you set this to 1, you must have [lwippools.h](#) in your include path somewhere.

Definition at line 184 of file lwipopts.h.

9.9.2.145 #define NETIF_DEBUG LWIP_DBG_OFF

NETIF_DEBUG: Enable debugging in netif.c.

Definition at line 1907 of file lwipopts.h.

9.9.2.146 #define NO_SYS 0

NO_SYS==1: Provides VERY minimal functionality. Otherwise, use lwIP facilities.

Definition at line 62 of file lwipopts.h.

9.9.2.147 #define NO_SYS_NO_TIMERS 0

NO_SYS_NO_TIMERS==1: Drop support for sys_timeout when NO_SYS==1. Mainly for compatibility to old versions.

Definition at line 70 of file lwipopts.h.

9.9.2.148 #define PBUF_DEBUG LWIP_DBG_OFF

PBUF_DEBUG: Enable debugging in pbuf.c.

Definition at line 1914 of file lwipopts.h.

9.9.2.149 #define PBUF_LINK_HLEN (14 + ETH_PAD_SIZE)

PBUF_LINK_HLEN: the number of bytes that should be allocated for a link level header. The default is 14, the standard value for Ethernet.

Definition at line 1097 of file lwipopts.h.

9.9.2.150 #define PBUF_POOL_BUFSIZE LWIP_MEM_ALIGN_SIZE(TCP_MSS+40+PBUF_LINK_HLEN)

PBUF_POOL_BUFSIZE: the size of each pbuf in the pbuf pool. The default is designed to accomodate single full size TCP frame in one pbuf, including TCP_MSS, IP header, and link header.

Definition at line 1106 of file lwipopts.h.

9.9.2.151 #define PBUF_POOL_SIZE 16

PBUF_POOL_SIZE: the number of buffers in the pbuf pool.

Definition at line 408 of file lwipopts.h.

9.9.2.152 #define PPP_DEBUG LWIP_DBG_OFF

PPP_DEBUG: Enable debugging for PPP.

Definition at line 2090 of file lwipopts.h.

9.9.2.153 #define PPP_SUPPORT 0

PPP_SUPPORT==1: Enable PPP.

Definition at line 1629 of file lwipopts.h.

9.9.2.154 #define PPP_THREAD_NAME "pppInputThread"

PPP_THREAD_NAME: The name assigned to the pppInputThread.

Definition at line 1302 of file lwipopts.h.

9.9.2.155 #define PPP_THREAD_PRIO (LOWPRIO + 1)

PPP_THREAD_PRIO: The priority assigned to the pppInputThread. The priority value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1320 of file lwipopts.h.

9.9.2.156 #define PPP_THREAD_STACKSIZE 1024

PPP_THREAD_STACKSIZE: The stack size used by the pppInputThread. The stack size value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1311 of file lwipopts.h.

9.9.2.157 #define PPPOE_SUPPORT 0

PPPOE_SUPPORT==1: Enable PPP Over Ethernet

Definition at line 1636 of file lwipopts.h.

9.9.2.158 #define PPPOS_SUPPORT PPP_SUPPORT

PPPOS_SUPPORT==1: Enable PPP Over Serial

Definition at line 1643 of file lwipopts.h.

9.9.2.159 #define RAW_DEBUG LWIP_DBG_OFF

RAW_DEBUG: Enable debugging in raw.c.

Definition at line 1977 of file lwipopts.h.

9.9.2.160 #define RAW_TTL (IP_DEFAULT_TTL)

LWIP_RAW==1: Enable application layer to hook into the IP layer itself.

Definition at line 664 of file lwipopts.h.

9.9.2.161 #define RECV_BUFSIZE_DEFAULT INT_MAX

If LWIP_SO_RCVBUF is used, this is the default value for recv_bufsize.

Definition at line 1484 of file lwipopts.h.

9.9.2.162 #define SLIP_DEBUG LWIP_DBG_OFF

SLIP_DEBUG: Enable debugging in slipif.c.

Definition at line 2097 of file lwipopts.h.

9.9.2.163 #define SLIPIF_THREAD_NAME "slipif_loop"

SLIPIF_THREAD_NAME: The name assigned to the slipif_loop thread.

Definition at line 1277 of file lwipopts.h.

9.9.2.164 #define SLIPIF_THREAD_PRIO (LOWPRIO + 1)

SLIPIF_THREAD_PRIO: The priority assigned to the slipif_loop thread. The priority value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1295 of file lwipopts.h.

9.9.2.165 #define SLIPIF_THREAD_STACKSIZE 1024

SLIP_THREAD_STACKSIZE: The stack size used by the slipif_loop thread. The stack size value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1286 of file lwipopts.h.

9.9.2.166 #define SMEMCPY(dst, src, len) memcpy(dst,src,len)

SMEMCPY: override this with care! Some compilers (e.g. gcc) can inline a call to memcpy() if the length is known at compile time and is small.

Definition at line 86 of file lwipopts.h.

9.9.2.167 #define SNMP_CONCURRENT_REQUESTS 1

SNMP_CONCURRENT_REQUESTS: Number of concurrent requests the module will allow. At least one request buffer is required. Does not have to be changed unless external MIBs answer request asynchronously

Definition at line 744 of file lwipopts.h.

9.9.2.168 #define SNMP_MAX_OCTET_STRING_LEN 127

The maximum length of strings used. This affects the size of MEMP_SNMP_VALUE elements.

Definition at line 778 of file lwipopts.h.

9.9.2.169 #define SNMP_MAX_TREE_DEPTH 15

The maximum depth of the SNMP tree. With private MIBs enabled, this depends on your MIB! This affects the size of MEMP_SNMP_VALUE elements.

Definition at line 787 of file lwipopts.h.

9.9.2.170 #define SNMP_MAX_VALUE_SIZE LWIP_MAX((SNMP_MAX_OCTET_STRING_LEN)+1, sizeof(s32_t)*(SNMP_MAX_TREE_DEPTH))

The size of the MEMP_SNMP_VALUE elements, normally calculated from SNMP_MAX_OCTET_STRING_LEN and SNMP_MAX_TREE_DEPTH.

Definition at line 795 of file lwipopts.h.

9.9.2.171 #define SNMP_MIB_DEBUG LWIP_DBG_OFF

SNMP_MIB_DEBUG: Enable debugging for SNMP MIBs.

Definition at line 2125 of file lwipopts.h.

9.9.2.172 #define SNMP_MSG_DEBUG LWIP_DBG_OFF

SNMP_MSG_DEBUG: Enable debugging for SNMP messages.

Definition at line 2118 of file lwipopts.h.

9.9.2.173 #define SNMP_PRIVATE_MIB 0

SNMP_PRIVATE_MIB: When using a private MIB, you have to create a file 'private_mib.h' that contains a 'struct mib_array_node mib_private' which contains your MIB.

Definition at line 761 of file lwipopts.h.

9.9.2.174 #define SNMP_SAFE_REQUESTS 1

Only allow SNMP write actions that are 'safe' (e.g. disabling netifs is not a safe action and disabled when SNMP_SAFE_REQUESTS = 1). Unsafe requests are disabled by default!

Definition at line 770 of file lwipopts.h.

9.9.2.175 #define SNMP_TRAP_DESTINATIONS 1

SNMP_TRAP_DESTINATIONS: Number of trap destinations. At least one trap destination is required

Definition at line 752 of file lwipopts.h.

9.9.2.176 #define SO_REUSE 0

SO_REUSE==1: Enable SO_REUSEADDR option.

Definition at line 1491 of file lwipopts.h.

9.9.2.177 #define SO_REUSE_RXTOALL 0

SO_REUSE_RXTOALL==1: Pass a copy of incoming broadcast/multicast packets to all local matches if SO_REUSEADDR is turned on. WARNING: Adds a memcpy for every packet if passing to more than one pcb!

Definition at line 1500 of file lwipopts.h.

9.9.2.178 #define SOCKETS_DEBUG LWIP_DBG_OFF

SOCKETS_DEBUG: Enable debugging in sockets.c.

Definition at line 1935 of file lwipopts.h.

9.9.2.179 #define SYS_DEBUG LWIP_DBG_OFF

SYS_DEBUG: Enable debugging in sys.c.

Definition at line 1998 of file lwipopts.h.

9.9.2.180 #define SYS_LIGHTWEIGHT_PROT 0

SYS_LIGHTWEIGHT_PROT==1: if you want inter-task protection for certain critical regions during buffer allocation, deallocation and memory allocation and deallocation.

Definition at line 54 of file lwipopts.h.

9.9.2.181 #define SYS_STATS (NO_SYS == 0)

SYS_STATS==1: Enable system stats (sem and mbox counts, etc).

Definition at line 1601 of file lwipopts.h.

9.9.2.182 #define TCP_CALCULATE_EFF_SEND_MSS 1

TCP_CALCULATE_EFF_SEND_MSS: "The maximum size of a segment that TCP really sends, the 'effective send MSS,' MUST be the smaller of the send MSS (which reflects the available reassembly buffer size at the remote host) and the largest size permitted by the IP layer" (RFC 1122) Setting this to 1 enables code that checks TCP_MSS against the MTU of the netif used for a connection and limits the MSS if it would be too big otherwise.

Definition at line 970 of file lwipopts.h.

9.9.2.183 #define TCP_CWND_DEBUG LWIP_DBG_OFF

TCP_CWND_DEBUG: Enable debugging for TCP congestion window.

Definition at line 2041 of file lwipopts.h.

9.9.2.184 #define TCP_DEBUG LWIP_DBG_OFF

TCP_DEBUG: Enable debugging for TCP.

Definition at line 2012 of file lwipopts.h.

9.9.2.185 #define TCP_DEFAULT_LISTEN_BACKLOG 0xff

The maximum allowed backlog for TCP listen netconns. This backlog is used unless another is explicitly specified. 0xff is the maximum (u8_t).

Definition at line 1037 of file lwipopts.h.

9.9.2.186 #define TCP_FR_DEBUG LWIP_DBG_OFF

TCP_FR_DEBUG: Enable debugging in tcp_in.c for fast retransmit.

Definition at line 2026 of file lwipopts.h.

9.9.2.187 #define TCP_INPUT_DEBUG LWIP_DBG_OFF

TCP_INPUT_DEBUG: Enable debugging in tcp_in.c for incoming debug.

Definition at line 2019 of file lwipopts.h.

9.9.2.188 #define TCP_LISTEN_BACKLOG 0

TCP_LISTEN_BACKLOG: Enable the backlog option for tcp listen pcb.

Definition at line 1028 of file lwipopts.h.

9.9.2.189 #define TCP_MAXRTX 12

TCP_MAXRTX: Maximum number of retransmissions of data segments.

Definition at line 932 of file lwipopts.h.

9.9.2.190 #define TCP_MSS 536

TCP_MSS: TCP Maximum segment size. (default is 536, a conservative default, you might want to increase this.) For the receive side, this MSS is advertised to the remote side when opening a connection. For the transmit size, this MSS sets an upper limit on the MSS advertised by the remote host.

Definition at line 958 of file lwipopts.h.

9.9.2.191 `#define TCP_OOSEQ_MAX_BYTES 0`

`TCP_OOSEQ_MAX_BYTES`: The maximum number of bytes queued on ooseq per pcb. Default is 0 (no limit). Only valid for `TCP_QUEUE_OOSEQ==0`.

Definition at line 1013 of file lwipopts.h.

9.9.2.192 `#define TCP_OOSEQ_MAX_PBUFS 0`

`TCP_OOSEQ_MAX_PBUFS`: The maximum number of pbufs queued on ooseq per pcb. Default is 0 (no limit). Only valid for `TCP_QUEUE_OOSEQ==0`.

Definition at line 1021 of file lwipopts.h.

9.9.2.193 `#define TCP_OUTPUT_DEBUG LWIP_DBG_OFF`

`TCP_OUTPUT_DEBUG`: Enable debugging in tcp_out.c output functions.

Definition at line 2055 of file lwipopts.h.

9.9.2.194 `#define TCP_OVERSIZE TCP_MSS`

`TCP_OVERSIZE`: The maximum number of bytes that tcp_write may allocate ahead of time in an attempt to create shorter pbuf chains for transmission. The meaningful range is 0 to `TCP_MSS`. Some suggested values are:

0: Disable oversized allocation. Each tcp_write() allocates a new pbuf (old behaviour). 1: Allocate size-aligned pbufs with minimal excess. Use this if your scatter-gather DMA requires aligned fragments. 128: Limit the pbuf/memory overhead to 20%. `TCP_MSS`: Try to create unfragmented TCP packets. `TCP_MSS/4`: Try to create 4 fragments or less per TCP packet.

Definition at line 1055 of file lwipopts.h.

9.9.2.195 `#define TCP_QLEN_DEBUG LWIP_DBG_OFF`

`TCP_QLEN_DEBUG`: Enable debugging for TCP queue lengths.

Definition at line 2069 of file lwipopts.h.

9.9.2.196 `#define TCP_QUEUE_OOSEQ (LWIP_TCP)`

`TCP_QUEUE_OOSEQ==1`: TCP will queue segments that arrive out of order. Define to 0 if your device is low on memory.

Definition at line 947 of file lwipopts.h.

9.9.2.197 `#define TCP_RST_DEBUG LWIP_DBG_OFF`

`TCP_RST_DEBUG`: Enable debugging for TCP with the RST message.

Definition at line 2062 of file lwipopts.h.

9.9.2.198 `#define TCP_RTO_DEBUG LWIP_DBG_OFF`

`TCP_RTO_DEBUG`: Enable debugging in TCP for retransmit timeout.

Definition at line 2034 of file lwipopts.h.

9.9.2.199 `#define TCP_SND_BUF (2 * TCP_MSS)`

`TCP_SND_BUF`: TCP sender buffer space (bytes). To achieve good performance, this should be at least $2 * TCP_MSS$.

Definition at line 979 of file lwipopts.h.

9.9.2.200 `#define TCP_SND_QUEUELEN ((4 * (TCP_SND_BUF) + (TCP_MSS - 1))/(TCP_MSS))`

`TCP_SND_QUEUELEN`: TCP sender buffer space (pbufs). This must be at least as much as $(2 * TCP_SND_BUF / TCP_MSS)$ for things to work.

Definition at line 987 of file lwipopts.h.

9.9.2.201 `#define TCP_SNDLOWAT LWIP_MIN(LWIP_MAX(((TCP_SND_BUF)/2), (2 * TCP_MSS) + 1), (TCP_SND_BUF) - 1)`

`TCP_SNDLOWAT`: TCP writable space (bytes). This must be less than `TCP_SND_BUF`. It is the amount of space which must be available in the TCP `snd_buf` for select to return writable (combined with `TCP_SNDQUEUELOWAT`).

Definition at line 996 of file lwipopts.h.

9.9.2.202 `#define TCP_SNDQUEUELOWAT LWIP_MAX(((TCP_SND_QUEUELEN)/2), 5)`

`TCP_SNDQUEUELOWAT`: TCP writable bufs (pbuf count). This must be less than `TCP_SND_QUEUELEN`. If the number of pbufs queued on a pcb drops below this number, select returns writable (combined with `TCP_SNDLOWAT`).

Definition at line 1005 of file lwipopts.h.

9.9.2.203 `#define TCP_STATS (LWIP_TCP)`

`TCP_STATS==1`: Enable TCP stats. Default is on if TCP enabled, otherwise off.

Definition at line 1580 of file lwipopts.h.

9.9.2.204 `#define TCP_SYNMAXRTX 6`

`TCP_SYNMAXRTX`: Maximum number of retransmissions of SYN segments.

Definition at line 939 of file lwipopts.h.

9.9.2.205 `#define TCP_TTL (IP_DEFAULT_TTL)`

`TCP_TTL`: Default Time-To-Live value.

Definition at line 917 of file lwipopts.h.

9.9.2.206 `#define TCP_WND (4 * TCP_MSS)`

`TCP_WND`: The size of a TCP window. This must be at least $(2 * TCP_MSS)$ for things to work well

Definition at line 925 of file lwipopts.h.

9.9.2.207 #define TCP_WND_DEBUG LWIP_DBG_OFF

TCP_WND_DEBUG: Enable debugging in tcp_in.c for window updating.

Definition at line 2048 of file lwipopts.h.

9.9.2.208 #define TCP_WND_UPDATE_THRESHOLD (TCP_WND / 4)

TCP_WND_UPDATE_THRESHOLD: difference in window to trigger an explicit window update

Definition at line 1070 of file lwipopts.h.

9.9.2.209 #define TCPIP_DEBUG LWIP_DBG_OFF

TCPIP_DEBUG: Enable debugging in tcpip.c.

Definition at line 2083 of file lwipopts.h.

9.9.2.210 #define TCPIP_MBOX_SIZE MEMP_NUM_PBUF

TCPIP_MBOX_SIZE: The mailbox size for the tcpip thread messages. The queue size value itself is platform-dependent, but is passed to sys_mbox_new() when tcpip_init is called.

Definition at line 1270 of file lwipopts.h.

9.9.2.211 #define TCPIP_THREAD_NAME "tcpip_thread"

TCPIP_THREAD_NAME: The name assigned to the main tcpip thread.

Definition at line 1243 of file lwipopts.h.

9.9.2.212 #define TCPIP_THREAD_PRIO (LOWPRIO + 1)

TCPIP_THREAD_PRIO: The priority assigned to the main tcpip thread. The priority value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1261 of file lwipopts.h.

9.9.2.213 #define TCPIP_THREAD_STACKSIZE 1024

TCPIP_THREAD_STACKSIZE: The stack size used by the main tcpip thread. The stack size value itself is platform-dependent, but is passed to sys_thread_new() when the thread is created.

Definition at line 1252 of file lwipopts.h.

9.9.2.214 #define TIMERS_DEBUG LWIP_DBG_OFF

TIMERS_DEBUG: Enable debugging in timers.c.

Definition at line 2005 of file lwipopts.h.

9.9.2.215 #define UDP_DEBUG LWIP_DBG_OFF

UDP_DEBUG: Enable debugging in UDP.

Definition at line 2076 of file lwipopts.h.

The maximum permitted size of a packet.

Typedefs

- typedef struct [_packet_obj_t](#) **packet_obj_t**
- typedef struct [_relay_queue_t](#) **relay_queue_t**

Functions

- void [packet_linked_list_init](#) (void)
Initialize the empty linked list.
- bool [packet_write](#) ([packet_obj_t](#) *packet, uint8_t *data, int32_t length)
Copy the packet to the packet object.
- bool [packet_read](#) ([packet_obj_t](#) *packet, uint8_t *data, int32_t *length)
Copy the data from the packet object.
- bool [packet_transfer](#) ([relay_queue_t](#) *src, [relay_queue_t](#) *dst)
Transfer a packet resource from one queue to another.
- [packet_obj_t](#) * [packet_get_front](#) ([relay_queue_t](#) *queue)
Obtain the front item but don't pop it.

Variables

- [relay_queue_t](#) [g_empty_queue](#)
The empty queue where all packet resources are initially stored.
- [relay_queue_t](#) [g_relay_queue](#)
A queue of packets that must be processed by the relay logic.

9.10.1 Detailed Description

A linked list resource used to share packet storage across the comms interface.

Author

neil

Definition in file [packet_linked_list.h](#).

9.10.2 Function Documentation

9.10.2.1 [packet_obj_t](#)* [packet_get_front](#) ([relay_queue_t](#) * *queue*)

Obtain the front item but don't pop it.

Returns

Return the front packet object

Definition at line 157 of file [packet_linked_list.c](#).

9.10.2.2 `bool packet_read ( packet_obj_t * packet, uint8_t * data, int32_t * length )`

Copy the data from the packet object.

Returns

Returns true if successful, false otherwise

Definition at line 125 of file packet_linked_list.c.

9.10.2.3 `bool packet_transfer ( relay_queue_t * src, relay_queue_t * dst )`

Transfer a packet resource from one queue to another.

Returns

Returns true if successful, false otherwise

Definition at line 138 of file packet_linked_list.c.

9.10.2.4 `bool packet_write ( packet_obj_t * packet, uint8_t * data, int32_t length )`

Copy the packet to the packet object.

Returns

Returns true if successful, false otherwise

Definition at line 112 of file packet_linked_list.c.

9.11 comms/rs485_if.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "comms/rs485_if.h"
#include "osdp_header.h"
#include "comms/comms_hub.h"
#include "comms/comms_relay.h"
#include "comms/packet_linked_list.h"
#include "global.h"
#include <string.h>
```

Include dependency graph for rs485_if.c:



Macros

- `#define RS485_ITERATION_DELAY 1`
Delay between checks on whether to switch from receive to transmit.
- `#define RS485_NUM_BUFFERS 8`
The number of transmit buffers.
- `#define RS485_BUFFER_SIZE COMMS_HUB_MAX_PACKET_SIZE`
The depth of each transmit buffer.

Functions

- **WORKING_AREA** (rs485_thread, [RS485_THREAD_SIZE](#))
- void **txend1** (UARTDriver *uartp)
- void **txend2** (UARTDriver *uartp)
- void **rxerr** (UARTDriver *uartp, uartflags_t e)
- void **rxchar** (UARTDriver *uartp, uint16_t c)
- void **rxend** (UARTDriver *uartp)
- void [rs485_start](#) (uint32_t baud)

An RS485 initialisation function for use in system initialisation.

- void [rs485_stop](#) (void)

Stop RS485 communications.

- msg_t [rs485_entry](#) (void *p)

The rs485 thread function.

Variables

- [relay_queue_t g_rs485_queue](#)
- [relay_queue_t g_rs485_incoming](#)
- Mailbox [mb_relay_comms](#)

9.11.1 Detailed Description

Author

neil

Definition in file [rs485_if.c](#).

9.11.2 Function Documentation

9.11.2.1 void rs485_start (uint32_t baud)

An RS485 initialisation function for use in system initialisation.

Parameters

<i>The</i>	selected baud rate
------------	--------------------

Definition at line 233 of file [rs485_if.c](#).

9.11.2.2 void rs485_stop (void)

Stop RS485 communications.

Parameters

<i>The</i>	selected baud rate
------------	--------------------

Definition at line 252 of file [rs485_if.c](#).

9.11.3 Variable Documentation

9.11.3.1 Mailbox mb_relay_comms

A working area for the relay mailbox

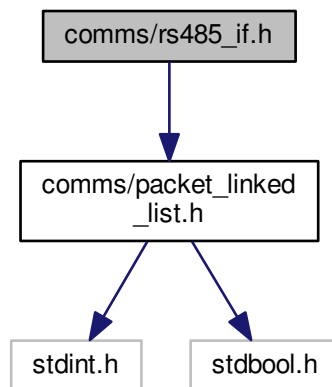
Definition at line 31 of file comms_relay.c.

9.12 comms/rs485_if.h File Reference

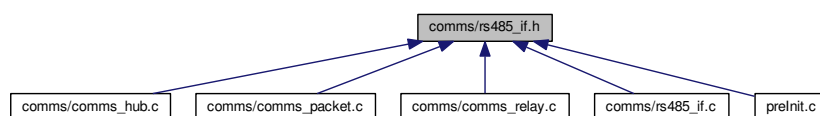
A C library for RS485 half-duplex communication.

```
#include "comms/packet_linked_list.h"
```

Include dependency graph for rs485_if.h:



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



Macros

- `#define RS485_THREAD_SIZE 1024`
Specify external linkage for `rs485_thread`.

Functions

- **WORKING_AREA** (`rs485_thread`, `RS485_THREAD_SIZE`)
- void `rs485_start` (`uint32_t` baud)

An RS485 initialisation function for use in system initialisation.

- void [rs485_stop](#) (void)

Stop RS485 communications.

- msg_t [rs485_entry](#) (void *p)

The rs485 thread function.

Variables

- [relay_queue_t g_rs485_queue](#)

9.12.1 Detailed Description

A C library for RS485 half-duplex communication.

Author

neil

Definition in file [rs485_if.h](#).

9.12.2 Function Documentation

9.12.2.1 void rs485_start (uint32_t baud)

An RS485 initialisation function for use in system initialisation.

Parameters

<i>The</i>	selected baud rate
------------	--------------------

Definition at line 233 of file [rs485_if.c](#).

9.12.2.2 void rs485_stop (void)

Stop RS485 communications.

Parameters

<i>The</i>	selected baud rate
------------	--------------------

Definition at line 252 of file [rs485_if.c](#).

9.13 comms/tcpclient_thread.c File Reference

```
#include "ch.h"
```

```

#include "hal.h"
#include <string.h>
#include "comms/tcpclient_thread.h"
#include "comms/comms_hub.h"
#include "comms/comms_relay.h"
#include "comms/packet_linked_list.h"
#include "comms/lwipopts.h"
#include "osdp_header.h"
#include "trace/trace.h"
#include "global.h"
#include "stringutils.h"
#include "lwip/tcp.h"
#include "lwip/sockets.h"
#include "osdp_ack_base.h"
#include "osdp_nak_base.h"
#include "osdp_comset_base.h"
#include "osdp_id_base.h"
#include "osdp_ms100_mfg_base.h"
#include "osdp_ms100_mfgrep_base.h"
#include "osdp_pdid_base.h"
#include "osdp_poll_base.h"
#include "osdp_cap_base.h"
#include "osdp_pdcap_base.h"
#include "osdp_lstat_base.h"
#include "osdp_lstatr_base.h"
#include "osdp_istat_base.h"
#include "osdp_istatr_base.h"

```

Include dependency graph for tcpclient_thread.c:



Macros

- **#define TCP_NUM_BUFFERS 8**
The number of transmit buffers.
- **#define TCP_BUFFER_SIZE COMMS_HUB_MAX_PACKET_SIZE**
The depth of each transmit buffer.
- **#define MS100_TCP_RECV_TIMEOUT 5**
The timeout used in TCP/IP receive operations.
- **#define MS100_TCP_SEND_TIMEOUT 5000**
The timeout used in TCP/IP send operations.
- **#define TCPIP_CLIENT_PORT 1234**
The TCP port to be used for the client socket connection.

Functions

- **WORKING_AREA** (wa_tcpclient_thread, **TCCLIENT_THREAD_SIZE**)
- void **tcpclient_init_ipv4** (uint32_t ipv4, int32_t port)
Declare an TCP/IP initialisation function.
- void **tcpclient_restart_ipv4** (uint32_t ipv4, int32_t port)
Restart TCP/IP communications with defined parameters.

- `msg_t tcpclient_entry` (void *p)
The TCP/IP Client thread function.

Variables

- `relay_queue_t g_eth_queue`
- `relay_queue_t g_eth_incoming`
- volatile bool `osdp_client_connection_flag` = false
A flag used to indicate that a OSDP Control Panel client connection has been made.

9.13.1 Detailed Description

Author

neil

Definition in file `tcpclient_thread.c`.

9.13.2 Function Documentation

9.13.2.1 void tcpclient_init_ipv4 (uint32_t *ipv4*, int32_t *port*)

Declare an TCP/IP initialisation function.

Parameters

<i>ipv4</i>	The selected IP address represented as a 32-bit word (big endian)
<i>port</i>	The selected port

Definition at line 230 of file `tcpclient_thread.c`.

Here is the call graph for this function:



9.13.2.2 void tcpclient_restart_ipv4 (uint32_t *ipv4*, int32_t *port*)

Restart TCP/IP communications with defined parameters.

Parameters

<i>ipv4</i>	The selected IP address represented as a 32-bit word (big endian)
<i>port</i>	The selected port

Todo Shutdown the thread, reconfigure the parameters and then create it again

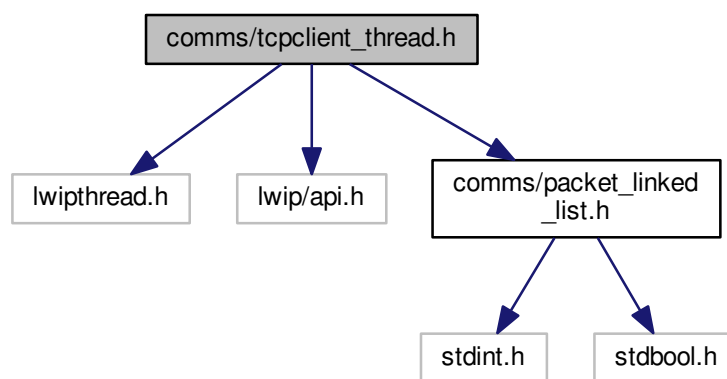
Definition at line 259 of file tcpclient_thread.c.

9.14 comms/tcpclient_thread.h File Reference

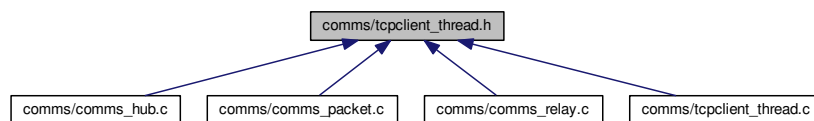
A C library for a TCP client using lwIP. Currently this library supports IPv4 addresses only.

```
#include "lwipthread.h"
#include "lwip/api.h"
#include "comms/packet_linked_list.h"
```

Include dependency graph for tcpclient_thread.h:



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



Macros

- `#define LWIP_SO_SNDTIMEO 1`
- `#define TCPLIENT_THREAD_SIZE 2048`
Specify external linkage for `tcpclient_thread`.

Functions

- **WORKING_AREA** (`wa_tcpclient_thread`, `TCPLIENT_THREAD_SIZE`)

- void [tcpclient_init_ipv4](#) (uint32_t ipv4, int32_t port)
Declare an TCP/IP initialisation function.
- void [tcpclient_restart_ipv4](#) (uint32_t ipv4, int32_t port)
Restart TCP/IP communications with defined parameters.
- msg_t [tcpclient_entry](#) (void *p)
The TCP/IP Client thread function.

Variables

- [relay_queue_t g_eth_queue](#)

9.14.1 Detailed Description

A C library for a TCP client using lwIP. Currently this library supports IPv4 addresses only.

Author

neil

Definition in file [tcpclient_thread.h](#).

9.14.2 Function Documentation

9.14.2.1 void tcpclient_init_ipv4 (uint32_t *ipv4*, int32_t *port*)

Declare an TCP/IP initialisation function.

Parameters

<i>ipv4</i>	The selected IP address represented as a 32-bit word (big endian)
<i>port</i>	The selected port

Definition at line 230 of file [tcpclient_thread.c](#).

Here is the call graph for this function:



9.14.2.2 void tcpclient_restart_ipv4 (uint32_t *ipv4*, int32_t *port*)

Restart TCP/IP communications with defined parameters.

Parameters

<i>ipv4</i>	The selected IP address represented as a 32-bit word (big endian)
<i>port</i>	The selected port

Todo Shutdown the thread, reconfigure the parameters and then create it again

Definition at line 259 of file tcpclient_thread.c.

9.15 comms/tcpserver.c File Reference

```
#include "tcpserver.h"
#include "ch.h"
#include "hal.h"
#include "string.h"
#include "global.h"
#include "comms/packet_linked_list.h"
#include "comms/comms_hub.h"
#include "comms/comms_packet.h"
#include "comms/comms_relay.h"
#include "osdp_hub_messages_base.h"
#include "link/link_if.h"
#include "link/link_alarm_list.h"
#include "lwip/api.h"
#include "lwip/tcp.h"
#include "lwip/sockets.h"
```

Include dependency graph for tcpserver.c:



Data Structures

- struct [_msg_ip_worker_t](#)

Macros

- #define [MS_DEBUG_LEN](#)(s) ((sizeof(s)/sizeof(s[0])) - 1)
This macro will return the equivalent of strlen()
- #define [DEBUG_BUFFER_SIZE](#) 80
The size of a buffer used to store formatted debug strings.
- #define [MAX_IP_CLIENTS](#) 1
The maximum supported number of clients supported by the threadpool.
- #define [MAX_IP_CLIENTS_FIELD](#) ((1 << [MAX_IP_CLIENTS](#)) - 1)
A bitfield used to encode which threads are busy/free.
- #define [IP_BUFFER_SIZE](#) 1024
The size of the message buffer.
- #define [B2B_TCP_RECV_TIMEOUT](#) 50
The timeout used in TCP/IP receive operations.
- #define [B2B_TCP_SEND_TIMEOUT](#) 5000
The timeout used in TCP/IP send operations.

Typedefs

- typedef struct [_msg_ip_worker_t](#) [msg_ip_worker_t](#)

Enumerations

- enum [b2b_auth_state_e](#) {
`ST_B2B_AUTH_IDLE = 0, ST_B2B_AUTH_0, ST_B2B_AUTH_1, ST_B2B_AUTH_1_RDY,`
`ST_B2B_AUTH_1_GET, ST_B2B_AUTH_2, ST_B2B_AUTH_3, ST_B2B_AUTH_3_RDY,`
`ST_B2B_AUTH_3_GET, ST_B2B_AUTH_OK }`

An enumerated type used to describe the current authentication status of a back-to-back connection.

Functions

- bool [send_auth](#) (int32_t interface, bool *is_authenticated)
- void [tcp_server](#) (int port)
- msg_t [server_thread](#) (void *p)

Variables

- [msg_ip_worker_t](#) [msg_ip](#) [4]
- volatile bool [b2b_server_connection_flag](#) = false
A flag used to indicate that a back-to-back connection has been made.
- [relay_queue_t](#) [g_server_in_queue](#)
- [relay_queue_t](#) [g_server_out_queue](#)
- [network_alarms_t](#) [g_network_alarms](#)
- [network_alarms_t](#) [g_known_alarms](#)

9.15.1 Detailed Description

Author

neil

Definition in file [tcpserver.c](#).

9.15.2 Typedef Documentation

9.15.2.1 typedef struct [_msg_ip_worker_t](#) [msg_ip_worker_t](#)

This struct is used to store the client socket, a flag to indicate the progress of the thread and a mutex controlling access to the struct contents

9.15.3 Function Documentation

9.15.3.1 bool [send_auth](#) (int32_t *interface*, bool * *is_authenticated*)

Todo

Todo

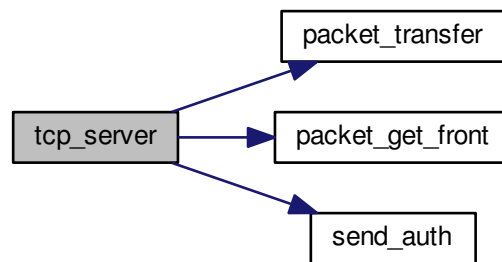
Definition at line 390 of file [tcpserver.c](#).

9.15.3.2 void tcp_server (int port)

A flag indicating if the connection is authenticated

Definition at line 664 of file tcpserver.c.

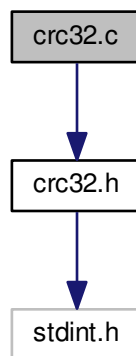
Here is the call graph for this function:



9.16 crc32.c File Reference

```
#include "crc32.h"
```

Include dependency graph for `crc32.c`:



Functions

- void [crc32_init](#) (uint32_t *crc)
Initialize a LUT and the CRC initial value.
- void [crc32_update](#) (uint32_t *crc, uint8_t data)
Byte oriented update of the CRC.

9.16.1 Detailed Description

Author

neil

Definition in file [crc32.c](#).

9.16.2 Function Documentation

9.16.2.1 void crc32_init (uint32_t * *crc*)

Initialize a LUT and the CRC initial value.

Parameters

<i>crc</i>	A pointer to the CRC value to be modified
------------	---

Todo Add the ability to dynamically create the CRC table or place it in ROM

Definition at line 13 of file [crc32.c](#).

9.16.2.2 void crc32_update (uint32_t * *crc*, uint8_t *data*)

Byte oriented update of the CRC.

Parameters

<i>crc</i>	A pointer to the CRC value to be modified
<i>data</i>	The byte to be consumed by the CRC calculation

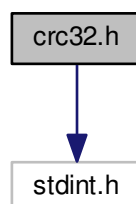
Definition at line 41 of file [crc32.c](#).

9.17 crc32.h File Reference

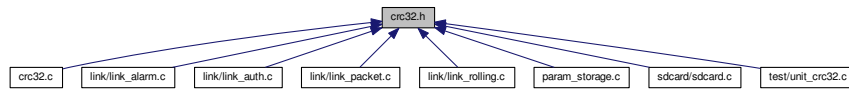
C functions for IEEE CRC-32 (reversed, non-inverted)

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

Include dependency graph for [crc32.h](#):



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



Functions

- void [crc32_init](#) (uint32_t *crc)
Initialize a LUT and the CRC initial value.
- void [crc32_update](#) (uint32_t *crc, uint8_t data)
Byte oriented update of the CRC.

9.17.1 Detailed Description

C functions for IEEE CRC-32 (reversed, non-inverted)

Author

neil

Definition in file [crc32.h](#).

9.17.2 Function Documentation

9.17.2.1 void [crc32_init](#) (uint32_t * *crc*)

Initialize a LUT and the CRC initial value.

Parameters

<i>crc</i>	A pointer to the CRC value to be modified
------------	---

Todo Add the ability to dynamically create the CRC table or place it in ROM

Definition at line 13 of file [crc32.c](#).

9.17.2.2 void [crc32_update](#) (uint32_t * *crc*, uint8_t *data*)

Byte oriented update of the CRC.

Parameters

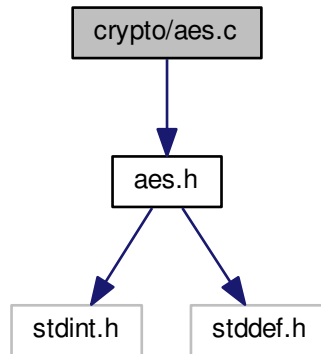
<i>crc</i>	A pointer to the CRC value to be modified
<i>data</i>	The byte to be consumed by the CRC calculation

Definition at line 41 of file [crc32.c](#).

9.18 crypto/aes.c File Reference

```
#include "aes.h"
```

Include dependency graph for aes.c:



Macros

- `#define AES_SMALL_TABLES`
- `#define RCON(i) (rcons[(i)] << 24)`
- `#define TE0(i) Te0[(i) >> 24] & 0xff`
- `#define TE1(i) rotr(Te0[(i) >> 16] & 0xff, 8)`
- `#define TE2(i) rotr(Te0[(i) >> 8] & 0xff, 16)`
- `#define TE3(i) rotr(Te0[(i) & 0xff], 24)`
- `#define TE41(i) ((Te0[(i) >> 24] & 0xff) << 8) & 0xff000000`
- `#define TE42(i) (Te0[(i) >> 16] & 0xff) & 0x00ff0000`
- `#define TE43(i) (Te0[(i) >> 8] & 0xff) & 0x0000ff00`
- `#define TE44(i) ((Te0[(i) & 0xff] >> 8) & 0x000000ff)`
- `#define TE421(i) ((Te0[(i) >> 16] & 0xff) << 8) & 0xff000000`
- `#define TE432(i) (Te0[(i) >> 8] & 0xff) & 0x00ff0000`
- `#define TE443(i) (Te0[(i) & 0xff] & 0x0000ff00)`
- `#define TE414(i) ((Te0[(i) >> 24] & 0xff) >> 8) & 0x000000ff`
- `#define TE4(i) ((Te0[(i)] >> 8) & 0x000000ff)`
- `#define TD0(i) Td0[(i) >> 24] & 0xff`
- `#define TD1(i) rotr(Td0[(i) >> 16] & 0xff, 8)`
- `#define TD2(i) rotr(Td0[(i) >> 8] & 0xff, 16)`
- `#define TD3(i) rotr(Td0[(i) & 0xff], 24)`
- `#define TD41(i) (Td4s[(i) >> 24] & 0xff) << 24`
- `#define TD42(i) (Td4s[(i) >> 16] & 0xff) << 16`
- `#define TD43(i) (Td4s[(i) >> 8] & 0xff) << 8`
- `#define TD44(i) (Td4s[(i) & 0xff])`
- `#define TD0_(i) Td0[(i) & 0xff]`
- `#define TD1_(i) rotr(Td0[(i) & 0xff], 8)`
- `#define TD2_(i) rotr(Td0[(i) & 0xff], 16)`
- `#define TD3_(i) rotr(Td0[(i) & 0xff], 24)`
- `#define SWAP(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)`

- **#define GETU32**(pt)
- **#define PUTU32**(ct, st)
- **#define ROUND**(i, d, s)
- **#define ROUND**(i, d, s)

Functions

- void **rijndaelKeySetupEnc** (uint32_t rk[], const uint8_t cipherKey[])
- void **rijndaelKeySetupDec** (uint32_t rk[], const uint8_t cipherKey[])
- void **rijndaelEncrypt** (const uint32_t rk[], const uint8_t pt[16], uint8_t ct[16])
- void **rijndaelDecrypt** (const uint32_t rk[], const uint8_t ct[16], uint8_t pt[16])
- void * **aes_encrypt_init** (const uint8_t *key, uint32_t *rk, size_t len)
Initialise the keyspace for AES encryption.
- void **aes_encrypt** (void *ctx, const uint8_t *plain, uint8_t *crypt)
Encrypt a block of data.
- void * **aes_decrypt_init** (const uint8_t *key, uint32_t *rk, size_t len)
Initialise the keyspace for AES decryption.
- void **aes_decrypt** (void *ctx, const uint8_t *crypt, uint8_t *plain)
Decrypt a block of data.

9.18.1 Macro Definition Documentation

9.18.1.1 #define GETU32(pt)

Value:

```
((uint32_t)(pt)[0] << 24) ^ ((uint32_t)(pt)[1] << 16) ^ \
((uint32_t)(pt)[2] << 8) ^ ((uint32_t)(pt)[3]))
```

Definition at line 860 of file aes.c.

9.18.1.2 #define PUTU32(ct, st)

Value:

```
{ \
(ct)[0] = (uint8_t)((st) >> 24); (ct)[1] = (uint8_t)((st) >> 16); \
(ct)[2] = (uint8_t)((st) >> 8); (ct)[3] = (uint8_t)(st); }
```

Definition at line 862 of file aes.c.

9.18.1.3 #define ROUND(i, d, s)

Value:

```
d##0 = TE0(s##0) ^ TE1(s##1) ^ TE2(s##2) ^ TE3(s##3) ^ rk[4 * i]; \
d##1 = TE0(s##1) ^ TE1(s##2) ^ TE2(s##3) ^ TE3(s##0) ^ rk[4 * i + 1]; \
d##2 = TE0(s##2) ^ TE1(s##3) ^ TE2(s##0) ^ TE3(s##1) ^ rk[4 * i + 2]; \
d##3 = TE0(s##3) ^ TE1(s##0) ^ TE2(s##1) ^ TE3(s##2) ^ rk[4 * i + 3]
```

9.18.1.4 #define ROUND(i, d, s)

Value:

```
d##0 = TD0(s##0) ^ TD1(s##3) ^ TD2(s##2) ^ TD3(s##1) ^ rk[4 * i]; \
d##1 = TD0(s##1) ^ TD1(s##0) ^ TD2(s##3) ^ TD3(s##2) ^ rk[4 * i + 1]; \
d##2 = TD0(s##2) ^ TD1(s##1) ^ TD2(s##0) ^ TD3(s##3) ^ rk[4 * i + 2]; \
d##3 = TD0(s##3) ^ TD1(s##2) ^ TD2(s##1) ^ TD3(s##0) ^ rk[4 * i + 3]
```

9.18.2 Function Documentation

9.18.2.1 `void rijndaelKeySetupDec ( uint32_t rk[], const uint8_t cipherKey[] )`

Expand the cipher key into the decryption key schedule.

Returns

the number of rounds for the given cipher key size.

Definition at line 898 of file aes.c.

Here is the call graph for this function:



9.18.2.2 `void rijndaelKeySetupEnc ( uint32_t rk[], const uint8_t cipherKey[] )`

Expand the cipher key into the encryption key schedule.

Returns

the number of rounds for the given cipher key size.

Definition at line 872 of file aes.c.

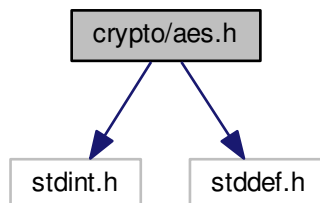
9.19 crypto/aes.h File Reference

C functions for AES (LUT accelerated)

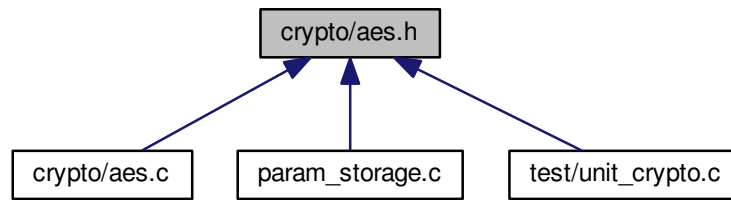
```
#include <stdint.h>
```

```
#include <stddef.h>
```

Include dependency graph for aes.h:



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



Functions

- void * [aes_encrypt_init](#) (const uint8_t *key, uint32_t *rk, size_t len)
Initialise the keyspace for AES encryption.
- void [aes_encrypt](#) (void *ctx, const uint8_t *plain, uint8_t *crypt)
Encrypt a block of data.
- void * [aes_decrypt_init](#) (const uint8_t *key, uint32_t *rk, size_t len)
Initialise the keyspace for AES decryption.
- void [aes_decrypt](#) (void *ctx, const uint8_t *crypt, uint8_t *plain)
Decrypt a block of data.

9.19.1 Detailed Description

C functions for AES (LUT accelerated)

Author

Jouni Malinen jkmaline@cc.hut.fi

Definition in file [aes.h](#).

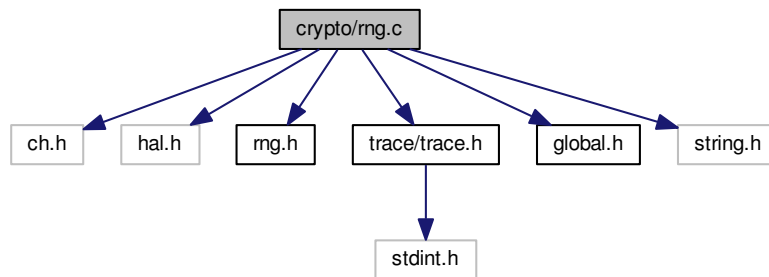
9.20 crypto/rng.c File Reference

```

#include "ch.h"
#include "hal.h"
#include "rng.h"
#include "trace/trace.h"
#include "global.h"
#include <string.h>

```

Include dependency graph for rng.c:



Functions

- uint32_t **gen_random_word** (void)
- void **gen_random_start** (void)
- void **gen_random_stop** (void)
- void **gen_random** (uint8_t *random, int32_t length)
Generate an array of random bytes.
- void **gen_random_i** (uint8_t *random, int32_t length)
Generate an array of random bytes.

9.20.1 Detailed Description

Author

neil

Definition in file [rng.c](#).

9.20.2 Function Documentation

9.20.2.1 void **gen_random** (uint8_t * *random*, int32_t *length*)

Generate an array of random bytes.

Parameters

<i>random</i>	A pointer to an array of random numbers to be used as output
<i>length</i>	The number of bytes to be generated

Definition at line 55 of file rng.c.

9.20.2.2 void **gen_random_i** (uint8_t * *random*, int32_t *length*)

Generate an array of random bytes.

Parameters

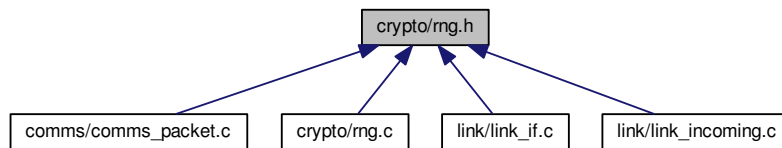
<i>random</i>	A pointer to an array of random numbers to be used as output
<i>length</i>	The number of bytes to be generated

Definition at line 81 of file rng.c.

9.21 crypto/rng.h File Reference

C functions for random number generation.

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



Functions

- void [gen_random](#) (uint8_t *random, int32_t length)
Generate an array of random bytes.
- void [gen_random_i](#) (uint8_t *random, int32_t length)
Generate an array of random bytes.

9.21.1 Detailed Description

C functions for random number generation.

Author

neil

Definition in file [rng.h](#).

9.21.2 Function Documentation

9.21.2.1 void [gen_random](#) (uint8_t * *random*, int32_t *length*)

Generate an array of random bytes.

Parameters

<i>random</i>	A pointer to an array of random numbers to be used as output
<i>length</i>	The number of bytes to be generated

Definition at line 55 of file rng.c.

9.21.2.2 void gen_random_i (uint8_t * *random*, int32_t *length*)

Generate an array of random bytes.

Parameters

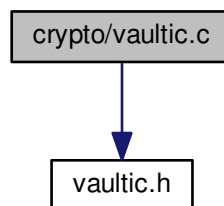
<i>random</i>	A pointer to an array of random numbers to be used as output
<i>length</i>	The number of bytes to be generated

Definition at line 81 of file rng.c.

9.22 crypto/vaultic.c File Reference

```
#include "vaultic.h"
```

Include dependency graph for vaultic.c:



9.22.1 Detailed Description

Author

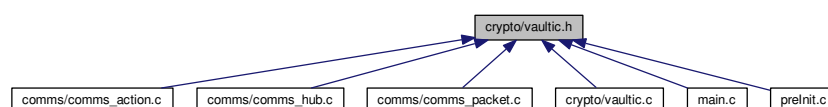
neil

Definition in file [vaultic.c](#).

9.23 crypto/vaultic.h File Reference

A thread used to provide an interface to the VaultIC device and perform the required cryptographic actions.

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



9.23.1 Detailed Description

A thread used to provide an interface to the VaultIC device and perform the required cryptographic actions.

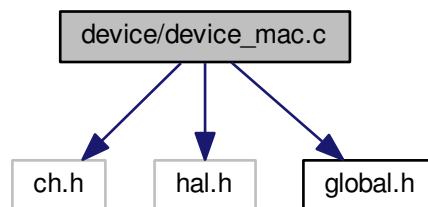
Author

neil

Definition in file [vaultic.h](#).

9.24 device/device_mac.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "global.h"
Include dependency graph for device_mac.c:
```



Functions

- void [set_mac](#) (uint32_t serial, uint8_t *mac)
Generate a MAC address from the UID and the predefined OUI.

9.24.1 Detailed Description

Author

neil

Definition in file [device_mac.c](#).

9.24.2 Function Documentation

9.24.2.1 void `set_mac` (uint32_t *serial*, uint8_t * *mac*)

Generate a MAC address from the UID and the predefined OUI.

Parameters

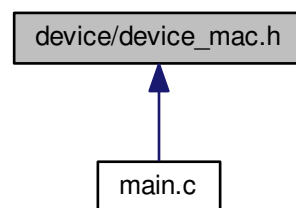
<i>serial</i>	The device serial number
<i>mac</i>	Pointer to the MAC array

Definition at line 14 of file device_mac.c.

9.25 device/device_mac.h File Reference

Function to configure a unique MAC address from a Microsense OUI and the STM32 OUI.

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



Functions

- void [set_mac](#) (uint32_t serial, uint8_t *mac)
Generate a MAC address from the UID and the predefined OUI.

9.25.1 Detailed Description

Function to configure a unique MAC address from a Microsense OUI and the STM32 OUI.

Author

neil

Definition in file [device_mac.h](#).

9.25.2 Function Documentation

9.25.2.1 void set_mac (uint32_t serial, uint8_t * mac)

Generate a MAC address from the UID and the predefined OUI.

Parameters

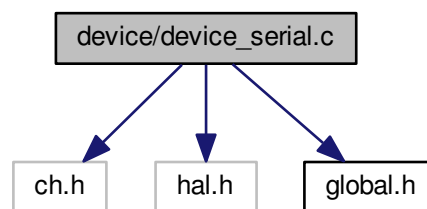
<i>serial</i>	The device serial number
---------------	--------------------------

<i>mac</i>	Pointer to the MAC array
------------	--------------------------

Definition at line 14 of file device_mac.c.

9.26 device/device_serial.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "global.h"
Include dependency graph for device_serial.c:
```



Functions

- void [set_serial](#) (uint32_t *serial)
Generate a serial number from the CRC32 of the UID.

9.26.1 Detailed Description

Author

neil

Definition in file [device_serial.c](#).

9.26.2 Function Documentation

9.26.2.1 void set_serial (uint32_t * serial)

Generate a serial number from the CRC32 of the UID.

Parameters

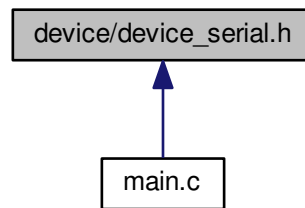
<i>serial</i>	A pointer to the 32-bit serial number
---------------	---------------------------------------

Definition at line 14 of file device_serial.c.

9.27 device/device_serial.h File Reference

Function to generate a unique 32-bit serial number based upon its STM32 UID.

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



Functions

- void [set_serial](#) (uint32_t *serial)

Generate a serial number from the CRC32 of the UID.

9.27.1 Detailed Description

Function to generate a unique 32-bit serial number based upon its STM32 UID.

Author

neil

Definition in file [device_serial.h](#).

9.27.2 Function Documentation

9.27.2.1 void set_serial (uint32_t * serial)

Generate a serial number from the CRC32 of the UID.

Parameters

<i>serial</i>	A pointer to the 32-bit serial number
---------------	---------------------------------------

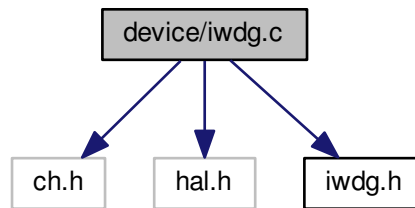
Definition at line 14 of file device_serial.c.

9.28 device/iwdg.c File Reference

IWDG Driver code.

```
#include "ch.h"
#include "hal.h"
#include "iwdg.h"
```

Include dependency graph for `iwdg.c`:



9.28.1 Detailed Description

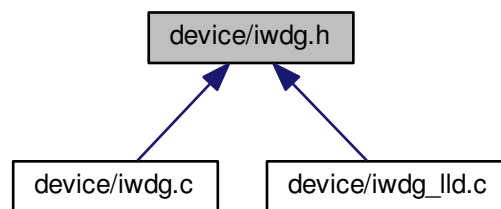
IWDG Driver code.

Definition in file [iwdg.c](#).

9.29 `device/iwdg.h` File Reference

IWDG Driver macros and structures.

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



9.29.1 Detailed Description

IWDG Driver macros and structures.

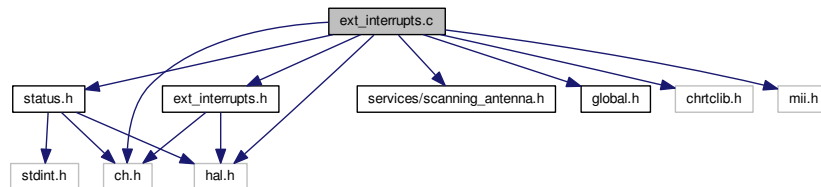
Definition in file [iwdg.h](#).

9.30 `ext_interrupts.c` File Reference

```
#include "ext_interrupts.h"
```

```
#include "services/scanning_antenna.h"
#include "global.h"
#include "status.h"
#include "hal.h"
#include "chrtclib.h"
#include "ch.h"
#include "mii.h"
```

Include dependency graph for ext_interrupts.c:



Functions

- void [gps_pps_irq](#) (EXTDriver *extp, expchannel_t channel)
An interrupt handler for the GPS PPS signal.
- void [accel_irq](#) (EXTDriver *extp, expchannel_t channel)
An interrupt handler for the accelerometer interrupt signal.
- void [start_ext_interrupts](#) (void)
Start the EXT interrupts.
- void [add_ext_interrupt](#) (uint8_t pin, uint32_t mode, extcallback_t cb)
Add a new EXT interrupt to be monitored.

Variables

- Thread * [tp_gps](#)
- EventSource [gps_pps](#)
An EventSource used to pass PPS events to the GPS thread.
- Thread * [accel_tp](#)
A thread pointer used to wakeup the accelerometer thread on an interrupt.
- uint32_t [ssr_rtc](#) = 0
- uint32_t [shift_rtc](#) = 0
- uint32_t [wakeup_rtc](#) = 0
- uint32_t [gps_pps_uptime](#) = 0
A counter that increments each time the GPS PPS interrupt occurs.
- bool [flag_first_gps_pps](#) = false
A flag used to indicate if the GPS PPS is to be used to set off rtc interrupt.

9.30.1 Detailed Description

Author

neil

Definition in file [ext_interrupts.c](#).

9.30.2 Function Documentation

9.30.2.1 void accel_irq (EXTDriver * *extp*, expchannel_t *channel*)

An interrupt handler for the accelerometer interrupt signal.

Parameters

<i>extp</i>	A pointer to the EXT driver
<i>channel</i>	The channel configuration to be used

Definition at line 120 of file ext_interrupts.c.

9.30.2.2 void add_ext_interrupt (uint8_t *pin*, uint32_t *mode*, extcallback_t *cb*)

Add a new EXT interrupt to be monitored.

Parameters

<i>pin</i>	The pin on which the external interrupt is present
<i>mode</i>	The mode of the interrupt (e.g. negative edge event)
<i>cb</i>	The callback function used to handle the external interrupt event

Definition at line 148 of file ext_interrupts.c.

9.30.2.3 void gps_pps_irq (EXTDriver * *extp*, expchannel_t *channel*)

An interrupt handler for the GPS PPS signal.

Parameters

<i>extp</i>	A pointer to the EXT driver
<i>channel</i>	The channel configuration to be used

Definition at line 65 of file ext_interrupts.c.

9.30.3 Variable Documentation

9.30.3.1 bool flag_first_gps_pps = false

A flag used to indicate if the GPS PPS is to be used to set off rtc interrupt.

A flag used to indicate if the GPS PPS is firing for the second time.

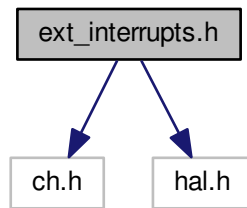
Definition at line 30 of file ext_interrupts.c.

9.31 ext_interrupts.h File Reference

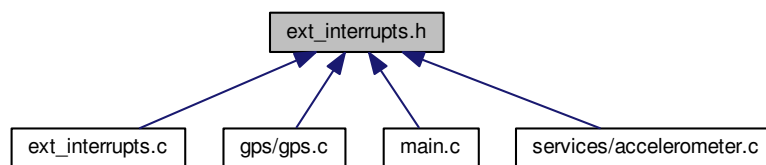
External interrupts used by the MS100 system.

```
#include "ch.h"
#include "hal.h"
```

Include dependency graph for ext_interrupts.h:



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



Functions

- void [gps_pps_irq](#) (EXTDriver *extp, expchannel_t channel)
An interrupt handler for the GPS PPS signal.
- void [accel_irq](#) (EXTDriver *extp, expchannel_t channel)
An interrupt handler for the accelerometer interrupt signal.
- void [start_ext_interrupts](#) (void)
Start the EXT interrupts.
- void [add_ext_interrupt](#) (uint8_t pin, uint32_t mode, extcallback_t cb)
Add a new EXT interrupt to be monitored.

Variables

- EventSource [gps_pps](#)
An EventSource used to pass PPS events to the GPS thread.
- Thread * [accel_tp](#)
A thread pointer used to wakeup the accelerometer thread on an interrupt.
- bool [flag_first_gps_pps](#)
A flag used to indicate if the GPS PPS is firing for the second time.

9.31.1 Detailed Description

External interrupts used by the MS100 system.

Author

neil

Definition in file [ext_interrupts.h](#).

9.31.2 Function Documentation

9.31.2.1 void accel_irq (EXTDriver * *extp*, expchannel_t *channel*)

An interrupt handler for the accelerometer interrupt signal.

Parameters

<i>extp</i>	A pointer to the EXT driver
<i>channel</i>	The channel configuration to be used

Definition at line 120 of file [ext_interrupts.c](#).

9.31.2.2 void add_ext_interrupt (uint8_t *pin*, uint32_t *mode*, extcallback_t *cb*)

Add a new EXT interrupt to be monitored.

Parameters

<i>pin</i>	The pin on which the external interrupt is present
<i>mode</i>	The mode of the interrupt (e.g. negative edge event)
<i>cb</i>	The callback function used to handle the external interrupt event

Definition at line 148 of file [ext_interrupts.c](#).

9.31.2.3 void gps_pps_irq (EXTDriver * *extp*, expchannel_t *channel*)

An interrupt handler for the GPS PPS signal.

Parameters

<i>extp</i>	A pointer to the EXT driver
<i>channel</i>	The channel configuration to be used

Definition at line 65 of file [ext_interrupts.c](#).

9.31.3 Variable Documentation

9.31.3.1 bool flag_first_gps_pps

A flag used to indicate if the GPS PPS is firing for the second time.

A flag used to indicate if the GPS PPS is firing for the second time.

Definition at line 30 of file [ext_interrupts.c](#).

9.32 flash/external_eeprom.c File Reference

9.32.1 Detailed Description

Author

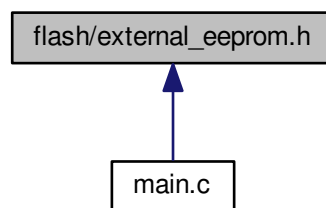
neil
tony

Definition in file [external_eeprom.c](#).

9.33 flash/external_eeprom.h File Reference

Functions to enable the external EEPROM device to be used for storage.

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



9.33.1 Detailed Description

Functions to enable the external EEPROM device to be used for storage.

Author

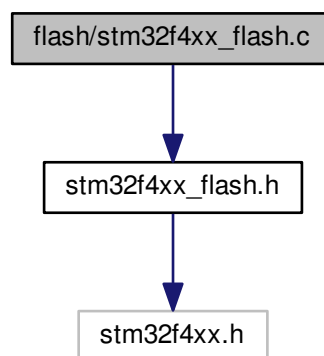
neil

Definition in file [external_eeprom.h](#).**9.34 flash/stm32f4xx_flash.c File Reference**

This file provides firmware functions to manage the following functionalities of the FLASH peripheral:

```
#include "stm32f4xx_flash.h"
```

Include dependency graph for stm32f4xx_flash.c:

**Macros**

- #define **assert_param**(x)
- #define **SECTOR_MASK** ((uint32_t)0xFFFFFFF07)

Functions

- void **FLASH_SetLatency** (uint32_t FLASH_Latency)
Sets the code latency value.
- void **FLASH_PrefetchBufferCmd** (FunctionalState NewState)
Enables or disables the Prefetch Buffer.
- void **FLASH_InstructionCacheCmd** (FunctionalState NewState)
Enables or disables the Instruction Cache feature.
- void **FLASH_DataCacheCmd** (FunctionalState NewState)
Enables or disables the Data Cache feature.
- void **FLASH_InstructionCacheReset** (void)
Resets the Instruction Cache.
- void **FLASH_DataCacheReset** (void)
Resets the Data Cache.
- void **FLASH_Unlock** (void)
Unlocks the FLASH control register access.

- void [FLASH_Lock](#) (void)
Locks the FLASH control register access.
- [FLASH_Status FLASH_EraseSector](#) (uint32_t FLASH_Sector, uint8_t VoltageRange)
Erases a specified FLASH Sector.
- [FLASH_Status FLASH_EraseAllSectors](#) (uint8_t VoltageRange)
Erases all FLASH Sectors.
- [FLASH_Status FLASH_ProgramDoubleWord](#) (uint32_t Address, uint64_t Data)
Programs a double word (64-bit) at a specified address.
- [FLASH_Status FLASH_ProgramWord](#) (uint32_t Address, uint32_t Data)
Programs a word (32-bit) at a specified address.
- [FLASH_Status FLASH_ProgramHalfWord](#) (uint32_t Address, uint16_t Data)
Programs a half word (16-bit) at a specified address.
- [FLASH_Status FLASH_ProgramByte](#) (uint32_t Address, uint8_t Data)
Programs a byte (8-bit) at a specified address.
- void [FLASH_OB_Unlock](#) (void)
Unlocks the FLASH Option Control Registers access.
- void [FLASH_OB_Lock](#) (void)
Locks the FLASH Option Control Registers access.
- void [FLASH_OB_WRPConfig](#) (uint32_t OB_WRP, FunctionalState NewState)
Enables or disables the write protection of the desired sectors.
- void [FLASH_OB_RDPCConfig](#) (uint8_t OB_RDP)
Sets the read protection level.
- void [FLASH_OB_UserConfig](#) (uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)
Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.
- void [FLASH_OB_BORConfig](#) (uint8_t OB_BOR)
Sets the BOR Level.
- [FLASH_Status FLASH_OB_Launch](#) (void)
Launch the option byte loading.
- uint8_t [FLASH_OB_GetUser](#) (void)
Returns the FLASH User Option Bytes values.
- uint16_t [FLASH_OB_GetWRP](#) (void)
Returns the FLASH Write Protection Option Bytes value.
- FlagStatus [FLASH_OB_GetRDP](#) (void)
Returns the FLASH Read Protection level.
- uint8_t [FLASH_OB_GetBOR](#) (void)
Returns the FLASH BOR level.
- void [FLASH_ITConfig](#) (uint32_t FLASH_IT, FunctionalState NewState)
Enables or disables the specified FLASH interrupts.
- FlagStatus [FLASH_GetFlagStatus](#) (uint32_t FLASH_FLAG)
Checks whether the specified FLASH flag is set or not.
- void [FLASH_ClearFlag](#) (uint32_t FLASH_FLAG)
Clears the FLASH's pending flags.
- [FLASH_Status FLASH_GetStatus](#) (void)
Returns the FLASH Status.
- [FLASH_Status FLASH_WaitForLastOperation](#) (void)
Waits for a FLASH operation to complete.
- uint32_t [FLASH_GetSector](#) (uint32_t Address)
Return the Flash sector bitmask of the address.
- uint32_t [FLASH_GetSectorNumber](#) (uint32_t Address)
Return the Flash sector Number of the address.

9.34.1 Detailed Description

This file provides firmware functions to manage the following functionalities of the FLASH peripheral:

Author

MCD Application Team

Version

V1.0.0

Date

30-September-2011

- FLASH Interface configuration
- FLASH Memory Programming
- Option Bytes Programming
- Interrupts and flags management

```

*
*  =====
*                      How to use this driver
*  =====
*
*  This driver provides functions to configure and program the FLASH
*  memory of all STM32F4xx devices.
*  These functions are split in 4 groups:
*
*  1. FLASH Interface configuration functions: this group includes the
*     management of the following features:
*      - Set the latency
*      - Enable/Disable the prefetch buffer
*      - Enable/Disable the Instruction cache and the Data cache
*      - Reset the Instruction cache and the Data cache
*
*  2. FLASH Memory Programming functions: this group includes all needed
*     functions to erase and program the main memory:
*      - Lock and Unlock the FLASH interface
*      - Erase function: Erase sector, erase all sectors
*      - Program functions: byte, half word, word and double word
*
*  3. Option Bytes Programming functions: this group includes all needed
*     functions to manage the Option Bytes:
*      - Set/Reset the write protection
*      - Set the Read protection Level
*      - Set the BOR level
*      - Program the user Option Bytes
*      - Launch the Option Bytes loader
*
*  4. Interrupts and flags management functions: this group
*     includes all needed functions to:
*      - Enable/Disable the FLASH interrupt sources
*      - Get flags status
*      - Clear flags
*      - Get FLASH operation status
*      - Wait for last FLASH operation
*
*

```

Attention

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© COPYRIGHT 2011 STMicroelectronics

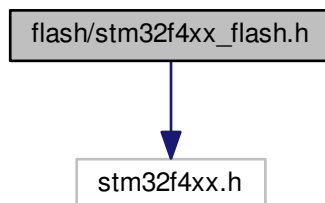
Definition in file [stm32f4xx_flash.c](#).

9.35 flash/stm32f4xx_flash.h File Reference

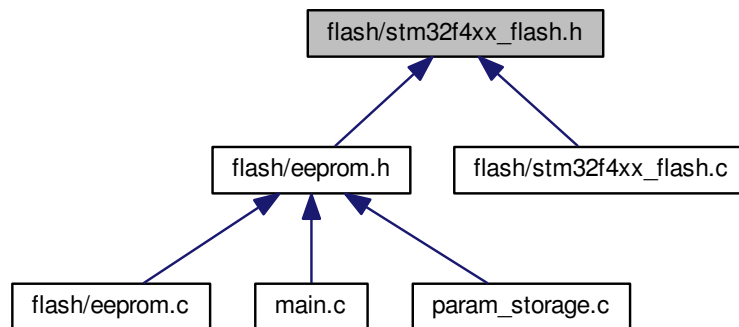
This file contains all the functions prototypes for the FLASH firmware library.

```
#include "stm32f4xx.h"
```

Include dependency graph for stm32f4xx_flash.h:



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



Macros

- `#define FLASH_Latency_0 ((uint8_t)0x0000)`
- `#define FLASH_Latency_1 ((uint8_t)0x0001)`
- `#define FLASH_Latency_2 ((uint8_t)0x0002)`
- `#define FLASH_Latency_3 ((uint8_t)0x0003)`
- `#define FLASH_Latency_4 ((uint8_t)0x0004)`
- `#define FLASH_Latency_5 ((uint8_t)0x0005)`
- `#define FLASH_Latency_6 ((uint8_t)0x0006)`
- `#define FLASH_Latency_7 ((uint8_t)0x0007)`
- `#define IS_FLASH_LATENCY(LATENCY)`
- `#define VoltageRange_1 ((uint8_t)0x00)`
- `#define VoltageRange_2 ((uint8_t)0x01)`
- `#define VoltageRange_3 ((uint8_t)0x02)`
- `#define VoltageRange_4 ((uint8_t)0x03)`
- `#define IS_VOLTAGERANGE(RANGE)`
- `#define FLASH_Sector_0 ((uint16_t)0x0000)`
- `#define FLASH_Sector_1 ((uint16_t)0x0008)`
- `#define FLASH_Sector_2 ((uint16_t)0x0010)`
- `#define FLASH_Sector_3 ((uint16_t)0x0018)`
- `#define FLASH_Sector_4 ((uint16_t)0x0020)`
- `#define FLASH_Sector_5 ((uint16_t)0x0028)`
- `#define FLASH_Sector_6 ((uint16_t)0x0030)`
- `#define FLASH_Sector_7 ((uint16_t)0x0038)`
- `#define FLASH_Sector_8 ((uint16_t)0x0040)`
- `#define FLASH_Sector_9 ((uint16_t)0x0048)`
- `#define FLASH_Sector_10 ((uint16_t)0x0050)`
- `#define FLASH_Sector_11 ((uint16_t)0x0058)`
- `#define IS_FLASH_SECTOR(SECTOR)`
- `#define IS_FLASH_ADDRESS(ADDRESS)`
- `#define ADDR_FLASH_SECTOR_0 ((uint32_t)0x08000000) /* Base @ of Sector 0, 16 Kbyte */`
- `#define ADDR_FLASH_SECTOR_1 ((uint32_t)0x08004000) /* Base @ of Sector 1, 16 Kbyte */`
- `#define ADDR_FLASH_SECTOR_2 ((uint32_t)0x08008000) /* Base @ of Sector 2, 16 Kbyte */`
- `#define ADDR_FLASH_SECTOR_3 ((uint32_t)0x0800C000) /* Base @ of Sector 3, 16 Kbyte */`
- `#define ADDR_FLASH_SECTOR_4 ((uint32_t)0x08010000) /* Base @ of Sector 4, 64 Kbyte */`

- #define **ADDR_FLASH_SECTOR_5** ((uint32_t)0x08020000) /* Base @ of Sector 5, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_6** ((uint32_t)0x08040000) /* Base @ of Sector 6, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_7** ((uint32_t)0x08060000) /* Base @ of Sector 7, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_8** ((uint32_t)0x08080000) /* Base @ of Sector 8, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_9** ((uint32_t)0x080A0000) /* Base @ of Sector 9, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_10** ((uint32_t)0x080C0000) /* Base @ of Sector 10, 128 Kbyte */
- #define **ADDR_FLASH_SECTOR_11** ((uint32_t)0x080E0000) /* Base @ of Sector 11, 128 Kbyte */
- #define **OB_WRP_Sector_0** ((uint32_t)0x00000001)
- #define **OB_WRP_Sector_1** ((uint32_t)0x00000002)
- #define **OB_WRP_Sector_2** ((uint32_t)0x00000004)
- #define **OB_WRP_Sector_3** ((uint32_t)0x00000008)
- #define **OB_WRP_Sector_4** ((uint32_t)0x00000010)
- #define **OB_WRP_Sector_5** ((uint32_t)0x00000020)
- #define **OB_WRP_Sector_6** ((uint32_t)0x00000040)
- #define **OB_WRP_Sector_7** ((uint32_t)0x00000080)
- #define **OB_WRP_Sector_8** ((uint32_t)0x00000100)
- #define **OB_WRP_Sector_9** ((uint32_t)0x00000200)
- #define **OB_WRP_Sector_10** ((uint32_t)0x00000400)
- #define **OB_WRP_Sector_11** ((uint32_t)0x00000800)
- #define **OB_WRP_Sector_All** ((uint32_t)0x00000FFF)
- #define **IS_OB_WRP**(SECTOR) (((SECTOR) & (uint32_t)0xFFFFF000) == 0x00000000) && ((SECTOR) != 0x00000000)
- #define **OB_RDP_Level_0** ((uint8_t)0xAA)
- #define **OB_RDP_Level_1** ((uint8_t)0x55)
- #define **IS_OB_RDP**(LEVEL)
- #define **OB_IWDG_SW** ((uint8_t)0x20)
- #define **OB_IWDG_HW** ((uint8_t)0x00)
- #define **IS_OB_IWDG_SOURCE**(SOURCE) (((SOURCE) == **OB_IWDG_SW**) || ((SOURCE) == **OB_IWDG_←**
G_HW))
- #define **OB_STOP_NoRST** ((uint8_t)0x40)
- #define **OB_STOP_RST** ((uint8_t)0x00)
- #define **IS_OB_STOP_SOURCE**(SOURCE) (((SOURCE) == **OB_STOP_NoRST**) || ((SOURCE) == **OB_S_←**
TOP_RST))
- #define **OB_STDBY_NoRST** ((uint8_t)0x80)
- #define **OB_STDBY_RST** ((uint8_t)0x00)
- #define **IS_OB_STDBY_SOURCE**(SOURCE) (((SOURCE) == **OB_STDBY_NoRST**) || ((SOURCE) == **OB_←**
_STDBY_RST))
- #define **OB_BOR_LEVEL3** ((uint8_t)0x00)
- #define **OB_BOR_LEVEL2** ((uint8_t)0x04)
- #define **OB_BOR_LEVEL1** ((uint8_t)0x08)
- #define **OB_BOR_OFF** ((uint8_t)0x0C)
- #define **IS_OB_BOR**(LEVEL)
- #define **FLASH_IT_EOP** ((uint32_t)0x01000000)
- #define **FLASH_IT_ERR** ((uint32_t)0x02000000)
- #define **IS_FLASH_IT**(IT) (((IT) & (uint32_t)0xFCFFFFFF) == 0x00000000) && ((IT) != 0x00000000)
- #define **FLASH_FLAG_EOP** ((uint32_t)0x00000001)
- #define **FLASH_FLAG_OPERR** ((uint32_t)0x00000002)
- #define **FLASH_FLAG_WRPERR** ((uint32_t)0x00000010)
- #define **FLASH_FLAG_PGERR** ((uint32_t)0x00000020)
- #define **FLASH_FLAG_PGPERR** ((uint32_t)0x00000040)
- #define **FLASH_FLAG_PGSERR** ((uint32_t)0x00000080)
- #define **FLASH_FLAG_BSY** ((uint32_t)0x00010000)
- #define **IS_FLASH_CLEAR_FLAG**(FLAG) (((FLAG) & (uint32_t)0xFFFFF0C) == 0x00000000) && ((FL_←
AG) != 0x00000000)
- #define **IS_FLASH_GET_FLAG**(FLAG)

- #define **FLASH_PSIZE_BYTE** ((uint32_t)0x00000000)
- #define **FLASH_PSIZE_HALF_WORD** ((uint32_t)0x00000100)
- #define **FLASH_PSIZE_WORD** ((uint32_t)0x00000200)
- #define **FLASH_PSIZE_DOUBLE_WORD** ((uint32_t)0x00000300)
- #define **CR_PSIZE_MASK** ((uint32_t)0xFFFFFCFF)
- #define **RDP_KEY** ((uint16_t)0x00A5)
- #define **FLASH_KEY1** ((uint32_t)0x45670123)
- #define **FLASH_KEY2** ((uint32_t)0xCDEF89AB)
- #define **FLASH_OPT_KEY1** ((uint32_t)0x08192A3B)
- #define **FLASH_OPT_KEY2** ((uint32_t)0x4C5D6E7F)
- #define **ACR_BYTE0_ADDRESS** ((uint32_t)0x40023C00)
ACR register byte 0 (Bits[8:0]) base address.
- #define **OPTCR_BYTE0_ADDRESS** ((uint32_t)0x40023C14)
OPTCR register byte 3 (Bits[24:16]) base address.
- #define **OPTCR_BYTE1_ADDRESS** ((uint32_t)0x40023C15)
- #define **OPTCR_BYTE2_ADDRESS** ((uint32_t)0x40023C16)

Enumerations

- enum **FLASH_Status** {
FLASH_BUSY = 1, **FLASH_ERROR_PGS**, **FLASH_ERROR_PGP**, **FLASH_ERROR_PGA**,
FLASH_ERROR_WRP, **FLASH_ERROR_PROGRAM**, **FLASH_ERROR_OPERATION**, **FLASH_COMPLETE** }
FLASH Status.

Functions

- void **FLASH_SetLatency** (uint32_t FLASH_Latency)
Sets the code latency value.
- void **FLASH_PrefetchBufferCmd** (FunctionalState NewState)
Enables or disables the Prefetch Buffer.
- void **FLASH_InstructionCacheCmd** (FunctionalState NewState)
Enables or disables the Instruction Cache feature.
- void **FLASH_DataCacheCmd** (FunctionalState NewState)
Enables or disables the Data Cache feature.
- void **FLASH_InstructionCacheReset** (void)
Resets the Instruction Cache.
- void **FLASH_DataCacheReset** (void)
Resets the Data Cache.
- uint32_t **FLASH_GetSector** (uint32_t Address)
Return the Flash sector bitmask of the address.
- uint32_t **FLASH_GetSectorNumber** (uint32_t Address)
Return the Flash sector Number of the address.
- void **FLASH_Unlock** (void)
Unlocks the FLASH control register access.
- void **FLASH_Lock** (void)
Locks the FLASH control register access.
- **FLASH_Status** **FLASH_EraseSector** (uint32_t FLASH_Sector, uint8_t VoltageRange)
Erases a specified FLASH Sector.
- **FLASH_Status** **FLASH_EraseAllSectors** (uint8_t VoltageRange)
Erases all FLASH Sectors.

- [FLASH_Status FLASH_ProgramDoubleWord](#) (uint32_t Address, uint64_t Data)
Programs a double word (64-bit) at a specified address.
- [FLASH_Status FLASH_ProgramWord](#) (uint32_t Address, uint32_t Data)
Programs a word (32-bit) at a specified address.
- [FLASH_Status FLASH_ProgramHalfWord](#) (uint32_t Address, uint16_t Data)
Programs a half word (16-bit) at a specified address.
- [FLASH_Status FLASH_ProgramByte](#) (uint32_t Address, uint8_t Data)
Programs a byte (8-bit) at a specified address.
- void [FLASH_OB_Unlock](#) (void)
Unlocks the FLASH Option Control Registers access.
- void [FLASH_OB_Lock](#) (void)
Locks the FLASH Option Control Registers access.
- void [FLASH_OB_WRPConfig](#) (uint32_t OB_WRP, FunctionalState NewState)
Enables or disables the write protection of the desired sectors.
- void [FLASH_OB_RDPConfig](#) (uint8_t OB_RDP)
Sets the read protection level.
- void [FLASH_OB_UserConfig](#) (uint8_t OB_IWDG, uint8_t OB_STOP, uint8_t OB_STDBY)
Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.
- void [FLASH_OB_BORConfig](#) (uint8_t OB_BOR)
Sets the BOR Level.
- [FLASH_Status FLASH_OB_Launch](#) (void)
Launch the option byte loading.
- uint8_t [FLASH_OB_GetUser](#) (void)
Returns the FLASH User Option Bytes values.
- uint16_t [FLASH_OB_GetWRP](#) (void)
Returns the FLASH Write Protection Option Bytes value.
- FlagStatus [FLASH_OB_GetRDP](#) (void)
Returns the FLASH Read Protection level.
- uint8_t [FLASH_OB_GetBOR](#) (void)
Returns the FLASH BOR level.
- void [FLASH_ITConfig](#) (uint32_t FLASH_IT, FunctionalState NewState)
Enables or disables the specified FLASH interrupts.
- FlagStatus [FLASH_GetFlagStatus](#) (uint32_t FLASH_FLAG)
Checks whether the specified FLASH flag is set or not.
- void [FLASH_ClearFlag](#) (uint32_t FLASH_FLAG)
Clears the FLASH's pending flags.
- [FLASH_Status FLASH_GetStatus](#) (void)
Returns the FLASH Status.
- [FLASH_Status FLASH_WaitForLastOperation](#) (void)
Waits for a FLASH operation to complete.

9.35.1 Detailed Description

This file contains all the functions prototypes for the FLASH firmware library.

Author

MCD Application Team

Version

V1.0.0

Date

30-September-2011

Attention

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

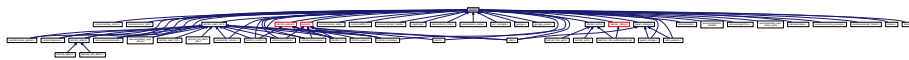
© COPYRIGHT 2011 STMicroelectronics

Definition in file [stm32f4xx_flash.h](#).

9.36 global.h File Reference

A global include file for distribution of common definitions and variables.

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



Data Structures

- [struct _alarms_t](#)
A struct definition used to provide flags for alarms.
- [struct _InitCfg](#)

Macros

- [#define TCPIP_B2B_PORT](#) 27000
The hardcoded port to be used for back-to-back communications over Ethernet.
- [#define DEFAULT_RS485_BAUD](#) 115200
The selected default baud rate for RS485 communications.
- [#define DEFAULT_LINK_BAUD](#) 38400
The selected default baud rate for microwave link communications.
- [#define INSTALLATION_TIMEOUT](#) (15)
- [#define INSTALLATION_TIMEOUT_MINUTE](#) (60000)
A timeout value that is called every minute from startup.

- #define [INSTALLATION_IDLE_TIMEOUT](#) 300000
The period of inactivity within installation mode before it expires.
- #define [NORMAL_IDLE_TIMEOUT](#) 60000
- #define [STM32_UID_ADDR](#) 0x1FFF7A10
ST Microelectronics Unique Identifier address (LSW, 96 bits available)
- #define [EEPROM_READ_OK](#) 0x00000001
The bits associated with each hardware peripheral.
- #define [EEPROM_WRITE_OK](#) 0x00000002
- #define [GPS_STREAM_OK](#) 0x00000004
- #define [ACCELEROMETER_CONFIG_OK](#) 0x00000008
- #define [OPTICAL_STATUS_OK](#) 0x00000010
- #define [SDCARD_MOUNTED_OK](#) 0x00000020
- #define [SDCARD_READ_OK](#) 0x00000040
- #define [SDCARD_WRITE_OK](#) 0x00000080
- #define [RF_DIRECT_PATH_OK](#) 0x00000100
- #define [RF_GAIN_PATH_OK](#) 0x00000200
- #define [UPPER_DEADZONE_PRESENT](#) 0x00000400
- #define [LOWER_DEADZONE_PRESENT](#) 0x00000800
- #define [ADC_0_ENABLED](#) 0x00001000
- #define [ADC_1_ENABLED](#) 0x00002000
- #define [ADC_2_ENABLED](#) 0x00004000
- #define [ADC_3_ENABLED](#) 0x00008000
- #define [TEMPERATURE_CONFIG_OK](#) 0x00010000
- #define [TEMPERATURE_READING_OK](#) 0x00020000
- #define [PARAMETER_LOAD_OK](#) 0x00040000
- #define [FACTORY_RESET_DISABLED](#) 0x00080000
- #define [BOOT_VERSION_ADDR](#) 0x7FC0
The addresses of fixed parameters.
- #define [RECORD_INDEX_ADDR](#) 0x7F80
- #define [PRODUCT_VERSION_ADDR](#) 0x7F40
- #define [TEST_EEPROM_ADDR](#) 0x7F00

Firmware version

The firmware version number (syntactic)

- #define [MAJOR_VERSION](#) 1
Major version - changes on major firmware updates.
- #define [MINOR_VERSION](#) 35
Minor version - changes when features are implemented or changed.
- #define [BUILD_VERSION](#) HALO_BUILD_VERSION
Build version - indicates status of the code base.
- #define [REVISION_VERSION](#) 25
Revision version - changes when bugs are fixed, code is restructured, etc.
- #define [STRINGIZE2\(s\) #s](#)
- #define [STRINGIZE\(s\) STRINGIZE2\(s\)](#)
- #define [RELEASE_STRING](#) "alpha"
- #define [BUILD_STRING](#) STRINGIZE([MAJOR_VERSION](#)) "." STRINGIZE([MINOR_VERSION](#)) "." STRI↵
NGIZE([REVISION_VERSION](#)) "-" RELEASE_STRING

OUI

Todo *This is IOT's OUI, this must be changed...*

The OUI assigned to the device (used with Ethernet MAC etc.)

- #define [MICROSENSE_OUI_0](#) 0xAC

- #define MICROSENSE_OUI_1 0xE9
- #define MICROSENSE_OUI_2 0x7F

MS100-A1 Operating Modes

The operating mode of the device

- #define TX_OPERATING_MODE 0
- #define RX_OPERATING_MODE 1
- #define BI_MASTER_MODE 2
- #define BI_SLAVE_MODE 3
- #define UNDEFINED_OPERATING_MODE 4

Global event id's

These are global events used for inter-thread communication

- #define EVT_SDCARD_INSERTION 0x01
- #define EVT_SDCARD_REMOVAL 0x02
- #define EVT_TX_SWITCH 0x04
- #define EVT_RX_SWITCH 0x08
- #define EVT_DIRECT_PATH_SYNC 0x10
- #define EVT_TX_SWITCH_START 0x20
- #define EVT_RX_SWITCH_START 0x40

EEPROM virtual addresses

The virtual address lookup table values used for the persistent variables

- #define EEPROM_OSDP_ADDR 0
- #define EEPROM_RS485_BAUD_L 1
- #define EEPROM_RS485_BAUD_H 2
- #define EEPROM_OPERATING_MODE 3
- #define EEPROM_RF_INT_FREQ 4
- #define EEPROM_RF_DIFF_FREQ 5
- #define EEPROM_RF_RANGE 6
- #define EEPROM_RF_SENSITIVITY 7
- #define EEPROM_RELAYS 8
- #define EEPROM_ALARMS 9
- #define EEPROM_DEADZONE 10
- #define EEPROM_ADC 11
- #define EEPROM_IPV4_ADDRESS_L 12
- #define EEPROM_IPV4_ADDRESS_H 13
- #define EEPROM_IPV4_NETMASK_L 14
- #define EEPROM_IPV4_NETMASK_H 15
- #define EEPROM_IPV4_GATEWAY_L 16
- #define EEPROM_IPV4_GATEWAY_H 17
- #define EEPROM_IPV4_SERVER_ADDR_L 18
- #define EEPROM_IPV4_SERVER_ADDR_H 19
- #define EEPROM_SERVER_PORT 20
- #define EEPROM_SCBK_0 21
- #define EEPROM_SCBK_1 22
- #define EEPROM_SCBK_2 23
- #define EEPROM_SCBK_3 24
- #define EEPROM_SCBK_4 25
- #define EEPROM_SCBK_5 26
- #define EEPROM_SCBK_6 27
- #define EEPROM_SCBK_7 28

Relay indexing definitions

- #define RELAY_OUT_0 0
- #define RELAY_OUT_1 1

Alarm class definitions

Human readable definitions of the alarm classes used in InitCfg

- #define **ALARM_CLASS_INTRUDER** 0
- #define **ALARM_CLASS_ENVIRONMENTAL** 1
- #define **ALARM_CLASS_ERROR** 2
- #define **ALARM_CLASS_POWER** 3
- #define **ALARM_CLASS_TAMPER** 4
- #define **ALARM_CLASS_EXTERNAL** 5
- #define **ALARM_NUM_CLASSES** 6

Paired device modes

@]

Helper macros that define the paired device modes

- #define **PAIRING_DISABLED** 0
- #define **PAIRING_UNIDIRECTIONAL_ALARMS** 1

Relay helper macros

Helper macros for relay mode and duration

- #define **RELAY_1_MODE**(x) (((x)>>14)&0x03)
- #define **RELAY_1_DUR**(x) (((x)>>6)&0x3F)
- #define **RELAY_2_MODE**(x) (((x)>>12)&0x03)
- #define **RELAY_2_DUR**(x) ((x)&0x3F)

Typedefs

- typedef struct [_alarms_t](#) **alarms_t**
- typedef struct [_InitCfg](#) **InitCfg**

Enumerations

- enum [ms100_state_e](#) {
ST_MS100_IDLE = 0, **ST_MS100_INIT**, **ST_MS100_INIT_WAIT**, **ST_MS100_RECONFIGURE**,
ST_MS100_RECONFIGURE_SETUP, **ST_MS100_INSTALLATION**, **ST_MS100_COMMS_SELECT**, **ST_↵**
_MS100_NORMAL,
ST_MS100_ERROR }

Enumeration for the MS100 Operational state machine.

- enum [relay_mode_e](#) { **RELAY_DISABLED** = 0, **RELAY_ENABLED_NORM_ON**, **RELAY_ENABLED_NO_↵**
RM_OFF, **RELAY_ENABLED** }

Enumeration for relay modes.

Functions

- bool [channel_to_int_frac](#) (void)
A channel to integer/fraction conversion function.
- void [alarms_disable](#) (void)
A function used to temporarily enable/disable alarms.

EEPROM storage

Functions used to store data sets in the EEPROM

- void **write_eeprom_osdp_com** (void)
- void **write_eeprom_osdp_scbk** (void)

- void **write_eeprom_osdp_security_params** (void)
- void **write_eeprom_trace_level** (void)
- void **write_eeprom_osdp_ms100** (void)
- void **write_eeprom_osdp_ipv4** (void)
- void **write_eeprom_record_index** (void)
- void **write_eeprom_paired** (void)

Variables

- [alarms_t g_alarms_osdp](#)
A struct used to provide flags for alarms over OSDP.
- [alarms_t g_alarms_paired](#)
A struct used to provide flags for alarms in the paired transmitter.
- [uint32_t g_alarms_relay](#) [2]
A struct used to provide flags for alarms to the relay.
- [uint32_t g_bi_operating_mode](#)
A flag indicating the bidirectional transceiver mode.
- [uint16_t g_device_alarms](#)
A bit field used to simulate triggering device alarms.
- [uint8_t g_microwave_algorithm](#)
A variable describing the selected microwave intruder algorithm.
- [bool g_upper_deadzone_enable](#)
Test variables used to test out setting of deadzone functionality.
- [uint8_t g_upper_deadzone_sensitivity](#)
- [bool g_lower_deadzone_enable](#)
- [uint8_t g_lower_deadzone_sensitivity](#)
- [ms100_state_e g_ms100_state](#)
The device state.
- [uint32_t g_serial](#)
The device serial number.
- [uint8_t g_mac](#) [6]
The device MAC address.
- [void * g_uwave_samples](#)
An array of microwave samples that are buffered for transmission or manipulation.
- [Mutex g_mtx_dma1_s7](#)
A global mutex to control access to DMAC1 stream 7.
- [Mutex g_mtx_dma1_s0](#)
A global mutex to control access to DMAC1 stream 0.
- [bool g_installation_timeout](#)
A flag to indicate that an installation timeout has occurred and the default key is invalid.
- [uint32_t g_uptime](#)
A system uptime counter (seconds)
- [volatile uint32_t g_osdp_scs_uptime](#)
A counter used to indicate the OSDP SCS running time.
- [InitCfg init_default](#)
Structure that holds default values for the system.
- [InitCfg init_user](#)
Structure that holds user defined settings from persistent storage.
- [uint32_t hardware_status](#)
A bitfield used to store the system hardware status.
- [bool g_accel_interrupt](#)
A flag used to indicate that an accelerometer interrupt has triggered.

- bool [relay_2_enabled](#)
A flag used to indicate if the 2nd relay is available.
- uint32_t [alarm_timeout](#)
A timeout value used to indicate how many seconds the alarms should be disabled.
- uint16_t **THRESHOLD_SIGNAL_LOW**
- uint16_t **THRESHOLD_SIGNAL_HIGH**

Microwave control signals

- int32_t **thresh_high**
- int32_t **thresh_low**
- int32_t [filtered_signal](#)
The current filtered signal obtained from current and past samples.
- uint32_t [dir_path_rssi](#)
The receiver direct path RSSI obtained prior to the transmitter being determined as active.
- int32_t [threshold](#)
The instantaneous threshold in adaptive alarm thresholding [int32].
- bool **thresh_ok**
- bool [tx_active](#)
A flag indicating if a receiver has detected a transmitter signal (always true if a transmitter)
- int32_t [temp_value](#)
The temperature of the board.
- int32_t **temp_status**

Model identification

Identifying numbers for the product

- #define [MODEL_NUMBER\(x\)](#) ((x) & 0xFF)
The model number associated with the build (e.g. 0 == MS100)
- #define [IS_A_HALO\(x\)](#) ([MODEL_NUMBER\(x\)](#) == 0)
Check if a HALO model.
- #define [IS_A_VIGIL\(x\)](#) ([MODEL_NUMBER\(x\)](#) == 1)
Check if a VIGIL model.
- #define [VERSION_NUMBER](#) [g_version_number](#)
The version number associated with the build (e.g. 0 == A1)
- #define [VIGIL_RELAY_2_ENABLED](#) (([g_version_number](#) & 0x01)?true:false)
A boolean flag indicating if the 2nd relay is enabled on a Vigil.
- #define [VIGIL_ETHERNET_ENABLED](#) (([g_version_number](#) & 0x02)?true:false)
A boolean flag indicating if Ethernet is enabled on a Vigil.
- uint8_t [g_version_number](#)
The default version number for the device (this is dynamically updated)
- #define [CTRL_MB_SIZE](#) 5
- Mailbox [mb_ctrl](#)
A mailbox used to send messages to the control FSM.
- msg_t [wa_ctrl](#) [[CTRL_MB_SIZE](#)]
A working area for the control mailbox.

9.36.1 Detailed Description

A global include file for distribution of common definitions and variables.

Author

maria
neil
tony

Definition in file [global.h](#).

9.36.2 Macro Definition Documentation

9.36.2.1 `#define CTRL_MB_SIZE 5`

A mailbox used to send messages to the state machine

Definition at line 264 of file [global.h](#).

9.36.2.2 `#define INSTALLATION_TIMEOUT (15)`

A timeout used to control the number of ms the device initially has to recognise a command to enter installation mode

Definition at line 122 of file [global.h](#).

9.36.2.3 `#define NORMAL_IDLE_TIMEOUT 60000`

The period of inactivity the device must endure before it re-enters a communications selection state

Definition at line 132 of file [global.h](#).

9.36.3 Function Documentation

9.36.3.1 `bool channel_to_int_frac ( void )`

A channel to integer/fraction conversion function.

A channel to integer/fraction conversion function.

Parameters

<i>channel</i>	The channel number (1 to 37 inclusive are valid)
<i>integer</i>	The integer-divider register value
<i>fraction</i>	The fractional-divider register value

Returns

True if successful, false otherwise

Definition at line 397 of file [preInit.c](#).

9.36.4 Variable Documentation

9.36.4.1 `bool g_accel_interrupt`

A flag used to indicate that an accelerometer interrupt has triggered.

A flag used to indicate that an accelerometer interrupt has triggered.

A flag used to indicate that an accelerometer interrupt has triggered

Definition at line 251 of file main.c.

9.36.4.2 `alarms_t g_alarms_osdp`

A struct used to provide flags for alarms over OSDP.

A struct used to provide flags for alarms over OSDP.

Definition at line 157 of file main.c.

9.36.4.3 `alarms_t g_alarms_paired`

A struct used to provide flags for alarms in the paired transmitter.

A struct used to provide flags for alarms in the paired transmitter.

Definition at line 160 of file main.c.

9.36.4.4 `uint32_t g_alarms_relay[2]`

A struct used to provide flags for alarms to the relay.

A struct used to provide flags for alarms to the relay.

Definition at line 163 of file main.c.

9.36.4.5 `uint32_t g_bi_operating_mode`

A flag indicating the bidirectional transceiver mode.

A flag indicating the bidirectional transceiver mode.

Definition at line 154 of file main.c.

9.36.4.6 `uint16_t g_device_alarms`

A bit field used to simulate triggering device alarms.

A bit field used to simulate triggering device alarms.

Definition at line 122 of file microwave.c.

9.36.4.7 `uint8_t g_mac[6]`

The device MAC address.

The device MAC address.

Definition at line 144 of file main.c.

9.36.4.8 `ms100_state_e g_ms100_state`

The device state.

The device state.

Definition at line 166 of file main.c.

9.36.4.9 Mutex g_mtx_dma1_s0

A global mutex to control access to DMAC1 stream 0.

A global mutex to control access to DMAC1 stream 0.

Definition at line 233 of file main.c.

9.36.4.10 Mutex g_mtx_dma1_s7

A global mutex to control access to DMAC1 stream 7.

A global mutex to control access to DMAC1 stream 7.

Definition at line 230 of file main.c.

9.36.4.11 uint32_t g_serial

The device serial number.

The device serial number.

Definition at line 141 of file main.c.

9.36.4.12 uint32_t g_uptime

A system uptime counter (seconds)

A system uptime counter (seconds)

Definition at line 189 of file main.c.

9.36.4.13 void* g_uwave_samples

An array of microwave samples that are buffered for transmission or manipulation.

An array of microwave samples that are buffered for transmission or manipulation.

Definition at line 221 of file main.c.

9.36.4.14 uint32_t hardware_status

A bitfield used to store the system hardware status.

A bitfield used to store the system hardware status.

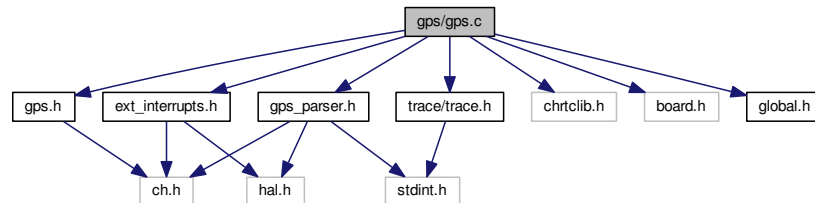
Definition at line 218 of file main.c.

9.37 gps/gps.c File Reference

```
#include "gps.h"
```

```
#include "gps_parser.h"
#include "ext_interrupts.h"
#include "trace/trace.h"
#include "chrtclib.h"
#include "board.h"
#include "global.h"
```

Include dependency graph for gps.c:



Macros

- `#define SPI_DRIVER SPID3`
- `#define SPI_BaudRatePrescaler_2 ((uint16_t)0x0000)`
- `#define SPI_BaudRatePrescaler_4 ((uint16_t)0x0008)`
- `#define SPI_BaudRatePrescaler_8 ((uint16_t)0x0010)`
- `#define SPI_BaudRatePrescaler_16 ((uint16_t)0x0018)`
- `#define SPI_BaudRatePrescaler_32 ((uint16_t)0x0020)`
- `#define SPI_BaudRatePrescaler_64 ((uint16_t)0x0028)`
- `#define SPI_BaudRatePrescaler_128 ((uint16_t)0x0030)`
- `#define SPI_BaudRatePrescaler_256 ((uint16_t)0x0038)`
- `#define GPS_BUFFER_DEPTH 128`

Functions

- **WORKING_AREA** (gps_thread, GPS_THREAD_SIZE)
- void **gps_update_cb** (void *p)
- int **gps_receive_data** (SPIDriver *spip, uint8_t *rx, size_t size)
- int **sirf_add_crc** (uint8_t *data, uint32_t length)
- int **gps_nmea_cmd** (SPIDriver *spip)
- void **gps_reset** (void)
- msg_t **gps_entry** (void *arg)
 - A thread function used parse the incoming GPS NMEA 0183 stream.*
- bool **gps_is_fixed** (void)
 - Determine if there is currently a GPS fix.*
- float **gps_latitude** (void)
 - Obtain the last known latitude.*
- float **gps_longitude** (void)
 - Obtain the last known longitude.*
- float **gps_altitude** (void)
 - Obtain the last known altitude.*
- void **gps_date** (int *day, int *month, int *year)
 - Obtain the last known date.*
- void **gps_utc_time** (int *hour, int *min, int *sec)

Obtain the last known time (UTC)

- uint32_t [gps_num_sentences](#) (void)

Obtain the number of parsed NMEA sentences.

Variables

- Thread * [tp_gps](#) = NULL
- VirtualTimer [gps_t](#)
- Semaphore [g_spimux_sem](#)

A semaphore used to guard access to the SPI MUX.

9.37.1 Detailed Description

Author

neil

Definition in file [gps.c](#).

9.37.2 Function Documentation

9.37.2.1 float [gps_altitude](#) (void)

Obtain the last known altitude.

Returns

Altitude in meters

Definition at line 327 of file [gps.c](#).

9.37.2.2 void [gps_date](#) (int * *day*, int * *month*, int * *year*)

Obtain the last known date.

Parameters

<i>day</i>	Day of the month
<i>month</i>	Month of the year
<i>year</i>	Calendar year

Definition at line 334 of file [gps.c](#).

9.37.2.3 msg_t [gps_entry](#) (void * *arg*)

A thread function used parse the incoming GPS NMEA 0183 stream.

Parameters

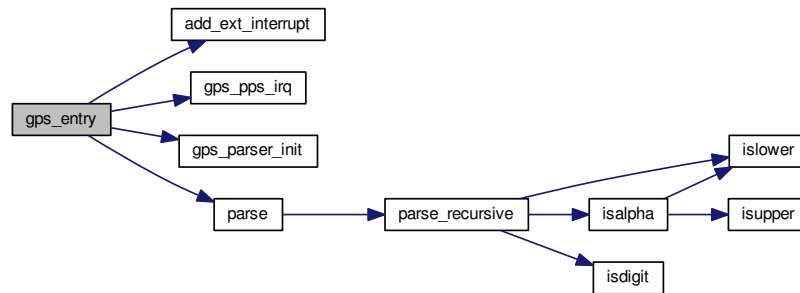
<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 209 of file gps.c.

Here is the call graph for this function:

**9.37.2.4 bool gps_is_fixed (void)**

Determine if there is currently a GPS fix.

Returns

true if fixed, false otherwise

Definition at line 305 of file gps.c.

9.37.2.5 float gps_latitude (void)

Obtain the last known latitude.

Returns

Latitude in decimal degrees

Definition at line 313 of file gps.c.

9.37.2.6 float gps_longitude (void)

Obtain the last known longitude.

Returns

Longitude in decimal degrees

Definition at line 320 of file gps.c.

9.37.2.7 uint32_t gps_num_sentences (void)

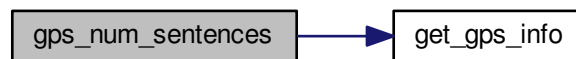
Obtain the number of parsed NMEA sentences.

Returns

Number of sentences that have been processed

Definition at line 352 of file gps.c.

Here is the call graph for this function:



9.37.2.8 void gps_utc_time (int * *hour*, int * *min*, int * *sec*)

Obtain the last known time (UTC)

Parameters

<i>hour</i>	Hour
<i>min</i>	Minute
<i>sec</i>	Second

Definition at line 343 of file gps.c.

9.37.3 Variable Documentation

9.37.3.1 Thread* tp_gps = NULL

Declaration of the static thread used for GPS processing

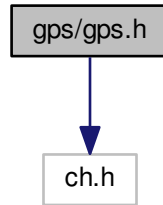
Definition at line 20 of file gps.c.

9.38 gps/gps.h File Reference

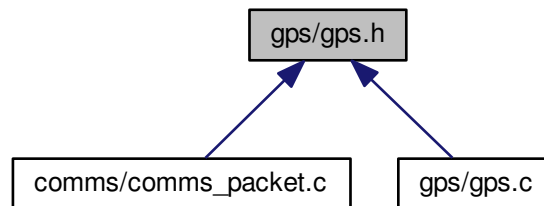
A thread used to provide a GPS location and manage the PPS.

```
#include "ch.h"
```

Include dependency graph for gps.h:



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



Macros

- `#define` [GPS_UPDATE_PERIOD](#) 15000
The delay (ms) between successive GPS time/date updates.

Functions

- void [gps_reset](#) (void)
- msg_t [gps_entry](#) (void *arg)
A thread function used parse the incoming GPS NMEA 0183 stream.
- bool [gps_is_fixed](#) (void)
Determine if there is currently a GPS fix.
- float [gps_latitude](#) (void)
Obtain the last known latitude.
- float [gps_longitude](#) (void)
Obtain the last known longitude.
- float [gps_altitude](#) (void)
Obtain the last known altitude.
- void [gps_date](#) (int *day, int *month, int *year)

Obtain the last known date.

- void [gps_utc_time](#) (int *hour, int *min, int *sec)

Obtain the last known time (UTC)

- uint32_t [gps_num_sentences](#) (void)

Obtain the number of parsed NMEA sentences.

- #define **GPS_THREAD_SIZE** 768
- Thread * [tp_gps](#)
- **WORKING_AREA** (gps_thread, GPS_THREAD_SIZE)

9.38.1 Detailed Description

A thread used to provide a GPS location and manage the PPS.

Author

Neil Smyth

Definition in file [gps.h](#).

9.38.2 Function Documentation

9.38.2.1 float [gps_altitude](#) (void)

Obtain the last known altitude.

Returns

Altitude in meters

Definition at line 327 of file [gps.c](#).

9.38.2.2 void [gps_date](#) (int * *day*, int * *month*, int * *year*)

Obtain the last known date.

Parameters

<i>day</i>	Day of the month
<i>month</i>	Month of the year
<i>year</i>	Calendar year

Definition at line 334 of file [gps.c](#).

9.38.2.3 msg_t [gps_entry](#) (void * *arg*)

A thread function used parse the incoming GPS NMEA 0183 stream.

Parameters

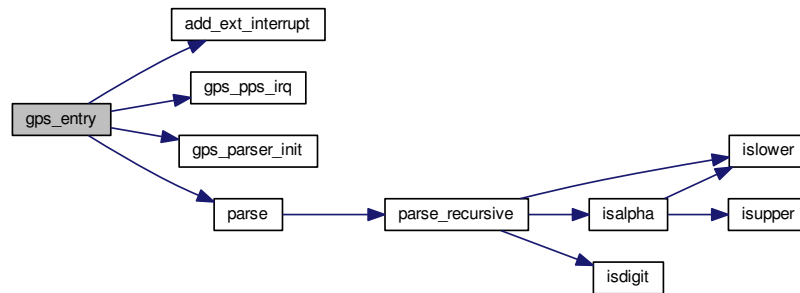
<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 209 of file gps.c.

Here is the call graph for this function:

**9.38.2.4 bool gps_is_fixed (void)**

Determine if there is currently a GPS fix.

Returns

true if fixed, false otherwise

Definition at line 305 of file gps.c.

9.38.2.5 float gps_latitude (void)

Obtain the last known latitude.

Returns

Latitude in decimal degrees

Definition at line 313 of file gps.c.

9.38.2.6 float gps_longitude (void)

Obtain the last known longitude.

Returns

Longitude in decimal degrees

Definition at line 320 of file gps.c.

9.38.2.7 uint32_t gps_num_sentences (void)

Obtain the number of parsed NMEA sentences.

Returns

Number of sentences that have been processed

Definition at line 352 of file gps.c.

Here is the call graph for this function:

**9.38.2.8 void gps_utc_time (int * *hour*, int * *min*, int * *sec*)**

Obtain the last known time (UTC)

Parameters

<i>hour</i>	Hour
<i>min</i>	Minute
<i>sec</i>	Second

Definition at line 343 of file gps.c.

9.38.3 Variable Documentation**9.38.3.1 Thread* tp_gps**

Declaration of the static thread used for GPS processing

Definition at line 20 of file gps.c.

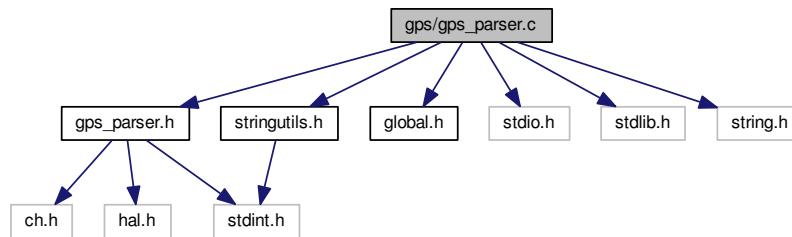
9.39 gps/gps_parser.c File Reference

```

#include "gps_parser.h"
#include "stringutils.h"
#include "global.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

Include dependency graph for `gps_parser.c`:



Macros

- `#define ADDRESS_FIELD_MAX_LENGTH 10`
- `#define NMEA_SEQUENCE_MAX_LENGTH 81`

Functions

- void `gps_parser_init` (void)
Initialization function to reset variables to starting values.
- `int32_t axtoi` (const char *hex_string)
- void `parse` (const char *buf, const uint32_t buf_size)
The principal parsing function, consumes input ASCII text.
- `gps_info_t * get_gps_info` (void)
Function to return a pointer to the decoded GPS data.
- void `parse_recursive` (const char ch)
Private functions.
- void `parse_nmea_sentence` (const char *address_field, const char *buf, const uint32_t buf_size)
- bool `process_gpgga` (const char *buf, const uint32_t buf_size)
- bool `process_gpgsa` (const char *buf, const uint32_t buf_size)
- bool `process_gpgsv` (const char *buf, const uint32_t buf_size)
- bool `process_gprmb` (const char *buf, const uint32_t buf_size)
- bool `process_gprmc` (const char *buf, const uint32_t buf_size)
- bool `process_gpzda` (const char *buf, const uint32_t buf_size)
- bool `process_gpgll` (const char *buf, const uint32_t buf_size)

Variables

- `gps_info_t g_gps_info`

9.39.1 Detailed Description

Author

neil

Definition in file `gps_parser.c`.

9.39.2 Function Documentation

9.39.2.1 `gps_info_t*` `get_gps_info ( void )`

Function to return a pointer to the decoded GPS data.

Returns

Pointer to the GPS data

Definition at line 247 of file `gps_parser.c`.

9.39.2.2 `void` `parse ( const char * buf, const uint32_t buf_size )`

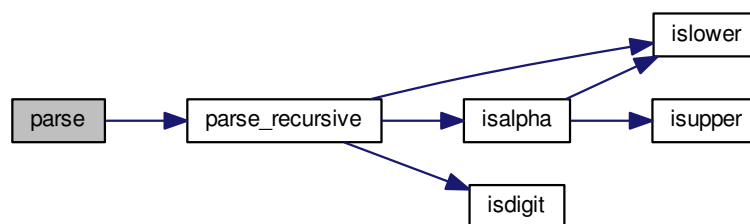
The principal parsing function, consumes input ASCII text.

Parameters

<i>buf</i>	Input buffer pointer for NMEA sentences
<i>buf_size</i>	Number of input bytes to consume

Definition at line 82 of file `gps_parser.c`.

Here is the call graph for this function:



9.40 `gps/gps_parser.h` File Reference

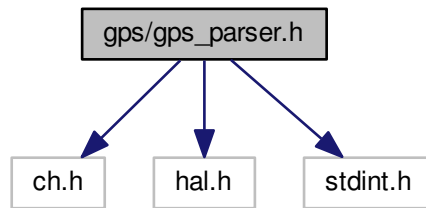
A NMEA sentence parser with guarded access to decoded data using a mutex.

```

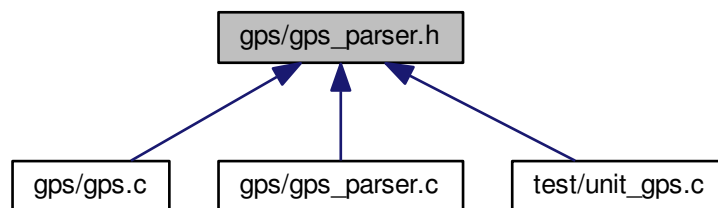
#include "ch.h"
#include "hal.h"
#include <stdint.h>

```

Include dependency graph for gps_parser.h:



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



Data Structures

- struct [_nmea_utc_t](#)
- struct [_gps_info_t](#)

Typedefs

- typedef struct [_nmea_utc_t](#) [nmea_utc_t](#)
- typedef struct [_gps_info_t](#) [gps_info_t](#)

Functions

- void [gps_parser_init](#) (void)
Initialization function to reset variables to starting values.
- void [parse](#) (const char *buf, const uint32_t buf_size)
The principal parsing function, consumes input ASCII text.
- [gps_info_t](#) * [get_gps_info](#) (void)
Function to return a pointer to the decoded GPS data.

9.40.1 Detailed Description

A NMEA sentence parser with guarded access to decoded data using a mutex.

Author

Neil Smyth

Definition in file [gps_parser.h](#).

9.40.2 Function Documentation

9.40.2.1 `gps_info_t*` `get_gps_info ( void )`

Function to return a pointer to the decoded GPS data.

Returns

Pointer to the GPS data

Definition at line 247 of file `gps_parser.c`.

9.40.2.2 `void` `parse ( const char * buf, const uint32_t buf_size )`

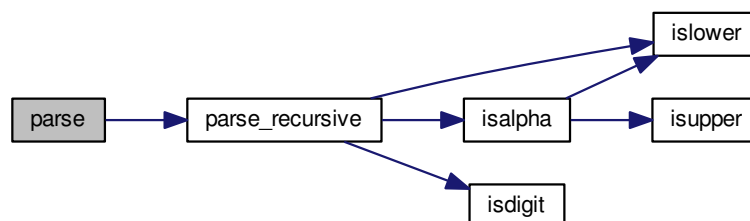
The principal parsing function, consumes input ASCII text.

Parameters

<i>buf</i>	Input buffer pointer for NMEA sentences
<i>buf_size</i>	Number of input bytes to consume

Definition at line 82 of file `gps_parser.c`.

Here is the call graph for this function:



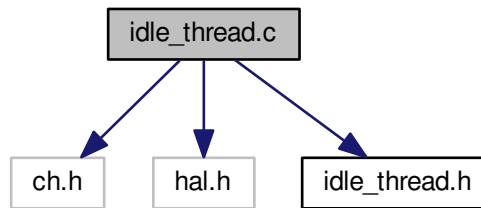
9.41 idle_thread.c File Reference

```

#include <ch.h>
#include <hal.h>
#include "idle_thread.h"

```

Include dependency graph for idle_thread.c:



Functions

- **WORKING_AREA** (`idle_thread`, `IDLE_THREAD_SIZE`)
- `msg_t idle_entry` (`void *arg`)

Variables

- `uint32_t g_idle_time` = 0

9.41.1 Detailed Description

Author

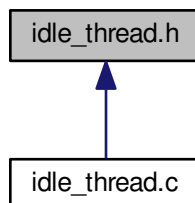
neil

Definition in file [idle_thread.c](#).

9.42 idle_thread.h File Reference

A custom idle thread used to estimate CPU usage.

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



9.42.1 Detailed Description

A custom idle thread used to estimate CPU usage.

Author

neil

Definition in file [idle_thread.h](#).

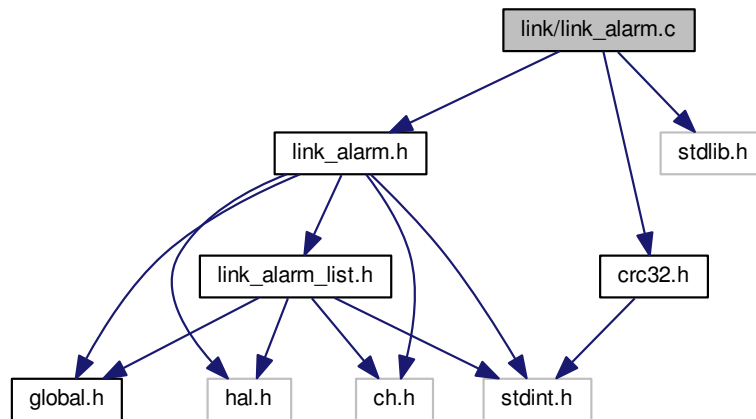
9.43 link/link_alarm.c File Reference

```
#include "link_alarm.h"
```

```
#include "crc32.h"
```

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

Include dependency graph for link_alarm.c:



Functions

- `uint16_t manc_encode_byte (uint8_t byte)`
- `uint8_t manc_decode_byte (uint16_t halfword)`
- `int32_t link_alarm_create (uint8_t *packet)`
Create a blank link alarm packet.
- `int32_t link_alarm_finalize (uint8_t *packet, int32_t length)`
Perform the relevant security and error detection processing.
- `int32_t link_alarm_encode (uint8_t *data, int32_t *data_length)`
Encode the packet, appending any additional data defined by g_link_data.
- `int32_t link_alarm_check_hdr (uint8_t *data, int32_t length)`
Verify if the header is valid.
- `int32_t link_alarm_ready (uint8_t *data, int32_t length)`
Compute the size of the link packet and determine if it is all present.
- `int32_t link_alarm_decode (uint8_t *data, int32_t length)`
Decode the packet, retrieving information regarding the specified id.

9.43.1 Detailed Description

Author

neil

Definition in file [link_alarm.c](#).

9.43.2 Function Documentation

9.43.2.1 `int32_t link_alarm_check_hdr ( uint8_t * data, int32_t length )`

Verify if the header is valid.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

0 on success

Definition at line 138 of file link_alarm.c.

9.43.2.2 `int32_t link_alarm_create ( uint8_t * packet )`

Create a blank link alarm packet.

Parameters

<i>packet</i>	A pointer to the packet to be created
---------------	---------------------------------------

Returns

Returns length of the packet

Definition at line 40 of file link_alarm.c.

9.43.2.3 `int32_t link_alarm_decode ( uint8_t * data, int32_t length )`

Decode the packet, retrieving information regarding the specified id.

Parameters

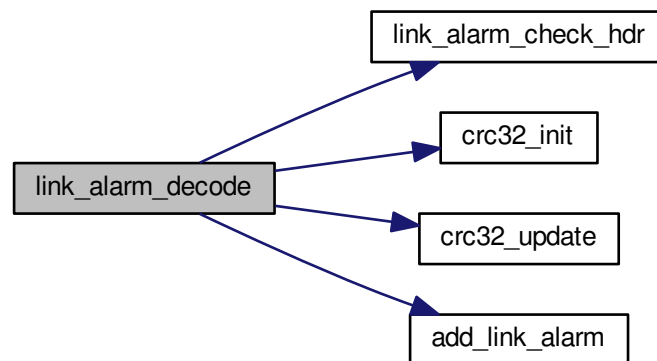
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 172 of file link_alarm.c.

Here is the call graph for this function:



9.43.2.4 `int32_t link_alarm_encode ( uint8_t* data, int32_t* data_length )`

Encode the packet, appending any additional data defined by `g_link_data`.

Parameters

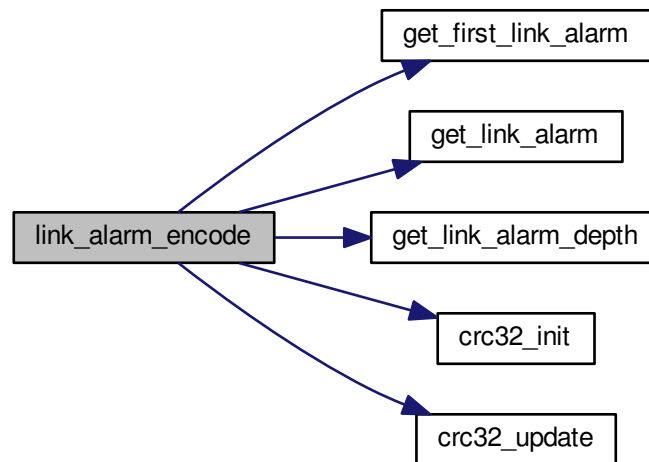
<i>data</i>	The current packet payload
<i>data_length</i>	The length of the packet payload

Returns

0 on success

Definition at line 81 of file link_alarm.c.

Here is the call graph for this function:



9.43.2.5 `int32_t link_alarm_finalize ( uint8_t * packet, int32_t length )`

Perform the relevant security and error detection processing.

Parameters

<i>packet</i>	A pointer to the packet to be created
<i>length</i>	The length of the packet

Returns

Returns length of the packet

Definition at line 72 of file `link_alarm.c`.

9.43.2.6 `int32_t link_alarm_ready ( uint8_t * data, int32_t length )`

Compute the size of the link packet and determine if it is all present.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

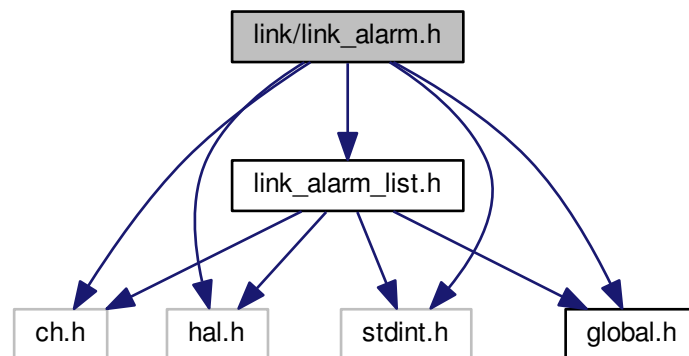
size of the link packet

Definition at line 152 of file `link_alarm.c`.

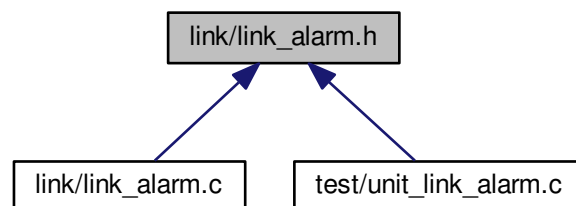
9.44 link/link_alarm.h File Reference

Encode/decode microwave link alarm packets for transmission between MS100 devices.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
#include "global.h"
#include "link_alarm_list.h"
Include dependency graph for link_alarm.h:
```



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



Macros

- `#define LINK_ALARM_VERSION 0`
Version of the link packet.
- `#define LINK_START_DEPTH 5`
The number of start bytes in the header.
- `#define LINK_ALARM_HDR_DEPTH 10`
The total size of the base header (without further extension fields)

Functions

- `int32_t link_alarm_create (uint8_t *packet)`
Create a blank link alarm packet.
- `int32_t link_alarm_finalize (uint8_t *packet, int32_t length)`
Perform the relevant security and error detection processing.
- `int32_t link_alarm_encode (uint8_t *data, int32_t *data_length)`
Encode the packet, appending any additional data defined by `g_link_data`.
- `int32_t link_alarm_check_hdr (uint8_t *data, int32_t length)`
Verify if the header is valid.
- `int32_t link_alarm_ready (uint8_t *data, int32_t length)`
Compute the size of the link packet and determine if it is all present.
- `int32_t link_alarm_decode (uint8_t *data, int32_t length)`
Decode the packet, retrieving information regarding the specified id.

9.44.1 Detailed Description

Encode/decode microwave link alarm packets for transmission between MS100 devices.

Author

neil

Definition in file [link_alarm.h](#).

9.44.2 Function Documentation

9.44.2.1 `int32_t link_alarm_check_hdr ( uint8_t * data, int32_t length )`

Verify if the header is valid.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

0 on success

Definition at line 138 of file [link_alarm.c](#).

9.44.2.2 `int32_t link_alarm_create ( uint8_t * packet )`

Create a blank link alarm packet.

Parameters

<i>packet</i>	A pointer to the packet to be created
---------------	---------------------------------------

Returns

Returns length of the packet

Definition at line 40 of file [link_alarm.c](#).

9.44.2.3 `int32_t link_alarm_decode ( uint8_t * data, int32_t length )`

Decode the packet, retrieving information regarding the specified id.

Parameters

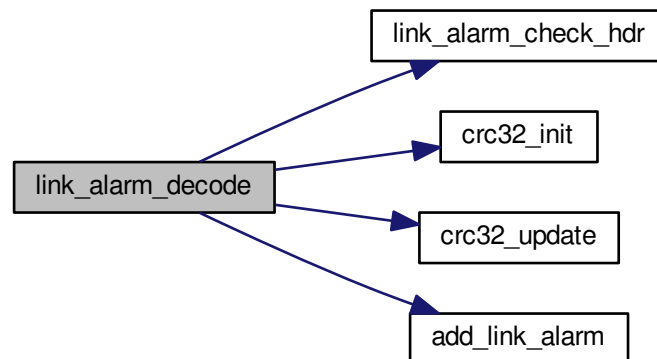
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 172 of file link_alarm.c.

Here is the call graph for this function:



9.44.2.4 int32_t link_alarm_encode (uint8_t * data, int32_t * data_length)

Encode the packet, appending any additional data defined by `g_link_data`.

Parameters

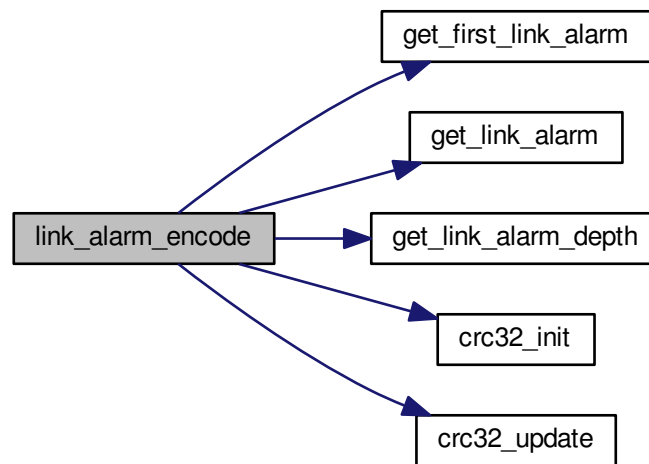
<i>data</i>	The current packet payload
<i>data_length</i>	The length of the packet payload

Returns

0 on success

Definition at line 81 of file link_alarm.c.

Here is the call graph for this function:



9.44.2.5 `int32_t link_alarm_finalize ( uint8_t * packet, int32_t length )`

Perform the relevant security and error detection processing.

Parameters

<i>packet</i>	A pointer to the packet to be created
<i>length</i>	The length of the packet

Returns

Returns length of the packet

Definition at line 72 of file link_alarm.c.

9.44.2.6 `int32_t link_alarm_ready ( uint8_t * data, int32_t length )`

Compute the size of the link packet and determine if it is all present.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

size of the link packet

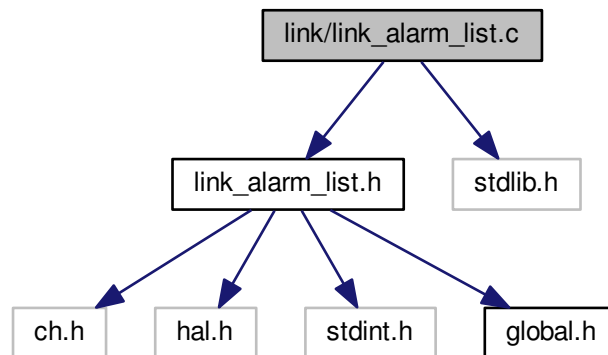
Definition at line 152 of file link_alarm.c.

9.45 link/link_alarm_list.c File Reference

```
#include "link_alarm_list.h"
```

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

Include dependency graph for link_alarm_list.c:

**Functions**

- void [init_link_alarms](#) (void)
Initialize the alarm list.
- int32_t [add_link_alarm](#) (uint8_t id, [alarm_types_e](#) code, uint16_t t_off)
Add a link alarm to the list, discarding the oldest alarm if necessary.
- int32_t [update_link_alarms](#) (void)
Update the alarm list, decrementing the time offset by 1.
- [alarm_item_t](#) * [get_first_link_alarm](#) (void)
Return the first item in the list.
- [alarm_item_t](#) * [get_link_alarm](#) ([alarm_item_t](#) *ptr)
Return the next item in the list pointed to by the current item.
- int32_t [get_link_alarm_depth](#) (void)
Return the depth of the linked list.

9.45.1 Detailed Description**Author**

neil

Definition in file [link_alarm_list.c](#).

9.45.2 Function Documentation

9.45.2.1 `int32_t add_link_alarm ( uint8_t id, alarm_types_e code, uint16_t t_off )`

Add a link alarm to the list, discarding the oldest alarm if necessary.

Returns

0 on success, failure otherwise

Definition at line 89 of file `link_alarm_list.c`.

9.45.2.2 `int32_t update_link_alarms ( void )`

Update the alarm list, decrementing the time offset by 1.

Returns

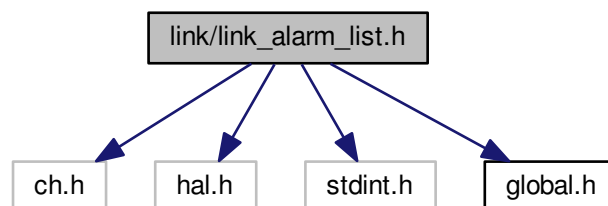
0 on success, failure otherwise

Definition at line 232 of file `link_alarm_list.c`.

9.46 `link/link_alarm_list.h` File Reference

A doubly-linked list used to store link alarms.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
#include "global.h"
Include dependency graph for link_alarm_list.h:
```



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



Data Structures

- struct [_alarm_item_t](#)

Macros

- #define **ALARM_LINK_SECONDS** 3
- #define **MAX_LINK_ALARM_ITEM_DEPTH** 32

Typedefs

- typedef struct [_alarm_item_t](#) **alarm_item_t**

Enumerations

- enum [alarm_types_e](#) {
ALARM_LINK_POWER = 0, **ALARM_LINK_TAMPER**, **ALARM_LINK_MICROWAVE**, **ALARM_LINK_U↵**
PPER_DEADZONE,
ALARM_LINK_LOWER_DEADZONE, **ALARM_LINK_ENVIRONMENTAL**, **ALARM_LINK_ERROR**, **ALA↵**
RM_LINK_AUX_0,
ALARM_LINK_AUX_1, **ALARM_LINK_AUX_2**, **ALARM_LINK_AUX_3**, **ALARM_LINK_MICROWAVE↵**
AUTH,
ALARM_LINK_B2B_LINK_DISCONNECT, **ALARM_LINK_NONE** = 255 }

Functions

- void [init_link_alarms](#) (void)
Initialize the alarm list.
- int32_t [add_link_alarm](#) (uint8_t id, [alarm_types_e](#) code, uint16_t t_off)
Add a link alarm to the list, discarding the oldest alarm if necessary.
- int32_t [update_link_alarms](#) (void)
Update the alarm list, decrementing the time offset by 1.
- [alarm_item_t](#) * [get_first_link_alarm](#) (void)
Return the first item in the list.
- [alarm_item_t](#) * [get_link_alarm](#) ([alarm_item_t](#) *ptr)
Return the next item in the list pointed to by the current item.
- int32_t [get_link_alarm_depth](#) (void)
Return the depth of the linked list.

9.46.1 Detailed Description

A doubly-linked list used to store link alarms.

Author

neil

Definition in file [link_alarm_list.h](#).

9.46.2 Enumeration Type Documentation

9.46.2.1 enum alarm_types_e

Alarm codes for all recognised alarm types

Definition at line 30 of file link_alarm_list.h.

9.46.3 Function Documentation

9.46.3.1 int32_t add_link_alarm (uint8_t id, alarm_types_e code, uint16_t t_off)

Add a link alarm to the list, discarding the oldest alarm if necessary.

Returns

0 on success, failure otherwise

Definition at line 89 of file link_alarm_list.c.

9.46.3.2 int32_t update_link_alarms (void)

Update the alarm list, decrementing the time offset by 1.

Returns

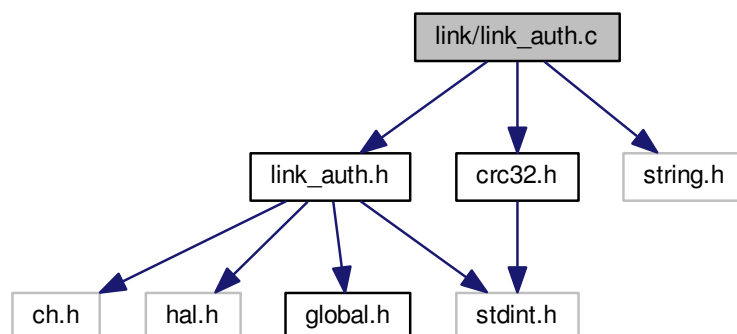
0 on success, failure otherwise

Definition at line 232 of file link_alarm_list.c.

9.47 link/link_auth.c File Reference

```
#include "link_auth.h"
#include "crc32.h"
#include <string.h>
```

Include dependency graph for link_auth.c:



Functions

- void [set_link_payload](#) (int32_t size)
Set the maximum packet payload (bytes)
- int32_t [link_auth_get_num_packets](#) ()
Return the number of packets that must be encoded.
- int32_t [link_auth_encode](#) (uint8_t *data, int32_t *length)
Encode the packet.
- int32_t [link_auth_decode](#) (uint8_t *data, int32_t length)
Decode the packet, retrieving information regarding the specified id.

Variables

- [link_auth_t g_link_auth](#)
Definition of the link authentication struct used to pass link auth about.
- int32_t [g_link_auth_packet_size](#) = 64

9.47.1 Detailed Description

Author

neil

Definition in file [link_auth.c](#).

9.47.2 Function Documentation

9.47.2.1 int32_t link_auth_decode (uint8_t * data, int32_t length)

Decode the packet, retrieving information regarding the specified id.

Parameters

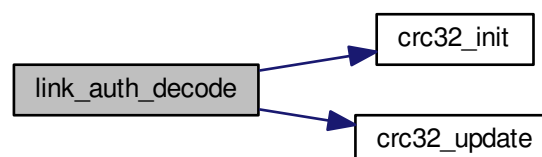
<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 81 of file link_auth.c.

Here is the call graph for this function:



9.47.2.2 `int32_t link_auth_encode ( uint8_t * data, int32_t * length )`

Encode the packet.

Parameters

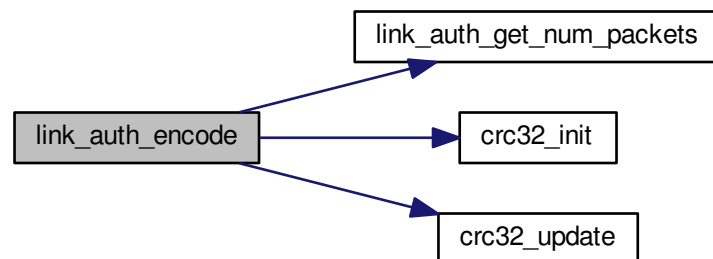
<i>id</i>	The unique sequence number of the packet
<i>data</i>	The current packet payload
<i>length</i>	The length of the packet payload

Returns

0 on success

Definition at line 35 of file `link_auth.c`.

Here is the call graph for this function:



9.47.3 Variable Documentation

9.47.3.1 `link_auth_t g_link_auth`

Definition of the link authentication struct used to pass link auth about.

An extern declaration of the link data struct which is defined in [link_packet.c](#).

Definition at line 10 of file `link_auth.c`.

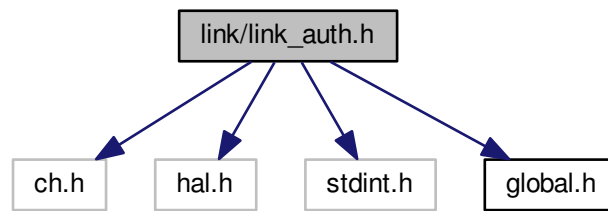
9.48 `link/link_auth.h` File Reference

Encode/decode microwave link authentication messages.

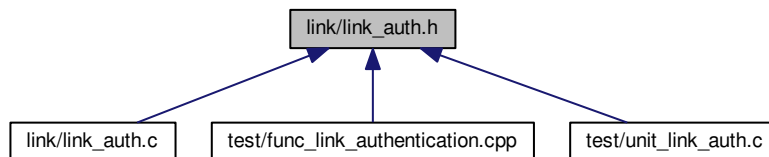
```

#include "ch.h"
#include "hal.h"
#include <stdint.h>
#include "global.h"
  
```

Include dependency graph for link_auth.h:



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



Data Structures

- struct [_link_auth_t](#)
A struct used to store the id and alarm information for a specific device.

Macros

- #define [LINK_AUTH_VERSION](#) 0
Version number of the link authentication packet.
- #define [MAX_LINK_AUTH_SIZE](#) 1024
The maximum permitted size of a link authentication packet.

Typedefs

- typedef struct [_link_auth_t](#) [link_auth_t](#)
A type declaration of the id and alarm information.

Functions

- void [set_link_payload](#) (int32_t size)
Set the maximum packet payload (bytes)
- int32_t [link_auth_get_num_packets](#) ()

Return the number of packets that must be encoded.

- `int32_t link_auth_encode (uint8_t *data, int32_t *length)`

Encode the packet.

- `int32_t link_auth_decode (uint8_t *data, int32_t length)`

Decode the packet, retrieving information regarding the specified id.

Variables

- `int32_t g_link_auth_packet_size`
- `link_auth_t g_link_auth`

An extern declaration of the link data struct which is defined in [link_packet.c](#).

9.48.1 Detailed Description

Encode/decode microwave link authentication messages.

Author

neil

Definition in file [link_auth.h](#).

9.48.2 Function Documentation

9.48.2.1 `int32_t link_auth_decode ( uint8_t * data, int32_t length )`

Decode the packet, retrieving information regarding the specified id.

Parameters

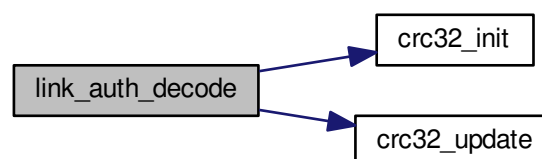
<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 81 of file [link_auth.c](#).

Here is the call graph for this function:



9.48.2.2 `int32_t link_auth_encode ( uint8_t * data, int32_t * length )`

Encode the packet.

Parameters

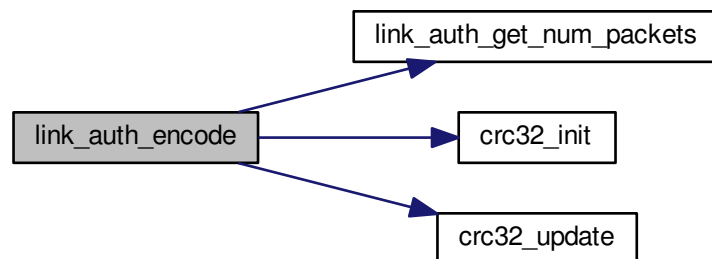
<i>id</i>	The unique sequence number of the packet
<i>data</i>	The current packet payload
<i>length</i>	The length of the packet payload

Returns

0 on success

Definition at line 35 of file `link_auth.c`.

Here is the call graph for this function:



9.48.3 Variable Documentation

9.48.3.1 `link_auth_t g_link_auth`

An extern declaration of the link data struct which is defined in [link_packet.c](#).

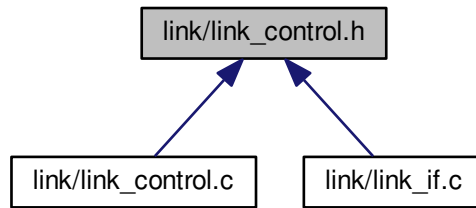
An extern declaration of the link data struct which is defined in [link_packet.c](#).

Definition at line 10 of file `link_auth.c`.

9.49 `link/link_control.h` File Reference

Microwave link control.

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



9.49.1 Detailed Description

Microwave link control.

Author

neil

Definition in file [link_control.h](#).

9.50 link/link_if.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "link_if.h"
#include "link_packet.h"
#include "link_alarm_list.h"
#include "halo_alarms.h"
#include "stringutils.h"
#include "osdp_header.h"
#include "preInit.h"
#include "crypto/sha256.h"
#include "crypto/rng.h"
#include "comms/comms_hub.h"
#include "comms/comms_relay.h"
#include "comms/packet_linked_list.h"
#include "trace/trace.h"
#include "services/microwave.h"
#include "global.h"
#include <string.h>
#include "osdp_master_beacon_base.h"
#include "osdp_halo_sync_base.h"
#include "osdp_halo_alarm_base.h"
#include "link_control.h"
#include "link_incoming.h"
```

Include dependency graph for `link_if.c`:



9.50.1 Detailed Description

Author

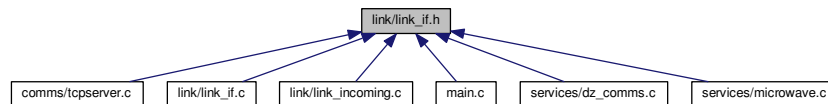
neil

Definition in file [link_if.c](#).

9.51 link/link_if.h File Reference

A C library for microwave link half-duplex communication.

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



9.51.1 Detailed Description

A C library for microwave link half-duplex communication.

Author

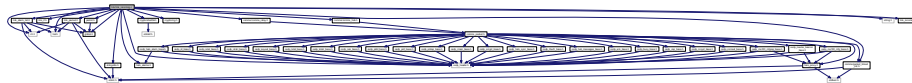
neil

Definition in file [link_if.h](#).

9.52 link/link_incoming.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "link_if.h"
#include "link_packet.h"
#include "link_alarm_list.h"
#include "halo_alarms.h"
#include "stringutils.h"
#include "osdp_header.h"
#include "preInit.h"
#include "crypto/sha256.h"
#include "crypto/rng.h"
#include "comms/comms_hub.h"
#include "comms/comms_relay.h"
#include "comms/packet_linked_list.h"
#include "global.h"
#include <string.h>
#include "link_incoming.h"
```

Include dependency graph for link_incoming.c:



9.52.1 Detailed Description

Author

neil

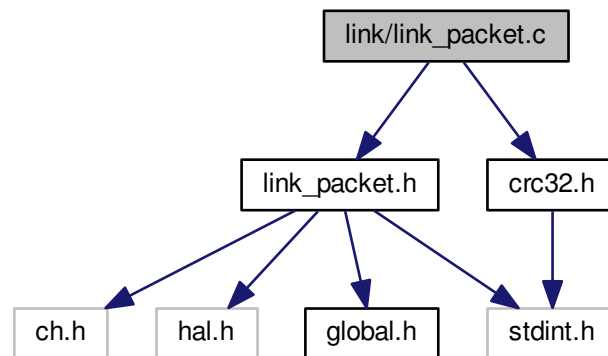
Definition in file [link_incoming.c](#).

9.53 link/link_packet.c File Reference

```
#include "link_packet.h"
```

```
#include "crc32.h"
```

Include dependency graph for link_packet.c:



Functions

- `int32_t link_packet_create` (uint8_t *packet, [link_packet_e](#) type)
Create a blank link alarm packet.
- void **calc_indices** (int32_t id, int32_t *word_index, int32_t *bit_offset, int32_t *byte_index, int32_t *bit_index)
- `int32_t link_packet_check_som` (uint8_t data)
Verify if the SOM is present.
- `int32_t link_packet_check_hdr` (uint8_t *data, int32_t length)
Verify if the header is valid.
- `int32_t link_packet_ready` (uint8_t *data, int32_t length)
Compute the size of the link packet and determine if it is all present.
- void **add_addr_to_hdr** (int32_t word_index, int32_t bit_offset, uint8_t *hdr, int32_t *hdr_length)
- uint32_t **append_crc** (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)
- uint32_t **crc_verify** (uint8_t *data, int32_t length)
- `int32_t check_addr_in_hdr` (int32_t id, uint8_t *data, int32_t length, int32_t *found)
- `int32_t link_packet_alarm_encode` (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)
Encode an alarm packet, appending any additional data defined by g_link_data.
- `int32_t link_packet_alarm_decode` (uint8_t id, uint8_t *data, int32_t length)
Decode an alarm packet, retrieving information regarding the specified id.
- `int32_t link_packet_sync_encode` (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)
Encode a synchronization packet, appending any additional data defined by g_link_data.
- `int32_t link_packet_sync_decode` (uint8_t id, uint8_t *data, int32_t length)
Decode a synchronization packet, retrieving information regarding the specified id.

Variables

- [link_packet_data_t g_link_data](#)

Definition of the link data struct used to pass link data about.

9.53.1 Detailed Description

Author

neil

Definition in file [link_packet.c](#).

9.53.2 Function Documentation

9.53.2.1 `int32_t link_packet_alarm_decode ( uint8_t id, uint8_t * data, int32_t length )`

Decode an alarm packet, retrieving information regarding the specified id.

Parameters

<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 266 of file [link_packet.c](#).

Here is the call graph for this function:



9.53.2.2 `int32_t link_packet_alarm_encode ( uint8_t * hdr, int32_t * hdr_length, uint8_t * data, int32_t * data_length )`

Encode an alarm packet, appending any additional data defined by [g_link_data](#).

Parameters

<i>hdr</i>	The current packet header
<i>hdr_length</i>	The length of the packet hader
<i>data</i>	The current packet payload

<i>data_length</i>	The length of the packet payload
--------------------	----------------------------------

Returns

0 on success

Todo Create a global temperature variable

Definition at line 191 of file link_packet.c.

9.53.2.3 int32_t link_packet_check_hdr (uint8_t * *data*, int32_t *length*)

Verify if the header is valid.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

0 on success

Definition at line 77 of file link_packet.c.

9.53.2.4 int32_t link_packet_check_som (uint8_t *data*)

Verify if the SOM is present.

Parameters

<i>data</i>	The possible SOM byte
-------------	-----------------------

Returns

0 on success

Definition at line 65 of file link_packet.c.

9.53.2.5 int32_t link_packet_create (uint8_t * *packet*, link_packet_e *type*)

Create a blank link alarm packet.

Parameters

<i>packet</i>	A pointer to the packet to be created
<i>type</i>	The type of link packet

Returns

Returns length of the packet

Definition at line 14 of file link_packet.c.

9.53.2.6 int32_t link_packet_ready (uint8_t * *data*, int32_t *length*)

Compute the size of the link packet and determine if it is all present.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

size of the link packet

Definition at line 92 of file link_packet.c.

9.53.2.7 int32_t link_packet_sync_decode (uint8_t *id*, uint8_t * *data*, int32_t *length*)

Decode a synchronization packet, retrieving information regarding the specified id.

Parameters

<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 368 of file link_packet.c.

Here is the call graph for this function:



9.53.2.8 int32_t link_packet_sync_encode (uint8_t * *hdr*, int32_t * *hdr_length*, uint8_t * *data*, int32_t * *data_length*)

Encode a synchronization packet, appending any additional data defined by `g_link_data`.

Parameters

<i>hdr</i>	The current packet header
<i>hdr_length</i>	The length of the packet hader
<i>data</i>	The current packet payload
<i>data_length</i>	The length of the packet payload

Returns

0 on success

Definition at line 331 of file link_packet.c.

9.53.3 Variable Documentation

9.53.3.1 `link_packet_data_t g_link_data`

Definition of the link data struct used to pass link data about.

An extern declaration of the link data struct which is defined in [link_packet.c](#).

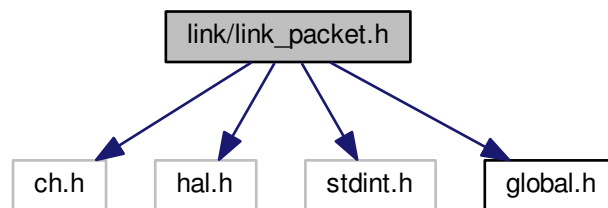
Definition at line 9 of file link_packet.c.

9.54 `link/link_packet.h` File Reference

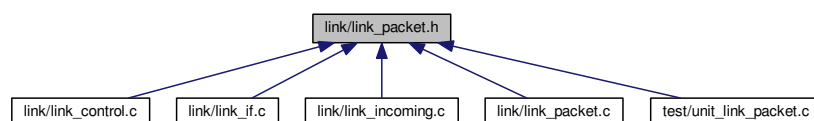
Encode/decode microwave link alarm packets for transmission between MS100 devices.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
#include "global.h"
```

Include dependency graph for link_packet.h:



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



Data Structures

- struct [_link_packet_data_t](#)

A struct used store the id and alarm information for a specific device.

Macros

- `#define LINK_PACKET_VERSION 1`

Version of the link packet.

- `#define LINK_START_DEPTH 5`
The number of start bytes in the header.
- `#define LINK_HDR_DEPTH 9`
The total size of the base header (without further extension fields)

Typedefs

- `typedef struct _link_packet_data_t link_packet_data_t`
A type declaration of the id and alarm information.

Enumerations

- `enum link_packet_e {`
`LINK_PKT_ALARM = 0, LINK_PKT_SYNC, LINK_PKT_AUTH_MASTER_0, LINK_PKT_AUTH_CLIENT_1,`
`LINK_PKT_AUTH_MASTER_2, LINK_PKT_AUTH_CLIENT_3 }`
An enumerated type describing the packet type.

Functions

- `int32_t link_packet_create (uint8_t *packet, link_packet_e type)`
Create a blank link alarm packet.
- `int32_t link_packet_check_som (uint8_t data)`
Verify if the SOM is present.
- `int32_t link_packet_check_hdr (uint8_t *data, int32_t length)`
Verify if the header is valid.
- `int32_t link_packet_ready (uint8_t *data, int32_t length)`
Compute the size of the link packet and determine if it is all present.
- `int32_t link_packet_alarm_encode (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)`
Encode an alarm packet, appending any additional data defined by g_link_data.
- `int32_t link_packet_alarm_decode (uint8_t id, uint8_t *data, int32_t length)`
Decode an alarm packet, retrieving information regarding the specified id.
- `int32_t link_packet_sync_encode (uint8_t *hdr, int32_t *hdr_length, uint8_t *data, int32_t *data_length)`
Encode a synchronization packet, appending any additional data defined by g_link_data.
- `int32_t link_packet_sync_decode (uint8_t id, uint8_t *data, int32_t length)`
Decode a synchronization packet, retrieving information regarding the specified id.

Variables

- `link_packet_data_t g_link_data`
An extern declaration of the link data struct which is defined in [link_packet.c](#).

9.54.1 Detailed Description

Encode/decode microwave link alarm packets for transmission between MS100 devices.

Author

neil

Definition in file [link_packet.h](#).

9.54.2 Function Documentation

9.54.2.1 `int32_t link_packet_alarm_decode ( uint8_t id, uint8_t * data, int32_t length )`

Decode an alarm packet, retrieving information regarding the specified id.

Parameters

<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 266 of file link_packet.c.

Here is the call graph for this function:



9.54.2.2 `int32_t link_packet_alarm_encode ( uint8_t * hdr, int32_t * hdr_length, uint8_t * data, int32_t * data_length )`

Encode an alarm packet, appending any additional data defined by `g_link_data`.

Parameters

<i>hdr</i>	The current packet header
<i>hdr_length</i>	The length of the packet header
<i>data</i>	The current packet payload
<i>data_length</i>	The length of the packet payload

Returns

0 on success

Todo Create a global temperature variable

Definition at line 191 of file link_packet.c.

9.54.2.3 `int32_t link_packet_check_hdr ( uint8_t * data, int32_t length )`

Verify if the header is valid.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

0 on success

Definition at line 77 of file link_packet.c.

9.54.2.4 int32_t link_packet_check_som (uint8_t *data*)

Verify if the SOM is present.

Parameters

<i>data</i>	The possible SOM byte
-------------	-----------------------

Returns

0 on success

Definition at line 65 of file link_packet.c.

9.54.2.5 int32_t link_packet_create (uint8_t * *packet*, link_packet_e *type*)

Create a blank link alarm packet.

Parameters

<i>packet</i>	A pointer to the packet to be created
<i>type</i>	The type of link packet

Returns

Returns length of the packet

Definition at line 14 of file link_packet.c.

9.54.2.6 int32_t link_packet_ready (uint8_t * *data*, int32_t *length*)

Compute the size of the link packet and determine if it is all present.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

size of the link packet

Definition at line 92 of file link_packet.c.

9.54.2.7 int32_t link_packet_sync_decode (uint8_t *id*, uint8_t * *data*, int32_t *length*)

Decode a synchronization packet, retrieving information regarding the specified id.

Parameters

<i>id</i>	The id of the node to be extracted
<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

1 if an alarm relevant to the specified id is present, 0 otherwise

Definition at line 368 of file link_packet.c.

Here is the call graph for this function:



9.54.2.8 `int32_t link_packet_sync_encode ( uint8_t * hdr, int32_t * hdr_length, uint8_t * data, int32_t * data_length )`

Encode a synchronization packet, appending any additional data defined by `g_link_data`.

Parameters

<i>hdr</i>	The current packet header
<i>hdr_length</i>	The length of the packet hader
<i>data</i>	The current packet payload
<i>data_length</i>	The length of the packet payload

Returns

0 on success

Definition at line 331 of file link_packet.c.

9.54.3 Variable Documentation**9.54.3.1** `link_packet_data_t g_link_data`

An extern declaration of the link data struct which is defined in [link_packet.c](#).

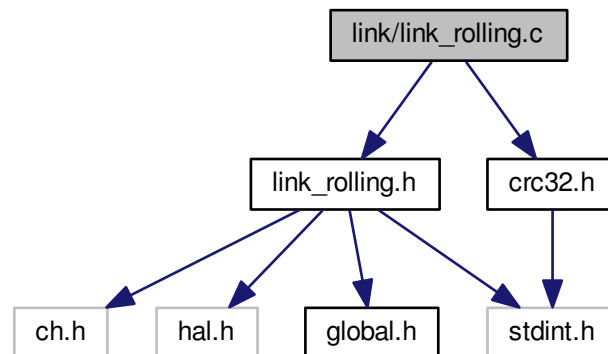
An extern declaration of the link data struct which is defined in [link_packet.c](#).

Definition at line 9 of file link_packet.c.

9.55 link/link_rolling.c File Reference

```
#include "link_rolling.h"
#include "crc32.h"
```

Include dependency graph for link_rolling.c:



Functions

- `int32_t link_rolling_encode (uint8_t *data, int32_t *length)`
Encode the packet.
- `int32_t link_rolling_decode (uint8_t *data, int32_t length)`
Decode the packet.

Variables

- `link_rolling_t g_link_rolling`
Definition of the link rolling code struct used to transport rolling codes.

9.55.1 Detailed Description

Author

neil

Definition in file [link_rolling.c](#).

9.55.2 Function Documentation

9.55.2.1 `int32_t link_rolling_decode ( uint8_t * data, int32_t length )`

Decode the packet.

Parameters

<code>data</code>	The array of bytes to be read
-------------------	-------------------------------

<i>length</i>	The maximum number of bytes to be read
---------------	--

Returns

0 on success

Definition at line 57 of file link_rolling.c.

9.55.2.2 `int32_t link_rolling_encode ( uint8_t * data, int32_t * length )`

Encode the packet.

Parameters

<i>data</i>	The current packet payload
<i>length</i>	The length of the packet payload

Returns

0 on success

Definition at line 14 of file link_rolling.c.

9.55.3 Variable Documentation

9.55.3.1 `link_rolling_t g_link_rolling`

Definition of the link rolling code struct used to transport rolling codes.

An extern declaration of the link rolling code struct which is defined in [link_rolling.c](#).

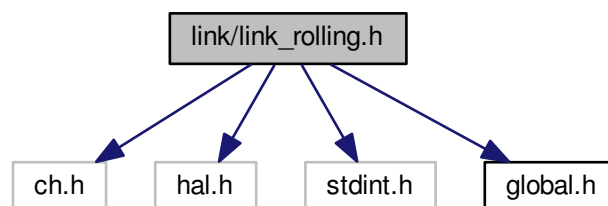
Definition at line 9 of file link_rolling.c.

9.56 link/link_rolling.h File Reference

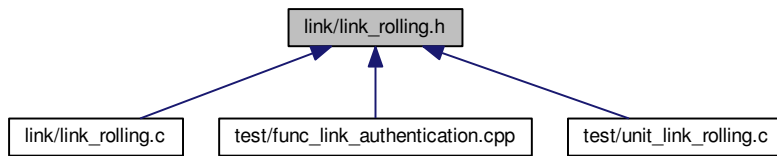
Encode/decode microwave link rolling codes.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
#include "global.h"
```

Include dependency graph for link_rolling.h:



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



Data Structures

- [struct `\_link\_rolling\_t`](#)
A struct used to store the rolling code data.

Macros

- `#define LINK_ROLLING_VERSION 0`

Typedefs

- `typedef int32_t(* enc_cb )(uint8_t *, int32_t *)`
- `typedef int32_t(* dec_cb )(uint8_t *, int32_t)`
- `typedef struct \_link\_rolling\_t link\_rolling\_t`
A type declaration of the rolling code struct.

Functions

- `int32_t link\_rolling\_encode (uint8_t *data, int32_t *length)`
Encode the packet.
- `int32_t link\_rolling\_decode (uint8_t *data, int32_t length)`
Decode the packet.

Variables

- [link_rolling_t g_link_rolling](#)
An extern declaration of the link rolling code struct which is defined in [link_rolling.c](#).

9.56.1 Detailed Description

Encode/decode microwave link rolling codes.

Author

neil

Definition in file [link_rolling.h](#).

9.56.2 Function Documentation

9.56.2.1 `int32_t link_rolling_decode ( uint8_t * data, int32_t length )`

Decode the packet.

Parameters

<i>data</i>	The array of bytes to be read
<i>length</i>	The maximum number of bytes to be read

Returns

0 on success

Definition at line 57 of file link_rolling.c.

9.56.2.2 int32_t link_rolling_encode (uint8_t * *data*, int32_t * *length*)

Encode the packet.

Parameters

<i>data</i>	The current packet payload
<i>length</i>	The length of the packet payload

Returns

0 on success

Definition at line 14 of file link_rolling.c.

9.56.3 Variable Documentation

9.56.3.1 link_rolling_t g_link_rolling

An extern declaration of the link rolling code struct which is defined in [link_rolling.c](#).

An extern declaration of the link rolling code struct which is defined in [link_rolling.c](#).

Definition at line 9 of file link_rolling.c.

9.57 main.c File Reference

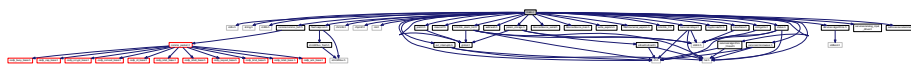
The main process and executable.

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "ch.h"
#include "hal.h"
#include "chrtclib.h"
#include "chprintf.h"
#include "mii.h"
#include "param_storage.h"
#include "ext_interrupts.h"
#include "preInit.h"
#include "maxconf.h"
#include "comms/comms_hub.h"
#include "web/web.h"
#include "flash/eeprom.h"
#include "device/device_serial.h"
#include "device/device_mac.h"
#include "write_eeprom.h"
#include "flash/external_eeprom.h"
#include "link/link_if.h"
#include "link/link_alarm_list.h"
#include "crypto/vaultic.h"
#include "trace/trace.h"
#include "stringutils.h"
#include "status.h"
#include "services/algorithms.h"
#include "services/relays.h"
#include "services/microwave.h"
#include "services/algorithm_thread.h"
#include "global.h"
#include "services/analog_input_driver.h"
#include "services/accelerometer.h"

```

Include dependency graph for main.c:



Macros

- `#define SAMPLE_BUFFER_DEPTH 4096`
The size of the CCM memory used to buffer captured microwave samples.
- `#define DIP_SWITCH_PROGRAM_SEL1 1`
The external DIP switch settings.
- `#define DIP_SWITCH_PROGRAM_SEL0 2`
- `#define DIP_SWITCH_RESERVED 4`
- `#define DIP_SWITCH_RESET 8`

Functions

- `uint16_t uwave_buffer[SAMPLE_BUFFER_DEPTH] __attribute__((section(".ccm")))`
A pointer to the area of CCM RAM used for storing microwave samples.
- `void prvGetRegistersFromStack (uint32_t *pulFaultStackAddress)`
- `void HardFaultVector (void)`
- `void rtc_irq (EXTDriver *extp, expchannel_t channel)`

An interrupt handler for RTC interrupts.

- void **pids_rtc_setup** (void)
- void **installation_cb** (void *p)
- void **alarms_timeout_cb** (void *p)
- void **alarms_disable** (void)

A function used to temporarily enable/disable alarms.

- void **check_reset_condition** (void)
- void **print_startup_greeting** (void)
- void **start_sdcard_and_events** (void)
- void **init_parameters** (uint32_t dip)
- void **init_version_number** (void)
- void **start_relays** (void)
- void **start_real_time_clock** (void)
- void **start_pids_threads** (void)
- void **power_alarm_check** (void)
- void **sample_buffer_init** (void)
- void **temperature_update** (void)
- void **log_gps_info** (void)
- void **log_date_time** (void)
- void **log_direct_path** (void)
- void **log_sample_rate** (void)
- int **main** (void)

Variables

- const uint16_t **VirtAddVarTab** [NB_OF_VAR]

This table is used to store the virtual address of the persistent parameters.

- EventSource **es_record_event**

The event source for a microwave sample record event.

- uint32_t **g_serial**

The serial number of this device.

- uint8_t **g_mac** [6] = { MICROSENSE_OUI_0, MICROSENSE_OUI_1, MICROSENSE_OUI_2, 00, 00, 00 }

The MAC address of this device.

- uint32_t **g_bi_operating_mode** = UNDEFINED_OPERATING_MODE

A flag used to indicate the instantaneous operating mode.

- **alarms_t g_alarms_osdp**

Alarms intended for OSDP polling responses.

- **alarms_t g_alarms_paired**

Alarms for a paired device (obfuscated)

- uint32_t **g_alarms_relay** [2]

Alarms intended for the two relays.

- **ms100_state_e g_ms100_state**

The current control FSM state.

- bool **g_installation_timeout**

A flag to indicate that an installation timeout has occurred and the default key is invalid.

- struct EventListener el0 **el1**

Event listeners for SD card insertion/removal.

- Mailbox **mb_ctrl**

A mailbox used to send messages to the control FSM.

- msg_t **wa_ctrl** [CTRL_MB_SIZE]

A working area for the control mailbox.

- uint32_t **alarm_timeout** = 0

- A timeout value used to indicate how many seconds the alarms should be disabled.*
- uint32_t [g_uptime](#)

A counter used to indicate the application running time.
- uint32_t [rtc_uptime_uwave](#)

A counter that increments each time the RTC wakeup triggers.
- time_t [g_timestamp_start](#)

The system start time.
- time_t [g_timestamp](#)

A periodically updated timestamp value from the RTC.
- volatile uint32_t [g_osdp_scs_uptime](#)

A counter used to indicate the OSDP SCS running time.
- uint32_t [g_count_temp](#)

A counter used to indicate when the temperature should be updated.
- uint32_t [g_count_stat](#) [4]

A counter used to indicate when statistics should be logged.
- int32_t [temp_value](#)

The temperature of the board.
- uint32_t [hardware_status](#)

The hardware status bit field.
- void * [g_uwave_samples](#)

A global pointer to the microwave samples stored in [l_uwave_samples](#).
- volatile [osdp_sample_buffer_t](#) [l_uwave_samples](#)

A struct used to store microwave samples for display.
- Mutex [g_mtx_dma1_s7](#)

A mutex used to guard access to DMAC1 stream 7.
- Mutex [g_mtx_dma1_s0](#)

A mutex used to guard access to DMAC1 stream 0.
- Semaphore [g_spimux_sem](#)

A semaphore used to guard access to the SPI MUX.
- Mutex [g_gps_mtx](#)

A mutex used to guard access to the GPS device.
- volatile uint32_t [num_samples](#)

A counter used to accumulate the number of samples taken.
- uint8_t [dz_present](#)

A flag indicating the presence of deadzone hardware.
- bool [g_accel_interrupt](#)

Receiver direct path RSSI.
- bool [relay_2_enabled](#) = false

A flag used to indicate if the 2nd relay is available.
- VirtualTimer [vt3](#)

System timers.
- VirtualTimer [vt4](#)
- [InitCfg](#) [init_default](#)

Structure that holds default values for the system.
- [InitCfg](#) [init_user](#)

Structure that holds user defined settings from persistent storage.
- uint8_t [g_version_number](#) = 0

The default version number for the device (this is dynamically updated)
- volatile bool [osdp_client_connection_flag](#)

A flag used to indicate that a OSDP CControl Panel client connection has been made.

9.57.1 Detailed Description

The main process and executable.

Author

maria
neil
tony
peter

Definition in file [main.c](#).

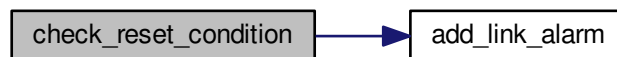
9.57.2 Function Documentation

9.57.2.1 void check_reset_condition (void)

Todo Alarm behaviour to be determined [modified and checked by PL on 9/10/15 and now is triggering error alarms as expected when a watchdog reset occurs]

Definition at line 694 of file main.c.

Here is the call graph for this function:



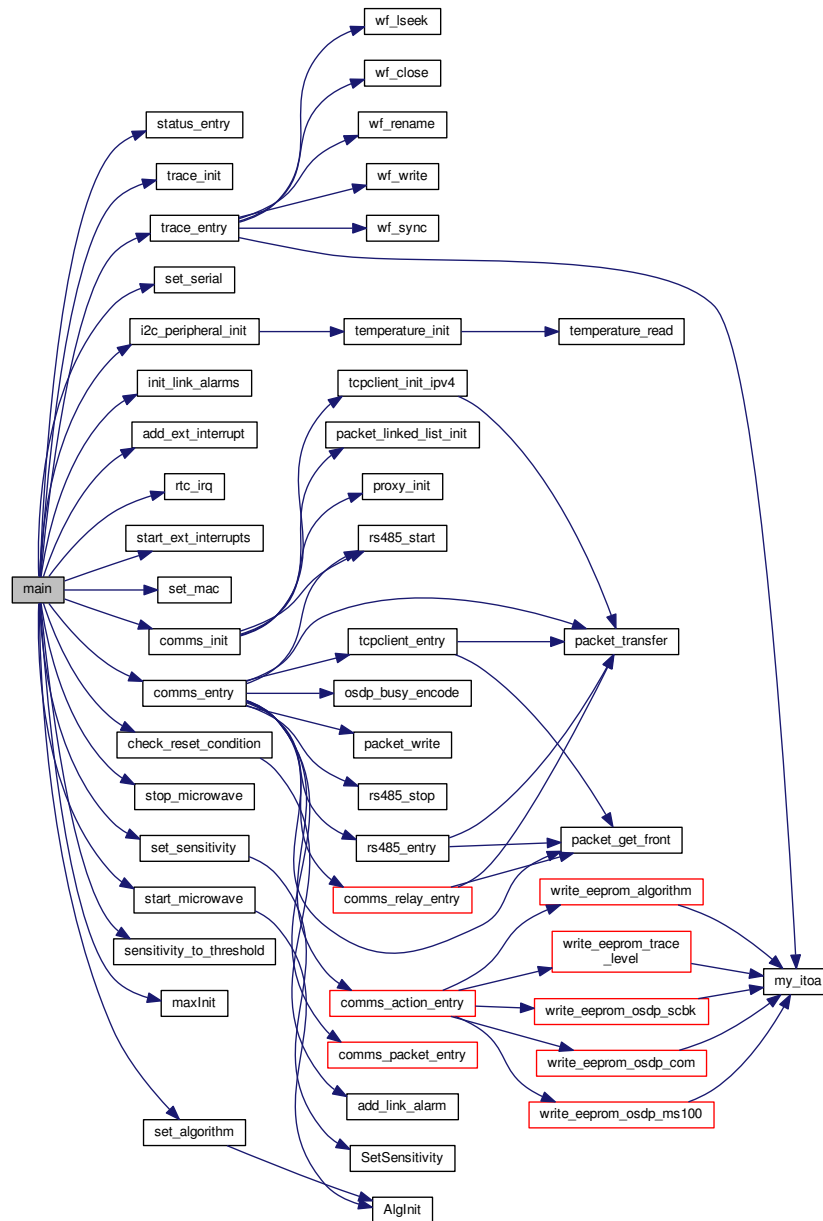
9.57.2.2 int main (void)

System Initialisation

Todo Change the microwave link enable to a programmable feature

Definition at line 1140 of file main.c.

Here is the call graph for this function:



9.57.2.3 void rtc_irq (EXTDriver * extp, expchannel_t channel)

An interrupt handler for RTC interrupts.

Parameters

<i>extp</i>	A pointer to the EXT driver
<i>channel</i>	The channel configuration to be used

Definition at line 409 of file main.c.

9.57.3 Variable Documentation

9.57.3.1 `bool g_accel_interrupt`

Receiver direct path RSSI.

A flag used to indicate that an accelerometer interrupt has triggered.

A flag used to indicate that an accelerometer interrupt has triggered

Definition at line 251 of file main.c.

9.57.3.2 `alarms_t g_alarms_osdp`

Alarms intended for OSDP polling responses.

A struct used to provide flags for alarms over OSDP.

Definition at line 157 of file main.c.

9.57.3.3 `alarms_t g_alarms_paired`

Alarms for a paired device (obfuscated)

A struct used to provide flags for alarms in the paired transmitter.

Definition at line 160 of file main.c.

9.57.3.4 `uint32_t g_alarms_relay[2]`

Alarms intended for the two relays.

A struct used to provide flags for alarms to the relay.

Definition at line 163 of file main.c.

9.57.3.5 `uint32_t g_bi_operating_mode = UNDEFINED_OPERATING_MODE`

A flag used to indicate the instantaneous operating mode.

A flag indicating the bidirectional transceiver mode.

Definition at line 154 of file main.c.

9.57.3.6 `uint8_t g_mac[6] = { MICROSENSE_OUI_0, MICROSENSE_OUI_1, MICROSENSE_OUI_2, 00, 00, 00 }`

The MAC address of this device.

The device MAC address.

Definition at line 144 of file main.c.

9.57.3.7 `ms100_state_e g_ms100_state`

The current control FSM state.

The device state.

Definition at line 166 of file main.c.

9.57.3.8 `Mutex g_mtx_dma1_s0`

A mutex used to guard access to DMAC1 stream 0.

A global mutex to control access to DMAC1 stream 0.

Definition at line 233 of file main.c.

9.57.3.9 `Mutex g_mtx_dma1_s7`

A mutex used to guard access to DMAC1 stream 7.

A global mutex to control access to DMAC1 stream 7.

Definition at line 230 of file main.c.

9.57.3.10 `uint32_t g_serial`

The serial number of this device.

The device serial number.

Definition at line 141 of file main.c.

9.57.3.11 `uint32_t g_uptime`

A counter used to indicate the application running time.

A system uptime counter (seconds)

Definition at line 189 of file main.c.

9.57.3.12 `void* g_uwave_samples`

A global pointer to the microwave samples stored in `l_uwave_samples`.

An array of microwave samples that are buffered for transmission or manipulation.

Definition at line 221 of file main.c.

9.57.3.13 `uint32_t hardware_status`

The hardware status bit field.

A bitfield used to store the system hardware status.

Definition at line 218 of file main.c.

9.57.3.14 `volatile osdp_sample_buffer_t l_uwave_samples`

A struct used to store microwave samples for display.

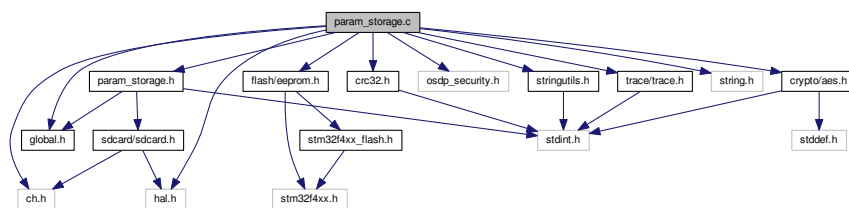
CCM microwave samples array.

Definition at line 224 of file main.c.

9.58 param_storage.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "param_storage.h"
#include "crc32.h"
#include "flash/EEPROM.h"
#include "osdp_security.h"
#include "stringutils.h"
#include "trace/trace.h"
#include "global.h"
#include <string.h>
#include "crypto/aes.h"
```

Include dependency graph for param_storage.c:



Macros

- #define **PARAM_STRING**(x) (x), sizeof(x)

Functions

- void **param_encrypt_16** (uint16_t *data, int32_t offset)
Update the encryption buffer with 16 consumed bits.
- void **param_encrypt_32** (uint32_t *data, int32_t offset)
Update the encryption buffer with 32 consumed bits.
- bool **create_param_file** (sdcard_file *sdfile)
Create a parameter file and fill with the parameter contents.
- int32_t **read_param_file** (sdcard_file *sdfile)
Read the SD Card contents to obtain the parameters.
- void **param_key_init** (void)
Initialise the parameter private key.
- void **param_key_destroy** (void)
Erase the parameter key from RAM.
- bool **param_write_16** (sdcard_file *sdfile, char *name, uint16_t val)
Modify a 16-bit parameter by name.
- uint32_t **param_read_plaintext** (int32_t addr)
Read a 32-bit parameter by address.
- bool **param_write_plaintext** (int32_t addr, uint32_t val)
Modify a 32-bit parameter by address.
- bool **param_write_32** (sdcard_file *sdfile, char *name, uint32_t val, bool encrypt)
Modify a 32-bit parameter by name.
- void **create_param_eeprom** (void)

Fill the emulated EEPROM with the parameters indexed from the table.

- void [read_param_eeprom](#) (void)

Read the EEPROM contents to obtain the parameters.

- bool [create_param_crc](#) (uint32_t *crc)

Generate the CRC from the working RAM contents and validate.

- bool [sanity_check_param](#) (void)

Check the parameters to determine if they are valid and lie within constraints.

- void **reinstate_params** (void)
- void **test_eeprom** (void)
- void **test_sdcard** (void)
- int32_t [param_init](#) (bool reset)

Initialise parameters from EEPROM or SD card.

- bool [param_open](#) (sdcard_file *sdfile, uint32_t flag)

Prepare the EEPROM and SD Card for access.

- void [param_close](#) (sdcard_file *sdfile, bool crc_update)

Close the EEPROM and SD Card, finalising changes.

Variables

- const uint16_t [VirtAddVarTab](#) [NB_OF_VAR]

This table is used to store the virtual address of the persistent parameters.

9.58.1 Detailed Description

Author

neil

Definition in file [param_storage.c](#).

9.58.2 Function Documentation

9.58.2.1 bool create_param_crc (uint32_t * crc)

Generate the CRC from the working RAM contents and validate.

Parameters

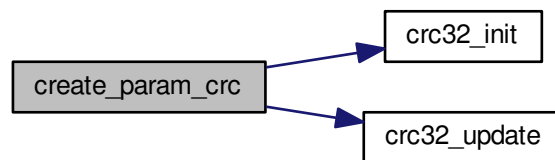
<i>crc</i>	The 32-bit CRC of the parameter list obtained elsewhere
------------	---

Returns

True if CRC matches, false if there is a mismatch

Definition at line 498 of file param_storage.c.

Here is the call graph for this function:



9.58.2.2 bool create_param_file (sdcard_file * *sdfile*)

Create a parameter file and fill with the parameter contents.

Parameters

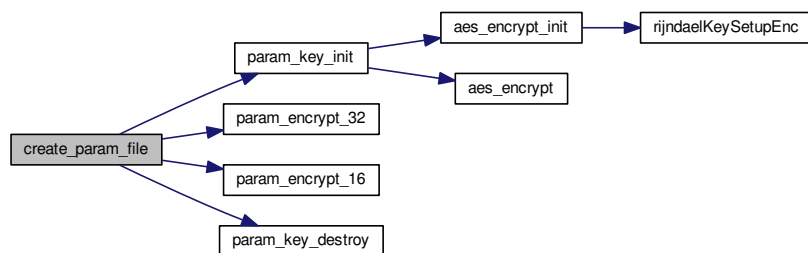
<i>sdfile</i>	A pointer to an SD card object
---------------	--------------------------------

Returns

A flag indicating success (true) or failure (false)

Definition at line 294 of file param_storage.c.

Here is the call graph for this function:



9.58.2.3 void param_close (sdcard_file * *sdfile*, bool *crc_update*)

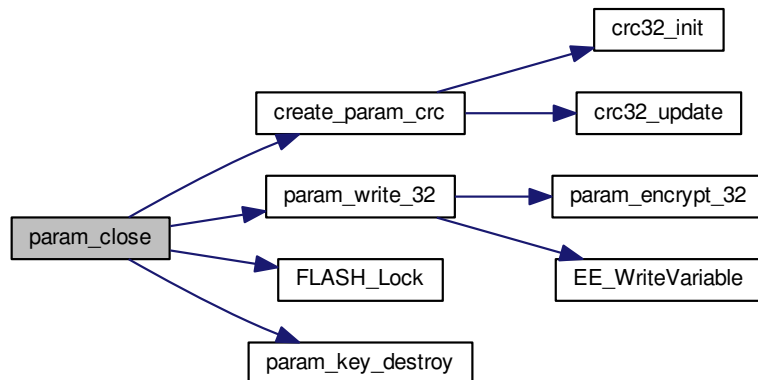
Close the EEPROM and SD Card, finalising changes.

Parameters

<i>sdf</i>	A pointer to an SD card object
<i>crc_update</i>	Flag indicating if CRC should be updated

Definition at line 973 of file param_storage.c.

Here is the call graph for this function:



9.58.2.4 int32_t param_init (bool reset)

Initialise parameters from EEPROM or SD card.

Parameters

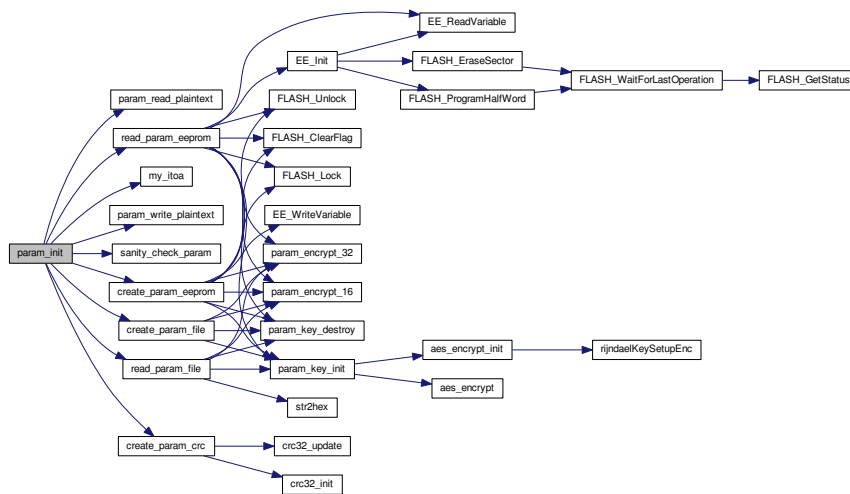
<i>reset</i>	A flag indicating if the parameter list must be reset
--------------	---

Returns

Indication of success or failure

Definition at line 669 of file param_storage.c.

Here is the call graph for this function:



9.58.2.5 bool param_open (sdcard_file * *sdfile*, uint32_t *flag*)

Prepare the EEPROM and SD Card for access.

Parameters

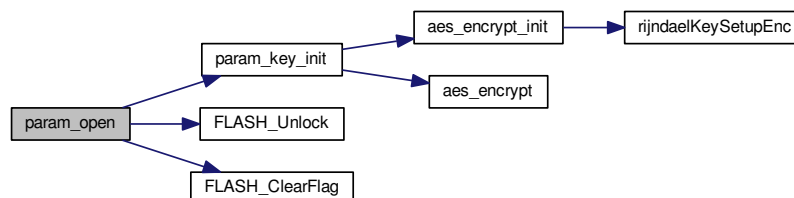
<i>sdfile</i>	A pointer to an SD card object
<i>flag</i>	How to open the file,PARAM_OPEN_DEFAULT or PARAM_OPEN_CREATE

Returns

True for success, false for failure

Definition at line 935 of file param_storage.c.

Here is the call graph for this function:



9.58.2.6 uint32_t param_read_plaintext (int32_t *addr*)

Read a 32-bit parameter by address.

Parameters

<i>addr</i>	EEPROM address
-------------	----------------

Returns

The parameter value

Definition at line 244 of file param_storage.c.

9.58.2.7 bool param_write_16 (sdcard_file * *sdf*file, char * *name*, uint16_t *val*)

Modify a 16-bit parameter by name.

Parameters

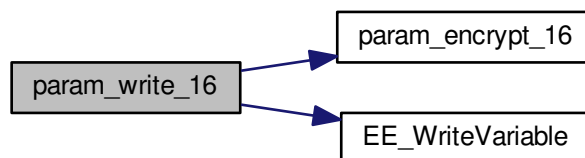
<i>sdf</i> file	A pointer to an SD card object
<i>name</i>	The name of the parameter
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 218 of file param_storage.c.

Here is the call graph for this function:



9.58.2.8 bool param_write_32 (sdcard_file * *sdf*file, char * *name*, uint32_t *val*, bool *encrypt*)

Modify a 32-bit parameter by name.

Parameters

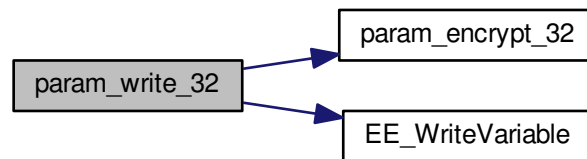
<i>sdf</i> file	A pointer to an SD card object
<i>name</i>	The name of the parameter
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 266 of file param_storage.c.

Here is the call graph for this function:



9.58.2.9 bool param_write_plaintext (int32_t *addr*, uint32_t *val*)

Modify a 32-bit parameter by address.

Parameters

<i>addr</i>	EEPROM address
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 255 of file param_storage.c.

9.58.2.10 int32_t read_param_file (sdcard_file * *sdf*)

Read the SD Card contents to obtain the parameters.

Parameters

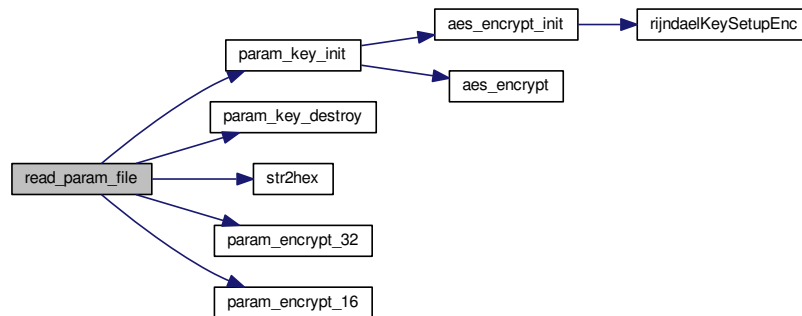
<i>sdf</i>	A pointer to an SD card object
------------	--------------------------------

Returns

Indication of success or failure

Definition at line 334 of file param_storage.c.

Here is the call graph for this function:

**9.58.2.11 bool sanity_check_param (void)**

Check the parameters to determine if they are valid and lie within constraints.

Returns

True indicates parameters are sane, false otherwise

Definition at line 536 of file param_storage.c.

9.59 param_storage.h File Reference

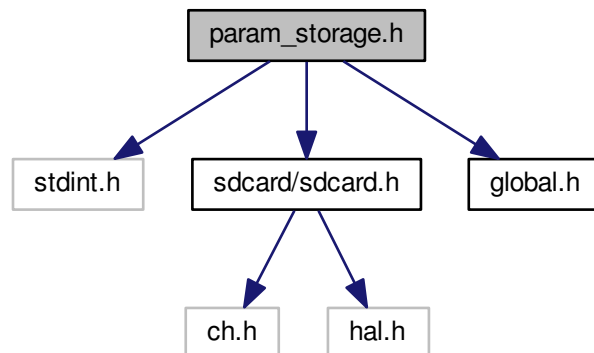
Functions for storing the (encrypted) parameters in emulated EEPROM and an SD card file.

```

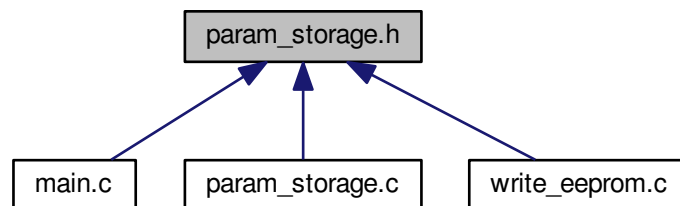
#include <stdint.h>
#include "sdcard/sdcard.h"
#include "global.h"

```

Include dependency graph for param_storage.h:



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



Data Structures

- struct [_param_name_t](#)

Macros

- #define **ENABLE_PARAM_ENCRYPTION**
- #define **PARAM_OPEN_DEFAULT** 0
- #define **PARAM_OPEN_CREATE** 1
- #define [PARAM_MAX](#) 42

The number of parameter values stored in EEPROM/SD card.

- #define [PARAM_OK](#) 0
- #define **PARAM_EEPROM_ERROR** 1
- #define **PARAM_SDCARD_ERROR** 2

Typedefs

- typedef struct [_param_name_t](#) [param_name_t](#)

Functions

- int32_t [param_init](#) (bool reset)
Initialise parameters from EEPROM or SD card.
- bool [param_open](#) (sdcard_file *sdfile, uint32_t flag)
Prepare the EEPROM and SD Card for access.
- void [param_close](#) (sdcard_file *sdfile, bool crc_update)
Close the EEPROM and SD Card, finalising changes.
- bool [param_write_16](#) (sdcard_file *sdfile, char *name, uint16_t val)
Modify a 16-bit parameter by name.
- bool [param_write_32](#) (sdcard_file *sdfile, char *name, uint32_t val, bool encrypt)
Modify a 32-bit parameter by name.
- bool [param_write_plaintext](#) (int32_t addr, uint32_t val)
Modify a 32-bit parameter by address.
- uint32_t [param_read_plaintext](#) (int32_t addr)
Read a 32-bit parameter by address.

9.59.1 Detailed Description

Functions for storing the (encrypted) parameters in emulated EEPROM and an SD card file.

Author

neil

Definition in file [param_storage.h](#).

9.59.2 Macro Definition Documentation

9.59.2.1 #define PARAM_OK 0

Return codes for parameter storage functions

Definition at line 21 of file [param_storage.h](#).

9.59.3 Function Documentation

9.59.3.1 void param_close (sdcard_file * sdfile, bool crc_update)

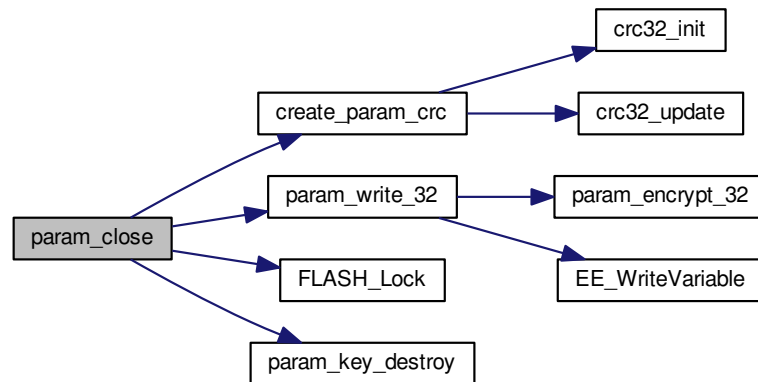
Close the EEPROM and SD Card, finalising changes.

Parameters

<i>sdfile</i>	A pointer to an SD card object
<i>crc_update</i>	Flag indicating if CRC should be updated

Definition at line 973 of file param_storage.c.

Here is the call graph for this function:



9.59.3.2 int32_t param_init (bool reset)

Initialise parameters from EEPROM or SD card.

Parameters

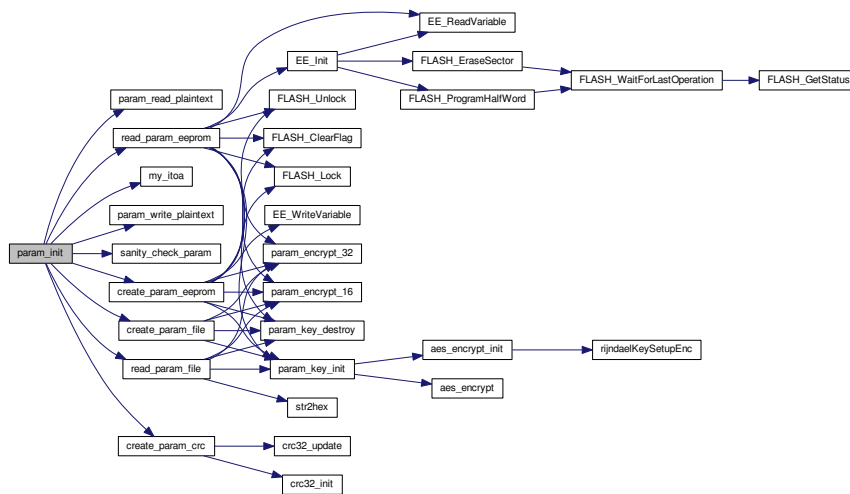
<i>reset</i>	A flag indicating if the parameter list must be reset
--------------	---

Returns

Indication of success or failure

Definition at line 669 of file param_storage.c.

Here is the call graph for this function:



9.59.3.3 bool param_open (sdcard_file * *sdfile*, uint32_t *flag*)

Prepare the EEPROM and SD Card for access.

Parameters

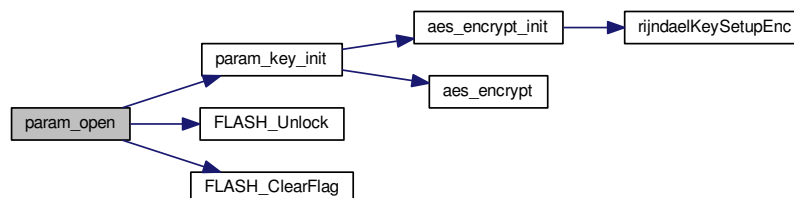
<i>sdfile</i>	A pointer to an SD card object
<i>flag</i>	How to open the file,PARAM_OPEN_DEFAULT or PARAM_OPEN_CREATE

Returns

True for success, false for failure

Definition at line 935 of file param_storage.c.

Here is the call graph for this function:



9.59.3.4 uint32_t param_read_plaintext (int32_t *addr*)

Read a 32-bit parameter by address.

Parameters

<i>addr</i>	EEPROM address
-------------	----------------

Returns

The parameter value

Definition at line 244 of file param_storage.c.

9.59.3.5 bool param_write_16 (sdcard_file * *sdfile*, char * *name*, uint16_t *val*)

Modify a 16-bit parameter by name.

Parameters

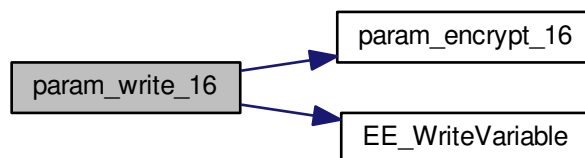
<i>sdfile</i>	A pointer to an SD card object
<i>name</i>	The name of the parameter
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 218 of file param_storage.c.

Here is the call graph for this function:



9.59.3.6 bool param_write_32 (sdcard_file * *sdfile*, char * *name*, uint32_t *val*, bool *encrypt*)

Modify a 32-bit parameter by name.

Parameters

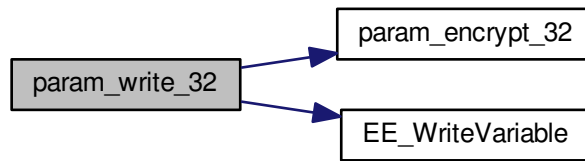
<i>sdfile</i>	A pointer to an SD card object
<i>name</i>	The name of the parameter
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 266 of file param_storage.c.

Here is the call graph for this function:



9.59.3.7 bool param_write_plaintext (int32_t addr, uint32_t val)

Modify a 32-bit parameter by address.

Parameters

<i>addr</i>	EEPROM address
<i>The</i>	value to be written

Returns

True for success, false for failure

Definition at line 255 of file param_storage.c.

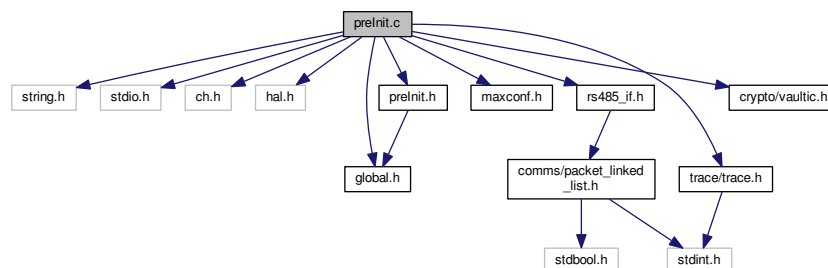
9.60 preInit.c File Reference

```

#include <string.h>
#include <stdio.h>
#include "ch.h"
#include "hal.h"
#include "global.h"
#include "preInit.h"
#include "maxconf.h"
#include "rs485_if.h"
#include "trace/trace.h"
#include "crypto/vaultic.h"

```

Include dependency graph for preInit.c:



Data Structures

- struct [_int_frac_t](#)
A struct used to define an integer/fractional divider pair.

Macros

- #define [FREQ_2_INT](#)(x)
Preprocessor conversion of floating-point frequency to integer component.

Typedefs

- typedef struct [_int_frac_t](#) [int_frac_t](#)
A typedef of the [_int_frac_t](#) struct.

Functions

- void [increment_mode](#) (ioportid_t ud_port, ioportmask_t ud_pad)
Rheostat Increment mode function.
- void [decrement_mode](#) (ioportid_t ud_port, ioportmask_t ud_pad)
Rheostat Decrement mode function.
- void [inc_dec](#) (ioportid_t cs_port, ioportmask_t cs_pad, ioportid_t ud_port, ioportmask_t ud_pad, int wiper)
Changes the value of the rheostat.
- bool [channel_to_int_frac](#) (void)
Function to convert a channel number to Max2828 register values.
- void [maxbufConfig](#) ([InitCfg](#) cfg)
- void [maxInit](#) ([InitCfg](#) cfg)
Max2828 Initialisation Function.
- void [maxSwitch](#) (uint8_t operating_mode)
Max2828 Initialisation Function.
- msg_t [accelerometerInit](#) ()
RS485 Initialisation Function.
- void [relayInit](#) ()
Relay Initialisation Function.
- void [curSenseInit](#) (void)
Current Sensor Initialisation Function.
- void [gpsInit](#) ()
GPS Initialisation.
- void [eepromInit](#) ()
EEPROM Initialisation.
- void [tamperInit](#) (void)
Board Initialisation.
- void [i2c_peripheral_init](#) (void)
Board Initialisation.
- void [rf_init](#) ()

Variables

- Semaphore [g_spimux_sem](#)
A semaphore used to guard access to the SPI MUX.
- [int_frac_t channel_int_frac_tx](#) [38]
An array of int_frac_t structs used to convert a transmit channel number.
- [int_frac_t channel_int_frac_rx](#) [38]
An array of int_frac_t structs used to convert a receive channel number.
- `adcsample_t vigil_tx_rx_buf` [VIGIL_TX_RX_SAMPLES]
- `adcsample_t vigil_tx_rx_average` = 0
- `bool flag_vigil_equals_tx` = false
- `bool flag_vigil_equals_rx` = false
- `bool flag_vigil_equals_tx_and_rx` = true

9.60.1 Function Documentation

9.60.1.1 `msg_t accelerometerInit ( void )`

RS485 Initialisation Function.

Note

Sends "RS485 OK" over RS485 to indicate Comms ok Accelerometer Initialisation function

Definition at line 628 of file preInit.c.

9.60.1.2 `bool channel_to_int_frac ( void )`

Function to convert a channel number to Max2828 register values.

A channel to integer/fraction conversion function.

Parameters

<i>channel</i>	The channel number (1 to 37 inclusive are valid)
<i>integer</i>	The integer-divider register value
<i>fraction</i>	The fractional-divider register value

Returns

True if successful, false otherwise

Definition at line 397 of file preInit.c.

9.60.1.3 `void decrement_mode ( ioportid_t ud_port, ioportmask_t ud_pad )`

Rheostat Decrement mode function.

Note

Sets the rheostat to operate in Decrement mode

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin

Definition at line 355 of file preInit.c.

9.60.1.4 void eeepromInit (void)

EEPROM Initialisation.

Note

Saves "Hello Microsense" to EEPROM, reads it back and sends the text over RS485

Definition at line 692 of file preInit.c.

9.60.1.5 void gpsInit (void)

GPS Initialisation.

Note

Definition at line 681 of file preInit.c.

9.60.1.6 void i2c_peripheral_init (void)

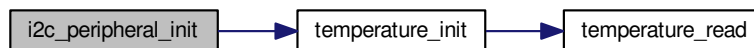
Board Initialisation.

Note

Initialises the MS100-A1 board to an operational State.

Definition at line 753 of file preInit.c.

Here is the call graph for this function:



9.60.1.7 void inc_dec (ioportid_t cs_port, ioportmask_t cs_pad, ioportid_t ud_port, ioportmask_t ud_pad, int wiper)

Changes the value of the rheostat.

Note

Changes the rheostat value - will increment or Decrement depending on whether increment_mode function or decrement_mode was previously called

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin
<i>cs_port</i>	the rheostat cs pin port (chip select)
<i>cs_pad</i>	the rheostat cs pin
<i>wiper</i>	defines the number of steps to increment or decrement 64 steps total - default to mid range value i.e. 10K rheostate defaults to 5k

Definition at line 372 of file preInit.c.

9.60.1.8 void increment_mode (ioportid_t *ud_port*, ioportmask_t *ud_pad*)

Rheostat Increment mode function.

Note

Sets the rheostat to operate in Increment mode

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin

Definition at line 343 of file preInit.c.

9.60.1.9 void maxInit (InitCfg *cfg*)

Max2828 Initialisation Function.

Note

Sets up the max registers

Parameters

<i>maxCfg</i>	initialisation configuration structure
---------------	--

Definition at line 453 of file preInit.c.

9.60.1.10 void maxSwitch (uint8_t *operating_mode*)

Max2828 Initialisation Function.

Note

Sets up the max registers

Parameters

<i>maxCfg</i>	initialisation configuration structure
---------------	--

Definition at line 516 of file preInit.c.

9.60.1.11 void relayInit (void)

Relay Initialisation Function.

Note

Definition at line 666 of file preInit.c.

9.60.1.12 void tamperInit (void)

Board Initialisation.

Note

Initialises the MS100-A1 board to an operational State.

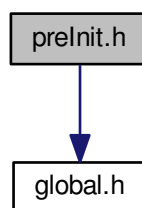
Definition at line 733 of file preInit.c.

9.61 preInit.h File Reference

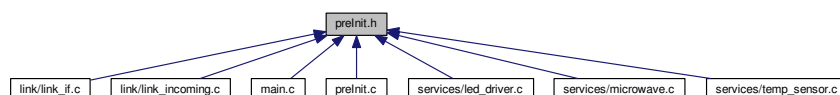
RF Initialization.

```
#include "global.h"
```

Include dependency graph for preInit.h:



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

**Macros**

- #define `rheostat_bank` GPIOE
Rheostat Bank.
- #define `rheostat_1_ck` GPIOE_PIN2
Rheostat 1 ck.
- #define `rheostat_1_ud` GPIOE_PIN1

- Rheostat 1 ud.*
- #define [rheostat_2_ck](#) GPIOE_PIN59
- Rheostat 2 ck.*
- #define [rheostat_2_ud](#) GPIOE_PIN58
- Rheostat 2 ud.*
- #define [ACC_OUT_ADDR](#) 0x1D
- Accelerometer I2C Address.*
- #define [ACC_CTRL_REG1](#) 0x20
- Accelerometer Control Register Address.*
- #define [ACC_FF_MT_CFG](#) 0x15
- Accelerometer Motion/ Freefall Configuration Register Address.*
- #define [ACC_FF_MT_THRS](#) 0x17
- Accelerometer Motion / Freefall threshold register address.*
- #define [ACC_FF_MT_COUNT](#) 0x18
- Accelerometer Motion / Freefall count register address - set the debounce counter.*
- #define [ACC_CTRL_REG4](#) 0x2D
- Accelerometer Control Register 4 Address.*
- #define [ACC_CTRL_REG5](#) 0x2E
- Accelerometer Control Register 5 Address.*
- #define [EEPROM_OUT_ADDR](#) 0xAE
- EEPROM I2C Address.*
- #define [VAULT_OUT_ADDR](#) 0xBE
- VaultIIC I2C Address.*
- #define [VIGIL_TX_RX_SAMPLES](#) 10
- #define [VIGIL_TX_LOWER](#) 310
- #define [VIGIL_TX_UPPER](#) 434
- #define [VIGIL_RX_LOWER](#) 695
- #define [VIGIL_RX_UPPER](#) 819
- #define [VIGIL_TX_AND_RX_LOWER](#) 1986
- #define [VIGIL_TX_AND_RX_UPPER](#) 2110
- #define [ADC_CH_NUMBER](#) 1

Functions

- void [vigil_transmit_receive_check](#) (void)
- void [i2c_peripheral_init](#) (void)
- Board Initialisation.*
- void [rf_init](#) (void)
- void [increment_mode](#) (ioportid_t ud_port, ioportmask_t ud_pad)
- Rheostat Increment mode function.*
- void [decrement_mode](#) (ioportid_t ud_port, ioportmask_t ud_pad)
- Rheostat Decrement mode function.*
- void [inc_dec](#) (ioportid_t cs_port, ioportmask_t cs_pad, ioportid_t ud_port, ioportmask_t ud_pad, int wiper)
- Changes the value of the rheostat.*
- void [rs485Init](#) (void)
- msg_t [accelerometerInit](#) (void)
- RS485 Initialisation Function.*
- void [maxInit](#) (InitCfg cfg)
- Max2828 Initialisation Function.*
- void [maxSwitch](#) (uint8_t operating_mode)
- Max2828 Initialisation Function.*

- void [relayInit](#) (void)
Relay Initialisation Function.
- void **curSensInit** (void)
- void [gpsInit](#) (void)
GPS Initialisation.
- void [eepromInit](#) (void)
EEPROM Initialisation.
- void [tamperInit](#) (void)
Board Initialisation.
- void **scanning_antenna_init** (void)
- uint32_t **tempAlarm** (int32_t temp)

Variables

- bool **flag_vigil_equals_tx**
- bool **flag_vigil_equals_rx**
- bool **flag_vigil_equals_tx_and_rx**

9.61.1 Detailed Description

RF Initialization.

Initialisation and configuration functions for the MS100 RF, definitions of values used during the setup procedure.

Definition in file [preInit.h](#).

9.61.2 Function Documentation

9.61.2.1 msg_t accelerometerInit (void)

RS485 Initialisation Function.

Note

Sends "RS485 OK" over RS485 to indicate Comms ok Accelerometer Initialisation function

Definition at line 628 of file preInit.c.

9.61.2.2 void decrement_mode (ioportid_t ud_port, ioportmask_t ud_pad)

Rheostat Decrement mode function.

Note

Sets the rheostat to operate in Decrement mode

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin

Definition at line 355 of file preInit.c.

9.61.2.3 void eepromInit (void)

EEPROM Initialisation.

Note

Saves "Hello Microsense" to EEPROM, reads it back and sends the text over RS485

Definition at line 692 of file preInit.c.

9.61.2.4 void gpsInit (void)

GPS Initialisation.

Note

Definition at line 681 of file preInit.c.

9.61.2.5 void i2c_peripheral_init (void)

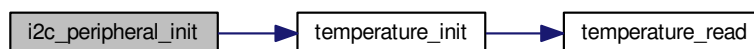
Board Initialisation.

Note

Initialises the MS100-A1 board to an operational State.

Definition at line 753 of file preInit.c.

Here is the call graph for this function:



9.61.2.6 void inc_dec (ioportid_t cs_port, ioportmask_t cs_pad, ioportid_t ud_port, ioportmask_t ud_pad, int wiper)

Changes the value of the rheostat.

Note

Changes the rheostat value - will increment or Decrement depending on whether `increment_mode` function or `decrement_mode` was previously called

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin
<i>cs_port</i>	the rheostat cs pin port (chip select)
<i>cs_pad</i>	the rheostat cs pin
<i>wiper</i>	defines the number of steps to increment or decrement 64 steps total - default to mid range value i.e. 10K rheostate defaults to 5k

Definition at line 372 of file preInit.c.

9.61.2.7 void increment_mode (ioportid_t *ud_port*, ioportmask_t *ud_pad*)

Rheostat Increment mode function.

Note

Sets the rheostat to operate in Increment mode

Parameters

<i>ud_port</i>	the rheostat u/d pin port
<i>ud_pad</i>	the rheostat u/d pin

Definition at line 343 of file preInit.c.

9.61.2.8 void maxInit (InitCfg *cfg*)

Max2828 Initialisation Function.

Note

Sets up the max registers

Parameters

<i>maxCfg</i>	initialisation configuration structure
---------------	--

Definition at line 453 of file preInit.c.

9.61.2.9 void maxSwitch (uint8_t *operating_mode*)

Max2828 Initialisation Function.

Note

Sets up the max registers

Parameters

<i>maxCfg</i>	initialisation configuration structure
---------------	--

Definition at line 516 of file preInit.c.

9.61.2.10 void relayInit (void)

Relay Initialisation Function.

Note

Definition at line 666 of file preInit.c.

9.61.2.11 void tamperInit (void)

Board Initialisation.

Note

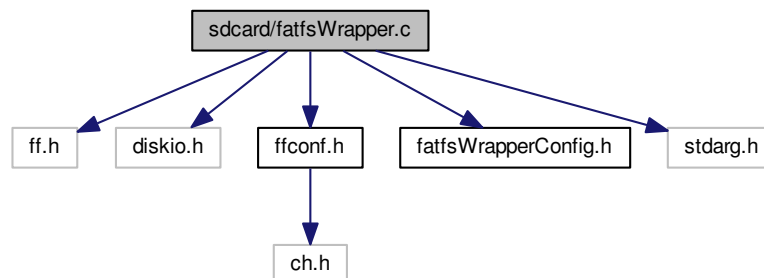
Initialises the MS100-A1 board to an operational State.

Definition at line 733 of file preInit.c.

9.62 sdcard/fatfsWrapper.c File Reference

```
#include "ff.h"
#include "diskio.h"
#include "ffconf.h"
#include "fatfsWrapperConfig.h"
#include <stdarg.h>
```

Include dependency graph for fatfsWrapper.c:



Data Structures

- struct [wrapper_msg_base](#)
- struct [wrapper_msg_vBYTEpFATFS](#)
- struct [wrapper_msg_pFILpTCHARvBYTE](#)
- struct [wrapper_msg_pFILpVOIDvUINTpUINT](#)
- struct [wrapper_msg_pFILvDWORD](#)
- struct [wrapper_msg_pFIL](#)
- struct [wrapper_msg_pDIRpTCHAR](#)
- struct [wrapper_msg_pDIRpFILINFO](#)
- struct [wrapper_msg_pTCHARpFILINFO](#)
- struct [wrapper_msg_pTCHARpDWORDppFATFS](#)
- struct [wrapper_msg_pTCHAR](#)
- struct [wrapper_msg_pTCHARvBYTEvBYTE](#)

- struct [wrapper_msg_pTCHARpTCHAR](#)
- struct [wrapper_msg_vBYTE](#)
- struct [wrapper_msg_pTCHARvUINT](#)
- struct [wrapper_msg_vBYTEvBYTEvUINT](#)
- struct [wrapper_msg_vBYTEpDWORDpVOID](#)
- struct [wrapper_msg_vTCHARpFIL](#)
- struct [wrapper_msg_pTCHARpFIL](#)
- struct [wrapper_msg_pTCHARvINTpFILpTCHAR](#)

Macros

- #define **HAS_MOUNT** (FATFS_WRP_ENABLE_MOUNT)
- #define **HAS_OPEN** (FATFS_WRP_ENABLE_OPEN)
- #define **HAS_READ** (FATFS_WRP_ENABLE_READ)
- #define **HAS_CLOSE** (FATFS_WRP_ENABLE_CLOSE)
- #define **HAS_WRITE** (FATFS_WRP_ENABLE_WRITE && !_FS_READONLY)
- #define **HAS_SYNC** (FATFS_WRP_ENABLE_SYNC && !_FS_READONLY)
- #define **HAS_CHDRIVE** (FATFS_WRP_ENABLE_CHDRIVE && (_FS_RPATH >= 1))
- #define **HAS_CHDIR** (FATFS_WRP_ENABLE_CHDIR && (_FS_RPATH >= 1))
- #define **HAS_GETCWD** (FATFS_WRP_ENABLE_GETCWD && (_FS_RPATH >= 2))
- #define **HAS_LSEEK** (FATFS_WRP_ENABLE_LSEEK && (_FS_MINIMIZE <= 2))
- #define **HAS_OPENDIR** (FATFS_WRP_ENABLE_OPENDIR && (_FS_MINIMIZE <= 1))
- #define **HAS_READDIR** (FATFS_WRP_ENABLE_READDIR && (_FS_MINIMIZE <= 1))
- #define **HAS_STAT** (FATFS_WRP_ENABLE_STAT && (_FS_MINIMIZE == 0))
- #define **HAS_GETFREE** (FATFS_WRP_ENABLE_GETFREE && (_FS_MINIMIZE == 0) && !_FS_READ←
ONLY)
- #define **HAS_TRUNCATE** (FATFS_WRP_ENABLE_TRUNCATE && (_FS_MINIMIZE == 0) && !_FS_RE←
ADONLY)
- #define **HAS_UNLINK** (FATFS_WRP_ENABLE_UNLINK && (_FS_MINIMIZE == 0) && !_FS_READONLY)
- #define **HAS_MKDIR** (FATFS_WRP_ENABLE_MKDIR && (_FS_MINIMIZE == 0) && !_FS_READONLY)
- #define **HAS_CHMOD** (FATFS_WRP_ENABLE_CHMOD && (_FS_MINIMIZE == 0) && !_FS_READONLY)
- #define **HAS_UTIME** (FATFS_WRP_ENABLE_UTIME && (_FS_MINIMIZE == 0) && !_FS_READONLY)
- #define **HAS_RENAME** (FATFS_WRP_ENABLE_RENAME && (_FS_MINIMIZE == 0) && !_FS_READON←
LY)
- #define **HAS_FORWARD** (FATFS_WRP_ENABLE_FORWARD && _USE_FORWARD && _FS_TINY)
- #define **HAS_MKFS** (FATFS_WRP_ENABLE_MKFS && _USE_MKFS && !_FS_READONLY)
- #define **HAS_FDISK** (FATFS_WRP_ENABLE_FDISK && _USE_MKFS && !_FS_READONLY && (_MUL←
TI_PARTITION == 2))
- #define **HAS_GETS** (FATFS_WRP_ENABLE_GETS && _USE_STRFUNC)
- #define **HAS_PUTC** (FATFS_WRP_ENABLE_PUTC && _USE_STRFUNC && !_FS_READONLY)
- #define **HAS_PUTS** (FATFS_WRP_ENABLE_PUTS && _USE_STRFUNC && !_FS_READONLY)
- #define **HAS_PRINTF** (FATFS_WRP_ENABLE_PRINTF && _USE_STRFUNC && !_FS_READONLY)
- #define **_wrapper_msg_base**
- #define **IsLower**(c) (((c)>='a')&&((c)<='z'))
- #define **IsDigit**(c) (((c)>='0')&&((c)<='9'))

Enumerations

- enum [wrapper_action](#) {
eFMOUNT, **eFOPEN**, **eFREAD**, **eFWRITE**,
eFSYNC, **eFCHDIR**, **eFLSEEK**, **eFCLOSE**,
eFOPENDIR, **eFREADDIR**, **eFSTAT**, **eFTRUNCATE**,
eFUNLINK, **eFMKDIR**, **eFRENAME**, **eFGETS**,
eFPUTC, **eFPUTS**, **eFPRINTF** }

Enumeration of all possible actions depending on activated functions.

Functions

- void **wf_init** (tprio_t priority)
- FRESULT **wf_mount** (BYTE vol, FATFS *fs)
Mount/Unmount a logical Drive.
- FRESULT **wf_open** (FIL *fp, const TCHAR *path, BYTE mode)
Open or Create a File.
- FRESULT **wf_read** (FIL *fp, void *buff, UINT btr, UINT *br)
Read File.
- FRESULT **wf_write** (FIL *fp, const void *buff, UINT btw, UINT *bw)
Write File.
- FRESULT **wf_sync** (FIL *fp)
Synchronize the File Object.
- FRESULT **wf_chdir** (const TCHAR *path)
Change current path.
- FRESULT **wf_lseek** (FIL *fp, DWORD ofs)
Seek File R/W Pointer.
- FRESULT **wf_close** (FIL *fp)
Close File.
- FRESULT **wf_opendir** (DIR *dj, const TCHAR *path)
Create a Directory Object.
- FRESULT **wf_readdir** (DIR *dj, FILINFO *fno)
Read Directory Entry in Sequence.
- FRESULT **wf_stat** (const TCHAR *path, FILINFO *fno)
Get File Status.
- FRESULT **wf_truncate** (FIL *fp)
Truncate File.
- FRESULT **wf_unlink** (const TCHAR *path)
Delete a File or Directory.
- FRESULT **wf_mkdir** (const TCHAR *path)
Create a Directory.
- FRESULT **wf_rename** (const TCHAR *path_old, const TCHAR *path_new)
Rename File/Directory.
- TCHAR * **wf_gets** (TCHAR *buff, int len, FIL *fil)
Get a string from the file.
- int **wf_putc** (TCHAR c, FIL *fil)
Put a character to the file.
- int **wf_puts** (const TCHAR *str, FIL *fil)
Put a string to the file.
- int **wf_printf** (FIL *fil, const TCHAR *str,...)
Put a string to the file.

9.62.1 Detailed Description

Author

Matthias Blaicher

Definition in file [fatfsWrapper.c](#).

9.62.2 Macro Definition Documentation

9.62.2.1 #define _wrapper_msg_base

Value:

```
enum wrapper_action action;          \
    FRESULT result;
```

Definition at line 138 of file fatfsWrapper.c.

9.62.3 Function Documentation

9.62.3.1 FRESULT wf_chdir (const TCHAR * *path*)

Change current path.

Parameters

<i>path</i>	Pointer to the directory path.
-------------	--------------------------------

Returns

Definition at line 711 of file fatfsWrapper.c.

9.62.3.2 FRESULT wf_close (FIL * *fp*)

Close File.

Parameters

<i>fp</i>	Pointer to the file object to be closed.
-----------	--

Returns

Definition at line 768 of file fatfsWrapper.c.

9.62.3.3 TCHAR* wf_gets (TCHAR * *buff*, int *len*, FIL * *fil*)

Get a string from the file.

Parameters

<i>buff</i>	Pointer to the string buffer to read.
<i>len</i>	Size of string buffer (characters).
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1024 of file fatfsWrapper.c.

9.62.3.4 FRESULT wf_lseek (FIL * *fp*, DWORD *ofs*)

Seek File R/W Pointer.

Parameters

<i>fp</i>	Pointer to the file object.
<i>ofs</i>	File pointer from top of file.

Returns

Definition at line 749 of file fatfsWrapper.c.

9.62.3.5 FRESULT wf_mkdir (const TCHAR * *path*)

Create a Directory.

Parameters

<i>path</i>	Pointer to the directory path.
-------------	--------------------------------

Returns

Definition at line 902 of file fatfsWrapper.c.

9.62.3.6 FRESULT wf_mount (BYTE *vol*, FATFS * *fs*)

Mount/Unmount a logical Drive.

Parameters

<i>vol</i>	Logical drive number to be mounted/unmounted.
<i>fs</i>	Pointer to new file system object (NULL for unmount).

Returns

Definition at line 585 of file fatfsWrapper.c.

9.62.3.7 FRESULT wf_open (FIL * *fp*, const TCHAR * *path*, BYTE *mode*)

Open or Create a File.

Parameters

<i>fp</i>	Pointer to the blank file object.
<i>path</i>	Pointer to the file name.
<i>mode</i>	Access mode and file open mode flags.

Returns

Definition at line 607 of file fatfsWrapper.c.

9.62.3.8 FRESULT wf_opendir (DIR * *dj*, const TCHAR * *path*)

Create a Directory Object.

Parameters

<i>dj</i>	Pointer to directory object to create.
<i>path</i>	Pointer to the directory path.

Returns

Definition at line 787 of file fatfsWrapper.c.

9.62.3.9 int wf_printf (FIL * *fil*, const TCHAR * *str*, ...)

Put a string to the file.

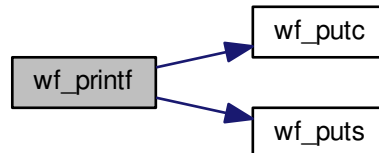
Parameters

<i>str</i>	Pointer to the string to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1085 of file fatfsWrapper.c.

Here is the call graph for this function:

**9.62.3.10** int wf_putc (TCHAR *c*, FIL * *fil*)

Put a character to the file.

Parameters

<i>c</i>	A character to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1044 of file fatfsWrapper.c.

9.62.3.11 int wf_puts (const TCHAR * *str*, FIL * *fil*)

Put a string to the file.

Parameters

<i>str</i>	Pointer to the string to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1064 of file fatfsWrapper.c.

9.62.3.12 FRESULT wf_read (FIL * *fp*, void * *buff*, UINT *btr*, UINT * *br*)

Read File.

Parameters

<i>fp</i>	Pointer to the file object.
<i>buff</i>	Pointer to data buffer.
<i>btr</i>	Number of bytes to read.
<i>br</i>	Pointer to number of bytes read.

Returns

Definition at line 630 of file fatfsWrapper.c.

9.62.3.13 FRESULT wf_readdir (DIR * *dj*, FILINFO * *fno*)

Read Directory Entry in Sequence.

Parameters

<i>dj</i>	Pointer to the open directory object.
<i>fno</i>	Pointer to file information to return.

Returns

Definition at line 806 of file fatfsWrapper.c.

9.62.3.14 FRESULT wf_rename (const TCHAR * *path_old*, const TCHAR * *path_new*)

Rename File/Directory.

Parameters

<i>path_old</i>	Pointer to the old name.
<i>path_new</i>	Pointer to the new name

Returns

Definition at line 962 of file fatfsWrapper.c.

9.62.3.15 FRESULT wf_stat (const TCHAR * *path*, FILINFO * *fno*)

Get File Status.

Parameters

<i>path</i>	Pointer to the file path.
<i>fno</i>	Pointer to file information to return.

Returns

Definition at line 826 of file fatfsWrapper.c.

9.62.3.16 FRESULT wf_sync (FIL * *fp*)

Synchronize the File Object.

Parameters

<i>fp</i>	Pointer to the file object.
-----------	-----------------------------

Returns

Definition at line 676 of file fatfsWrapper.c.

9.62.3.17 FRESULT wf_truncate (FIL * *fp*)

Truncate File.

Parameters

<i>fp</i>	Pointer to the file object.
-----------	-----------------------------

Returns

Definition at line 868 of file fatfsWrapper.c.

9.62.3.18 FRESULT wf_unlink (const TCHAR * *path*)

Delete a File or Directory.

Parameters

<i>path</i>	Pointer to the file or directory path.
-------------	--

Returns

Definition at line 885 of file fatfsWrapper.c.

9.62.3.19 FRESULT wf_write (FIL * *fp*, const void * *buff*, UINT *btw*, UINT * *bw*)

Write File.

Parameters

<i>fp</i>	Pointer to the file object.
<i>buff</i>	Pointer to the data to be written.
<i>btw</i>	Number of bytes to write.
<i>bw</i>	Pointer to number of bytes written.

Returns

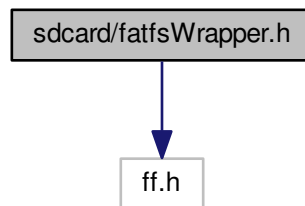
Definition at line 655 of file fatfsWrapper.c.

9.63 sdcard/fatfsWrapper.h File Reference

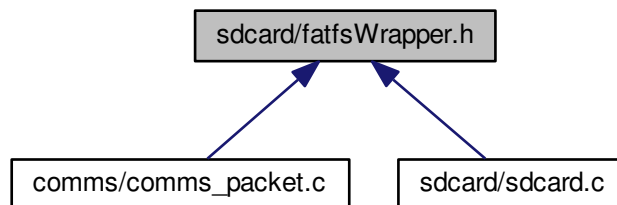
Wrapper around FatFS to localize all access over one single thread.

```
#include "ff.h"
```

Include dependency graph for fatfsWrapper.h:



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



Macros

- `#define wf_eof(fp) f_eof(fp)`
- `#define wf_error(fp) f_error(fp)`

- #define **wf_tell**(fp) f_tell(fp)
- #define **wf_size**(fp) f_size(fp)

Functions

- void **wf_init** (tprio_t priority)
- FRESULT **wf_mount** (BYTE, FATFS *)
Mount/Unmount a logical Drive.
- FRESULT **wf_open** (FIL *, const TCHAR *, BYTE)
Open or Create a File.
- FRESULT **wf_read** (FIL *, void *, UINT, UINT *)
Read File.
- FRESULT **wf_lseek** (FIL *, DWORD)
Seek File R/W Pointer.
- FRESULT **wf_close** (FIL *)
Close File.
- FRESULT **wf_opendir** (DIR *, const TCHAR *)
Create a Directory Object.
- FRESULT **wf_readdir** (DIR *, FILINFO *)
Read Directory Entry in Sequence.
- FRESULT **wf_stat** (const TCHAR *, FILINFO *)
Get File Status.
- FRESULT **wf_write** (FIL *, const void *, UINT, UINT *)
Write File.
- FRESULT **wf_getfree** (const TCHAR *, DWORD *, FATFS **)
- FRESULT **wf_truncate** (FIL *)
Truncate File.
- FRESULT **wf_sync** (FIL *)
Synchronize the File Object.
- FRESULT **wf_unlink** (const TCHAR *)
Delete a File or Directory.
- FRESULT **wf_mkdir** (const TCHAR *)
Create a Directory.
- FRESULT **wf_chmod** (const TCHAR *, BYTE, BYTE)
- FRESULT **wf_ftime** (const TCHAR *, const FILINFO *)
- FRESULT **wf_rename** (const TCHAR *, const TCHAR *)
Rename File/Directory.
- FRESULT **wf_chdrive** (BYTE)
- FRESULT **wf_chdir** (const TCHAR *)
Change current path.
- FRESULT **wf_getcwd** (TCHAR *, UINT)
- FRESULT **wf_forward** (FIL *, UINT*)(const BYTE *, UINT), UINT, UINT *)
- FRESULT **wf_mkfs** (BYTE, BYTE, UINT)
- FRESULT **wf_fdisk** (BYTE, const DWORD[], void *)
- int **wf_putc** (TCHAR, FIL *)
Put a character to the file.
- int **wf_puts** (const TCHAR *, FIL *)
Put a string to the file.
- int **wf_printf** (FIL *, const TCHAR *,...)
Put a string to the file.
- TCHAR * **wf_gets** (TCHAR *, int, FIL *)
Get a string from the file.

9.63.1 Detailed Description

Wrapper around FatFS to localize all access over one single thread.

Author

Matthias Blaicher

This greatly reduces stack usage when you use FatFS from several threads.

Definition in file [fatfsWrapper.h](#).

9.63.2 Function Documentation

9.63.2.1 FRESULT wf_chdir (const TCHAR * *path*)

Change current path.

Parameters

<i>path</i>	Pointer to the directory path.
-------------	--------------------------------

Returns

Definition at line 711 of file fatfsWrapper.c.

9.63.2.2 FRESULT wf_close (FIL * *fp*)

Close File.

Parameters

<i>fp</i>	Pointer to the file object to be closed.
-----------	--

Returns

Definition at line 768 of file fatfsWrapper.c.

9.63.2.3 TCHAR* wf_gets (TCHAR * *buff*, int *len*, FIL * *fil*)

Get a string from the file.

Parameters

<i>buff</i>	Pointer to the string buffer to read.
<i>len</i>	Size of string buffer (characters).
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1024 of file fatfsWrapper.c.

9.63.2.4 FRESULT wf_lseek (FIL * *fp*, DWORD *ofs*)

Seek File R/W Pointer.

Parameters

<i>fp</i>	Pointer to the file object.
<i>ofs</i>	File pointer from top of file.

Returns

Definition at line 749 of file fatfsWrapper.c.

9.63.2.5 FRESULT wf_mkdir (const TCHAR * *path*)

Create a Directory.

Parameters

<i>path</i>	Pointer to the directory path.
-------------	--------------------------------

Returns

Definition at line 902 of file fatfsWrapper.c.

9.63.2.6 FRESULT wf_mount (BYTE *vol*, FATFS * *fs*)

Mount/Unmount a logical Drive.

Parameters

<i>vol</i>	Logical drive number to be mounted/unmounted.
<i>fs</i>	Pointer to new file system object (NULL for unmount).

Returns

Definition at line 585 of file fatfsWrapper.c.

9.63.2.7 FRESULT wf_open (FIL * *fp*, const TCHAR * *path*, BYTE *mode*)

Open or Create a File.

Parameters

<i>fp</i>	Pointer to the blank file object.
<i>path</i>	Pointer to the file name.
<i>mode</i>	Access mode and file open mode flags.

Returns

Definition at line 607 of file fatfsWrapper.c.

9.63.2.8 FRESULT wf_opendir (DIR * *dj*, const TCHAR * *path*)

Create a Directory Object.

Parameters

<i>dj</i>	Pointer to directory object to create.
<i>path</i>	Pointer to the directory path.

Returns

Definition at line 787 of file fatfsWrapper.c.

9.63.2.9 int wf_printf (FIL * *fil*, const TCHAR * *str*, ...)

Put a string to the file.

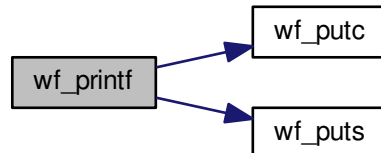
Parameters

<i>str</i>	Pointer to the string to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1085 of file fatfsWrapper.c.

Here is the call graph for this function:

**9.63.2.10** int wf_putc (TCHAR *c*, FIL * *fil*)

Put a character to the file.

Parameters

<i>c</i>	A character to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1044 of file fatfsWrapper.c.

9.63.2.11 int wf_puts (const TCHAR * *str*, FIL * *fil*)

Put a string to the file.

Parameters

<i>str</i>	Pointer to the string to be output.
<i>fil</i>	Pointer to the file object.

Returns

Definition at line 1064 of file fatfsWrapper.c.

9.63.2.12 FRESULT wf_read (FIL * *fp*, void * *buff*, UINT *btr*, UINT * *br*)

Read File.

Parameters

<i>fp</i>	Pointer to the file object.
<i>buff</i>	Pointer to data buffer.
<i>btr</i>	Number of bytes to read.
<i>br</i>	Pointer to number of bytes read.

Returns

Definition at line 630 of file fatfsWrapper.c.

9.63.2.13 FRESULT wf_readdir (DIR * *dj*, FILINFO * *fno*)

Read Directory Entry in Sequence.

Parameters

<i>dj</i>	Pointer to the open directory object.
<i>fno</i>	Pointer to file information to return.

Returns

Definition at line 806 of file fatfsWrapper.c.

9.63.2.14 FRESULT wf_rename (const TCHAR * *path_old*, const TCHAR * *path_new*)

Rename File/Directory.

Parameters

<i>path_old</i>	Pointer to the old name.
<i>path_new</i>	Pointer to the new name

Returns

Definition at line 962 of file fatfsWrapper.c.

9.63.2.15 FRESULT wf_stat (const TCHAR * *path*, FILINFO * *fno*)

Get File Status.

Parameters

<i>path</i>	Pointer to the file path.
<i>fno</i>	Pointer to file information to return.

Returns

Definition at line 826 of file fatfsWrapper.c.

9.63.2.16 FRESULT wf_sync (FIL * *fp*)

Synchronize the File Object.

Parameters

<i>fp</i>	Pointer to the file object.
-----------	-----------------------------

Returns

Definition at line 676 of file fatfsWrapper.c.

9.63.2.17 FRESULT wf_truncate (FIL * *fp*)

Truncate File.

Parameters

<i>fp</i>	Pointer to the file object.
-----------	-----------------------------

Returns

Definition at line 868 of file fatfsWrapper.c.

9.63.2.18 FRESULT wf_unlink (const TCHAR * *path*)

Delete a File or Directory.

Parameters

<i>path</i>	Pointer to the file or directory path.
-------------	--

Returns

Definition at line 885 of file fatfsWrapper.c.

9.63.2.19 FRESULT wf_write (FIL * *fp*, const void * *buff*, UINT *btw*, UINT * *bw*)

Write File.

Parameters

<i>fp</i>	Pointer to the file object.
<i>buff</i>	Pointer to the data to be written.
<i>btw</i>	Number of bytes to write.
<i>bw</i>	Pointer to number of bytes written.

Returns

Definition at line 655 of file fatfsWrapper.c.

9.64 sdcard/sample_record.c File Reference

9.64.1 Detailed Description

Author

neil

Definition in file [sample_record.c](#).

9.65 sdcard/sd_fw_download.c File Reference

9.65.1 Detailed Description

Author

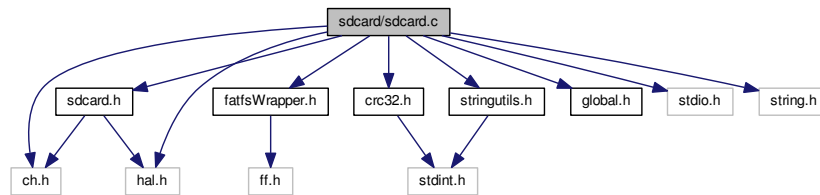
neil

Definition in file [sd_fw_download.c](#).

9.66 sdcard/sdcard.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "sdcard.h"
#include "fatfsWrapper.h"
#include "crc32.h"
#include "stringutils.h"
#include "global.h"
#include <stdio.h>
#include <string.h>
```

Include dependency graph for `sdcard.c`:



9.66.1 Detailed Description

Author

neil

Definition in file [sdcard.c](#).

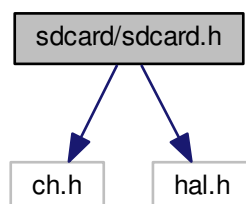
9.67 sdcard/sdcard.h File Reference

A thread used to save buffered FW images as a background task.

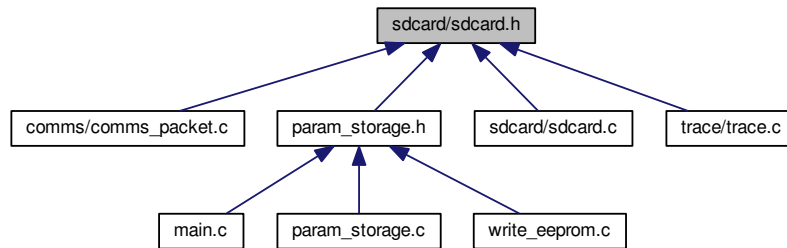
```
#include "ch.h"
```

```
#include "hal.h"
```

Include dependency graph for `sdcard.h`:



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



Macros

- `#define sdc card_file int32_t`

9.67.1 Detailed Description

A thread used to save buffered FW images as a background task.

A C library for SD card access.

Author

neil

Definition in file [sdc card.h](#).

9.68 services/accelerometer.c File Reference

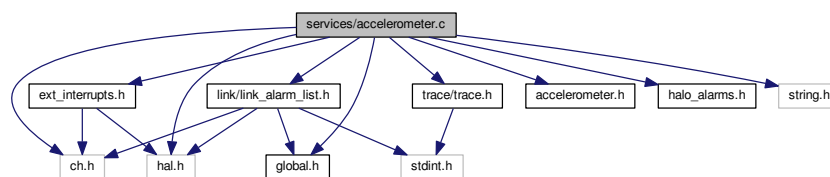
Accelerometer driver code.

```

#include "ch.h"
#include "hal.h"
#include "accelerometer.h"
#include "ext_interrupts.h"
#include "global.h"
#include "link/link_alarm_list.h"
#include "halo_alarms.h"
#include "trace/trace.h"
#include "string.h"

```

Include dependency graph for `accelerometer.c`:



Functions

- **WORKING_AREA** (accel_thread, [ACCEL_THREAD_SIZE](#))
- msg_t [accel_entry](#) (void *arg)

A thread function used to startup and process accelerometer events.

9.68.1 Detailed Description

Accelerometer driver code.

Author

Tony O'Callaghan

Version

1.0, Wed 26/11/14

Definition in file [accelerometer.c](#).

9.68.2 Function Documentation

9.68.2.1 msg_t [accel_entry](#) (void * *arg*)

A thread function used to startup and process accelerometer events.

Parameters

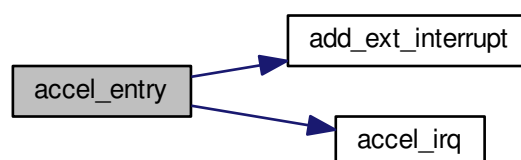
<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 204 of file [accelerometer.c](#).

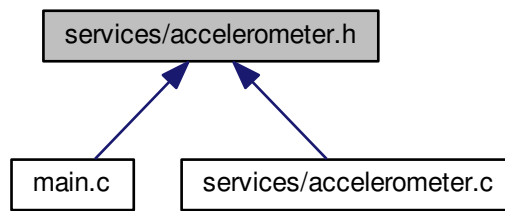
Here is the call graph for this function:



9.69 services/accelerometer.h File Reference

Accelerometer driver code.

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



Macros

- #define [MMA8451Q_ADDRESS](#) (0x1D)
MMA8451Q accelerometer specific address.
- #define [ACCEL_INTERRUPT_SOURCE](#) (0x0C)
MMA8451Q register definitions - see MMA8451Q Accelerometer datasheet for more detail.
- #define [ACCEL_TRANSIENT_CFG](#) (0x1D)
- #define [ACCEL_TRANSIENT_SRC](#) (0x1E)
- #define [ACCEL_TRANSIENT_THS](#) (0x1F)
- #define [ACCEL_TRANSIENT_COUNT](#) (0x20)
- #define [ACCEL_PULSE_CFG](#) (0x21)
- #define [ACCEL_PULSE_THSX](#) (0x23)
- #define [ACCEL_PULSE_THSY](#) (0x24)
- #define [ACCEL_PULSE_THSZ](#) (0x25)
- #define [ACCEL_PULSE_TMLT](#) (0x26)
- #define [ACCEL_PULSE_LTCY](#) (0x27)
- #define [ACCEL_CONTROL_REG1](#) (0x2A)
- #define [ACCEL_CONTROL_REG3](#) (0x2C)
- #define [ACCEL_CONTROL_REG4](#) (0x2D)
- #define [ACCEL_CONTROL_REG5](#) (0x2E)
- #define [ACCEL_INTERRUPT_FLAG](#) (0x20)

Functions

- msg_t [accel_entry](#) (void *arg)
A thread function used to startup and process accelerometer events.
- #define [ACCEL_THREAD_SIZE](#) (256)
- **WORKING_AREA** (accel_thread, [ACCEL_THREAD_SIZE](#))

9.69.1 Detailed Description

Accelerometer driver code.

Author

Tony O'Callaghan

Version

1.0, Wed 26/11/14

Definition in file [accelerometer.h](#).

9.69.2 Macro Definition Documentation

9.69.2.1 `#define ACCEL_THREAD_SIZE (256)`

Declaration of the static thread used for accelerometer processing

Definition at line 16 of file [accelerometer.h](#).

9.69.3 Function Documentation

9.69.3.1 `msg_t accel_entry ( void * arg )`

A thread function used to startup and process accelerometer events.

Parameters

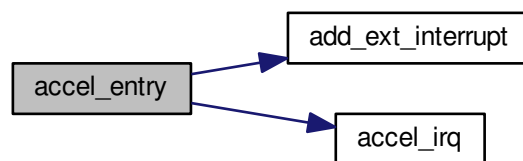
<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 204 of file [accelerometer.c](#).

Here is the call graph for this function:

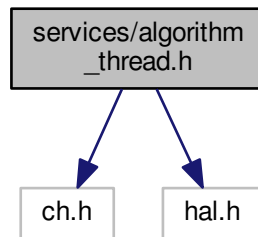


9.70 `services/algorithm_thread.h` File Reference

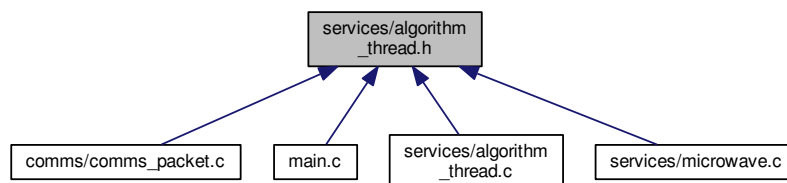
FIFO adapted for use with microwave intruder detection algorithms.

```
#include "ch.h"
#include "hal.h"
```

Include dependency graph for algorithm_thread.h:



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



Macros

- `#define ALGORITHM_THREAD_SIZE 1024`

Enumerations

- enum `microwave_algs_e` { `ALG_DEFAULT` = 0, `ALG_IIR`, `ALG_BANDPASS` }
Enumeration for the MS100 Operational state machine.

Functions

- void `algorithm_thread_init` (void)
Initialize the FIFO.
- void `push_alg_sample` (adcsample_t sample)
Add a microwave sample to the buffer for later analysis.
- void `set_algorithm` (uint8_t type)
Set the algorithm to be used for analysis.
- void `set_sensitivity` (uint8_t sensitivity)
Set the sensitivity to be used for analysis.
- `WORKING_AREA` (wa_algorithm_thread, ALGORITHM_THREAD_SIZE)
Instantiation of the algorithm thread memory resources.
- msg_t `algorithm_entry` (void *arg)
A thread function used for sample processing with various algorithms.

9.70.1 Detailed Description

FIFO adapted for use with microwave intruder detection algorithms.

Author

neil

Definition in file [algorithm_thread.h](#).

9.70.2 Function Documentation

9.70.2.1 msg_t algorithm_entry (void * arg)

A thread function used for sample processing with various algorithms.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

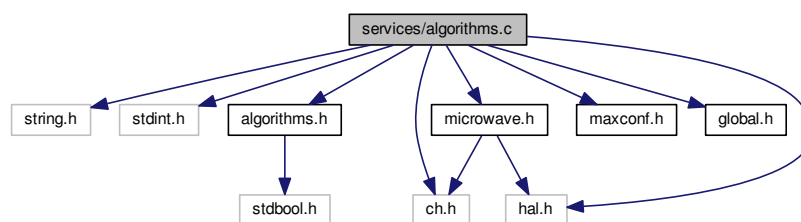
Standard ChibiOS thread return (0 on success)

Definition at line 91 of file [algorithm_thread.c](#).

9.71 services/algorithms.c File Reference

```
#include <string.h>
#include <stdint.h>
#include "ch.h"
#include "hal.h"
#include "algorithms.h"
#include "maxconf.h"
#include "global.h"
#include "microwave.h"
```

Include dependency graph for algorithms.c:



Macros

- `#define PTR_INC(c, l, s)`
– 256_HZ_SAMPLING ifdef = SAMPLE RATE REDUCED FROM 512 HZ TO 256 HZ TO REDUCE THE NUMBER OF INTERRUPTS/SECOND AND THEREBY REDUCE THE EFFECT ON OTHER OPERATIONS BEING CARRIED OUT BY THE CPU, E.G. MICROWAVE DATA LINK. THIS MEANS THAT FILTER COEFFICIENTS/SCALE FACTORS HAVE BEEN MODIFIED FOR THOSE USED WITHIN THE SUMMED_INTEGRATION_ALG ifdef.

Functions

- void [AlgInit](#) (int _algorithm, int _sensitivity)
Declare extern functions /// //////////////////////////////////////.
- float **Filter** (float sample, const float *coeff, const float *scalefactor)
- float **Filter2** (float sample, const float *coeff, const float *scalefactor)
- void [Algorithm](#) (float *sample, float *a, float *b)
Run the selected algorithm for the single sample input.
- bool [SetAlgorithm](#) (int value)
Set the selected algorithm.
- bool [SetSensitivity](#) (int value)
Set the selected algorithm.
- int16_t **SumArray** (int16_t Array[8], int16_t start_element, int16_t stop_element)
- bool **UpdateThresholds** (float sampleA, float sampleB)
- bool [CheckMicrowaveAlarm](#) (uint32_t *alarm, int timeout)
Determine if the microwave alarm should be asserted.
- int [GetUpperThreshold](#) (int path)
Return the upper threshold value.
- int [GetLowerThreshold](#) (int path)
Return the lower threshold value.

Variables

- int16_t **thresh_high_int16** [2]
- int16_t **thresh_low_int16** [2]
- int16_t [threshold_int16](#) [2]
The instantaneous threshold in adaptive alarm thresholding.
- bool **thresh_ok_int16** [2] = { false, false }
- int32_t **thresh_high**
- int32_t **thresh_low**
- int32_t [threshold](#)
The instantaneous threshold in adaptive alarm thresholding [int32].
- bool **thresh_ok**

9.71.1 Detailed Description

Author

Neil, Peter

Definition in file [algorithms.c](#).

9.71.2 Macro Definition Documentation

9.71.2.1 #define PTR_INC(c, l, s)

Value:

```
* (c) = * ((c) + 3) = (s); \
(c)++; \
if ((c) == (1)) { \
    (c) -= 3; \
}
```

– 256_HZ_SAMPLING ifdef = SAMPLE RATE REDUCED FROM 512 HZ TO 256 HZ TO REDUCE THE NUMBER OF INTERRUPTS/SECOND AND THEREBY REDUCE THE EFFECT ON OTHER OPERATIONS BEING CARRIED OUT BY THE CPU, E.G. MICROWAVE DATA LINK. THIS MEANS THAT FILTER COEFFICIENTS/SCALE FACTORS HAVE BEEN MODIFIED FOR THOSE USED WITHIN THE SUMMED_INTEGRATION_ALG ifdef.

– SUMMED_INTEGRATION_ALG ifdef = SUMMED INTEGRATION ALGORITHM IMPLEMENTED IN DECEMBER-FEBRUARY 2015 = ALGORITHMS B/C-V03-V07 IN ALGORITHMS B/C V03 = USED FILTERS OF 0-0.5 Hz (VERY SLOW CHANNEL), 1-2 Hz (SLOW CHANNEL) AND 3-5 Hz (MEDIUM/FAST CHANNELS). USED DIFFERENCE IN SUM OF ENVELOPE (RATHER THAN MEAN OF ENVELOPE) OVER DIFFERENT TIME PERIODS FOR EACH CHANNEL. IN ALGORITHMS B/C V04: MADE THRESHOLDS MORE SENSITIVE FOR VERY SLOW CHANNEL TO BETTER DETECT CRAWLING INTRUSIONS AND LESS SENSITIVE FOR SLOW CHANNEL SUCH THAT PARALLEL INTRUSIONS ARE NOT DETECTED. THE DIFFERENCE BETWEEN ALG B/C IS NOW THAT ALG B IS FOR WALKING/RUNNING DETECTION WHILE ALG C IS FOR CRAWLING/WALKING/RUNNING DETECTION. IN ALGORITHMS B/C V05: (I) CHANGED MEDIUM/FAST CHANNEL FILTER TO 3-8 HZ (FROM 3-5 HZ), (II) TIME AFTER ALARM FOR WHICH FURTHER ALARMS ARE DISABLED INCREASED TO 8 SECONDS, (III) ALARM TYPE IS NOW CLASSIFIED AS A RUNNING, WALKING OR STEALTH INTRUSION, (IV) SENSITIVITIES ADJUSTED SUCH THAT THERE IS A BIG DIFFERENCE FROM SENSITIVITY 0-1 AND 6-7, TO ACCOUNT FOR UNKNOWN SITE CONDITIONS, (V) FOR FAST/MEDIUM CHANNEL ALARMS THAT ARE OF LOW AMPLITUDE AN EXTRA CHECK IS MADE ON THE NUMBER OF ZERO CROSSINGS IN THE SIGNAL OVER A 4 SECOND PERIOD, THEREBY ENABLING BETTER DISTINGUISHING OF WALKING/RUNNING PERSON FROM ANIMAL, (VI) VERY SLOW CHANNEL ALARMS ARE NOW TRIGGERED BY THE AMPLITUDE OF A SINGLE PERIOD OF THE 0-0.5 Hz SIGNAL BEING GREATER THAN THRESHOLD VALUE IN ALGORITHMS B/C V06: (I) RECOMMENDED MOUNTING HEIGHT CHANGED TO 0.95M (FROM 1.1M) TO BETTER DETECT CRAWLING INTRUSIONS CLOSE TO TX/RX OF THE LINK, (II) THRESHOLDS DESENSITISED FOR ALL CHANNELS DUE TO INCREASE IN AMPLITUDE OF ALL SIGNATURES AT THE NEW MOUNTING HEIGHT, (III) NUMBER OF ZERO CROSSINGS IN BOTH MEDIUM/FAST CHANNELS AND SLOW CHANNEL NOW CHECKED TO BETTER DIFFERENTIATE WALKING/RUNNING INTRUSIONS FROM BIRDS AND SLOW INTRUSIONS (SUCH AS SLOW WALKS OR CRAWLS) FROM SMALL ANIMALS SUCH AS RABBITS, FOXES OR BADGERS MOVING ALONG THE GROUND IN ALGORITHMS B/C V07: (I) SENSITIVITY SCALE WIDENED FOR EACH CHANNEL, (II) DIFFERENT SENSITIVITY THRESHOLDS DEFINED FOR HALO/VIGIL SENSORS, (III) INSTALLATION AID WITH MINIMUM RECOMMENDED SENSITIVITY PRINTED AFTER EACH ALARM EVENT, (IV) RELAXED CONDITIONS FOR ALARM CHECK ON FRESNEL ZONE CROSSINGS, (V) REMOVED METHOD OF COPYING PRE-ALARM CH BUFFER WINDOW INTO POST-ALARM CH BUFFER WINDOW AFTER ALARM, (VI) USING DIFFERENT METHOD TO CLASSIFY INTRUSIONS - ONLY VERY SLOW CH ALARMS ARE IMMEDIATELY CLASSIFIED, AFTER 5 SECONDS OTHER CH ALARMS ARE CLASSIFIED DEPENDING ON PEAK SIGNAL LEVEL IN VERY SLOW CH, PEAK SIGNAL LEVEL IN FAST CHANNEL VS SLOW CHANNEL, NUMBER OF ZERO CROSSINGS IN 3-8 HZ BAND – AMSTERDAM_PORT_ALG ifdef = ALGORITHM IMPLEMENTED IN JANUARY 2015 = ALGORITHM D-V01 (ALTHOUGH IT USES ALGORITHM A IN THE EMBEDDED C CODE AT PRESENT AS THERE IS NO ALGORITHM D SELECTABLE IN THE FRONT-END) = DESIGNED TO ALARM ON DETECTION OF SMALL BOATS BUT DISABLE THE LINK WHEN A LARGER FERRY BOAT IS DETECTED. USES 1-5 HZ AND 6-9 HZ CHANNELS, POSSIBLE ALARM TRIGGERED BY A LOW THRESHOLD BEING EXCEEDED IN THE 1-5 HZ CHANNEL - IF IN THE NEXT 10 SECONDS A HIGH THRESHOLD IN THE 1-5 HZ CHANNEL AND A THRESHOLD IN THE 6-9 HZ CHANNEL ARE EXCEEDED THEN THE LINK IS DISABLED. IF THESE THRESHOLDS ARE NOT EXCEEDED IN THE 10 SECOND PERIOD THEN THE ALARM IS TRIGGERED AS A SMALL BOAT HAS BEEN DETECTED. IF THE LINK HAS BEEN DISABLED THEN IT IS NOT RE-ENABLED UNTIL THE SIGNAL PEAK-TO-PEAK LEVEL OVER A 4 SECOND PERIOD RETURNS TO A SIMILAR LEVEL TO THAT WHEN THE ALARM FIRST TRIGGERED. – ENABLE_256_HZ_SUMMED_INT_ALG ifdef = 256HZ SUMMED INTEGRATION ALGORITHM IMPLEMENTED IN FEBRUARY-MARCH 2015 IN ALGORITHMS B/C V08 = (I) USES 256 HZ SAMPLE RATE - REDUCED EFFECT ON OTHER OPERATIONS BEING CARRIED OUT BY CPU, ALLOWS MORE COMPLEX DSP OPERATIONS TO BE PERFORMED. (II) ADDED A 4TH FILTER BAND, WHICH HAS A DIFFERENT PASSBAND FOR DIFFERENT LINK RANGES (20-32Hz/18-30Hz/18-30Hz/6-16Hz/5-10Hz for 0-40m/40-80m/80-120m/120-160m/160-200m), TO ENABLE INSTANT CLASSIFICATION OF RUNNING INTRUSIONS AT ALARM TIME; THE FAST CHANNEL IS NOW EQUAL TO THE OUTPUT OF THIS FILTER BAND. (III) NOW USING SIMPLER CLASSIFICATION PROCESS - STEALTH SPEED INTRUSION = VSLOW CH ALARM, NORMAL SPEED INTRUSION = SLOW/MEDIUM CH ALARMS, FAST SPEED INTRUSION = FAST CH ALARMS. (IV) DUE TO ADDITION OF A 4TH FILTER BAND TO DETECT/CLASSIFY RUNNING INTRUSIONS, THE MEDIUM CH FILTER BAND IS NOW EQUAL TO 3-5 HZ (RATHER THAN 3-8 HZ). SLOW CH FILTER BAND IS NOW EQUAL TO 0.5-3 HZ. (V) ADDED A FURTHER RANGE CALIBRATION FIELD FOR 40-80M, SUCH THAT THERE ARE NOW CALIBRATION FIELDS AT 40M INCREMENTS. (VI) MODIFIED INTEGRATION TECHNIQUE USED

TO CALCULATE THE ENERGY IN FILTER OUTPUT SIGNALS, NOW SUM DIFFERENCE BETWEEN EVERY CONSECUTIVE SIGNAL PEAK, RATHER THAN JUST SUMMING THE LARGEST PEAK IN EACH 1 SECOND WINDOW, THEREFORE GIVING MORE DETAILED INFORMATION ABOUT THE SIZE OF OBJECT INTRUDING THROUGH THE LINK. (VII) NOW CALCULATING NUMBER OF FRESNEL ZONE CROSSINGS IN ALL 4 FILTER OUTPUTS, THEREBY ENABLING FURTHER DETERMINATION OF WHETHER POSSIBLE ALARMS ARE CAUSED BY PERSON OR ANIMAL INTRUDING. NOW CALCULATE ALARM CHECKS FOR VSLOW CH USING CROSSINGS IN SLOW CH, SLOW/MEDIUM CH USING CROSSINGS IN MEDIUM CH, FAST CH USING CROSSINGS IN FAST CH. (VIII) ENVELOPE DIFF VALUES NOW CALCULATED AS FOLLOWS: VSLOW CH = LAST 8 SEC WINDOW - PREVIOUS 8 SEC WINDOW, SLOW/MEDIUM CH = LAST 4 SEC WINDOW - PREVIOUS 4 SEC WINDOW, FAST CH = LAST 2 SEC WINDOW - PREVIOUS 2 SEC WINDOW. (IX) THRESHOLDS FOR EACH CH SET ACCORDING TO SCALED ADJUSTMENT OF SINGLE THRESHOLD VALUE FOR 100M RANGE/SENSITIVITY 1. (X) EXTRA CHECK ADDED FOR POSSIBLE ALARMS IN THE FAST CHANNEL - CHECK IS MADE ON MEDIUM CHANNEL AMPLITUDE, THIS HAS TO EXCEED LEVEL FOR MEDIUM CHANNEL/SENSITIVITY 7 TO TRIGGER ALARM.

Definition at line 64 of file algorithms.c.

9.71.3 Function Documentation

9.71.3.1 void Algnit (int *_algorithm*, int *_sensitivity*)

Declare extern functions /// //////////////////////////////////////.

Initialise the algorithm memory and control parameters

Definition at line 822 of file algorithms.c.

9.71.3.2 void Algorithm (float * *sample*, float * *a*, float * *b*)

Run the selected algorithm for the single sample input.

Parameters

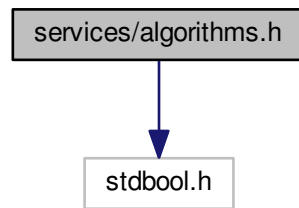
<i>samples</i>	A pointer to a sample
<i>a</i>	A pointer to output samples from path A
<i>b</i>	A pointer to output samples from path B (if exists, null pointer otherwise)
<i>c</i>	A pointer to output samples from path C (if exists, null pointer otherwise)
<i>d</i>	A pointer to output samples from path D (if exists, null pointer otherwise)

Definition at line 1347 of file algorithms.c.

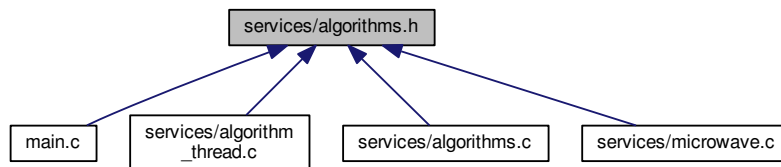
9.72 services/algorithms.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for algorithms.h:



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



Functions

- void **AlgInit** (int _algorithm, int _sensitivity)
Declare extern functions /// //////////////////////////////////////.
- bool **SetAlgorithm** (int value)
Set the selected algorithm.
- bool **SetSensitivity** (int value)
Set the selected algorithm.
- bool **CheckMicrowaveAlarm** (uint32_t *alarm, int timeout)
Determine if the microwave alarm should be asserted.
- int **GetUpperThreshold** (int path)
Return the upper threshold value.
- int **GetLowerThreshold** (int path)
Return the lower threshold value.
- bool **UpdateThresholds** (float sampleA, float sampleB)
- void **Algorithm** (float *sample, float *a, float *b)
Run the selected algorithm for the single sample input.

Variables

- uint32_t **timer_disable_channel_alarms**
Declare preprocessor values ///.
- int32_t **thresh_diff_mean**
- int **ml_threshold_variance_set**

9.72.1 Detailed Description

Author

Neil, Peter

Definition in file [algorithms.h](#).

9.72.2 Function Documentation

9.72.2.1 void AlgInit (int *_algorithm*, int *_sensitivity*)

Declare extern functions /// //////////////////////////////////////.

Initialise the algorithm memory and control parameters

Definition at line 822 of file algorithms.c.

9.72.2.2 void Algorithm (float * *sample*, float * *a*, float * *b*)

Run the selected algorithm for the single sample input.

Parameters

<i>samples</i>	A pointer to a sample
<i>a</i>	A pointer to output samples from path A
<i>b</i>	A pointer to output samples from path B (if exists, null pointer otherwise)
<i>c</i>	A pointer to output samples from path C (if exists, null pointer otherwise)
<i>d</i>	A pointer to output samples from path D (if exists, null pointer otherwise)

Definition at line 1347 of file algorithms.c.

9.72.3 Variable Documentation

9.72.3.1 uint32_t timer_disable_channel_alarms

Declare preprocessor values ///.

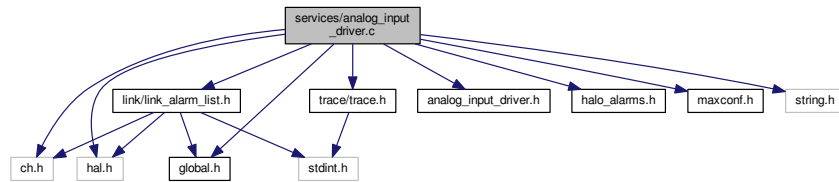
Declare extern variables ///

9.73 services/analog_input_driver.c File Reference

Analog input sampling and alarm service, currently samples analog channels ADCIN0, ADCIN1 and ADCIN2. Alarm generated depending on how many samples are above the required threshold which is 2.3 volts at PSU.

```
#include "ch.h"
#include "hal.h"
#include "global.h"
#include "analog_input_driver.h"
#include "link/link_alarm_list.h"
#include "halo_alarms.h"
#include "trace/trace.h"
#include "maxconf.h"
#include "string.h"
```

Include dependency graph for `analog_input_driver.c`:



Data Structures

- [struct `\_analog\_alarms\_t`](#)

Macros

- `#define AUX_ALARM_THREAD_SIZE (1024)`

Typedefs

- `typedef struct \_analog\_alarms\_t analog_alarms_t`

Functions

- [WORKING_AREA](#) (`aux_input_thread`, `AUX_ALARM_THREAD_SIZE`)
A thread used to sample three analogue input pins for auxiliary alarms.
- `msg_t sample\_aux\_alarm (void *p)`
- `void analog\_input\_init (void)`

9.73.1 Detailed Description

Analog input sampling and alarm service, currently samples analog channels ADCIN0, ADCIN1 and ADCIN2. Alarm generated depending on how many samples are above the required threshold which is 2.3 volts at PSU.

Author

Tony O'Callaghan

Version

1.0, Mon 29/09/14

Definition in file [analog_input_driver.c](#).

9.73.2 Function Documentation

9.73.2.1 `void analog_input_init ( void )`

Startup the analog input sampling thread

Definition at line 288 of file `analog_input_driver.c`.

9.73.2.2 msg_t sample_aux_alarm (void * p)

Note

Read a number of Analog pins based on ADC_AUX_GRP_NUM_CHANNELS Each pin will be sampled, if there are 5 consecutive samples above the threshold an alarm will be set provided there was 5 consecutive low samples to reset the alarm. If Analog Input is enabled in control panel then an osdp message will be sent and relay will be triggered

Definition at line 115 of file analog_input_driver.c.

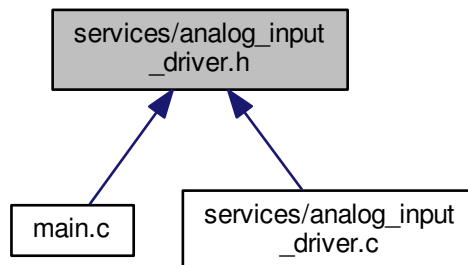
Here is the call graph for this function:



9.74 services/analog_input_driver.h File Reference

Settings used in the analog input sampling algorithm.

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



Macros

- #define **ADC_AUX_GRP_NUM_CHANNELS** (4)
- #define **NUM_OF_ADC_CHANNELS** (1)
- #define **ADC_AUX_BUFF_DEPTH** (1)
- #define **ADC_SAMPLE_TIME_INTERVAL** (99)
- #define **ADC_ALARM_THRESHOLD** (960)
- #define **NUM_ADC_HIGH_SAMPLES** (5)
- #define **NUM_ADC_LOW_SAMPLES** (5)
- #define **ADC_TEST_ARRAY_SIZE** (20)
- #define **ADC_ANALOG_CP_ENABLE**(i) (1 << i)

Functions

- void [analog_input_init](#) (void)

9.74.1 Detailed Description

Settings used in the analog input sampling algorithm.

Author

Tony O'Callaghan

Version

1.0, Mon 29/09/14

Definition in file [analog_input_driver.h](#).

9.74.2 Function Documentation

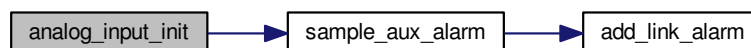
9.74.2.1 void [analog_input_init](#) (void)

Start analog inputs thread

Startup the analog input sampling thread

Definition at line 288 of file [analog_input_driver.c](#).

Here is the call graph for this function:



9.75 services/dz_comms.c File Reference

Communicates between STM32 CPU and lower/upper DZ CPUs to enable: (i) LDZ/UDZ Sensitivity selection via control panel, (ii) LDZ/UDZ steady-state level available via control panel, (iii) Detection of whether LDZ/UDZ sensors are present in a Halo unit, (iv) Single version of DZ firmware (rather than separate versions for LDZ/UDZ), (v) Error alarm if DZ loses initialised settings or cannot be communicated with.

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "stringutils.h"
#include "ch.h"
#include "hal.h"
#include "global.h"
#include "dz_comms.h"
#include "trace/trace.h"
#include "link/link_if.h"
#include "link/link_alarm_list.h"
#include "comms/comms_hub.h"

```

Include dependency graph for dz_comms.c:



Functions

- **WORKING_AREA** (dz_comms_thread, [DZ_COMMS_THREAD_SIZE](#))
- void **dz_comms_firmware_request** (void)
- void **dz_comms_sensor_type** (void)
- void **dz_comms_sensitivity_adjust** (void)
- void **dz_comms_steady_state_read** (void)
- void **dz_comms_counter_poll** (void)
- void **dz_comms_configure** (void)
- void **dz_comms_polling** (void)
- msg_t [dz_comms_entry](#) (void *arg)

A thread function used for DZ comms.

Variables

- uint8_t **dz_tx** [2] = {0x00, 0x00}
- uint8_t **dz_rx** [9] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
- uint8_t **ldz_firmware** [9] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
- uint8_t **udz_firmware** [9] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
- uint8_t **dz_ack** [1] = {0x00}
- uint8_t **ldz_or_udz** [1] = {0x00}
- uint8_t **ldz_tx_counter_poll** [1] = {0x00}
- uint8_t **udz_tx_counter_poll** [1] = {0x00}
- uint8_t **ldz_rx_counter_poll** [1] = {0x00}
- uint8_t **udz_rx_counter_poll** [1] = {0x00}
- uint16_t **ldz_rx_counter_poll_int** = 0
- uint16_t **udz_rx_counter_poll_int** = 0
- uint16_t **ldz_tx_counter_poll_int** = 0
- uint16_t **udz_tx_counter_poll_int** = 0
- uint8_t **ldz_sens_config** [1] = {0x06}
- uint8_t **udz_sens_config** [1] = {0x06}
- uint8_t **ldz_sens_check** [1] = {0x08}
- uint8_t **udz_sens_check** [1] = {0x08}
- uint8_t **ldz_ss** [2] = {0x00, 0x00}
- uint8_t **udz_ss** [2] = {0x00, 0x00}
- uint16_t **ldz_ss_int** = 0
- uint16_t **udz_ss_int** = 0
- int8_t **ldz_polling_count** = 0
- int8_t **udz_polling_count** = 0
- int32_t **ldz_firmware_int** [20] = {0}
- char **ldz_firmware_char** [20]
- char **ldz_firmware_string** [20]
- char * **ldz_firmware_string_ptr**
- int32_t **udz_firmware_int** [20] = {0}
- char **udz_firmware_char** [20]
- char **udz_firmware_string** [20]

- char * **udz_firmware_string_ptr**
- bool **flag_ldz_initialised** = false
- bool **flag_udz_initialised** = false
- bool **flag_ldz_set** = false
- bool **flag_udz_set** = false
- bool **flag_ldz_sens_change** = false
- bool **flag_udz_sens_change** = false
- bool **flag_ldz_comms_error** = false
- bool **flag_udz_comms_error** = false
- bool **flag_ldz_needs_reinitialised** = false
- bool **flag_udz_needs_reinitialised** = false
- bool **ldz** = false
- bool **udz** = false
- uint8_t **dz_present** = 0

A flag indicating the presence of deadzone hardware.

- uint32_t **deadzone_error_alarm** = 0
- systime_t **dz_timeout** = MS2ST(50)
- msg_t **status_l** = RDY_OK
- msg_t **status_u** = RDY_OK

9.75.1 Detailed Description

Communicates between STM32 CPU and lower/upper DZ CPUs to enable: (i) LDZ/UDZ Sensitivity selection via control panel, (ii) LDZ/UDZ steady-state level available via control panel, (iii) Detection of whether LDZ/UDZ sensors are present in a Halo unit, (iv) Single version of DZ firmware (rather than separate versions for LDZ/UDZ), (v) Error alarm if DZ loses initialised settings or cannot be communicated with.

Version

1.0, Thu 10/9/15

Author

Peter Ludlow

Definition in file [dz_comms.c](#).

9.75.2 Function Documentation

9.75.2.1 msg_t dz_comms_entry (void * arg)

A thread function used for DZ comms.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

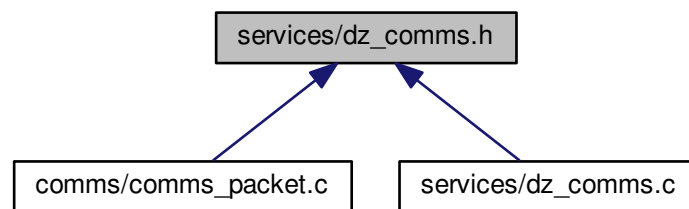
Definition at line 511 of file dz_comms.c.

Here is the call graph for this function:

**9.76 services/dz_comms.h File Reference**

Contains constants for other Dead zone comms C files.

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

**Macros**

- #define **MSP430_ADDRESS** (0x48)
MMA8451Q accelerometer specific address.
- #define **DZ_COUNTER_POLL_REQUEST** (0X43)
- #define **DZ_FIRMWARE_REQUEST** (0X46)
- #define **DZ_SENSITIVITY_REQUEST** (0x53)
- #define **DZ_STEADY_STATE_REQUEST** (0x4C)
- #define **DZ_SENSOR_TYPE** (0x54)
- #define **LDZ_SENSOR** (0x6C)
- #define **UDZ_SENSOR** (0x75)

Functions

- msg_t **dz_comms_entry** (void *arg)
A thread function used for DZ comms.

Variables

- bool **flag_ldz_initialised**
- bool **flag_udz_initialised**
- `#define DZ_COMMS_THREAD_SIZE 512`
- **WORKING_AREA** (dz_comms_thread, [DZ_COMMS_THREAD_SIZE](#))

9.76.1 Detailed Description

Contains constants for other Dead zone comms C files.

Author

Peter Ludlow

Definition in file [dz_comms.h](#).

9.76.2 Macro Definition Documentation

9.76.2.1 `#define DZ_COMMS_THREAD_SIZE 512`

Declaration of the memory resources used in the dz comms thread stack

Definition at line 12 of file dz_comms.h.

9.76.3 Function Documentation

9.76.3.1 `msg_t dz_comms_entry ( void * arg )`

A thread function used for DZ comms.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 511 of file dz_comms.c.

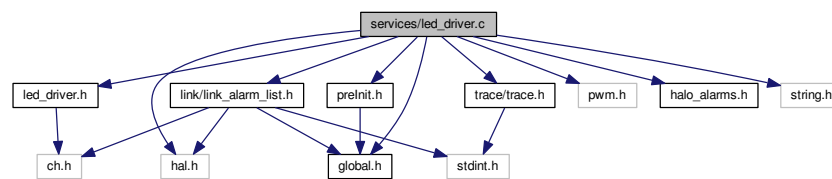
Here is the call graph for this function:



9.77 services/led_driver.c File Reference

```
#include "led_driver.h"
#include "hal.h"
#include "pwm.h"
#include "global.h"
#include "preInit.h"
#include "trace/trace.h"
#include "link/link_alarm_list.h"
#include "halo_alarms.h"
#include "string.h"
```

Include dependency graph for led_driver.c:



Functions

- **WORKING_AREA** (led_thread, LED_THREAD_SIZE)
- msg_t [led_entry](#) (void *arg)

A thread function used to the LED / PWM.

9.77.1 Detailed Description

Author

neil

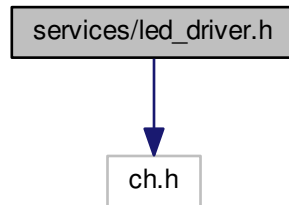
Definition in file [led_driver.c](#).

9.78 services/led_driver.h File Reference

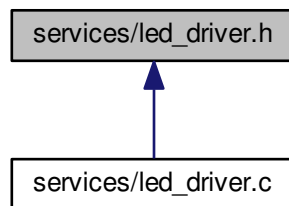
A control thread for the APDS-9700 optical sensor to provide tamper.

```
#include "ch.h"
```

Include dependency graph for led_driver.h:



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



Macros

- `#define LED_THREAD_SIZE 256`

Functions

- **WORKING_AREA** (led_thread, LED_THREAD_SIZE)
- msg_t **led_entry** (void *arg)

A thread function used to the LED / PWM.

9.78.1 Detailed Description

A control thread for the APDS-9700 optical sensor to provide tamper.

Author

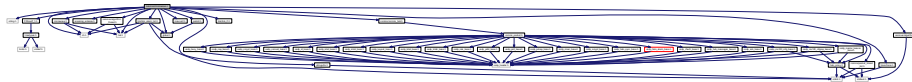
neil

Definition in file [led_driver.h](#).

9.79 services/microwave.c File Reference

```
#include <string.h>
#include "ch.h"
#include "hal.h"
#include "microwave.h"
#include "scanning_antenna.h"
#include "stringutils.h"
#include "FIRcoeff_1.h"
#include "preInit.h"
#include "maxconf.h"
#include "trace/trace.h"
#include "comms/comms_hub.h"
#include "link/link_if.h"
#include "link/link_alarm_list.h"
#include "global.h"
#include "services/algorithm_thread.h"
#include "services/algorithms.h"
```

Include dependency graph for microwave.c:



Macros

- `#define MICROWAVE_ALARM_DELAY 50`
Iteration delay between the microwave alarm thread's forever loop (milliseconds)

Functions

- **WORKING_AREA** (wa_microwave_thread, MICROWAVE_THREAD_SIZE)
- bool `sensitivity_to_threshold` (void)
Set the adaptive threshold according to the configured sensitivity.
- void `start_microwave` (bool pause)
Start the microwave capture and analysis features of the MS100.
- void `stop_microwave` (bool pause)
Stop the microwave capture and analysis features of the MS100.
- msg_t `microwave_thread` (void *arg)
A thread function used for raising microwave and deadzone alarms.

Variables

- int32_t `filtered_signal` = 0
The current filtered signal obtained from current and past samples.
- uint32_t `dir_path_rssi` = 0
The receiver direct path RSSI obtained prior to the transmitter being determined as active.
- bool `tx_active` = FALSE
A flag indicating if a receiver has detected a transmitter signal (always true if a transmitter)
- volatile uint32_t `num_samples` = 0
A counter used to accumulate the number of samples taken.
- uint32_t `microwave_alarm` = 0

- *A flag used to indicate that the microwave alarm has been triggered.*
- uint32_t [possible_microwave_alarm](#) = 0
- *A flag used to indicate that a possible microwave alarm has been triggered.*
- [osdp_sample_buffer_t](#) [l_uwave_samples](#)
- *CCM microwave samples array.*
- bool [is_sample_started](#) = false
- bool [g_link_stable](#)
- *A flag indicating if the signal is stable and ready for use.*
- bool [g_direct_path_sync](#)
- *A flag used to indicate synchronization timing.*
- bool [flag_sample_microwave](#) = false
- uint16_t [g_device_alarms](#)
- *This global variable is used to spoof alarms, useful for testing alarms.*
- bool [thresh_ok](#)
- adcsample_t [direct_buf](#) [DIR_PATH_NUM_SAMPLES]

9.79.1 Function Documentation

9.79.1.1 msg_t microwave_thread (void * arg)

A thread function used for raising microwave and deadzone alarms.

Parameters

arg	Arguments (unused)
---------------------	--------------------

Returns

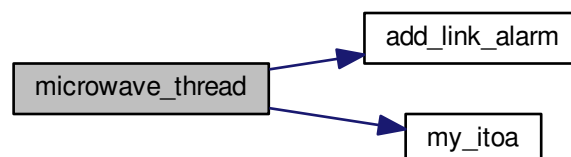
Standard ChibiOS thread return (0 on success)

Assert a flag to indicate to OSDP CPs that a microwave alarm has been generated.

These flags are reset in [comms_hub.c](#) after they have been packetised. The alarm is raised if the alarm conditions are met and the alarm disable countdown timer ([alarm_timeout](#)) is reset to zero.

Definition at line 590 of file [microwave.c](#).

Here is the call graph for this function:



9.79.1.2 bool sensitivity_to_threshold (void)

Set the adaptive threshold according to the configured sensitivity.

Returns

True if succeeded, false otherwise

Definition at line 502 of file microwave.c.

9.79.2 Variable Documentation

9.79.2.1 bool flag_sample_microwave = false

A flag used to indicate whether the microwave_callback function is to be called during the RTC interrupt

Definition at line 101 of file microwave.c.

9.79.2.2 uint16_t g_device_alarms

This global variable is used to spoof alarms, useful for testing alarms.

A bit field used to simulate triggering device alarms.

Definition at line 122 of file microwave.c.

9.79.2.3 osdp_sample_buffer_t l_uwave_samples

CCM microwave samples array.

CCM microwave samples array.

Definition at line 224 of file main.c.

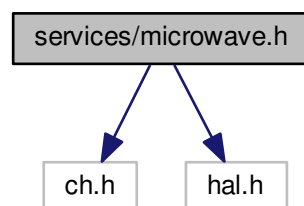
9.80 services/microwave.h File Reference

Functions and timer callbacks used to detect alarms from sampled RF.

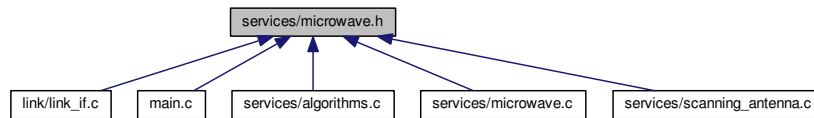
```
#include "ch.h"
```

```
#include "hal.h"
```

Include dependency graph for microwave.h:



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



Macros

- `#define MICROWAVE_THREAD_SIZE 2048`
The microwave thread stack size.

Functions

- bool `sensitivity_to_threshold` (void)
Set the adaptive threshold according to the configured sensitivity.
- void `start_microwave` (bool pause)
Start the microwave capture and analysis features of the MS100.
- void `stop_microwave` (bool pause)
Stop the microwave capture and analysis features of the MS100.
- void `scanning_sample_msg` (void)
- `WORKING_AREA` (wa_microwave_thread, `MICROWAVE_THREAD_SIZE`)
Instantiation of the microwave thread memory resources.
- msg_t `microwave_thread` (void *arg)
A thread function used for raising microwave and deadzone alarms.

Variables

- uint32_t `possible_microwave_alarm`
A flag used to indicate that a possible microwave alarm has been triggered.
- uint32_t `microwave_alarm`
A flag used to indicate that the microwave alarm has been triggered.
- int32_t `raw_signal_scan`
The current raw ADC signal obtained from current and past samples.

9.80.1 Detailed Description

Functions and timer callbacks used to detect alarms from sampled RF.

Author

neil

Definition in file `microwave.h`.

9.80.2 Function Documentation

9.80.2.1 msg_t microwave_thread (void * *arg*)

A thread function used for raising microwave and deadzone alarms.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

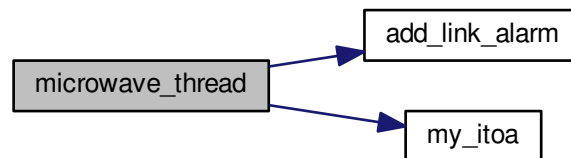
Standard ChibiOS thread return (0 on success)

Assert a flag to indicate to OSDP CPs that a microwave alarm has been generated.

These flags are reset in [comms_hub.c](#) after they have been packetised. The alarm is raised if the alarm conditions are met and the alarm disable countdown timer (`alarm_timeout`) is reset to zero.

Definition at line 590 of file `microwave.c`.

Here is the call graph for this function:



9.80.2.2 `bool sensitivity_to_threshold ( void )`

Set the adaptive threshold according to the configured sensitivity.

Returns

True if succeeded, false otherwise

Definition at line 502 of file `microwave.c`.

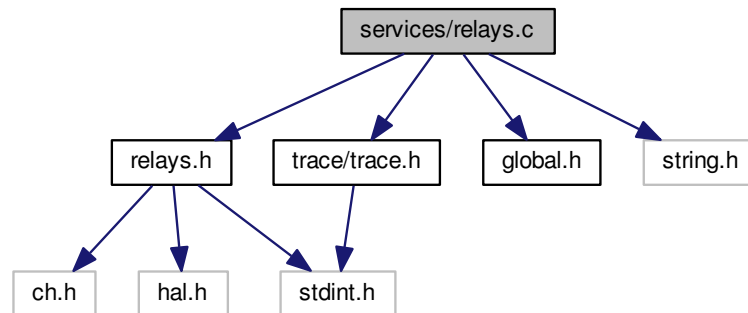
9.81 `services/relays.c` File Reference

```

#include "relays.h"
#include "trace/trace.h"
#include "global.h"
#include "string.h"

```

Include dependency graph for relays.c:



Functions

- **WORKING_AREA** (relay_thread_1, [RELAY_THREAD_SIZE](#))
- **WORKING_AREA** (relay_thread_2, [RELAY_THREAD_SIZE](#))
- msg_t [set_alarm_relay_1](#) (void *arg)
A thread function used for output relay 1 control.
- msg_t [set_alarm_relay_2](#) (void *arg)
A thread function used for output relay 2 control.

9.81.1 Detailed Description

Author

neil

Definition in file [relays.c](#).

9.81.2 Function Documentation

9.81.2.1 msg_t set_alarm_relay_1 (void * arg)

A thread function used for output relay 1 control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 19 of file relays.c.

9.81.2.2 msg_t set_alarm_relay_2 (void * arg)

A thread function used for output relay 2 control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

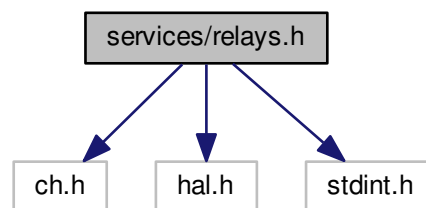
Definition at line 71 of file relays.c.

9.82 services/relays.h File Reference

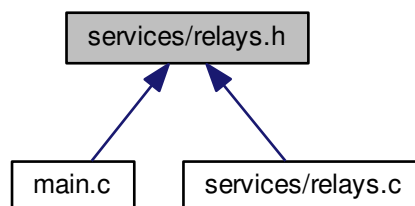
Threads used to operate the output relays.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
```

Include dependency graph for relays.h:



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

**Functions**

- msg_t [set_alarm_relay_1](#) (void *arg)
A thread function used for output relay 1 control.
- msg_t [set_alarm_relay_2](#) (void *arg)

A thread function used for output relay 2 control.

- `#define RELAY_THREAD_SIZE 256`
- **WORKING_AREA** (relay_thread_1, [RELAY_THREAD_SIZE](#))
- **WORKING_AREA** (relay_thread_2, [RELAY_THREAD_SIZE](#))

9.82.1 Detailed Description

Threads used to operate the output relays.

Author

neil

Definition in file [relays.h](#).

9.82.2 Macro Definition Documentation

9.82.2.1 `#define RELAY_THREAD_SIZE 256`

Declaration of the memory resources used for status LED control

Definition at line 21 of file relays.h.

9.82.3 Function Documentation

9.82.3.1 `msg_t set_alarm_relay_1 ( void * arg )`

A thread function used for output relay 1 control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 19 of file relays.c.

9.82.3.2 `msg_t set_alarm_relay_2 ( void * arg )`

A thread function used for output relay 2 control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

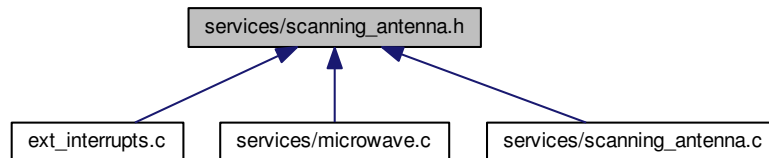
Standard ChibiOS thread return (0 on success)

Definition at line 71 of file relays.c.

9.83 services/scanning_antenna.h File Reference

Contains constants for other Scanning Antenna C files.

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



Functions

- void **increment_scan_angle** (void)
- void **scanning_algorithm** (void)
- msg_t **scanning_beam_entry** (void *arg)
A thread function used for scanning beam control.
- #define **SCANNING_BEAM_THREAD_SIZE** 1024
- Thread * **tp_sb**
- int **focal_point**
- **WORKING_AREA** (scanning_beam_thread, **SCANNING_BEAM_THREAD_SIZE**)

9.83.1 Detailed Description

Contains constants for other Scanning Antenna C files.

Author

Peter Ludlow

Definition in file [scanning_antenna.h](#).

9.83.2 Macro Definition Documentation

9.83.2.1 #define SCANNING_BEAM_THREAD_SIZE 1024

Declaration of the memory resources used the scanning beam thread stack

Definition at line 22 of file scanning_antenna.h.

9.83.3 Function Documentation

9.83.3.1 msg_t scanning_beam_entry (void * arg)

A thread function used for scanning beam control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 235 of file scanning_antenna.c.

9.83.4 Variable Documentation

9.83.4.1 Thread* tp_sb

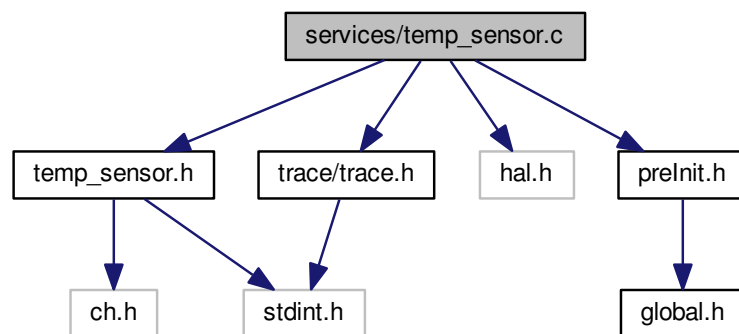
Declaration of the static thread used for scanning beam thread

Definition at line 233 of file scanning_antenna.c.

9.84 services/temp_sensor.c File Reference

```
#include "temp_sensor.h"
#include "trace/trace.h"
#include "hal.h"
#include "preInit.h"
```

Include dependency graph for temp_sensor.c:



Functions

- `msg_t temperature_init (void)`
Temperature Sensor Initialisation.
- `void temperature_read (int32_t *temp)`
Temperature Sensor Read.
- `uint32_t temperature_alarm (int32_t temp)`
Temperature Sensor Alarm.
- `void temperature_reset (int32_t temp)`
Temperature status reset.

9.84.1 Detailed Description

Author

neil

Definition in file [temp_sensor.c](#).

9.84.2 Function Documentation

9.84.2.1 `uint32_t temperature_alarm ( int32_t temp )`

Temperature Sensor Alarm.

Note

Determine if a temperature alarm should be changed

Returns

A coded bitfield indicating which temperature alarm has been triggered

Definition at line 161 of file `temp_sensor.c`.

9.84.2.2 `msg_t temperature_init ( void )`

Temperature Sensor Initialisation.

Note

Sends config data to the sensor and sets the alarm parameters

Returns

Returns RDY_OK if successful

Definition at line 32 of file `temp_sensor.c`.

Here is the call graph for this function:



9.84.2.3 `void temperature_read ( int32_t * temp )`

Temperature Sensor Read.

Note

Reads the temperature and status of the sensor

Definition at line 103 of file `temp_sensor.c`.

9.84.2.4 void temperature_reset (int32_t temp)

Temperature status reset.

Parameters

<i>temp</i>	Temperature used to initialise the variables
-------------	--

Definition at line 239 of file temp_sensor.c.

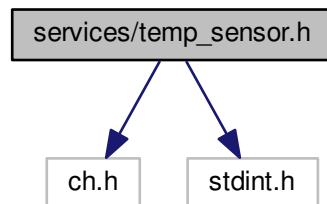
9.85 services/temp_sensor.h File Reference

API functions for reading and manipulating the board temperature.

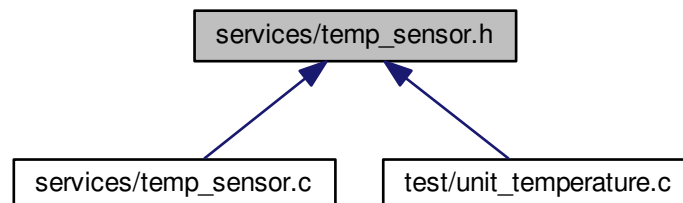
```
#include "ch.h"
```

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

Include dependency graph for temp_sensor.h:



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



Macros

- #define `TEMP_OUT_ADDR` 0x90
Temperature Sensor i2c address.
- #define `TEMP_TEMP_REG` 0x00
Temperature Sensor Temp Register Address.

- `#define TEMP_CONFIG_REG 0x01`
Temperature Sensor Configuration Register Address.
- `#define TEMP_TLOW_REG 0x02`
Temperature Sensor TLow Register Address.
- `#define TEMP_THIGH_REG 0x03`
Temperature Sensor THigh Register Address.
- `#define TEMP_PERIOD 5`
The period between temperature measurements (seconds)
- `#define TEMP_WINDOW 480`
The window within which the temperature gradient is monitored (seconds)
- `#define TEMP_WINDOW_NUM (TEMP_WINDOW/TEMP_PERIOD)`
The number of samples obtained within a window.
- `#define TEMP_MAX_ABS_DIFF (10<<4)`
The maximum absolute temperature difference between adjacent samples.
- `#define TEMP_MAX_ABS_GRADIENT (10<<4)`
The maximum absolute temperature gradient (degrees per window)
- `#define TEMP_MAX_THRESHOLD (65<<4)`
The maximum temperature threshold.
- `#define TEMP_MIN_THRESHOLD (-20<<4)`
The minimum temperature threshold.
- `#define TEMP_RANGE_THRESHOLD (TEMP_MAX_THRESHOLD-TEMP_MIN_THRESHOLD)`
The operating temperature range.
- `#define TEMP_THRESHOLD_BUFFER (5<<4)`
The temperature buffer before a threshold alarm is reactivated.

- `#define TEMP_ALARM_MAX 1`
- `#define TEMP_ALARM_MIN 2`
- `#define TEMP_ALARM_GRADIENT 4`
- `#define TEMP_ALARM_DIFF 8`

Functions

- `msg_t temperature_init (void)`
Temperature Sensor Initialisation.
- `void temperature_read (int32_t *temp)`
Temperature Sensor Read.
- `uint32_t temperature_alarm (int32_t temp)`
Temperature Sensor Alarm.
- `void temperature_reset (int32_t temp)`
Temperature status reset.

9.85.1 Detailed Description

API functions for reading and manipulating the board temperature.

Author

neil

Definition in file [temp_sensor.h](#).

9.85.2 Macro Definition Documentation

9.85.2.1 `#define TEMP_ALARM_MAX 1`

Temperature alarm flags (bitfield)

Definition at line 67 of file temp_sensor.h.

9.85.3 Function Documentation

9.85.3.1 `uint32_t temperature_alarm ( int32_t temp )`

Temperature Sensor Alarm.

Note

Determine if a temperature alarm should be changed

Returns

A coded bitfield indicating which temperature alarm has been triggered

Definition at line 161 of file temp_sensor.c.

9.85.3.2 `msg_t temperature_init ( void )`

Temperature Sensor Initialisation.

Note

Sends config data to the sensor and sets the alarm parameters

Returns

Returns RDY_OK if successful

Definition at line 32 of file temp_sensor.c.

Here is the call graph for this function:



9.85.3.3 `void temperature_read ( int32_t * temp )`

Temperature Sensor Read.

Note

Reads the temperature and status of the sensor

Definition at line 103 of file temp_sensor.c.

9.85.3.4 void temperature_reset (int32_t temp)

Temperature status reset.

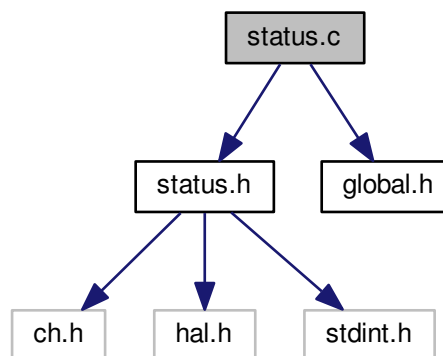
Parameters

<i>temp</i>	Temperature used to initialise the variables
-------------	--

Definition at line 239 of file temp_sensor.c.

9.86 status.c File Reference

```
#include "status.h"
#include "global.h"
Include dependency graph for status.c:
```



Functions

- **WORKING_AREA** (status_thread, [STATUS_THREAD_SIZE](#))
- msg_t [status_entry](#) (void *p)
A thread function used for status LED and watchdog control.

Variables

- [ms100_state_e g_ms100_state](#)
The current control FSM state.
- Semaphore [g_spimux_sem](#)
A semaphore used to guard access to the SPI MUX.

9.86.1 Detailed Description

Author

neil

Definition in file [status.c](#).

9.86.2 Function Documentation

9.86.2.1 msg_t status_entry (void * arg)

A thread function used for status LED and watchdog control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

Standard ChibiOS thread return (0 on success)

Definition at line 25 of file status.c.

9.86.3 Variable Documentation

9.86.3.1 ms100_state_e g_ms100_state

The current control FSM state.

The device state.

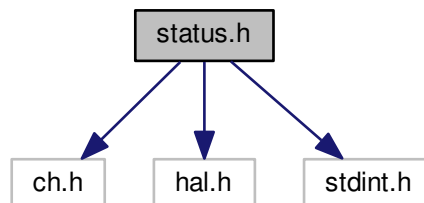
Definition at line 166 of file main.c.

9.87 status.h File Reference

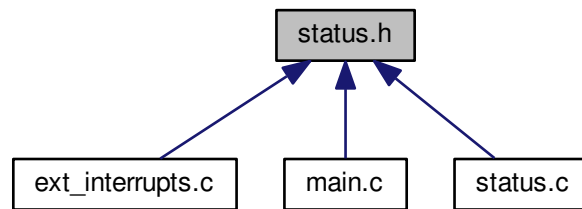
A thread used to operate the status LED on the rear of the board and the HW watchdog.

```
#include "ch.h"
#include "hal.h"
#include <stdint.h>
```

Include dependency graph for status.h:



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



Functions

- `msg_t status_entry (void *arg)`
A thread function used for status LED and watchdog control.
- `#define STATUS_THREAD_SIZE 256`
- **WORKING_AREA** (status_thread, STATUS_THREAD_SIZE)

9.87.1 Detailed Description

A thread used to operate the status LED on the rear of the board and the HW watchdog.

Author

neil

Definition in file [status.h](#).

9.87.2 Macro Definition Documentation

9.87.2.1 `#define STATUS_THREAD_SIZE 256`

Declaration of the static thread used for status LED and watchdog control

Definition at line 24 of file status.h.

9.87.3 Function Documentation

9.87.3.1 `msg_t status_entry ( void * arg )`

A thread function used for status LED and watchdog control.

Parameters

<i>arg</i>	Arguments (unused)
------------	--------------------

Returns

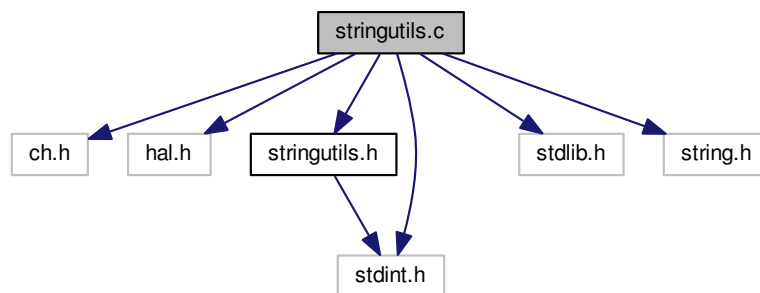
Standard ChibiOS thread return (0 on success)

Definition at line 25 of file status.c.

9.88 stringutils.c File Reference

```
#include <ch.h>
#include <hal.h>
#include "stringutils.h"
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for stringutils.c:



Functions

- int [isdigit](#) (int c)
Check if a character is a numeric ASCII number.
- int [islower](#) (int c)
Check if a character is a lower case alphabetic ASCII character.
- int [isupper](#) (int c)
Check if a character is a upper case alphabetic ASCII character.
- int [isalpha](#) (int c)
Check if a character is an alphabetic ASCII character.
- int [isalnum](#) (int c)
Check if a character is a numeric or alphabetic ASCII character.
- int [my_atoi](#) (const char *s)
Function to convert a string to an integer.
- float [my_atof](#) (const char *s)
Function to convert a string to a float.
- char * [my_itoa](#) (int32_t val, int base)
Function to convert a number to a formatted ASCII string.

- char [my_itochar](#) (int val, int base)
Function to convert a number to a formatted ASCII character.
- uint32_t [str2hex](#) (char *str, int len)
Convert a hex ASCII string to a 32-bit hex number.
- void [hex2str](#) (uint32_t hex, char *str)
Convert a 32-bit hex number to an 8 character capitalised ASCII string.

Variables

- const char * [calendar_day](#) [7]
An array of strings defining the days of the week.
- const char * [calendar_month](#) [12]
An array of strings defining the months of the year.
- const char * [temp_frac](#) [16]
A temperature gradient.

9.88.1 Detailed Description

Author

neil

Definition in file [stringutils.c](#).

9.88.2 Function Documentation

9.88.2.1 void hex2str (uint32_t hex, char * str)

Convert a 32-bit hex number to an 8 character capitalised ASCII string.

Parameters

<i>hex</i>	A 32-bit unsigned integer
<i>str</i>	A C-string containing a zero-padded hex string in ASCII

Definition at line 186 of file stringutils.c.

9.88.2.2 int isalnum (int c) [inline]

Check if a character is a numeric or alphabetic ASCII character.

Parameters

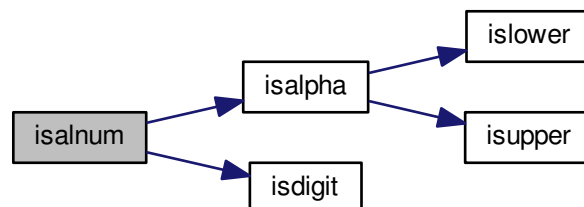
<i>c</i>	Character to be tested
----------	------------------------

Returns

0 if true

Definition at line 81 of file stringutils.c.

Here is the call graph for this function:

**9.88.2.3 int isalpha (int c) [inline]**

Check if a character is an alphabetic ASCII character.

Parameters

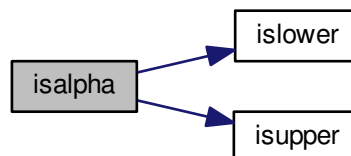
<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 74 of file stringutils.c.

Here is the call graph for this function:

**9.88.2.4 int isdigit (int c) [inline]**

Check if a character is a numeric ASCII number.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 53 of file stringutils.c.

9.88.2.5 `int islower ( int c ) [inline]`

Check if a character is a lower case alphabetic ASCII character.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 60 of file stringutils.c.

9.88.2.6 `int isupper ( int c ) [inline]`

Check if a character is a upper case alphabetic ASCII character.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 67 of file stringutils.c.

9.88.2.7 `float my_atof ( const char * s )`

Function to convert a string to a float.

Parameters

<code>s</code>	String to be converted
----------------	------------------------

Returns

Floating-point integer result

Definition at line 99 of file stringutils.c.

9.88.2.8 `int my_atoi ( const char * s )`

Function to convert a string to an integer.

Parameters

<i>s</i>	String to be converted
----------	------------------------

Returns

Integer result

Definition at line 88 of file stringutils.c.

Here is the call graph for this function:

**9.88.2.9** `char* my_itoa ( int32_t val, int base )`

Function to convert a number to a formatted ASCII string.

Parameters

<i>val</i>	Number to be converted
<i>base</i>	Base number to use for conversion

Returns

String array containing formatted number

Definition at line 124 of file stringutils.c.

9.88.2.10 `char my_itochar ( int val, int base )`

Function to convert a number to a formatted ASCII character.

Parameters

<i>val</i>	Number to be converted
<i>base</i>	Base number to use for conversion

Returns

ASCII code for number

Definition at line 150 of file stringutils.c.

9.88.2.11 `uint32_t str2hex ( char * str, int len )`

Convert a hex ASCII string to a 32-bit hex number.

Parameters

<i>str</i>	A C-string containing a zero-padded hex string in ASCII
<i>len</i>	The length of the input string

Returns

The hex string converted to a 32-bit unsigned integer

Definition at line 159 of file stringutils.c.

9.88.3 Variable Documentation**9.88.3.1 const char* calendar_day[7]****Initial value:**

```
= {
    "Sunday" , "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"
}
```

An array of strings defining the days of the week.

Definition at line 16 of file stringutils.c.

9.88.3.2 const char* calendar_month[12]**Initial value:**

```
= {
    "January", "February", "March", "April",
    "May", "June", "July", "August",
    "September", "October", "November", "December"
}
```

An array of strings defining the months of the year.

Definition at line 22 of file stringutils.c.

9.88.3.3 const char* temp_frac[16]**Initial value:**

```
=
{
    ".0",
    ".0625",
    ".125",
    ".1875",
    ".25",
    ".3125",
    ".375",
    ".4375",
    ".5",
    ".5625",
    ".625",
    ".6875",
    ".75",
    ".8125",
    ".875",
    ".9375",
}
```

A temperature gradient.

An array used to provide a string for 1/16 fractions.

Definition at line 29 of file stringutils.c.

Variables

- const char * [calendar_day](#) [7]

An array of strings defining the days of the week.

- const char * [calendar_month](#) [12]

An array of strings defining the months of the year.

- const char * [temp_frac](#) [16]

An array used to provide a string for 1/16 fractions.

9.89.1 Detailed Description

A thread used to provide an interface to the VaultIC device and perform the required cryptographic actions.

Author

neil

Definition in file [stringutils.h](#).

9.89.2 Function Documentation

9.89.2.1 void hex2str (uint32_t *hex*, char * *str*)

Convert a 32-bit hex number to an 8 character capitalised ASCII string.

Parameters

<i>hex</i>	A 32-bit unsigned integer
<i>str</i>	A C-string containing a zero-padded hex string in ASCII

Definition at line 186 of file stringutils.c.

9.89.2.2 int isalnum (int *c*) [inline]

Check if a character is a numeric or alphabetic ASCII character.

Parameters

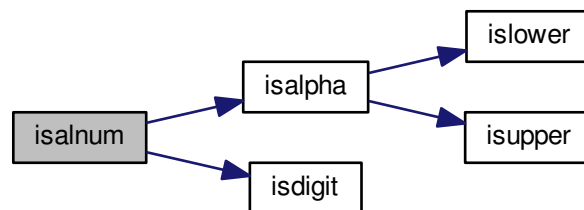
<i>c</i>	Character to be tested
----------	------------------------

Returns

0 if true

Definition at line 81 of file stringutils.c.

Here is the call graph for this function:

**9.89.2.3 int isalpha (int c) [inline]**

Check if a character is an alphabetic ASCII character.

Parameters

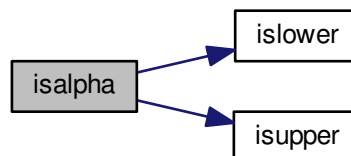
<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 74 of file stringutils.c.

Here is the call graph for this function:

**9.89.2.4 int isdigit (int c) [inline]**

Check if a character is a numeric ASCII number.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 53 of file stringutils.c.

9.89.2.5 `int islower ( int c )` `[inline]`

Check if a character is a lower case alphabetic ASCII character.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 60 of file stringutils.c.

9.89.2.6 `int isupper ( int c )` `[inline]`

Check if a character is a upper case alphabetic ASCII character.

Parameters

<code>c</code>	Character to be tested
----------------	------------------------

Returns

0 if true

Definition at line 67 of file stringutils.c.

9.89.2.7 `float my_atof ( const char * s )`

Function to convert a string to a float.

Parameters

<code>s</code>	String to be converted
----------------	------------------------

Returns

Floating-point integer result

Definition at line 99 of file stringutils.c.

9.89.2.8 `int my_atoi ( const char * s )`

Function to convert a string to an integer.

Parameters

<i>s</i>	String to be converted
----------	------------------------

Returns

Integer result

Definition at line 88 of file stringutils.c.

Here is the call graph for this function:

**9.89.2.9** `char* my_itoa ( int32_t val, int base )`

Function to convert a number to a formatted ASCII string.

Parameters

<i>val</i>	Number to be converted
<i>base</i>	Base number to use for conversion

Returns

String array containing formatted number

Definition at line 124 of file stringutils.c.

9.89.2.10 `char my_itochar ( int val, int base )`

Function to convert a number to a formatted ASCII character.

Parameters

<i>val</i>	Number to be converted
<i>base</i>	Base number to use for conversion

Returns

ASCII code for number

Definition at line 150 of file stringutils.c.

9.89.2.11 `uint32_t str2hex ( char * str, int len )`

Convert a hex ASCII string to a 32-bit hex number.

Parameters

<i>str</i>	A C-string containing a zero-padded hex string in ASCII
<i>len</i>	The length of the input string

Returns

The hex string converted to a 32-bit unsigned integer

Definition at line 159 of file stringutils.c.

9.89.3 Variable Documentation**9.89.3.1** `const char* temp_frac[16]`

An array used to provide a string for 1/16 fractions.

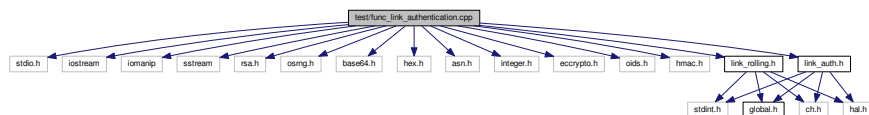
An array used to provide a string for 1/16 fractions.

Definition at line 29 of file stringutils.c.

9.90 test/func_link_authentication.cpp File Reference

```
#include <stdio.h>
#include <iostream>
#include <iomanip>
#include <sstream>
#include "rsa.h"
#include "osrng.h"
#include "base64.h"
#include "hex.h"
#include "asn.h"
#include "integer.h"
#include "eccrypto.h"
#include "oids.h"
#include "hmac.h"
#include "link_auth.h"
#include "link_rolling.h"
```

Include dependency graph for func_link_authentication.cpp:

**Data Structures**

- [struct _rsa_priv_t](#)

Macros

- `#define func_assert(message, test) do { if (!test) { cout << message << endl; return false; } } while (0)`
- `#define func_run_test(test)`

Typedefs

- typedef [_rsa_priv_t](#) [rsa_priv_t](#)

Functions

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

Variables

- int32_t **tests_run** = 0

9.90.1 Detailed Description

Author

neil

Definition in file [func_link_authentication.cpp](#).

9.90.2 Macro Definition Documentation

9.90.2.1 #define func_run_test(test)

Value:

```
do { bool ret = test(); tests_run++; \
    if (ret==false) return false; } while (0)
```

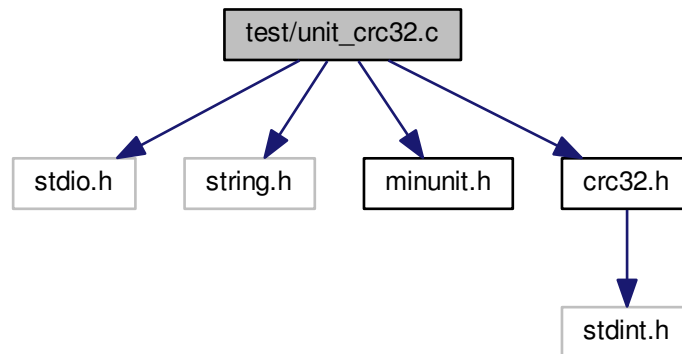
Definition at line 35 of file [func_link_authentication.cpp](#).

9.91 test/unit_crc32.c File Reference

Unit testing for CRC32 functions.

```
#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "crc32.h"
```

Include dependency graph for `unit_crc32.c`:



Functions

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

Variables

- `int tests_run = 0`

9.91.1 Detailed Description

Unit testing for CRC32 functions.

Author

neil

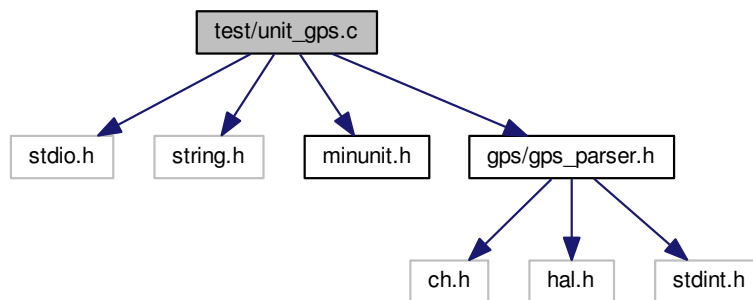
Definition in file [unit_crc32.c](#).

9.92 test/unit_gps.c File Reference

Unit testing for the GPS parser.

```
#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "gps/gps_parser.h"
```

Include dependency graph for unit_gps.c:



Functions

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

Variables

- int **tests_run** = 0

9.92.1 Detailed Description

Unit testing for the GPS parser.

Author

neil

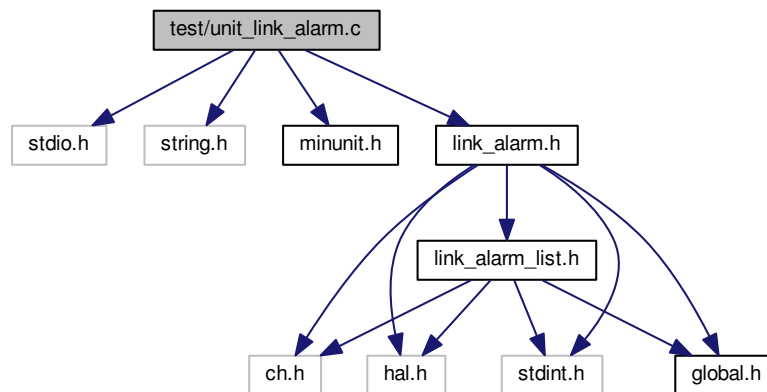
Definition in file [unit_gps.c](#).

9.93 test/unit_link_alarm.c File Reference

Unit testing for link alarm functions.

```
#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "link_alarm.h"
```

Include dependency graph for `unit_link_alarm.c`:



Functions

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

Variables

- `int tests_run = 0`

9.93.1 Detailed Description

Unit testing for link alarm functions.

Author

neil

Definition in file [unit_link_alarm.c](#).

9.94 test/unit_link_auth.c File Reference

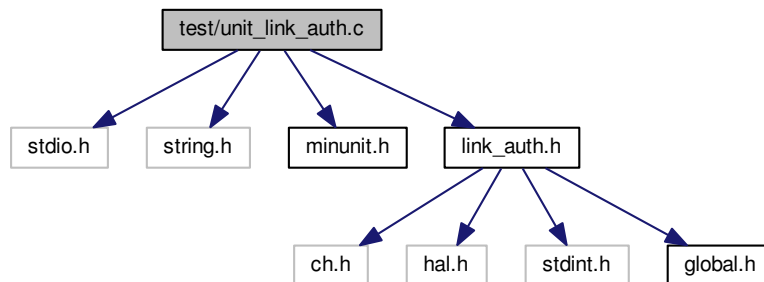
Unit testing for link authentication functions.

```

#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "link_auth.h"

```

Include dependency graph for unit_link_auth.c:



Functions

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

Variables

- int **tests_run** = 0

9.94.1 Detailed Description

Unit testing for link authentication functions.

Author

neil

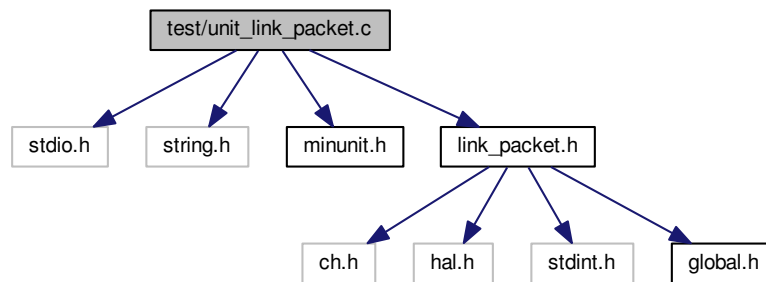
Definition in file [unit_link_auth.c](#).

9.95 test/unit_link_packet.c File Reference

Unit testing for link packet functions.

```
#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "link_packet.h"
```

Include dependency graph for `unit_link_packet.c`:



Functions

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

Variables

- `int tests_run = 0`

9.95.1 Detailed Description

Unit testing for link packet functions.

Unit testing for link rolling functions.

Author

neil

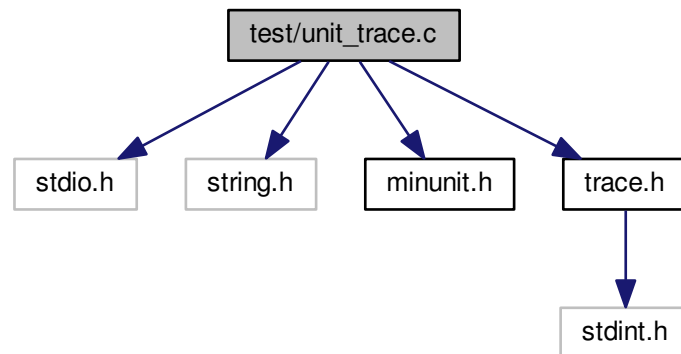
Definition in file [unit_link_packet.c](#).

9.96 test/unit_trace.c File Reference

Unit testing for the trace FIFO.

```
#include <stdio.h>
#include <string.h>
#include "minunit.h"
#include "trace.h"
```

Include dependency graph for unit_trace.c:



Functions

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

Variables

- int **tests_run** = 0

9.96.1 Detailed Description

Unit testing for the trace FIFO.

Author

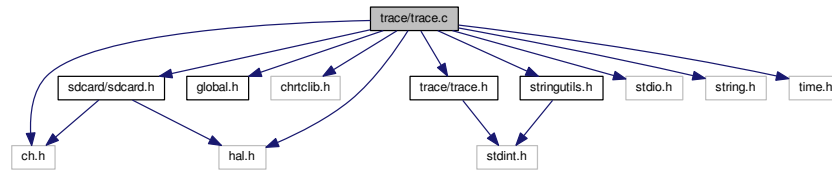
neil

Definition in file [unit_trace.c](#).

9.97 trace/trace.c File Reference

```
#include "ch.h"
#include "hal.h"
#include "global.h"
#include "chrtclib.h"
#include "trace/trace.h"
#include "sdcard/sdcard.h"
#include "stringutils.h"
#include <stdio.h>
#include <string.h>
#include <time.h>
```

Include dependency graph for trace.c:



Functions

- [WORKING_AREA](#) (trace_thread, TRACE_STACK_SIZE)

The trace thread.

- char * [get_rtc_time](#) (void)
- [msg_info_t](#) * [get_msg](#) (void)
- [msg_info_t](#) * [get_msg_sd](#) (void)
- void [remove_msg](#) (void)

Remove a message from the trace buffer.

- void [remove_msg_sd](#) (void)
- void [add_msg_impl](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)
- void [add_msg](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)
- void [add_msg_s](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)
- int [trace_write](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)

Write data to trace message storage.

- int [trace_write_s](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)

Write data to trace message storage.

- void [trace_init](#) ()
- void [trace_stop](#) (void)

Stop the tracing functionality.

- [msg_t](#) [trace_entry](#) (void *p)

The trace thread function.

Variables

- [time_t](#) [g_timestamp_start](#)

The system start time.

- [time_t](#) [g_timestamp](#)

A periodically updated timestamp value from the RTC.

- bool [g_fw_download_in_progress](#)
- [int32_t](#) [fifo_rdp_ptr_cp](#)
- [int32_t](#) [fifo_wr_ptr](#)
- [int32_t](#) [fifo_depth_cp](#)
- volatile [msg_info_t](#) [fifo](#)[MAX_MSG_FIFO_ENTRIES] [__attribute__\(\(section\(".ccm_trace"\)\)\)](#)

9.97.1 Detailed Description

Author

neil

Definition in file [trace.c](#).

9.97.2 Function Documentation

9.97.2.1 `volatile msg_info_t fifo [MAX_MSG_FIFO_ENTRIES] __attribute__ ( (section(".ccm_trace")) )`

An array of FIFO entries and the associated controls

9.97.2.2 `msg_t trace_entry ( void * p )`

The trace thread function.

Parameters

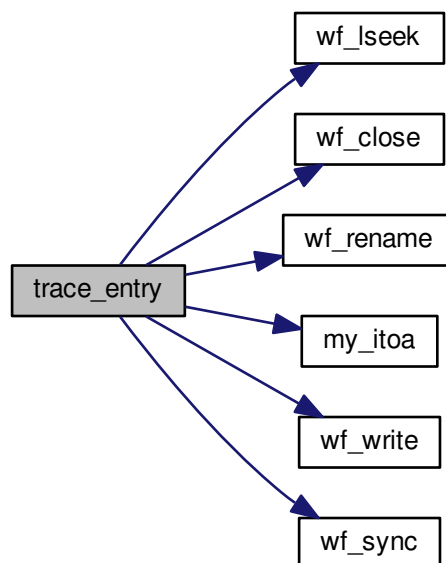
<i>p</i>	Ignored
----------	---------

Returns

A standard ChibiOS thread return structure

Definition at line 308 of file [trace.c](#).

Here is the call graph for this function:



9.97.2.3 `int trace_write ( trace_level_e level, const char * file, const char * function, int line, const char * msg, int length )`

Write data to trace message storage.

Parameters

<i>msg</i>	The buffer array of bytes to be transmitted
<i>length</i>	The length of the message to be stored

Returns

1 on success, 0 otherwise

Definition at line 260 of file trace.c.

9.97.2.4 `int trace_write_s ( trace_level_e level, const char * file, const char * function, int line, const char * msg, int length )`

Write data to trace message storage.

Parameters

<i>msg</i>	The buffer array of bytes to be transmitted
<i>length</i>	The length of the message to be stored

Returns

1 on success, 0 otherwise

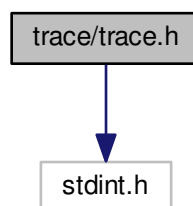
Definition at line 273 of file trace.c.

9.98 trace/trace.h File Reference

A C library for recording trace strings in RAM and SD card storage.

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

Include dependency graph for trace.h:



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



Data Structures

- struct [_msg_info_t](#)

A struct used to define the position and time of the message origin.

Macros

- #define **MAX_FILE_LENGTH** 32
- #define **MAX_FUNCTION_LENGTH** 32
- #define **MAX_MEM_DEPTH** 32768
- #define **MAX_MSG_LENGTH** 52
- #define **FIFO_ENTRY_SIZE** (76 + MAX_MSG_LENGTH)
- #define **MAX_MSG_FIFO_ENTRIES** (MAX_MEM_DEPTH / FIFO_ENTRY_SIZE)
- #define **TRACELOG**(l, m) [trace_write](#)(l, __FILE__, __FUNCTION__, __LINE__, m, strlen(m))
- #define **TRACELOGS**(l, m) [trace_write_s](#)(l, __FILE__, __FUNCTION__, __LINE__, m, strlen(m))
- #define **TRACE_STACK_SIZE** 2048

Typedefs

- typedef struct [_msg_info_t](#) **msg_info_t**

Enumerations

- enum [trace_level_e](#) {
TRACE_DISABLED = 0, **TRACE_ERROR**, **TRACE_WARNING**, **TRACE_NOTICE**,
TRACE_DEBUG }

The trace levels that can be configured.

Functions

- [WORKING_AREA](#) (trace_thread, TRACE_STACK_SIZE)
Specify external linkage for trace_thread.
- char * [get_rtc_time](#) (void)
- [msg_info_t](#) * [get_msg](#) (void)
- int [trace_write](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)
Write data to trace message storage.
- int [trace_write_s](#) ([trace_level_e](#) level, const char *file, const char *function, int line, const char *msg, int length)
Write data to trace message storage.
- void [remove_msg](#) (void)
Remove a message from the trace buffer.
- void [trace_init](#) (void)
An initialisation function that configures the trace thread.
- void [trace_stop](#) (void)
Stop the tracing functionality.
- [msg_t](#) [trace_entry](#) (void *p)
The trace thread function.

9.98.1 Detailed Description

A C library for recording trace strings in RAM and SD card storage.

Author

neil

Definition in file [trace.h](#).

9.98.2 Function Documentation

9.98.2.1 `msg_t trace_entry ( void * p )`

The trace thread function.

Parameters

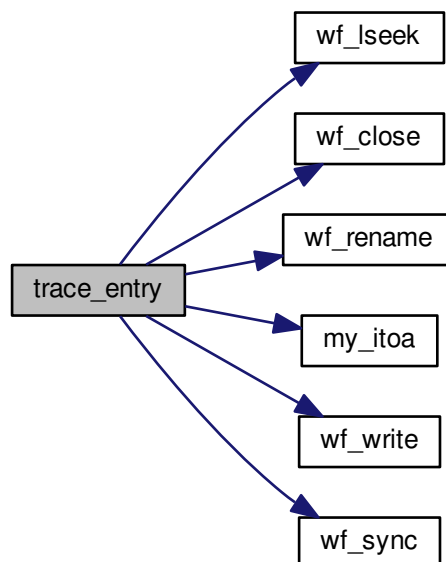
<i>p</i>	Ignored
----------	---------

Returns

A standard ChibiOS thread return structure

Definition at line 308 of file `trace.c`.

Here is the call graph for this function:



9.98.2.2 `int trace_write ( trace_level_e level, const char * file, const char * function, int line, const char * msg, int length )`

Write data to trace message storage.

Parameters

<i>msg</i>	The buffer array of bytes to be transmitted
<i>length</i>	The length of the message to be stored

Returns

1 on success, 0 otherwise

Definition at line 260 of file trace.c.

9.98.2.3 `int trace_write_s ( trace_level_e level, const char * file, const char * function, int line, const char * msg, int length )`

Write data to trace message storage.

Parameters

<i>msg</i>	The buffer array of bytes to be transmitted
<i>length</i>	The length of the message to be stored

Returns

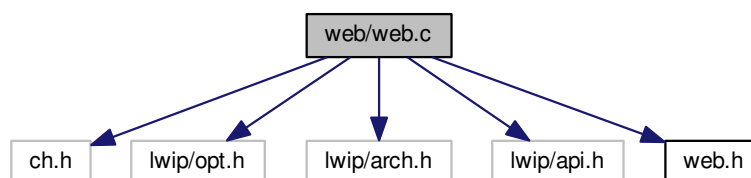
1 on success, 0 otherwise

Definition at line 273 of file trace.c.

9.99 web/web.c File Reference

HTTP server wrapper thread code.

```
#include "ch.h"
#include "lwip/opt.h"
#include "lwip/arch.h"
#include "lwip/api.h"
#include "web.h"
Include dependency graph for web.c:
```



9.99.1 Detailed Description

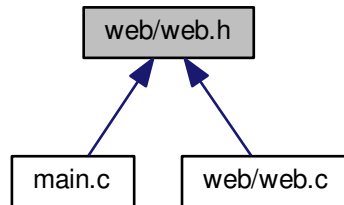
HTTP server wrapper thread code.

Definition in file [web.c](#).

9.100 web/web.h File Reference

HTTP server wrapper thread macros and structures.

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



Macros

- `#define WEB_THREAD_STACK_SIZE 1024`
- `#define WEB_THREAD_PORT 80`
- `#define WEB_THREAD_PRIORITY (LOWPRIO + 2)`

Functions

- **WORKING_AREA** (wa_http_server, WEB_THREAD_STACK_SIZE)
- `msg_t http_server` (void *p)

9.100.1 Detailed Description

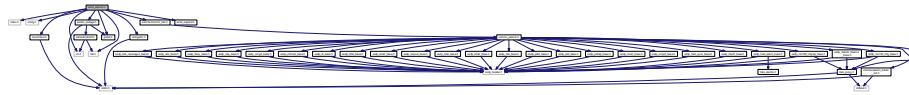
HTTP server wrapper thread macros and structures.

Definition in file [web.h](#).

9.101 write_eeprom.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "ch.h"
#include "hal.h"
#include "write_eeprom.h"
#include "comms/comms_hub.h"
#include "param_storage.h"
#include "trace/trace.h"
#include "stringutils.h"
#include "global.h"
```

Include dependency graph for `write_eeprom.c`:



Functions

- void `write_eeprom_osdp_com` (void)
Write the new OSDP address and baud rate to EEPROM.
- void `write_eeprom_osdp_scbk` (void)
Write the new SCBK to EEPROM.
- void `write_eeprom_osdp_security_params` (void)
Write the new SCBK to EEPROM.
- void `write_eeprom_trace_level` (void)
Write the new SCBK to EEPROM.
- void `write_eeprom_osdp_ms100` (void)
Write the new MS100 settings to EEPROM.
- void `write_eeprom_algorithm` (void)
Write the new microwave intruder algorithm selection to EEPROM.
- void `write_eeprom_osdp_ipv4` (void)
Store the new IPv4 settings to EEPROM.
- void `write_eeprom_relay_triggers` (void)
Store the relay trigger parameters.
- void `write_eeprom_paired` (void)
Store the paired devicer parameters.

9.101.1 Detailed Description

Author

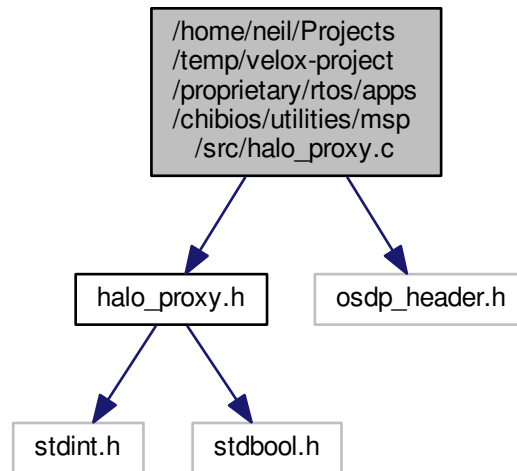
neil

Definition in file `write_eeprom.c`.

9.102 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/halo_proxy.c File Reference

```
#include "halo_proxy.h"
#include "osdp_header.h"
```

Include dependency graph for halo_proxy.c:



Functions

- void [proxy_init](#) ([proxy_list_t](#) *hdl)
Initialize the proxy list.
- bool [is_proxy_pd](#) ([proxy_list_t](#) *hdl, [uint8_t](#) addr)
Using the OSDP address, determine if the PD is on the proxy list.
- [int32_t](#) [get_num_proxy](#) ([proxy_list_t](#) *hdl)
Return the number of proxy PDs.
- [uint8_t](#) [get_first_proxy_addr](#) ([proxy_list_t](#) *hdl)
Return the OSDP address of the first proxy PD.
- [uint8_t](#) [get_next_proxy_addr](#) ([proxy_list_t](#) *hdl, [uint8_t](#) addr)
Return the OSDP address of the next proxy PD.
- [proxy_item_t](#) * [get_proxy_item](#) ([proxy_list_t](#) *hdl, [uint8_t](#) addr)
Add a device to the proxy list, associating a serial number with an OSDP address.
- bool [add_proxy](#) ([proxy_list_t](#) *hdl, [uint8_t](#) addr, [proxy_item_t](#) *item)
Add a device to the proxy list, associating a serial number with an OSDP address.
- bool [remove_proxy](#) ([proxy_list_t](#) *hdl, [uint8_t](#) addr)
Remove the PD selected by its OSDP address.
- bool [modify_addr](#) ([proxy_list_t](#) *hdl, [uint8_t](#) old_addr, [uint8_t](#) new_addr)
Modify the OSDP address used by a PD.

9.102.1 Detailed Description

Author

neil

Definition in file [halo_proxy.c](#).

9.102.2 Function Documentation

9.102.2.1 `uint8_t get_first_proxy_addr ( proxy_list_t * hdl )`

Return the OSDP address of the first proxy PD.

Returns

0-126 if it exists, 255 otherwise

Definition at line 43 of file `halo_proxy.c`.

9.102.2.2 `uint8_t get_next_proxy_addr ( proxy_list_t * hdl, uint8_t addr )`

Return the OSDP address of the next proxy PD.

Returns

0-126 if it exists, 255 otherwise

Definition at line 64 of file `halo_proxy.c`.

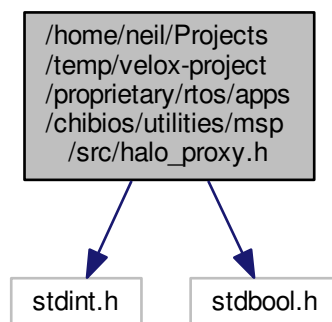
9.103 `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/halo_proxy.h` File Reference

A concrete implementation of `osdp_data` for ACK responses.

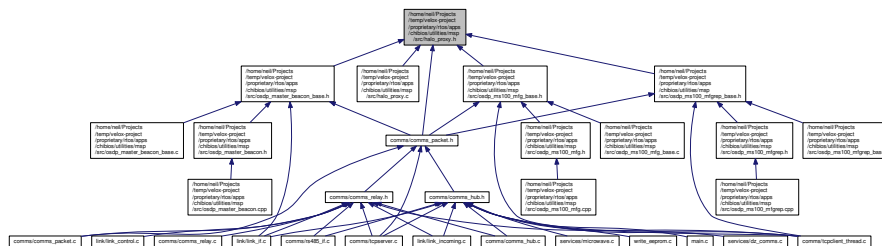
```
#include <stdint.h>
```

```
#include <stdbool.h>
```

Include dependency graph for `halo_proxy.h`:



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



Data Structures

- struct [proxy_item_t](#)
- struct [proxy_list_t](#)

Enumerations

- enum **proxy_status_e** { **PROXY_UNKNOWN** = 0, **PROXY_PRESENT**, **PROXY_ACTIVATED**, **PROXY_ENABLED** }

Functions

- void [proxy_init](#) ([proxy_list_t](#) *hdl)
Initialize the proxy list.
- bool [is_proxy_pd](#) ([proxy_list_t](#) *hdl, uint8_t addr)
Using the OSDP address, determine if the PD is on the proxy list.
- int32_t [get_num_proxy](#) ([proxy_list_t](#) *hdl)
Return the number of proxy PDs.
- uint8_t [get_first_proxy_addr](#) ([proxy_list_t](#) *hdl)
Return the OSDP address of the first proxy PD.
- uint8_t [get_next_proxy_addr](#) ([proxy_list_t](#) *hdl, uint8_t addr)
Return the OSDP address of the next proxy PD.
- [proxy_item_t](#) * [get_proxy_item](#) ([proxy_list_t](#) *hdl, uint8_t addr)
Add a device to the proxy list, associating a serial number with an OSDP address.
- bool [add_proxy](#) ([proxy_list_t](#) *hdl, uint8_t addr, [proxy_item_t](#) *item)
Add a device to the proxy list, associating a serial number with an OSDP address.
- bool [remove_proxy](#) ([proxy_list_t](#) *hdl, uint8_t addr)
Remove the PD selected by its OSDP address.
- bool [modify_addr](#) ([proxy_list_t](#) *hdl, uint8_t old_addr, uint8_t new_addr)
Modify the OSDP address used by a PD.

9.103.1 Detailed Description

A concrete implementation of osdp_data for ACK responses.

Author

neil

Definition in file [halo_proxy.h](#).

9.103.2 Function Documentation

9.103.2.1 `uint8_t get_first_proxy_addr ( proxy_list_t * hdl )`

Return the OSDP address of the first proxy PD.

Returns

0-126 if it exists, 255 otherwise

Definition at line 43 of file `halo_proxy.c`.

9.103.2.2 `uint8_t get_next_proxy_addr ( proxy_list_t * hdl, uint8_t addr )`

Return the OSDP address of the next proxy PD.

Returns

0-126 if it exists, 255 otherwise

Definition at line 64 of file `halo_proxy.c`.

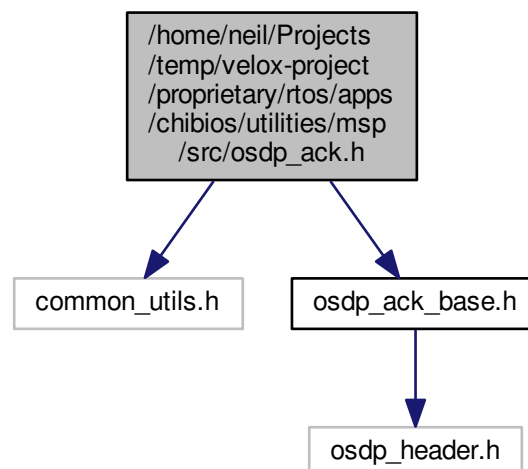
9.104 `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ack.h` File Reference

A concrete implementation of `osdp_data` for ACK responses.

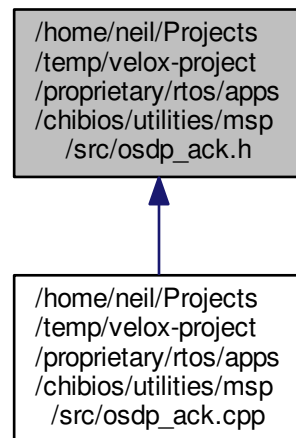
```
#include "common_utils.h"
```

```
#include "osdp_ack_base.h"
```

Include dependency graph for `osdp_ack.h`:



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



Data Structures

- class [osdp_ack](#)

9.104.1 Detailed Description

A concrete implementation of osdp_data for ACK responses.

A concrete implementation of osdp_data for NAK responses.

A concrete implementation of osdp_data for LSTATR responses.

A concrete implementation of osdp_data for ISTATR responses.

A concrete implementation of osdp_data for CAP requests.

Author

neil

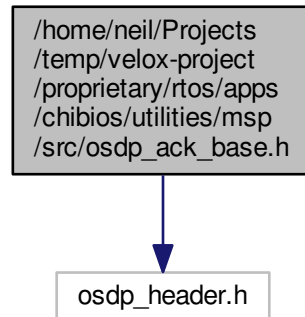
Definition in file [osdp_ack.h](#).

9.105 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ack_base.h](#) File Reference

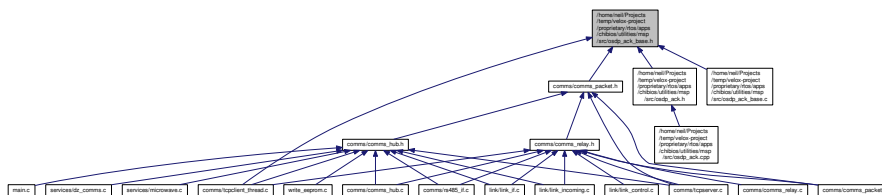
A C implementation of ACK packet encode function.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_ack_base.h:



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



Data Structures

- struct [_osdp_ack_t](#)

Macros

- #define [OSDP_ACK](#) 0x40
The ID used to define an ACK packet.

Typedefs

- typedef struct [_osdp_ack_t](#) [osdp_ack_t](#)

Functions

- void [osdp_ack_encode](#) ([osdp_ack_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.

9.105.1 Detailed Description

A C implementation of ACK packet encode function.

Author

neil

Definition in file [osdp_ack_base.h](#).

9.105.2 Function Documentation

9.105.2.1 void osdp_ack_encode (osdp_ack_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 10 of file osdp_ack_base.c.

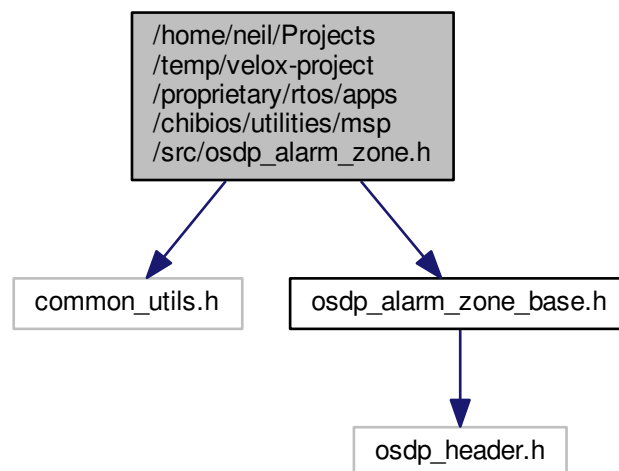
9.106 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_alarm_zone.h File Reference

A concrete implementation of osdp_data for ALARM_ZONE responses.

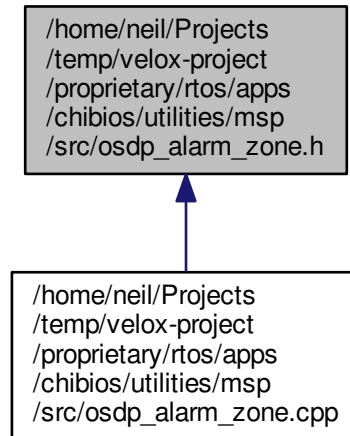
```
#include "common_utils.h"
```

```
#include "osdp_alarm_zone_base.h"
```

Include dependency graph for osdp_alarm_zone.h:



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



Data Structures

- class [osdp_alarm_zone](#)

9.106.1 Detailed Description

A concrete implementation of osdp_data for ALARM_ZONE responses.

Author

neil

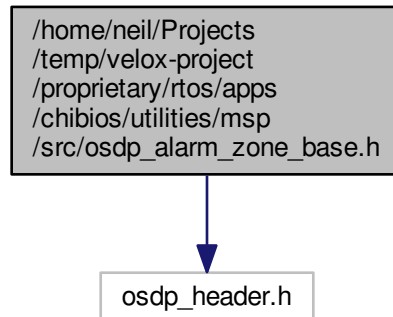
Definition in file [osdp_alarm_zone.h](#).

9.107 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_alarm_zone_base.h](#) File Reference

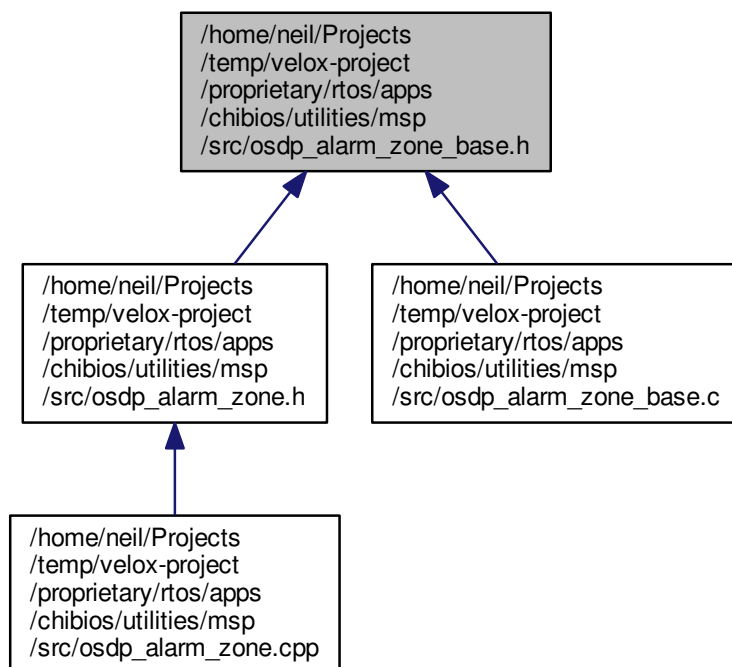
A C implementation of the proprietary Sensurity alarm zone packet.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_alarm_zone_base.h:



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



Data Structures

- [struct _osdp_alarm_zone_t](#)

Macros

- `#define OSDP_ALARM_ZONE 0xC3`
The ID used to define a ALARM_ZONE packet.

Typedefs

- `typedef struct _osdp_alarm_zone_t osdp_alarm_zone_t`

Functions

- `void osdp_alarm_zone_encode (osdp_alarm_zone_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_alarm_zone_decode (osdp_alarm_zone_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.107.1 Detailed Description

A C implementation of the proprietary Sensurity alarm zone packet.

Author

neil

Definition in file [osdp_alarm_zone_base.h](#).

9.107.2 Function Documentation

9.107.2.1 `int32_t osdp_alarm_zone_decode ( osdp_alarm_zone_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 48 of file [osdp_alarm_zone_base.c](#).

9.107.2.2 `void osdp_alarm_zone_encode ( osdp_alarm_zone_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
---------------	---

9.108

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling.h

File Reference

485

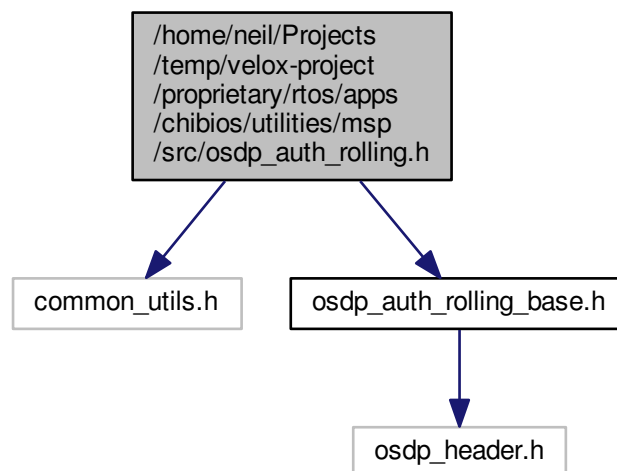
<i>len</i>	The length of the output packet
------------	---------------------------------

Definition at line 12 of file osdp_alarm_zone_base.c.

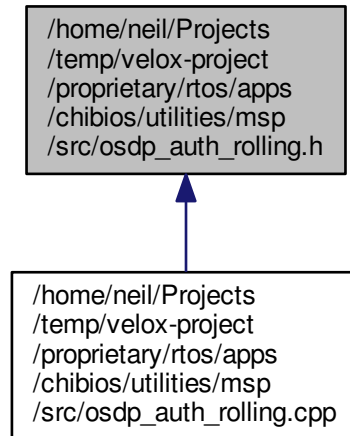
9.108 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling.h File Reference

A concrete implementation of osdp_data to send an MS100 rolling code.

```
#include "common_utils.h"
#include "osdp_auth_rolling_base.h"
Include dependency graph for osdp_auth_rolling.h:
```



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



Data Structures

- class [osdp_auth_rolling](#)

9.108.1 Detailed Description

A concrete implementation of `osdp_data` to send an MS100 rolling code.

Author

neil

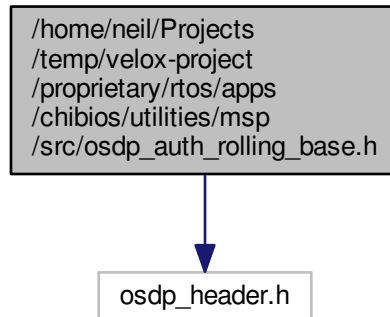
Definition in file [osdp_auth_rolling.h](#).

9.109 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_auth_rolling_base.h](#) File Reference

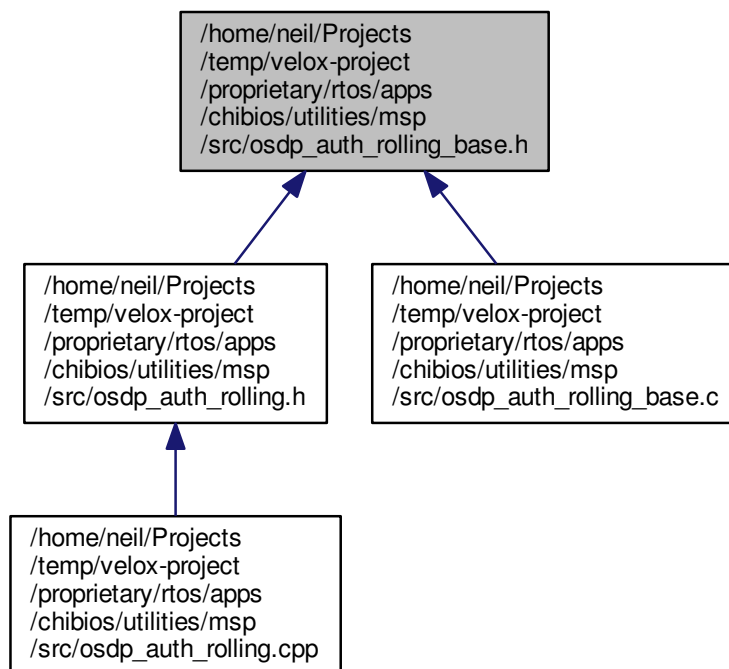
Encode and decode functionality for MS100 rolling codes.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_auth_rolling_base.h:



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



Data Structures

- struct `osdp_auth_rolling_t`

Macros

- `#define OSDP_AUTH_ROLLING 0x01`
The ID used to define a proprietary MS100 authentication rolling code.

Functions

- `void osdp_auth_rolling_encode (osdp_auth_rolling_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_auth_rolling_decode (osdp_auth_rolling_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.109.1 Detailed Description

Encode and decode functionality for MS100 rolling codes.

Author

neil

Definition in file [osdp_auth_rolling_base.h](#).

9.109.2 Function Documentation

9.109.2.1 `int32_t osdp_auth_rolling_decode ( osdp_auth_rolling_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

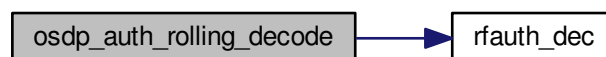
<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 32 of file [osdp_auth_rolling_base.c](#).

Here is the call graph for this function:



9.109.2.2 `void osdp_auth_rolling_encode ( osdp_auth_rolling_t * data, uint8_t * buffer, int32_t * len )`

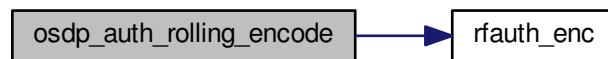
Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 13 of file osdp_auth_rolling_base.c.

Here is the call graph for this function:



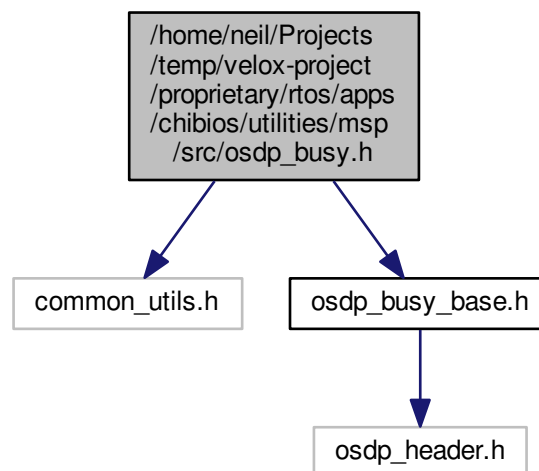
9.110 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_busy.h File Reference

Encode and decode functionality for MS100 rolling codes.

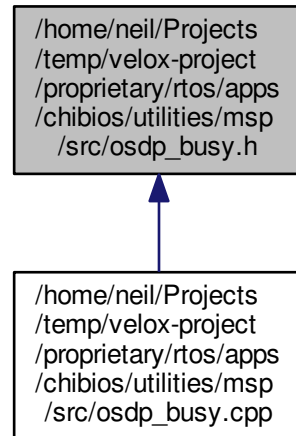
```
#include "common_utils.h"
```

```
#include "osdp_busy_base.h"
```

Include dependency graph for osdp_busy.h:



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



Data Structures

- class [osdp_busy](#)

9.110.1 Detailed Description

Encode and decode functionality for MS100 rolling codes.

Author

neil

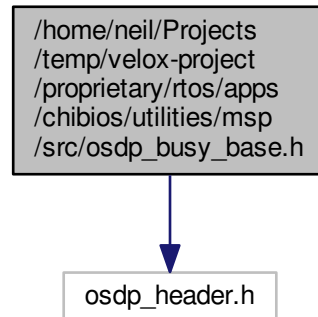
Definition in file [osdp_busy.h](#).

9.111 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_busy_base.h](#) File Reference

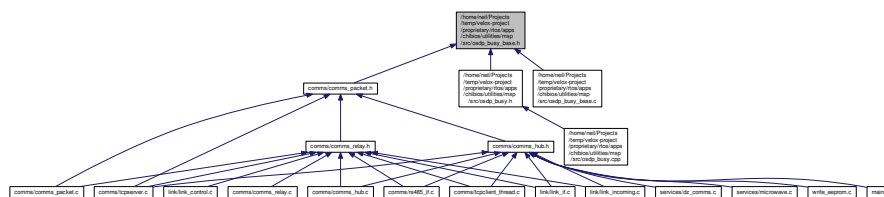
A C implementation of ACK packet encode function.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_busy_base.h:



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



Data Structures

- struct `_osdp_busy_t`

Macros

- `#define OSDP_BUSY 0x79`
The ID used to define a BUSY packet.

Typedefs

- `typedef struct _osdp_busy_t osdp_busy_t`

Functions

- `void osdp_busy_encode (osdp_busy_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.

9.111.1 Detailed Description

A C implementation of ACK packet encode function.

Author

neil

Definition in file [osdp_busy_base.h](#).

9.111.2 Function Documentation

9.111.2.1 void osdp_busy_encode (osdp_busy_t * *data*, uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

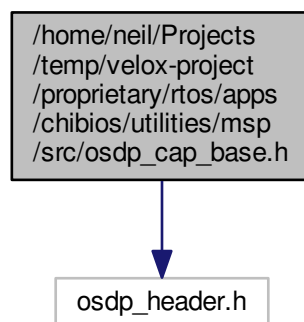
Definition at line 10 of file osdp_busy_base.c.

9.112 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↵ _cap_base.h File Reference

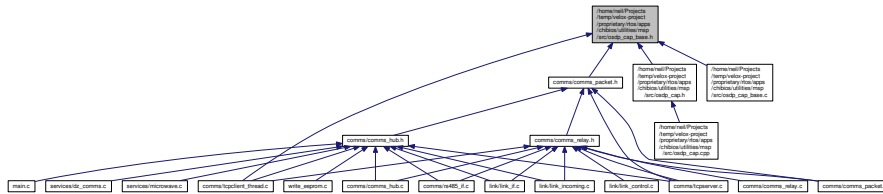
Encode and decode functionality for OSDP capabilty packets.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_cap_base.h:



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



Data Structures

- struct `_osdp_cap_t`

Macros

- #define **OSDP_CAP** 0x62
The ID used to define an CAP packet.

Typedefs

- typedef struct **_osdp_cap_t** **osdp_cap_t**

Enumerations

- enum { **SEND_STANDARD_REPLY** = 0 }

Functions

- void `osdp_cap_encode (osdp_cap_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- int32_t `osdp_cap_decode (osdp_cap_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.112.1 Detailed Description

Encode and decode functionality for OSDP capability packets.

Author

neil

Definition in file [osdp_cap_base.h](#).

9.112.2 Function Documentation

9.112.2.1 int32_t osdp_cap_decode (osdp_cap_t *data, uint8_t *buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 40 of file osdp_cap_base.c.

9.112.2.2 void osdp_cap_encode (osdp_cap_t * *data*, uint8_t * *buffer*, int32_t * *len*)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 10 of file osdp_cap_base.c.

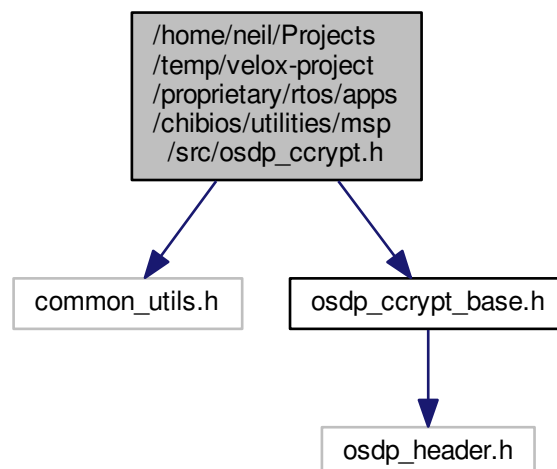
9.113 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ccrypt.h File Reference

A concrete implementation of osdp_data to transmit SCS-CS cryptograms.

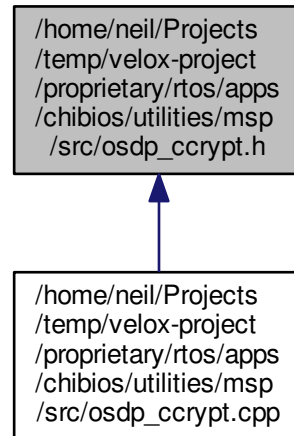
```
#include "common_utils.h"
```

```
#include "osdp_ccrypt_base.h"
```

Include dependency graph for osdp_ccrypt.h:



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



Data Structures

- class [osdp_ccrypt](#)

9.113.1 Detailed Description

A concrete implementation of osdp_data to transmit SCS-CS cryptograms.

Author

neil

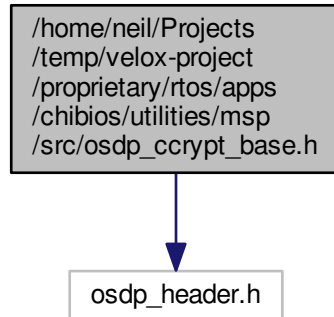
Definition in file [osdp_ccrypt.h](#).

9.114 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ccrypt_base.h](#) File Reference

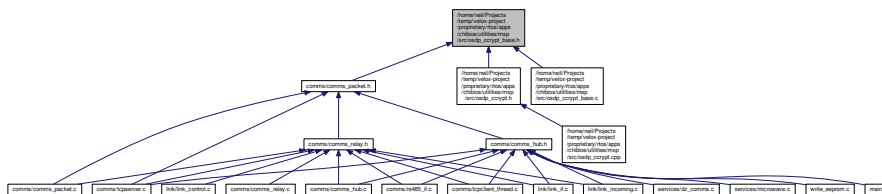
The low-level encode/decode functions for osdp_CCrypt functionality.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_ccrypt_base.h:



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



Data Structures

- struct [osdp_ccrypt_t](#)

Macros

- #define **OSDP_CCRIPT** 0x76
The ID used to define a SCS-CS Client Cryptogram packet.
- #define **OSDP_CHLNG** 0x76

Functions

- void [osdp_ccrypt_encode](#) ([osdp_ccrypt_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- void [osdp_chlng_encode](#) ([osdp_ccrypt_t](#) *data, uint8_t *buffer, int32_t *len)
- int32_t [osdp_ccrypt_decode](#) ([osdp_ccrypt_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.
- int32_t [osdp_chlng_decode](#) ([osdp_ccrypt_t](#) *data, uint8_t *buffer, int32_t len)

9.114.1 Detailed Description

The low-level encode/decode functions for osdp_CCrypt functionality.

Author

neil

Definition in file [osdp_ccrypt_base.h](#).

9.114.2 Function Documentation

9.114.2.1 `int32_t osdp_ccrypt_decode ( osdp_ccrypt_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 68 of file [osdp_ccrypt_base.c](#).

9.114.2.2 `void osdp_ccrypt_encode ( osdp_ccrypt_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 11 of file [osdp_ccrypt_base.c](#).

9.115 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↵ _comset_base.h File Reference

Encode and decode functionality for OSDP COMSET packets.

Author

neil

Definition in file [osdp_comset_base.h](#).

9.115.2 Function Documentation

9.115.2.1 int32_t osdp_comset_decode (osdp_comset_t * data, uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 46 of file [osdp_comset_base.c](#).

9.115.2.2 void osdp_comset_encode (osdp_comset_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

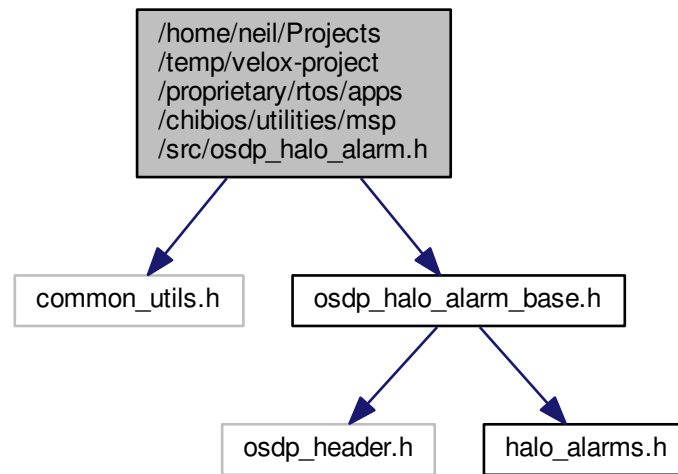
Definition at line 10 of file [osdp_comset_base.c](#).

9.116 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_alarm.h File Reference

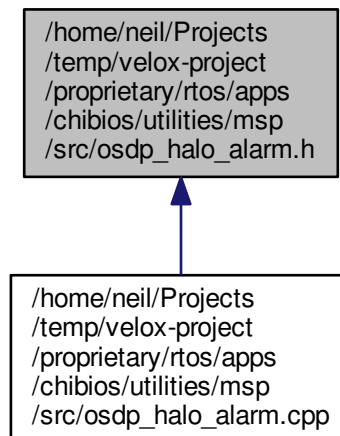
A concrete implementation of [osdp_data](#) to transfer alarms.

```
#include "common_utils.h"
#include "osdp_halo_alarm_base.h"
```

Include dependency graph for `osdp_halo_alarm.h`:



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



Data Structures

- class `osdp_halo_alarm`
- class `osdp_halo_alarm_request`

Macros

- `#define OSDP_HALO_ALARM 0xC1`
The ID used to define a HALO ALARM packet.
- `#define OSDP_HALO_ALARM_REQUEST 0xC2`
The ID used to define a HALO ALARM REQUEST packet.

Typedefs

- `typedef struct _osdp_halo_alarm_t osdp_halo_alarm_t`
- `typedef struct _osdp_halo_alarm_request_t osdp_halo_alarm_request_t`

Functions

- `void osdp_halo_alarm_encode (osdp_halo_alarm_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_halo_alarm_decode (osdp_halo_alarm_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.
- `void osdp_halo_alarm_request_encode (osdp_halo_alarm_request_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_halo_alarm_request_decode (osdp_halo_alarm_request_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.117.1 Detailed Description

A C implementation of the proprietary Halo alarm packet.

Author

neil

Definition in file [osdp_halo_alarm_base.h](#).

9.117.2 Function Documentation

9.117.2.1 `int32_t osdp_halo_alarm_decode ( osdp_halo_alarm_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 346 of file [osdp_halo_alarm_base.c](#).

9.117.2.2 `void osdp_halo_alarm_encode ( osdp_halo_alarm_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

9.118 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_sync_base.h File

Reference Parameters

503

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 297 of file osdp_halo_alarm_base.c.

9.117.2.3 `int32_t osdp_halo_alarm_request_decode ( osdp_halo_alarm_request_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 404 of file osdp_halo_alarm_base.c.

9.117.2.4 `void osdp_halo_alarm_request_encode ( osdp_halo_alarm_request_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

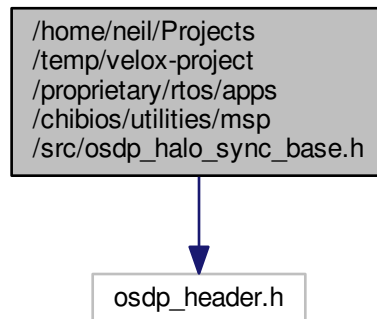
Definition at line 374 of file osdp_halo_alarm_base.c.

9.118 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_halo_sync_base.h File Reference

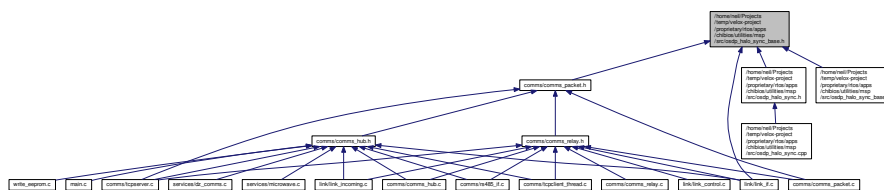
A C implementation of the proprietary Halo sync packet.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_halo_sync_base.h:



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



Data Structures

- struct [_osdp_halo_sync_t](#)

Macros

- #define [OSDP_HALO_SYNC](#) 0xC3
The ID used to define a HALO_SYNC packet.

Typedefs

- typedef struct [_osdp_halo_sync_t](#) [osdp_halo_sync_t](#)

Functions

- void [osdp_halo_sync_encode](#) ([osdp_halo_sync_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_halo_sync_decode](#) ([osdp_halo_sync_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.118.1 Detailed Description

A C implementation of the proprietary Halo sync packet.

Author

neil

Definition in file [osdp_halo_sync_base.h](#).

9.118.2 Function Documentation

9.118.2.1 `int32_t osdp_halo_sync_decode ( osdp_halo_sync_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 45 of file [osdp_halo_sync_base.c](#).

9.118.2.2 `void osdp_halo_sync_encode ( osdp_halo_sync_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

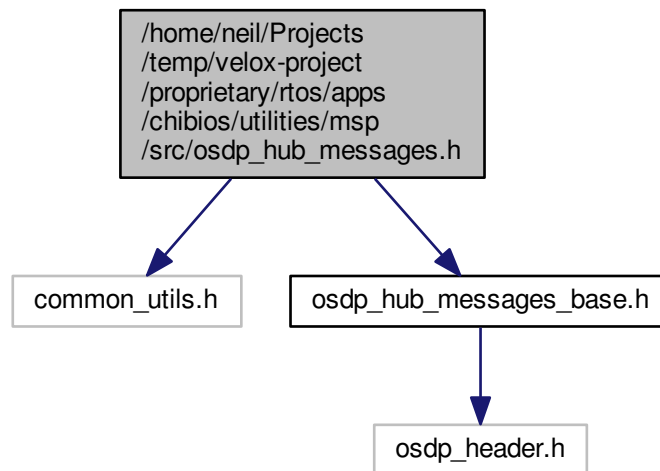
Definition at line 12 of file [osdp_halo_sync_base.c](#).

9.119 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_hub_messages.h File Reference

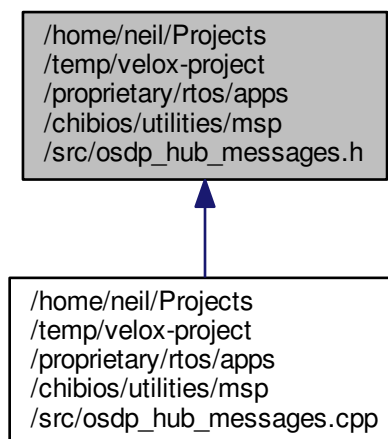
A concrete implementation of `osdp_data` to send OSDP Hub commands and replies.

```
#include "common_utils.h"
#include "osdp_hub_messages_base.h"
```

Include dependency graph for `osdp_hub_messages.h`:



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



Data Structures

- class `osdp_hub_messages`

9.119.1 Detailed Description

A concrete implementation of `osdp_data` to send OSDP Hub commands and replies.

Definition in file [osdp_hub_messages.h](#).

Encode and decode functionality for OSDP ID packets.

Include dependency graph for osdp_id_base.h:



- ## Macros

- The ID used to define an ID Report Request packet.*

Functions

- void [osdp_id_encode](#) ([osdp_id_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_id_decode](#) ([osdp_id_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.120.1 Detailed Description

Encode and decode functionality for OSDP ID packets.

Author

neil

Definition in file [osdp_id_base.h](#).

9.120.2 Function Documentation

9.120.2.1 int32_t osdp_id_decode (osdp_id_t * data, uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 47 of file [osdp_id_base.c](#).

9.120.2.2 void osdp_id_encode (osdp_id_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 15 of file [osdp_id_base.c](#).

9.121 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istat_base.h File Reference

A C implementation of ISTAT packet encode function.

Macros

- `#define OSDP_ISTATR 0x49`
The ID used to define an ISTATR packet.

Typedefs

- `typedef struct _osdp_istatr_t osdp_istatr_t`

Functions

- `void osdp_istatr_encode (osdp_istatr_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_istatr_decode (osdp_istatr_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.122.1 Detailed Description

A C implementation of ISTATR packet encode function.

Author

neil

Definition in file [osdp_istatr_base.h](#).

9.122.2 Function Documentation**9.122.2.1 int32_t osdp_istatr_decode (osdp_istatr_t * data, uint8_t * buffer, int32_t len)**

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 43 of file [osdp_istatr_base.c](#).

9.122.2.2 void osdp_istatr_encode (osdp_istatr_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
---------------	---

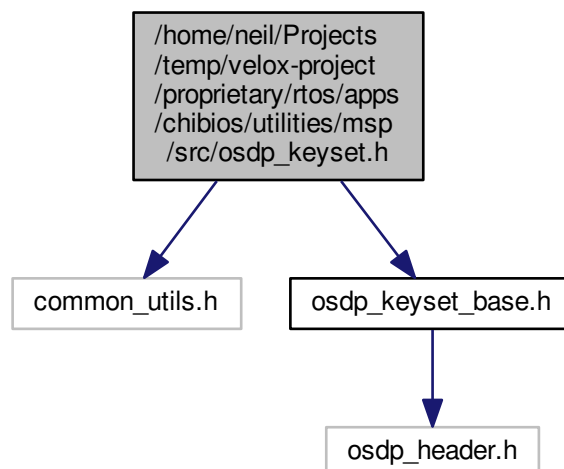
<i>len</i>	The length of the output packet
------------	---------------------------------

Definition at line 10 of file osdp_istatr_base.c.

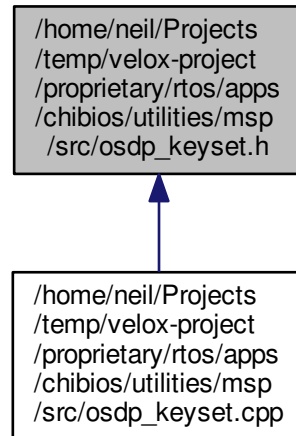
9.123 `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_keyset.h` File Reference

A concrete implementation of `osdp_data` to configure the SCBK.

```
#include "common_utils.h"
#include "osdp_keyset_base.h"
Include dependency graph for osdp_keyset.h:
```



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



Data Structures

- class [osdp_keyset](#)

9.123.1 Detailed Description

A concrete implementation of osdp_data to configure the SCBK.

Author

neil

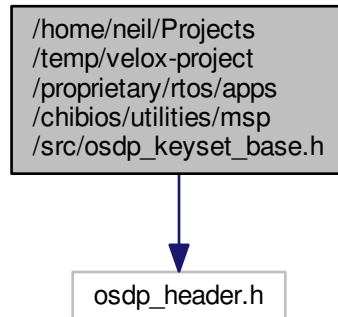
Definition in file [osdp_keyset.h](#).

9.124 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_keyset_base.h](#) File Reference

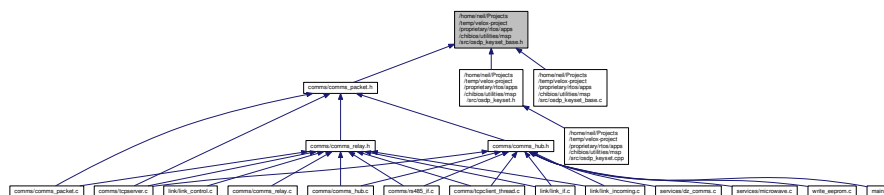
The low-level encode/decode functions for osdp_KEYSET functionality.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_keyset_base.h:



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



Data Structures

- struct [osdp_keyset_t](#)

Macros

- `#define` [OSDP_KEYSET](#) 0x75
The ID used to define a KEYSET packet.

Functions

- void [osdp_keyset_encode](#) ([osdp_keyset_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_keyset_decode](#) ([osdp_keyset_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.124.1 Detailed Description

The low-level encode/decode functions for osdp_KEYSET functionality.

Author

neil

Definition in file [osdp_keyset_base.h](#).

9.124.2 Function Documentation

9.124.2.1 `int32_t osdp_keyset_decode ( osdp_keyset_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 44 of file `osdp_keyset_base.c`.

9.124.2.2 `void osdp_keyset_encode ( osdp_keyset_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

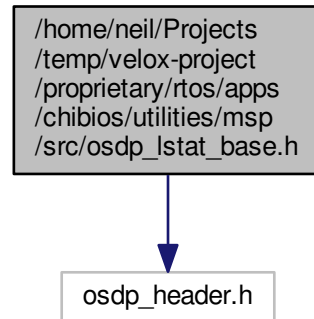
Definition at line 11 of file `osdp_keyset_base.c`.

9.125 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istat_base.h File Reference

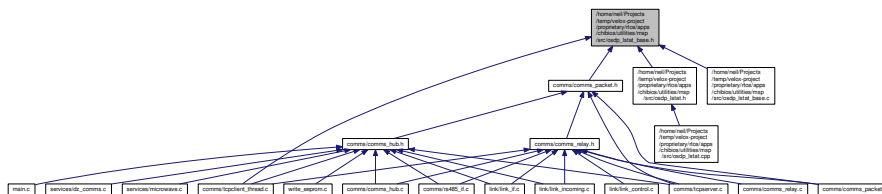
A C implementation of LSTAT packet encode functionality.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_lstat_base.h:



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



Data Structures

- struct [osdp_lstat_t](#)

Macros

- #define [OSDP_LSTAT](#) 0x64
The ID used to define an LSTAT packet.

Functions

- void [osdp_lstat_encode](#) ([osdp_lstat_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.

9.125.1 Detailed Description

A C implementation of LSTAT packet encode functionality.

Author

neil

Definition in file [osdp_lstat_base.h](#).

9.125.2 Function Documentation

9.125.2.1 void osdp_istat_encode (osdp_istat_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

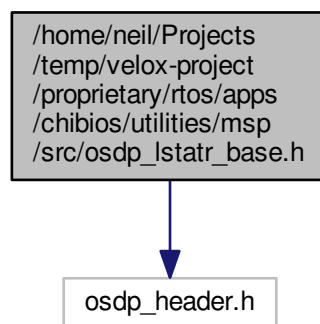
Definition at line 10 of file osdp_istat_base.c.

9.126 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_istatr_base.h File Reference

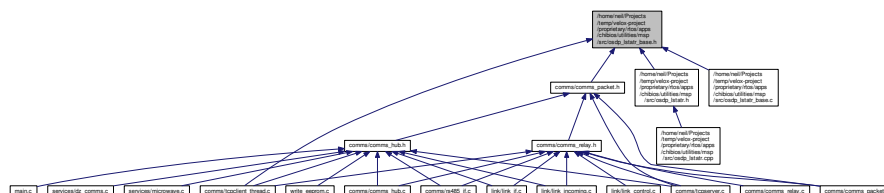
A C implementation of LSTATR packet encode function.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_istatr_base.h:



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



Data Structures

- struct `_osdp_istatr_t`

Macros

- `#define OSDP_LSTATR 0x48`
The ID used to define an LSTATR packet.

Typedefs

- `typedef struct _osdp_lstatr_t osdp_lstatr_t`

Functions

- `void osdp_lstatr_encode (osdp_lstatr_t *data, uint8_t *buffer, int32_t *len)`
Translate the data structure to a packet for transmission.
- `int32_t osdp_lstatr_decode (osdp_lstatr_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.126.1 Detailed Description

A C implementation of LSTATR packet encode function.

Author

neil

Definition in file [osdp_lstatr_base.h](#).

9.126.2 Function Documentation

9.126.2.1 `int32_t osdp_lstatr_decode ( osdp_lstatr_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 41 of file [osdp_lstatr_base.c](#).

9.126.2.2 `void osdp_lstatr_encode ( osdp_lstatr_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
---------------	---

<i>len</i>	The length of the output packet
------------	---------------------------------

Definition at line 10 of file osdp_istratr_base.c.

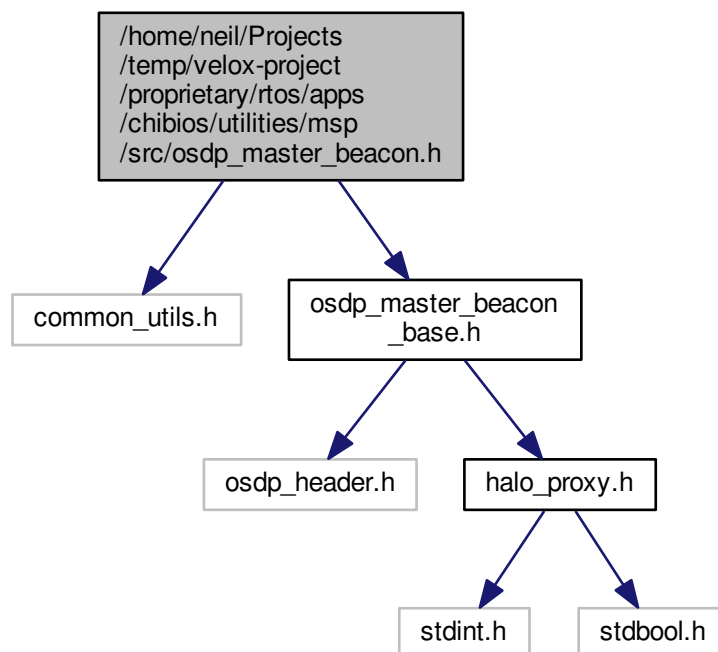
9.127 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp↵ _master_beacon.h File Reference

A concrete implementation of osdp_data for MASTER_BEACON responses.

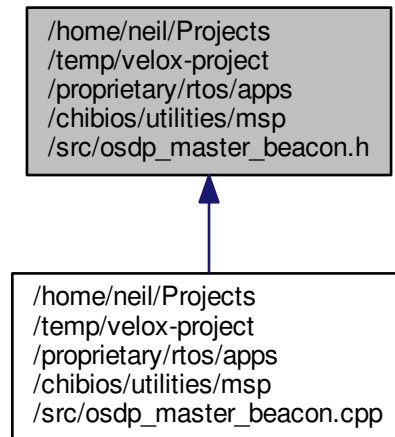
```
#include "common_utils.h"
```

```
#include "osdp_master_beacon_base.h"
```

Include dependency graph for osdp_master_beacon.h:



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



Data Structures

- class [osdp_master_beacon](#)

9.127.1 Detailed Description

A concrete implementation of osdp_data for MASTER_BEACON responses.

Author

neil

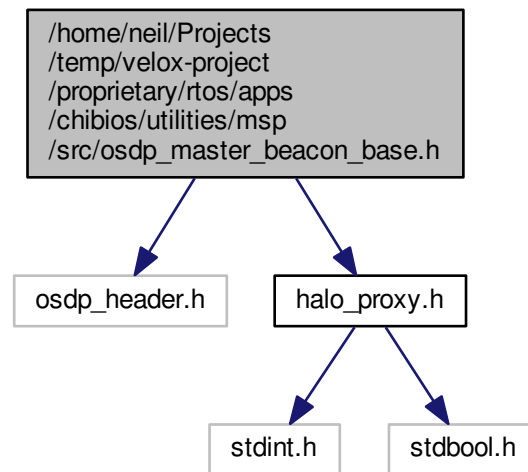
Definition in file [osdp_master_beacon.h](#).

9.128 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_master_beacon_base.h File Reference

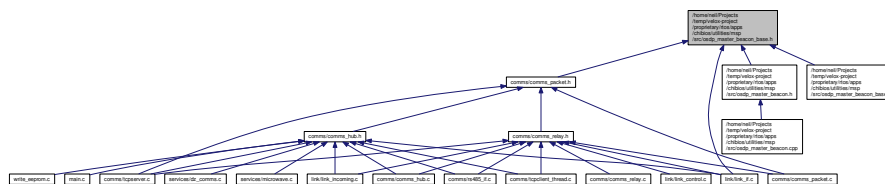
A C implementation of the proprietary master beacon packet.

```
#include "osdp_header.h"
#include "halo_proxy.h"
```

Include dependency graph for osdp_master_beacon_base.h:



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



Data Structures

- struct `_osdp_master_beacon_t`

Macros

- `#define OSDP_MASTER_BEACON 0xC0`
The ID used to define a MASTER BEACON packet.

Typedefs

- typedef struct _osdp_master_beacon_t **osdp_master_beacon_t**

Functions

- void `osdp_master_beacon_encode` (`osdp_master_beacon_t` *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.

- `int32_t osdp_master_beacon_decode (osdp_master_beacon_t *data, uint8_t *buffer, int32_t len)`
Translate a received packet into a data structure.

9.128.1 Detailed Description

A C implementation of the proprietary master beacon packet.

Author

neil

Definition in file [osdp_master_beacon_base.h](#).

9.128.2 Function Documentation

9.128.2.1 `int32_t osdp_master_beacon_decode ( osdp_master_beacon_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

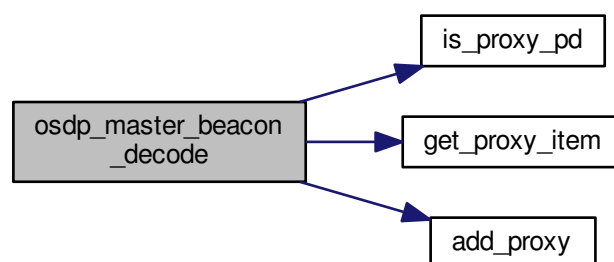
<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 65 of file `osdp_master_beacon_base.c`.

Here is the call graph for this function:



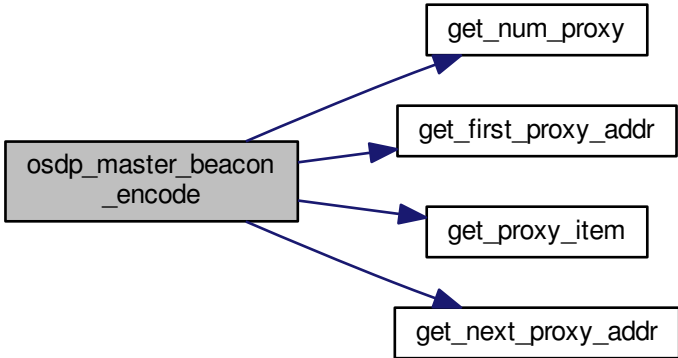
9.128.2.2 `void osdp_master_beacon_encode ( osdp_master_beacon_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 10 of file osdp_master_beacon_base.c.

Here is the call graph for this function:



9.129

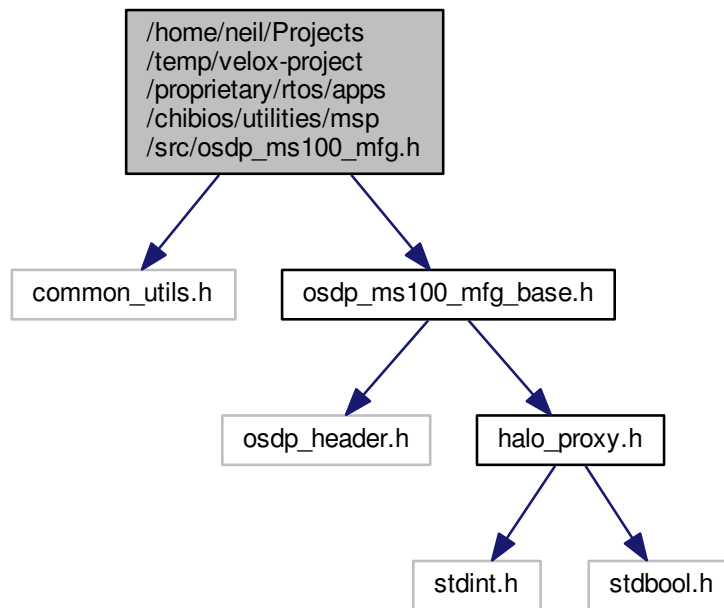
/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfg.h

File Reference

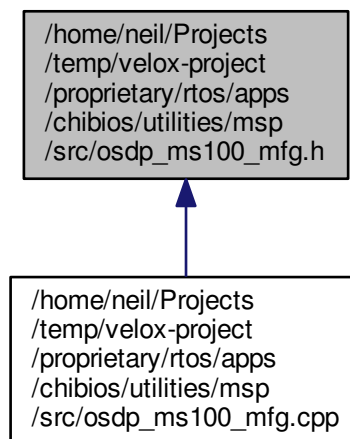
A concrete implementation of `osdp_data` to send an MS100 Configuration command.

```
#include "common_utils.h"
#include "osdp_ms100_mfg_base.h"
```

Include dependency graph for `osdp_ms100_mfg.h`:



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



Data Structures

- class `osdp_ms100_mfg`

Macros

- `#define OSDP_MFG 0x80`
The ID used to define a proprietary MS100 Configuration packet.

Enumerations

- enum `osdp_mfg_e` {
`MS100_CONFIG_CMD = 0, MS100_CONFIG_REQ, MS100_DEBUG_CMD, MS100_DEBUG_REQ,`
`MS100_ENTER_INSTALL_MODE, MS100_EXIT_INSTALL_MODE, MS100_SET_IPV4, MS100_GET_IPV4,`
`MS100_ENABLE_ALARMS, MS100_GET_FW_STATUS, MS100_FW_DOWNLOAD, MS100_SOFT_RESET,`
`MS100_GET_MICROWAVE_SAMPLES, MS100_SET_SECURITY, MS100_GET_SECURITY, MS100_GET_GPS,`
`MS100_SET_PKI, MS100_GET_PKI, MS100_GET_ALARM_TIMEOUT, MS100_SET_RELAY_TRIGGERS,`
`MS100_GET_RELAY_TRIGGERS, MS100_SET_DEBUG_ALARM, MS100_GET_DEBUG_ALARM, MS100_SET_DEADZONES,`
`MS100_GET_DEADZONES, MS100_SET_ALGORITHM, MS100_GET_ALGORITHM, MS100_GET_DEADZONE_FITTED,`
`MS100_SET_PROXY_PD_LIST, MS100_GET_PROXY_PD_LIST, MS100_SET_PAIRING, MS100_GET_PAIRING,`
`MS100_RUN_DIAGNOSTIC, MS100_GET_DIAGNOSTIC, MS100_GET_VIGIL_RELAY_FITTED, MS100_SET_SCANNABLE_BEAM,`
`MS100_GET_SCANNABLE_BEAM, MS100_GET_MICROWAVE_DETAIL, MS100_ENABLE_DHCP_AUTOIP }`

Functions

- void `osdp_ms100_mfg_encode` (`osdp_ms100_mfg_t` *data, `uint8_t` *buffer, `int32_t` *len)
Translate the data structure to a packet for transmission.
- `int32_t` `osdp_ms100_mfg_decode` (`osdp_ms100_mfg_t` *data, `uint8_t` *buffer, `int32_t` len)
Translate a received packet into a data structure.

9.130.1 Detailed Description

Encode and decode functionality for OSDP MFG packets.

Author

neil

Definition in file `osdp_ms100_mfg_base.h`.

9.130.2 Function Documentation

9.130.2.1 `int32_t osdp_ms100_mfg_decode ( osdp_ms100_mfg_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

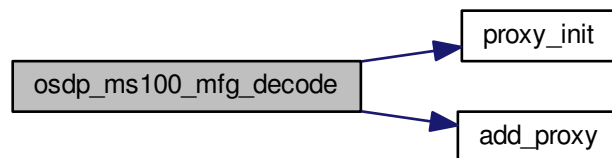
<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 197 of file osdp_ms100_mfg_base.c.

Here is the call graph for this function:



9.130.2.2 void osdp_ms100_mfg_encode (osdp_ms100_mfg_t * *data*, uint8_t * *buffer*, int32_t * *len*)

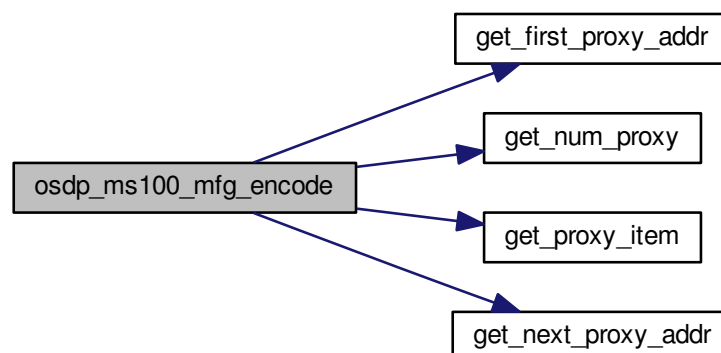
Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 11 of file osdp_ms100_mfg_base.c.

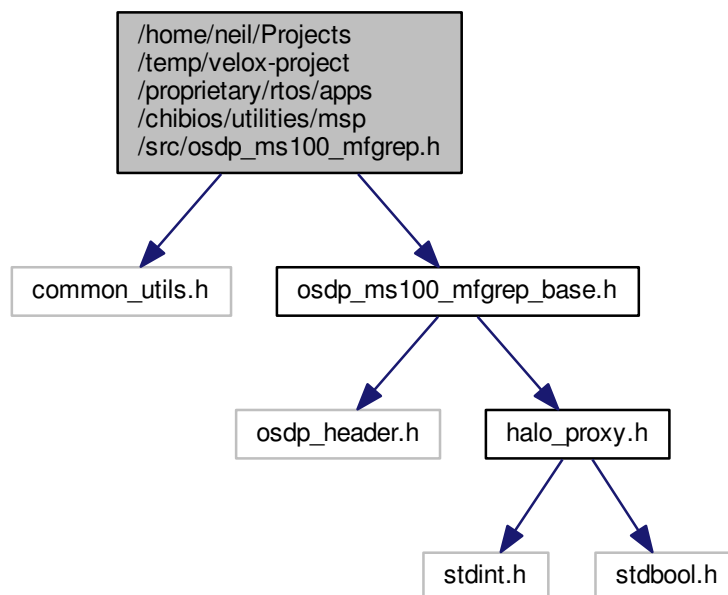
Here is the call graph for this function:



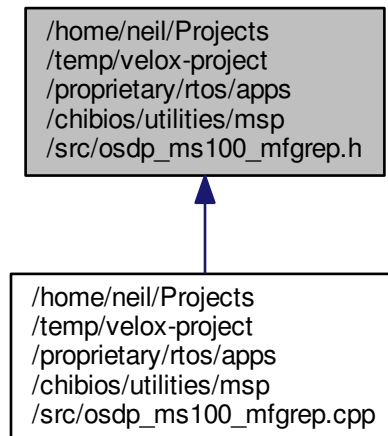
9.131 `/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_ms100_mfgrep.h` File Reference

A concrete implementation of `osdp_data` to set communication parameters.

```
#include "common_utils.h"
#include "osdp_ms100_mfgrep_base.h"
Include dependency graph for osdp_ms100_mfgrep.h:
```



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



Data Structures

- class [osdp_ms100_mfgrep](#)

9.131.1 Detailed Description

A concrete implementation of osdp_data to set communication parameters.

A concrete implementation of osdp_data to request data from an MS100.

Author

neil

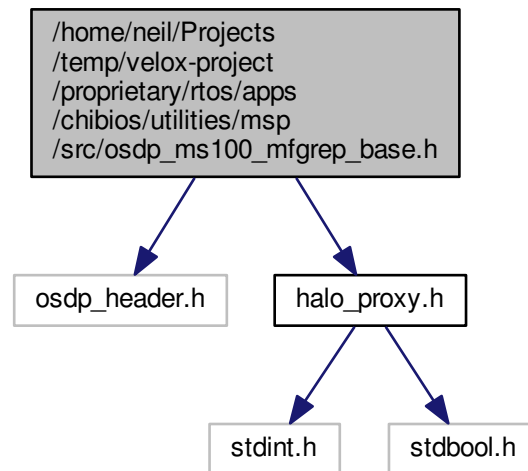
Definition in file [osdp_ms100_mfgrep.h](#).

9.132 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↔ _ms100_mfgrep_base.h File Reference

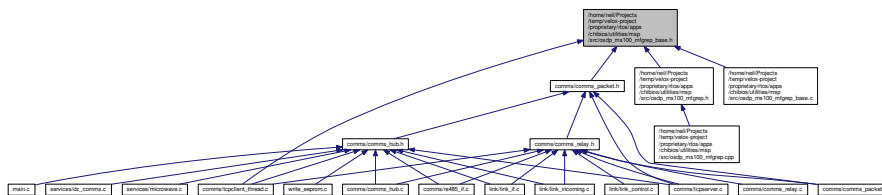
Encode and decode functionality for OSDP MFGREP packets.

```
#include "osdp_header.h"  
#include "halo_proxy.h"
```

Include dependency graph for `osdp_ms100_mfgrep_base.h`:



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



Data Structures

- struct `_osdp_sample_buffer_t`
- struct `osdp_ms100_mfgrep_t`

Macros

- `#define` `OSDP_MFGREP` 0x90
The ID used to define a proprietary MS100 Manufacturer Specific Reply.
- `#define` `MAX_DEBUG_PAYLOAD` 180
- `#define` `MAX_SAMPLE_PAYLOAD` 128

Typedefs

- typedef struct
 `_osdp_sample_buffer_t` `osdp_sample_buffer_t`

- enum **osdp_mfgrep_e** {
MS100_MFGREP_CMD = 0, MS100_MFGREP_REQ, MS100_MFGREP_DEBUG_CMD, MS100_MFGR↵
EP_DEBUG_REQ,
MS100_MFGREP_ENTER_INSTALL_MODE, MS100_MFGREP_EXIT_INSTALL_MODE, MS100_MFG↵
REP_SET_IPV4, MS100_MFGREP_GET_IPV4,
MS100_MFGREP_ENABLE_ALARMS, MS100_MFGREP_GET_FW_STATUS, MS100_MFGREP_FW_↵
DOWNLOAD, MS100_MFGREP_SOFT_RESET,
MS100_MFGREP_GET_MICROWAVE_SAMPLES, MS100_MFGREP_SET_SECURITY, MS100_MFGR↵
EP_GET_SECURITY, MS100_MFGREP_GET_GPS,
MS100_MFGREP_SET_PKI, MS100_MFGREP_GET_PKI, MS100_MFGREP_GET_ALARM_TIMEOUT,
MS100_MFGREP_SET_RELAY_TRIGGERS,
MS100_MFGREP_GET_RELAY_TRIGGERS, MS100_MFGREP_SET_DEBUG_ALARM, MS100_MFGR↵
EP_GET_DEBUG_ALARM, MS100_MFGREP_SET_DEADZONES,
MS100_MFGREP_GET_DEADZONES, MS100_MFGREP_SET_ALGORITHM, MS100_MFGREP_GET_↵
ALGORITHM, MS100_MFGREP_GET_DEADZONE_FITTED,
MS100_MFGREP_SET_PROXY_PD_LIST, MS100_MFGREP_GET_PROXY_PD_LIST, MS100_MFGR↵
EP_SET_PAIRED, MS100_MFGREP_GET_PAIRED,
MS100_MFGREP_RUN_DIAGNOSTIC, MS100_MFGREP_GET_DIAGNOSTIC, MS100_MFGREP_GET_↵
_VIGIL_RELAY_FITTED, MS100_MFGREP_SET_SCANNABLE_BEAM,
MS100_MFGREP_GET_SCANNABLE_BEAM, MS100_MFGREP_GET_MICROWAVE_DETAIL, MS100_↵
_MFGREP_ENABLE_DHCP_AUTOIP }

Functions

- void [osdp_ms100_mfgrep_encode](#) ([osdp_ms100_mfgrep_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_ms100_mfgrep_decode](#) ([osdp_ms100_mfgrep_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.132.1 Detailed Description

Encode and decode functionality for OSDP MFGREP packets.

Author

neil

Definition in file [osdp_ms100_mfgrep_base.h](#).

9.132.2 Function Documentation

9.132.2.1 int32_t osdp_ms100_mfgrep_decode (osdp_ms100_mfgrep_t * data, uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

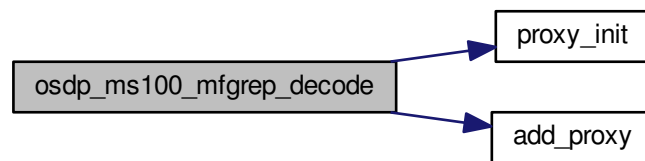
<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 356 of file osdp_ms100_mfgrep_base.c.

Here is the call graph for this function:



9.132.2.2 `void osdp_ms100_mfgrep_encode ( osdp_ms100_mfgrep_t * data, uint8_t * buffer, int32_t * len )`

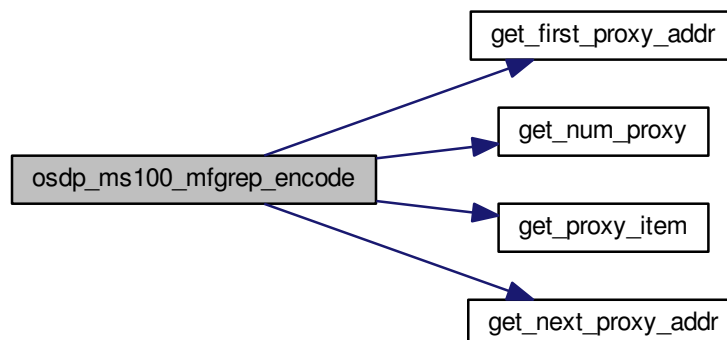
Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 19 of file osdp_ms100_mfgrep_base.c.

Here is the call graph for this function:



9.133

/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_nak_base.h

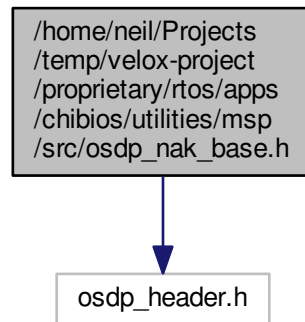
File Reference

9.133 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_nak_base.h File Reference 533

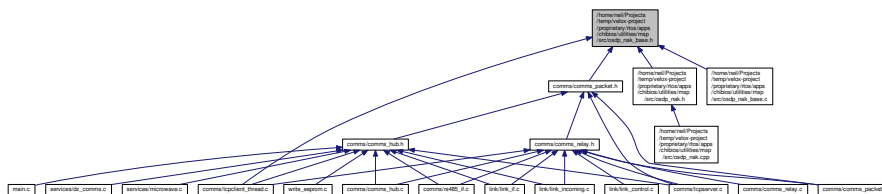
Encode and decode functionality for OSDP NAK packets.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_nak_base.h:



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



Data Structures

- struct `_osdp_nak_t`

Macros

- #define `OSDP_NAK` 0x41
The ID used to define an NAK packet.

Typedefs

- typedef struct `_osdp_nak_t` `osdp_nak_t`

Enumerations

- enum {
NAK_MSG_ERROR = 1, NAK_CMD_LENGTH_ERROR, NAK_CMD_UNKNOWN, NAK_UNEXPECTED

```

_SQN,
NAK_UNSUPPORTED_SCB, NAK_SECURITY_REQUIRED, NAK_BIO_TYPE_UNSUPPORTED, NAK_↵
BIO_FORMAT_UNSUPPORTED }

```

Functions

- void [osdp_nak_encode](#) ([osdp_nak_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_nak_decode](#) ([osdp_nak_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.133.1 Detailed Description

Encode and decode functionality for OSDP NAK packets.

Author

neil

Definition in file [osdp_nak_base.h](#).

9.133.2 Function Documentation

9.133.2.1 int32_t osdp_nak_decode (osdp_nak_t * data, uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 31 of file [osdp_nak_base.c](#).

9.133.2.2 void osdp_nak_encode (osdp_nak_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

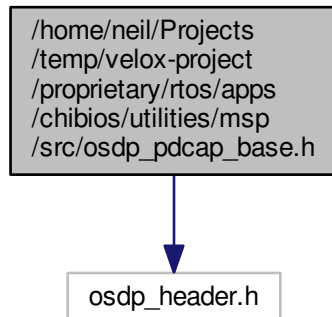
Definition at line 10 of file [osdp_nak_base.c](#).

9.134 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↵ _pdcap_base.h File Reference

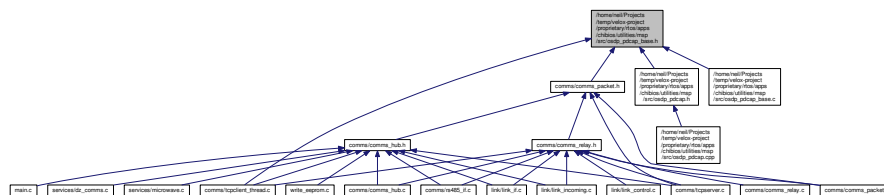
Encode and decode functionality for OSDP PDCAP packets.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_pdcap_base.h:



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



Data Structures

- struct [_default_vars_t](#)
- struct [_tuple_pdcap_t](#)
- struct [_osdp_pdcap_t](#)

Macros

- #define [OSDP_PDCAP](#) 0x46
The ID used to define a Device Capabilities Report packet.
- #define [MAX_PDCAP_TUPLES](#) 16

Typedefs

- typedef struct [_default_vars_t](#) [default_vars_t](#)
- typedef struct [_tuple_pdcap_t](#) [tuple_pdcap_t](#)
- typedef struct [_osdp_pdcap_t](#) [osdp_pdcap_t](#)

Functions

- void [osdp_pdcap_encode](#) ([osdp_pdcap_t](#) *data, [uint8_t](#) *buffer, [int32_t](#) *len)

Translate the data structure to a packet for transmission.

- `int32_t osdp_pdcap_decode (osdp_pdcap_t *data, uint8_t *buffer, int32_t len)`

Translate a received packet into a data structure.

9.134.1 Detailed Description

Encode and decode functionality for OSDP PDCAP packets.

Author

neil

Definition in file [osdp_pdcap_base.h](#).

9.134.2 Function Documentation

9.134.2.1 `int32_t osdp_pdcap_decode ( osdp_pdcap_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 46 of file `osdp_pdcap_base.c`.

9.134.2.2 `void osdp_pdcap_encode ( osdp_pdcap_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

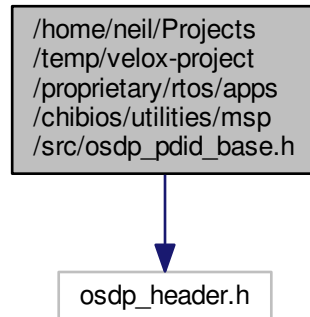
Definition at line 10 of file `osdp_pdcap_base.c`.

9.135 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_pdid_base.h](#) File Reference

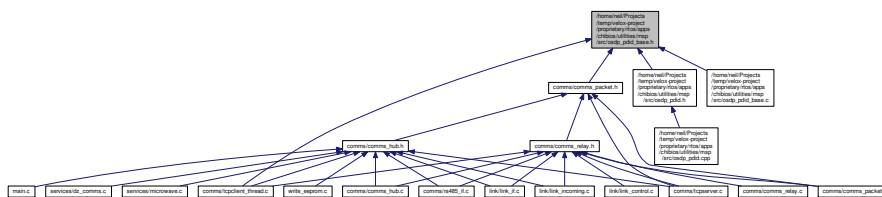
Encode and decode functionality for OSDP PDID packets.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_pdid_base.h:



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



Data Structures

- struct [osdp_pdid_t](#)

Macros

- `#define OSDP_PDID 0x45`
The ID used to define an ID Report Request packet.

Functions

- void [osdp_pdid_encode](#) ([osdp_pdid_t](#) *data, [uint8_t](#) *buffer, [int32_t](#) *len)
Translate the data structure to a packet for transmission.
- [int32_t](#) [osdp_pdid_decode](#) ([osdp_pdid_t](#) *data, [uint8_t](#) *buffer, [int32_t](#) len)
Translate a received packet into a data structure.

9.135.1 Detailed Description

Encode and decode functionality for OSDP PDID packets.

Author

neil

Definition in file [osdp_pdid_base.h](#).

9.135.2 Function Documentation

9.135.2.1 `int32_t osdp_pdid_decode ( osdp_pdid_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 52 of file `osdp_pdid_base.c`.

9.135.2.2 `void osdp_pdid_encode ( osdp_pdid_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

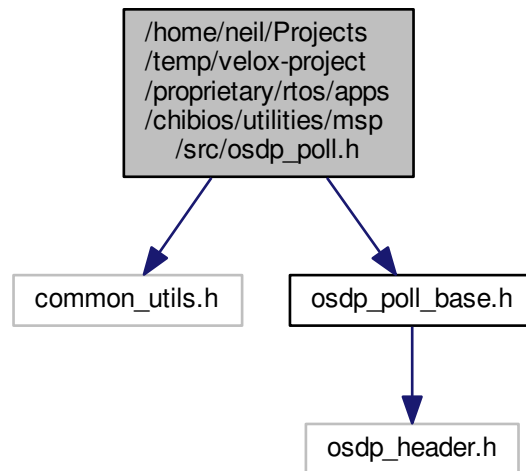
Definition at line 10 of file `osdp_pdid_base.c`.

9.136 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_poll.h](#) File Reference

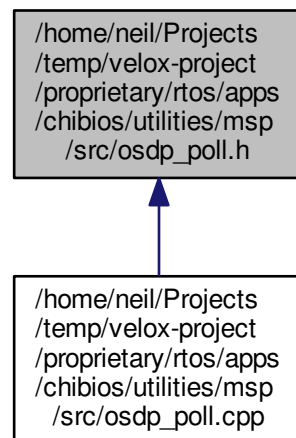
A concrete implementation of `osdp_data` for ISTAT requests.

```
#include "common_utils.h"
#include "osdp_poll_base.h"
```

Include dependency graph for osdp_poll.h:



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



Data Structures

- class [osdp_poll](#)

9.136.1 Detailed Description

A concrete implementation of `osdp_data` for ISTAT requests.

Functions

- void [osdp_poll_encode](#) (osdp_poll_t *data, uint8_t *buffer, int32_t *len)

Translate the data structure to a packet for transmission.

9.137.1 Detailed Description

A C implementation of POLL packet encode function.

Author

neil

Definition in file [osdp_poll_base.h](#).

9.137.2 Function Documentation

9.137.2.1 void [osdp_poll_encode](#) (osdp_poll_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

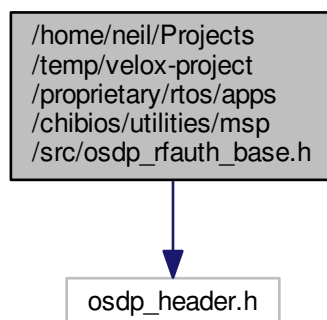
Definition at line 10 of file [osdp_poll_base.c](#).

9.138 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↵ _rfauth_base.h File Reference

A C implementation of the proprietary RF Authentication packet.

```
#include "osdp_header.h"
```

Include dependency graph for [osdp_rfauth_base.h](#):



<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 59 of file osdp_rfauth_base.c.

9.138.2.2 void osdp_rfauth_encode (osdp_rfauth_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

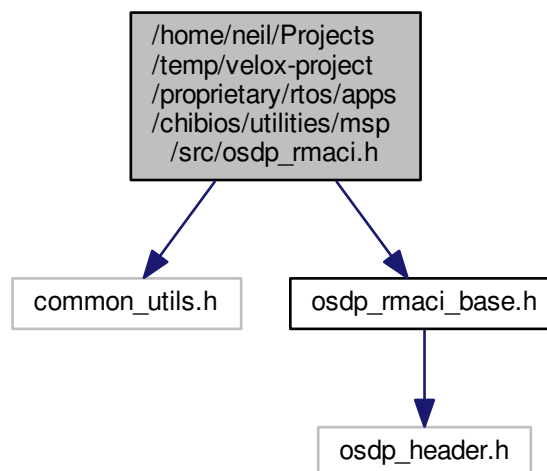
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

Definition at line 10 of file osdp_rfauth_base.c.

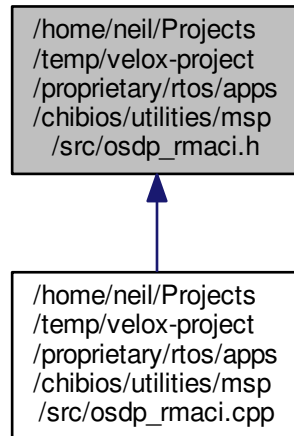
9.139 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_rmaci.h File Reference

A concrete implementation of osdp_data to transmit SCS initial MAC.

```
#include "common_utils.h"
#include "osdp_rmaci_base.h"
Include dependency graph for osdp_rmaci.h:
```



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



Data Structures

- class [osdp_rmaci](#)

9.139.1 Detailed Description

A concrete implementation of `osdp_data` to transmit SCS initial MAC.

Author

neil

Definition in file [osdp_rmaci.h](#).

9.140 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_rmaci_base.h](#) File Reference

The low-level encode/decode functions for `osdp_RMAC_I` functionality.

Author

neil

Definition in file [osdp_rmaci_base.h](#).

9.140.2 Function Documentation

9.140.2.1 `int32_t osdp_rmaci_decode ( osdp_rmaci_t * data, uint8_t * buffer, int32_t len )`

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 41 of file `osdp_rmaci_base.c`.

9.140.2.2 `void osdp_rmaci_encode ( osdp_rmaci_t * data, uint8_t * buffer, int32_t * len )`

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

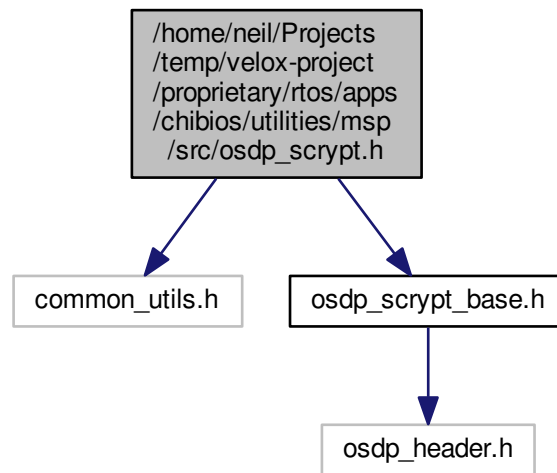
Definition at line 11 of file `osdp_rmaci_base.c`.

9.141 [/home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_↵](#) _script.h File Reference

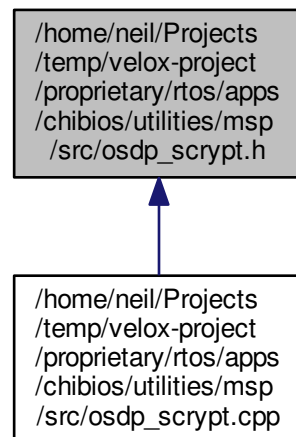
A concrete implementation of `osdp_data` to transmit SCS-CS cryptograms.

```
#include "common_utils.h"
#include "osdp_script_base.h"
```

Include dependency graph for osdp_scrypt.h:



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



Data Structures

- class `osdp_scrypt`

9.141.1 Detailed Description

A concrete implementation of `osdp_data` to transmit SCS-CS cryptograms.

Author

neil

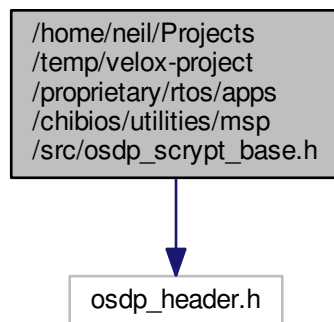
Definition in file [osdp_scrypt.h](#).

9.142 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/osdp_scrypt_base.h File Reference

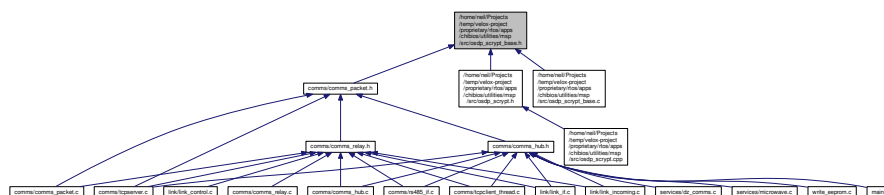
The low-level encode/decode functions for osdp_SCRYPT functionality.

```
#include "osdp_header.h"
```

Include dependency graph for osdp_scrypt_base.h:



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



Data Structures

- struct [osdp_scrypt_t](#)

Macros

- `#define` [OSDP_SCRYPT](#) 0x77

The ID used to define a SCS-CS Server Cryptogram packet.

Functions

- void [osdp_scrypt_encode](#) ([osdp_scrypt_t](#) *data, uint8_t *buffer, int32_t *len)
Translate the data structure to a packet for transmission.
- int32_t [osdp_scrypt_decode](#) ([osdp_scrypt_t](#) *data, uint8_t *buffer, int32_t len)
Translate a received packet into a data structure.

9.142.1 Detailed Description

The low-level encode/decode functions for osdp_SCRYPT functionality.

Author

neil

Definition in file [osdp_scrypt_base.h](#).

9.142.2 Function Documentation

9.142.2.1 int32_t osdp_scrypt_decode (osdp_scrypt_t * data, uint8_t * buffer, int32_t len)

Translate a received packet into a data structure.

Parameters

<i>data</i>	A struct containing the decoded output parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the input packet

Returns

The number of decoded bytes

Definition at line 40 of file [osdp_scrypt_base.c](#).

9.142.2.2 void osdp_scrypt_encode (osdp_scrypt_t * data, uint8_t * buffer, int32_t * len)

Translate the data structure to a packet for transmission.

Parameters

<i>data</i>	A struct containing the input parameters
<i>buffer</i>	An input data buffer used to store the packet
<i>len</i>	The length of the output packet

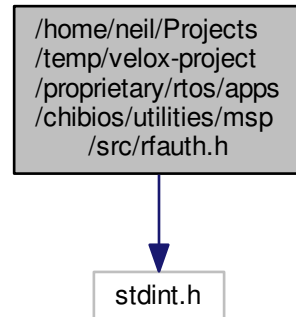
Definition at line 11 of file [osdp_scrypt_base.c](#).

9.143 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/rfauth.h File Reference

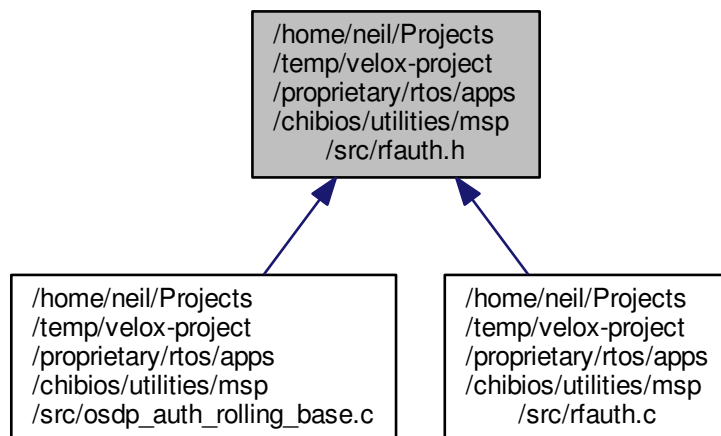
Encode and decode functionality for RF authentication.

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

Include dependency graph for rfauth.h:



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



Functions

- `int32_t rfauth_enc` (`uint8_t *key`, `uint32_t ctr`, `uint8_t *buffer`, `int32_t offset`, `uint32_t timestamp`, `uint32_t rolling`)
- `int32_t rfauth_dec` (`uint8_t *key`, `uint8_t *buffer`, `uint32_t *timestamp`, `uint32_t *rolling`)

9.143.1 Detailed Description

Encode and decode functionality for RF authentication.

Author

neil

Definition in file [rfauth.h](#).

9.143.2 Function Documentation

9.143.2.1 `int32_t rfauth_dec ( uint8_t * key, uint8_t * buffer, uint32_t * timestamp, uint32_t * rolling )`

Todo Switch to 256-bit keys using VaultIC

Definition at line 67 of file `rfauth.c`.

9.143.2.2 `int32_t rfauth_enc ( uint8_t * key, uint32_t ctr, uint8_t * buffer, int32_t offset, uint32_t timestamp, uint32_t rolling )`

Todo Switch to 256-bit keys using VaultIC

Definition at line 11 of file `rfauth.c`.

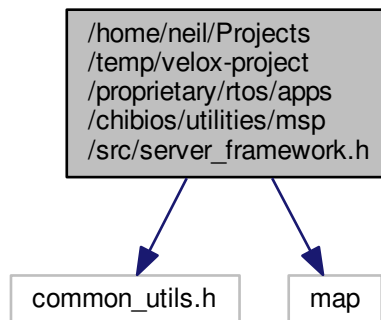
9.144 /home/neil/Projects/temp/velox-project/proprietary/rtos/apps/chibios/utilities/msp/src/server_framework.h File Reference

An abstract base class used to distribute resources throughout the system.

```
#include "common_utils.h"
```

```
#include <map>
```

Include dependency graph for `server_framework.h`:



Data Structures

- struct [ip_settings_t](#)
A structure defining IP settings.
- class [server_framework](#)
- class [server_framework_concrete](#)

9.144.1 Detailed Description

An abstract base class used to distribute resources throughout the system.

Author

neil

Definition in file [server_framework.h](#).