USER MANUAL

Software Development Kit (SDK) for the On-Board Computer (OBC)

OBC SDK – OPEN SOFTWARE

USER MANUAL

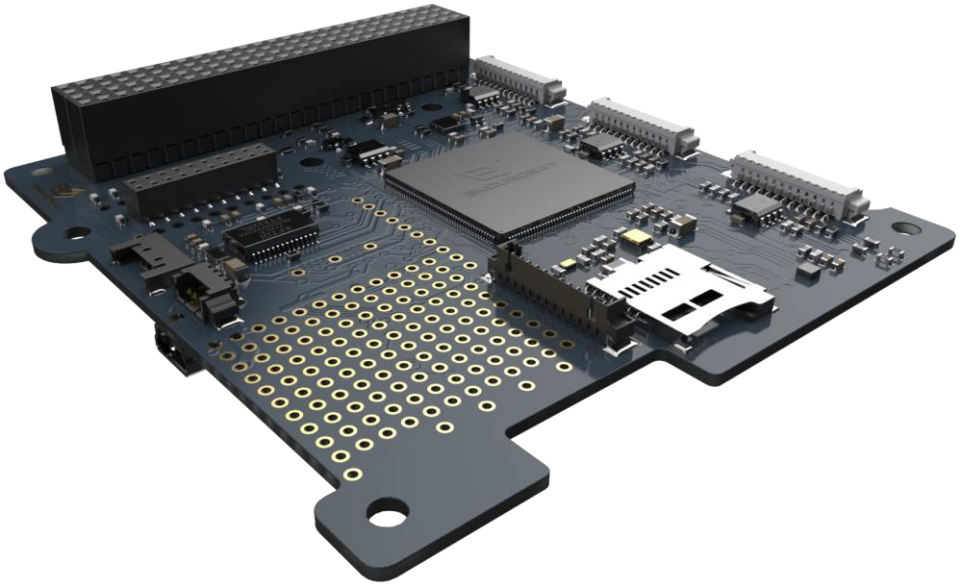


Figure 1 – EnduroSat OBC

1 CHANGE LOG

Date	Version	Note
22/06/2018	Rev 1	Initial document
01/10/2018	Rev 1.1	Section Real Time Clock, Program Control added, minor changes in overview section.
14/02/2019	Rev 1.2	Minor changes in section 4

2 LIST OF ABBREVIATIONS

A/D Converter: Analog Digital Converter
ADC : Analog Digital Converter
AHB : Advanced High Performance Bus
APB : Advanced Peripheral Bus
ART Accelator : Adaptive Real Time Accelator
B: Dedicated to BOOT0 Pin (Pin Abbreviation)
BCD : Binary Coded Decimal
BGA : Ball Grid Array
BJT : Bipolar Junction Transistor
BOR : Brownout Reset
BQFP : Bumpered Quad Flat Package
CAN : Controller Area Network
CF : Compact Flash
CMOS : Complementary Metal Oxide Semiconductor
CQFP : Ceramic Quad Flat Package
CRC : Cyclic Redundancy Check
CTS : Clear to Send
D/A Converter : Digital Analog Converter
DAC : Digital Analog Converter
DCE : Data Communication Equipment
DCMI : Digital Camera Interface
DFU : Device Firmware Upgrade
DMA : Direct Memory Access
DMIPS : Dhrystone Million Instructions Per Second
DSP : Digital Signal Processing
DTE : Data Terminal Equipment
EMI : Electromagnetic Interference
EMS : Electromagnetic Susceptibility
ESD : Electrostatic Discharge
ESR : Equivalent Series Resistance
ETM : Embedded Trace Macrocell
EXTI : External Interrupt
FET : Field Effect Transistor
FIFO : First In, First Out
FM+ : Fast Mode Plus (Pin Abbreviation)
FMC : Flexible Memory Controller
FPGA : Field Programmable Gate Array
FPU : Floating Point Unit
FPU : Floating Point Unit
FSMC : Flexible Static Memory Controller
FT : 5V Tolerant Input Output Pin (Pin Abbreviation)

FTB : Fast Transient Burst (Voltage)
FTf : 5V Tolerant Input Output Pin with FM+ capable (Pin Abbreviation)
GPIO : General Purpose Input Output
HSE : High Speed External (Oscillator/Clock)
HSI : High Speed Internal (Oscillator/Clock)
HVAC : Heating Ventilating and Air Conditioning
I : Input Only Pin (Pin Abbreviation)
I/O : Input Output Pin (Pin Abbreviation)
I/O : Input/Output
I2C : Inter Integrated Circuit, aka I squared C
I2S : Inter-IC Sound, Integrated Interchip Sound
IC : Input Capture
IC : Integrated Circuit
IRDA : Infrared Data Association
IWDG : Independent Watch Dog
JTAG : Joint Test Action Group
LAN : Local Area Network
LIN : Local Interconnect Network
LPR : Low Power Regulator
LQFP : Low Profile Quad Flat Package
LSE : Low Speed External (Oscillator/Clock)
LSI : Low Speed Internal (Oscillator/Clock)
MAC : Media Access Control (Address)
MCU : Micro Controller Unit
MII : Media Independent Interface
MIPS : Microprocessor without Interlocked Pipeline Stages
MIPS : Million Instructions Per Second
MISO : Master Input Slave Output (Serial Peripheral Interface Bus Abbreviation)
MMC : Multi Media Card
MOSI : Master Output Slave In (Serial Peripheral Interface Bus Abbreviation)
MPU : Memory Protection Unit
MR : Main Regulator
MSPS : Mega Sample Per Second
NC : Normally Closed
NC : Not Connected
NO : Normally Open
NRST : nRESET (Pin)
NVIC : Nested Vectored Interrupt Controller
OBC : On-Board Computer
OC : Output Compare
PCB: Printed Circuit Board
PDR : Power Down Reset
PHY : Physical (Layer)
PLC : Programmable Logic Controller
PLL : Phase Locked Loop
PM Bus : Power Management Bus

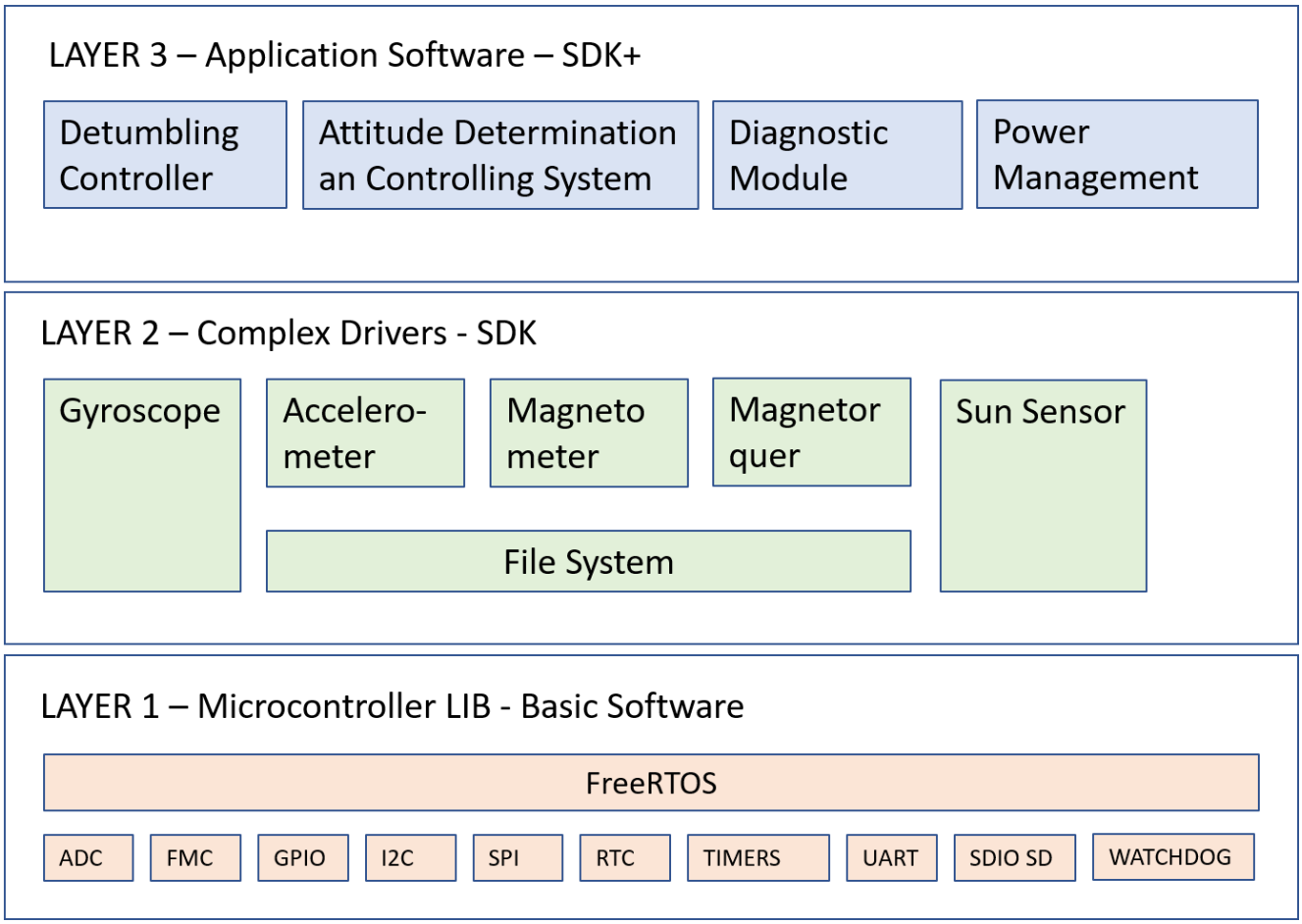
POR : Power On Reset
PPB : Private Peripheral Bus
PVD : Programmable Voltage Detector
PWM : Pulse Width Modulation
QFP : Quad Flat Package
RAM : Random Access Memory
RC : Resistor Capacitor
RMII : Reduced Media Independent Interface
RNG : Random Number Generator
RST : Bidirectional Reset Pin With Embedded Weak Pull Up Resistor
RTC : Real Time Clock
RTS : Request to Send
RTR : Ready to Receive
S : Supply Pin (Pin Abbreviation)
SCL : Serial Clock Line
SCLK : Serial Clock (Serial Peripheral Interface Bus Abbreviation)
SDA : Serial Data Line
SDK : Software Development Kit
SDIO : Secure Digital Input Output
SM Bus : System Management Bus
SMI : Serial Management Interface
SPI : Serial Peripheral Interface
SRAM : Static Random Access Memory
SS : Slave Select (Serial Peripheral Interface Bus Abbreviation)
SSCG : Spread Spectrum Clock Generation
SWD : Serial Wire Debug
TC: Standartd 3.3V Input Output Pin (Pin Abbreviation)
TIM : Timer
TPA : Trace Port Analyzer
TPIU : Trace Port Interface Unit
TQFP : Thin Quad Flat Package
TTa : 3.3V Tolerant Input Output Pin Directly Connected to ADC (Pin Abbreviation)
UART : Universal Asynchronous Receiver Transmitter
ULPI : Utmi Low Pin Interface
USART : Universal Synchronous Asynchronous Receiver Transmitter
USB OTG : USB On The Go
UTMI : USB 2.0 Transceiver Macrocell Interface
VBAT : Battery Voltage Supply (Pin)
VCC : Positive Supply Voltage (BJT)
VDD : Positive Supply Voltage (FET)
VEE : Negative Supply (Ground) (BJT)
VSS : Negative Supply (Ground) (FET)
WWDG : Windows Watch Dog

3 OVERVIEW

The software for the On-Board Computer (OBC) is designed in a 3-layer structure. The first layer is dedicated to software libraries for the microcontroller peripherals, including the real time operating system. This is everything required by a system developer.

Second layer – level 2 includes the software drivers for the sensors – gyroscope, accelerometer, magnetometer and sun sensors, and also for the actuators, in our case the magnetorquers. These are complex drivers and are designed specifically for the requirements of the EnduroSat satellite. This second level of abstraction provides complex, platform independent, encapsulated software drivers, which can be easily migrated to different microcontroller families for example. They are flexible and easily configurable, providing full functionality for the sensors. This package is a Software Development Kit (SDK) package.

The third level of abstraction is application layer 3. It can include any application software for satellite stabilization and control, diagnostics, power management etc. This level is not described in the current document because it is EnduroSat's intellectual property.



4 FIRST LAYER – BASIC SOFTWARE – MCU PERIPHERAL LIBRARY FUNCTIONS

static void MX_ADC1_Init(void)

brief	Make initialization of the Analog to Digital Converter Module 1
param[in]	-
param[out]	-
return	-

Three 12-bit analog-to-digital converters (ADCs) are embedded, and each ADC shares up to 16 external channels, performing conversions in the single-shot or scan mode. In scan mode, automatic conversion is performed on a selected group of analog inputs.
Additional logic functions embedded in the ADC interface allow:

- Simultaneous sample and hold
- Interleaved sample and hold

The ADC can be served by the DMA controller. An analog watchdog feature allows very precise monitoring of the converted voltage of one, some or all selected channels. An interrupt is generated when the converted voltage is outside the programmed thresholds. To synchronize A/D conversion and timers, the ADCs could be triggered by any of TIM1, TIM2, TIM3, TIM4, TIM5, or TIM8 timer.

static void MX_FMC_Init(void)

brief	Make initialization of the Flexible memory Controller Module
param[in]	-
param[out]	-
return	-

All devices embed an FMC. It has four Chip Select outputs supporting the following modes: PCCard/Compact Flash, SDRAM/LPSDR SDRAM, SRAM, PSRAM, NOR Flash and NAND Flash.
Functionality overview:

- 8-,16-, 32-bit data bus width
- Read FIFO for SDRAM controller
- Write FIFO
- Maximum FMC_CLK/FMC_SDCLK frequency for synchronous accesses is 90 MHz.

static void MX_GPIO_Init(void)

brief	Make initialization of the Flexible memory Controller Module
param[in]	-
param[out]	-
return	-

Each of the GPIO pins can be configured by software as output (push-pull or open-drain, with or without pull-up or pull-down), as input (floating, with or without pull-up or pull-down) or as peripheral alternate function. Most of the GPIO pins are shared with digital or analog alternate functions. All GPIOs are high-current-capable and have speed selection to better manage internal noise, power consumption and electromagnetic emission. The I/O configuration can be locked if needed by following a specific sequence in order to avoid spurious writing to the I/Os registers. Fast I/O handling allowing maximum I/O toggling up to 90 MHz.

static void MX_I2C1_Init(void)

brief	Make initialization of the Inter integrated Circuit Interface Module 1
param[in]	-
param[out]	-
return	-

static void MX_I2C2_Init(void)

brief	Make initialization of the Inter integrated Circuit Interface Module 2
param[in]	-
param[out]	-
return	-

static void MX_I2C3_Init(void)

brief	Make initialization of the Inter integrated Circuit Interface Module 3
param[in]	-
param[out]	-
return	-

Up to three I²C bus interfaces can operate in multimaster and slave modes. They can support the standard (up to 100 KHz), and fast (up to 400 KHz) modes. They support the 7/10-bit addressing mode and the 7-bit dual addressing mode (as slave). A hardware CRC generation/verification is embedded. They can be served by DMA and they support SMBus 2.0/PMBus.

static void MX_RTC_Init(void)

brief	Make initialization of the Inter integrated Circuit Interface Module 3
param[in]	-
param[out]	-
return	-

The real-time clock (RTC) is an independent BCD timer/counter. Dedicated registers contain the second, minute, hour (in 12/24 hour), week day, date, month, year, in BCD (binary coded decimal) format. Correction for 28, 29 (leap year), 30, and 31 day of the month are performed automatically. The RTC provides a programmable alarm and programmable periodic interrupts with wakeup from Stop and Standby modes. The sub-seconds value is also available in binary format. It is clocked

by a 32.768 kHz external crystal, resonator or oscillator, the internal low-power RC oscillator or the high-speed external clock divided by 128. The internal low-speed RC has a typical frequency of 32 kHz. The RTC can be calibrated using an external 512 Hz output to compensate for any natural quartz deviation.
Two alarm registers are used to generate an alarm at a specific time and calendar fields can be independently masked for alarm comparison. To generate a periodic interrupt, a 16-bit programmable binary auto-reload downcounter with programmable resolution is available and allows automatic wakeup and periodic alarms from every 120 µs to every 36 hours. A 20-bit prescaler is used for the time base clock. It is by default configured to generate a time base of 1 second from a clock at 32.768 kHz.

static void MX_SDIO_SD_Init(void)

brief	Make initialization of The Secure digital input/output interface (SDIO) Module
param[in]	-
param[out]	-
return	-

An SD/SDIO/MMC host interface is available, that supports MultiMediaCard System Specification Version 4.2 in three different databus modes: 1-bit (default), 4-bit and 8-bit. The interface allows data transfer at up to 48 MHz, and is compliant with the SD Memory Card Specification Version 2.0. The SDIO Card Specification Version 2.0 is also supported with two different databus modes: 1-bit (default) and 4-bit.
The current version supports only one SD/SDIO/MMC4.2 card at any one time and a stack of MMC4.1 or previous.
In addition to SD/SDIO/MMC, this interface is fully compliant with the CE-ATA digital protocol Rev1.1.

static void MX_SPI1_Init(void)

brief	Make initialization of The Serial peripheral interface (SPI) Module 1
param[in]	-
param[out]	-
return	-

static void MX_SPI2_Init(void)

brief	Make initialization of The Serial peripheral interface (SPI) Module 2
param[in]	-
param[out]	-
return	-

static void MX_SPI6_Init(void)

brief	Make initialization of The Serial peripheral interface (SPI) Module 6
param[in]	-
param[out]	-
return	-

The devices feature up to six SPIs in slave and master modes in full-duplex and simplex communication modes. SPI1, SPI4, SPI5, and SPI6 can communicate at up to 45 Mbits/s, SPI2 and SPI3 can communicate at up to 22.5 Mbit/s. The 3-bit prescaler gives 8 master mode frequencies and the frame is configurable to 8 bits or 16 bits. The hardware CRC generation/verification supports basic SD Card/MMC modes. All SPIs can be served by the DMA controller.
The SPI interface can be configured to operate in TI mode for communications in master mode and slave mode.

static void MX_TIM5_Init(void)

brief	Make initialization of The Timer Module 5
param[in]	-
param[out]	-
return	-

The STM32F42x include 4 full-featured general-purpose timers: TIM2, TIM5, TIM3, and TIM4.The TIM2 and TIM5 timers are based on a 32-bit auto-reload
up/downcounter and a 16-bit prescaler. The TIM2, TIM3, TIM4, TIM5 general-purpose timers can work together, or with the other general-purpose timers and the advanced-control timers TIM1 and TIM8 via the Timer Link feature for synchronization or event chaining. Any of these general-purpose timers can be used to generate PWM outputs. TIM2, TIM3, TIM4, TIM5 all have independent DMA request generation. They are capable of handling quadrature (incremental) encoder signals and the digital outputs
from 1 to 4 hall-effect sensors.

static void MX_UART4_Init(void)

brief	Make initialization of The Universal synchronous/asynchronous receiver transmitter Module 4
param[in]	-
param[out]	-
return	-

static void MX_USART1_UART_Init(void)

brief	Make initialization of The Universal synchronous/asynchronous receiver transmitter Module 4
param[in]	-
param[out]	-
return	-

static void MX_USART6_UART_Init(void)

brief	Make initialization of The Universal synchronous/asynchronous receiver transmitter Module 6
param[in]	-
param[out]	-
return	-

The devices embed four universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART3 and USART6) and four universal asynchronous receiver transmitters (UART4, UART5, UART7, and UART8).
These six interfaces provide asynchronous communication, IrDA SIR ENDEC support, multiprocessor communication mode, single-wire half-duplex communication mode and have LIN Master/Slave capability. The USART1 and USART6 interfaces are able to communicate at speeds of up to 11.25 Mbit/s. The other available interfaces communicate at up to 5.62 Mbit/s.
USART1, USART2, USART3 and USART6 also provide hardware management of the CTS and RTS signals, Smart Card mode (ISO 7816 compliant) and SPI-like communication capability. All interfaces can be served by the DMA controller.

void MX_WWDG_Init(void)

brief	Make initialization of The Window Watchdog Module
param[in]	-
param[out]	-
return	-

The window watchdog is based on a 7-bit downcounter that can be set as free-running. It can be used as a watchdog to reset the device when a problem occurs. It is clocked from the main clock. It has an early warning interrupt capability and the counter can be frozen in debug mode.

5 SECOND LAYER – DATA CONVERSION AND COMPLEX DRIVERS

void Process_Sensors (void const * argument)

brief	Make initialization of The OBC sensors and require status and values
param[in]	
param[out]	-
return	-

Function is called by tread with normal priority. It is executed initialization of magnetometer first and there are two options depending of the which magnetometer is mounted - HMC5883L or LIS3MDL.

5.1 Gyroscope functionality

There are upto three gyroscope sensors which measure rotational motion in each direction X,Y or Z and are mounted on specific EnduroSat panels. These are external sensors and are not placed on the OBC board. Please refer to the EnduroSat store to buy related panels if you want to use them.

status_t ADIS16265_Init(uint8_t Panel)

brief	Make initialization of Gyroscope_sensor mounted in panel (uint8_t Panel) and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	uint8_t Panel
param[out]	-
return	status_t param -

status_t ADIS16265_ReadReg16(uint8_t Address, uint16_t *data, uint8_t Panel)

brief	Register operation. Read 16 bit data (uint16_t *data) by address (uint8_t Address) from Gyroscope_sensor mounted on panel (uint8_t Panel) and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	uint8_t Address, uint8_t Panel
param[out]	uint16_t *data
return	status_t param -

status_t ADIS16265_WriteReg8(uint8_t Address, uint8_t data, uint8_t Panel)

brief	Refgister configuration. Write 8 bit data (uint8_t data) by address (uint8_t Address) on Gyroscope_sensor mounted on panel (uint8_t Panel) and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	uint8_t Address, uint8_t data, uint8_t Panel
param[out]	
return	status_t param -

status_t ADIS16265_WriteReg16(uint8_t Address, uint16_t data, uint8_t Panel)

brief	Register configuration. Write 16 bit data (uint16_t data) by address (uint8_t Address) on Gyroscope_sensor mounted on panel (uint8_t Panel) and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	uint8_t Address, uint16_t data, uint8_t Panel
param[out]	
return	status_t param -

status_t ADIS16265_GetTemperature(Temperature_t* tmp)

brief	Read Temperature from 3 Gyroscope_sensors X,Y,Z and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	-
param[out]	Temperature_t *tmp
return	status_t param -

status_t ADIS16265_GetAxesRate(AxesRaw_t* buff)

brief	Get raw values from 3 Gyroscope_sensors X,Y,Z and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	-
param[out]	AxesRaw_t* buff
return	status_t param -

status_t ADIS16265_GetAxesAngle(AxesRaw_t* buff)

brief	Get current angle values from 3 Gyroscope_sensors X,Y,Z and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	-
param[out]	AxesRaw_t* buff
return	status_t param -

5.2 Photosensor functionality

Photo sensors are not mounted on the OBC board, but can be accessed via OBC panel connectors if you buy EnduroSat 1U panels

void Panel_GetPhotodiodesLum(void)

brief	Get values form 6 photosensors mounted on the panels and update global array PanelLight[] with the data. Function itself is called subfunction Pan_PD_ADC_Measure(), which make ADC measurement on specific channel attached to each photo sensor
param[in]	-
param[out]	-
return	-

5.3 Magnetorquer functionality

Magnetorquers are inductive coils, integrated within specific EnduroSat panels. They are driven by a microcontroller PWM module using a power MOSFET in a H-bridge configuration. EnduroSat software can drive up to six magnetorquers X+,Y+,Z+,X-,Y- and Z-

status_t SetMagnetorque(uint8_t Panel, uint8_t perc, uint8_t dir)

brief	Set Magnetorquer in panel number (uint8_t Panel) with power 0-100% (uint8_t perc) and direction (uint8_t dir) and return status operation SEN_SUCCESS or SEN_ERROR
param[in]	uint8_t Panel, uint8_t perc, uint8_t dir
param[out]	-
return	status_t

void Magnetorquers_Update (Magnetorquer_Axis_t MT_level)

brief	Calculate magnetorquer value related to the dipole moment parameter, check direction and update power. Function called subfunction SetMagnetorque
param[in]	Magnetorquer_Axis_t MT_level – double X, Y, Z
param[out]	-
Return	-

void Boost_Magnetorquers (uint8_t Arrow)

brief	Put All Magnetorquers in maximum power in desired direction (uint8_t Arrow)
param[in]	uint8_t Arrow (Arrow =0 is minus Arrow >0 is plus)
param[out]	-
Return	-

void Stop_Magnetorquers (void)

brief	Turn off magnetorquers
param[in]	-
param[out]	-
Return	-

5.4 Magnetometer functionality

Please check if on your PCB is mounted HMC5883L or LIS3MDL to request proper software functions! Below are listed all software functions used in the current OBC software build, but on files *LIS3MDL_MAG_driver.h* and *.c* for LIS3MDL and *hmc5883l.c* for HMC5883L you can find more.

HMC5883L

ES_ReturnType Magnitometers_Init (Compass_mode_t CompassMode,

Compass_Measuring_mode_t CompassMeasuringMode,

Compass_ID_Def_t CompassIDDef,

Compass_Set_Range_t CompassSetRange,

Compass_Set_Bandwidth_t CompassSetBandwidth,

Compass_Samples_Averaged_t Compass_Samples_Averaged)

brief	Magnetometer sensor initialization. Function initialize magnetometer in continuous or sibgle operation, bias, range, bandwith and filter depth. At the end of operation is returned status ES_ReturnType
param[in]	Compass_mode_t CompassMode Compass_Measuring_mode_t CompassMeasuringMode, Compass_ID_Def_t CompassIDDef, Compass_Set_Range_t CompassSetRange, Compass_Set_Bandwidth_t CompassSetBandwidth, Compass_Samples_Averaged_t Compass_Samples_Averaged
param[out]	-
Return	ES_ReturnType

ES_ReturnType Magnitometers_Read_Data (Compass_Axis_t *MagOutData, Compass_Set_Range_t CompassSetRange)

brief	Magnetometer set range (Compass_Set_Range_t CompassSetRange) and data read (MagOutData) at same time. At the end of operation is returned status ES_ReturnType
param[in]	Compass_Set_Range_t CompassSetRange
param[out]	Compass_Axis_t *MagOutData
Return	ES_ReturnType

LIS3MDL

ES_ReturnType Magnitometers_LIS3MDL_Init(uint8_t Dev_No);

brief	Magnetometer sensor initialization. At the end of operation is returned status ES_ReturnType
param[in]	uint8_t Dev_No – I2C hexadecimal address of the magnetometer chip
param[out]	-
Return	ES_ReturnType

ES_ReturnType Magnitometers_LIS3MDL_Read_Data (Compass_Axis_t *MagOutData, uint8_t Dev_No);

brief	Magnetometer data read. At the end of operation is returned status ES_ReturnType
param[in]	uint8_t Dev_No – I2C hexadecimal address of the magnetometer chip
param[out]	Compass_Axis_t *MagOutData - value
Return	ES_ReturnType

5.5 Accelerometer functionality

Two accelerometer sensors are mounted on an OBC board. One is enough for the measurement of acceleration forces in all directions (X,Y and Z). The second can be used as backup. Full driver functions are presented in a software build – please refer to software functions in the *ais328dq_driver.h* and *.c* files.

status_t AIS328DQ_Init(uint8_t deviceAddress)

brief	Accelerometer sensor initialization
param[in]	uint8_t deviceAddress - I2C hexadecimal address of the accelerometer chip
param[out]	
Return	status_t

uint8_t AIS328DQ_ReadReg(uint8_t deviceAddr, uint8_t Reg, uint8_t* Data)

brief	Read data (uint8_t* Data) from accelerometer (uint8_t deviceAddr) on register address(uint8_t Reg)
param[in]	uint8_t deviceAddr, uint8_t Reg
param[out]	uint8_t* Data
Return	status_t

uint8_t AIS328DQ_WriteReg(uint8_t deviceAddress, uint8_t WriteAddr, uint8_t Data);

brief	Write data (uint8_t Data) to accelerometer (uint8_t deviceAddr) on address(uint8_t WriteAddr)
param[in]	uint8_t deviceAddress
param[out]	
Return	status_t

6 ESTTC COMMUNICATION PROTOCOL

EnduroSat has its own proprietary communication protocol called ESTTC (EnduroSat Telemetry and Telecommand). Communication between the OBC and external modules use this protocol over one of the suitable serial interfaces.

Read command

Byte No	0	1	2	3	4	5	6	7	…	n
	“E”	“S”	“+”	“R”	0x11	0…0xFF	DATA1	DATA2		DATAn
Description	symbol	symbol	Symbol	R- means that command is only for reading	OBC address in I2C bus	Command ID				

Write command

Byte No	0	1	2	3	4	5	6	7	8	…	n
	“E”	“S”	“+”	“W”	0x11	0…0xFF	0…0xFF	DATA1	DATA2		DATAn
Description	symbol	symbol	Symbol	W- means that command is updating parameters	OBC address in I2C bus	Command ID	Number of data bytes				

There is a task/tread parser for command requests and to execute activities related to command:

osThreadDef(myESTTC_UART, ESTTC_UART, osPriorityNormal, 0, 8*128);

osThreadCreate(osThread(myESTTC_UART), NULL);

Task is with normal priority and execute function: void ESTTC_UART(void const * argument)

A detailed list of commands can be found in file “OBC Sensor and actuators ESTTC protocol”.

7 FILE SYSTEM

I. OBC file system features

- OBC supports the FATFS file system, formatted on the MicroSD card. Communication is via the SDIO interface.
- FATFS reference: http://elm-chan.org/fsw/ff/00index_e.html
- Recommended card storage size is up to 4GB.
- The structure is a single-volume, folders and subfolders are not supported.
- File name format is ########.###.
- Only a Single file can be opened at a time.
- Supported interface is UART 115200,8,n,1. (Baud rate, Data bit, no parity control, one stop bit)
- Supported protocol is ESTTC.
- HEX format mentioned below is ASCII HEX representation of one byte
- Return Error codes description according FATFS:

FR_OK = 0,	(0) Succeeded
FR_DISK_ERR,	(1) A hard error occurred in the low level disk I/O layer
FR_INT_ERR,	(2) Assertion failed
FR_NOT_READY,	(3) The physical drive cannot work
FR_NO_FILE,	(4) Could not find the file
FR_NO_PATH,	(5) Could not find the path
FR_INVALID_NAME,	(6) The path name format is invalid
FR_DENIED,	(7) Access denied due to prohibited access or directory full
FR_EXIST,	(8) Access denied due to prohibited access
FR_INVALID_OBJECT,	(9) The file/directory object is invalid
FR_WRITE_PROTECTED,	(10) The physical drive is write protected
FR_INVALID_DRIVE,	(11) The logical drive number is invalid
FR_NOT_ENABLED,	(12) The volume has no work area
FR_NO_FILESYSTEM,	(13) There is no valid FAT volume
FR_MKFS_ABORTED,	(14) The f_mkfs() aborted due to any problem
FR_TIMEOUT,	(15) Could not get a grant to access the volume within defined period
FR_LOCKED,	(16) The operation is rejected according to the file sharing policy
FR_NOT_ENOUGH_CORE,	(17) LFN working buffer could not be allocated
FR_TOO_MANY_OPEN_FILES,	(18) Number of open files > _FS_LOCK
FR_INVALID_PARAMETER	(19) Given parameter is invalid

II. ESTTC file system commands description

1. Write the file list including file names, size and their total number to file “DirList.txt”. Any previously opened file is closed:

   Command: ES+D11FL[Wild Mask]<CR - carriage return > If omitted the default mask is *. [WildMask] format is according FATFS pattern conventions of f_findfirst() and f_findnext() functions.

Answer: ERR+LCF(code)<CR> - fail to create “DirList.txt” and error code.
Answer: ERR+LFF(code)<CR> - fail to find files and error code.
Answer: OK<CR> - “DirList.txt” created successfully.

Example:
ES+D11FL*<CR> - shows all files (no limitation is applied to name and extension of the files)
ES+D11FL*.jpg<CR> - shows all files with jpg extension
ES+D11FL[Wild Mask]<CR>

2. Close the currently opened file:
   Command: ES+D11FC<CR>
   Answer: OK<CR>

3. Create new file for writing with attributes FA_WRITE | FA_CREATE_ALWAYS and the specified file is opened:
   Command: ES+D11FO[filename]<CR>, where filename is according the above format.
   Answer: ERR+FNC(code)<CR> - fail to create the file with error code.v
   Answer: OK<CR> - file created successfully.

4. Write to the currently opened file specified number of bytes at specified position:
   Command: ES+D11FW[position][size]<CR>[data][fcs], where position is mandatory 0-padded 8-digit start position in the file for writing in HEX format, and size is mandatory 0-padded 4-digit number of bytes to be written in HEX format. Data is the bytes to be written to the file and fcs is one-byte checksum mod 256 over the previous data bytes in raw binary format.
   Answer: ERR+FIH<CR> - No opened file.
   Answer: ERR+FIP=fpos-fsize,par<CR> - invalid position/size parameters
   Answer: ERR+FIS(code)=fpos<CR> - invalid file position seek operation error code
   Answer: ERR+FTM<CR> - timeout in data reception
   Answer: ERR+FEC=num(fcs)<CR> – invalid received data checksum
   Answer: ERR+FWE=code<CR> - fail to write in file error code
   Answer: ERR+FWC=num(size)<CR> - invalid number of written bytes
   Answer: OK<CR>

   Example:
   ES+D11FOfile1.bin<CR>
   OK
   ES+D11FW000000000008<CR> 0x0, 0x1, 0x2, 0x3, 0x40, 0x50, 0x60, 0x70, 0x66
   Write at file position 00000000 8 bytes with values 0, 1,2,3,64,80,96,112
   and FCS = (1+2+3+64+80+96+112)%256 = 358%256 = 0x166%0x100 = 102 = 0x66
   OK
   ES+D11FC<CR>
   OK
   Create file “file1.bin”, write 8 bytes and close the file.

5. Find and open specified file for reading with attributes FA_READ | FA_OPEN_EXISTING:
   Command: ES+D11FI[filename]<CR>, where filename is according the above format.
   Answer: ERR+FNF(code)=filename<CR> - fail to find or open the file with error code
   Answer: OK<CR>

6. Read from currently opened file specified number of bytes at specified position:
   Command: ES+D11FR[position][size]<CR>, where position is mandatory 0-padded 8-digit start position in the file for writing in HEX format, and size is mandatory 0-padded 4-digit number of bytes to be written in HEX format.
   Answer: ERR+FIH<CR> - No opened file.
   Answer: ERR+FIP=fpos-fsize,size-num<CR> - invalid position/size parameters
   Answer: ERR+FIS(code)=fpos<CR> - invalid file position seek operation and error code
   Answer: ERR+FIR(code)=num<CR> - fail to read from file error code
   Answer: ERR+FRS(size)=num<CR> – invalid number of read bytes
   Answer: [data][fcs] - Data is the bytes read from the file and fcs is one-byte checksum mod 256 over the previous data bytes in raw binary format.

   Example:
   ES+D11FIfile1.bin<CR>
   OK
   ES+D11FR000000030005<CR>
   0x3, 0x40, 0x50, 0x60, 0x70, 0x63
   ES+D11FC<CR>
   OK
   Open file “file1.bin”, read 4 bytes from position 3 and close the file.

7. Delete a specified file:
   Command: ES+D11FD[filename]<CR>, where filename is according the above format.
   Answer: ERR+FDL[filename]<CR> - fail to delete the file.
   Answer: OK<CR> - file deleted successfully.

8. Calculate and report the checksum and size of a specified file:
   Command: ES+D11FS[filename]<CR>, where filename is according the above format.
   Answer: ERR+FNF(code)=filename<CR> - fail to open file and error code.
   Answer: ERR+FIR(code)=num<CR> - fail to read from file error code and read bytes number.
   Answer: ERR+FRS=num<CR> – invalid number of read bytes
   Answer: OK+[checksum] [size]<CR>, where checksum is 0-padded 8-digit sum mod 2^32 of the file content by bytes in HEX format, and size is 0-padded 8-digit size of the file in HEX format.

   ES+D11FSfile1.bin<CR>
   OK+00000166 00000008<CR>

8 REAL TIME CLOCK

Commands related to RTC according to ESTTC protocol are following:

- Command ID: 0x31 (Get Real Time from OBC Real time clock module)

ESTTC command: ES+R1131<CR> (No additional ”DATAn” is required)

Return: OK TIME HH : MM : SS, where HH is hours 0..24, MM are minutes 0..59, SS are seconds 00..59.

Example:
ES+R1131<CR>
Return: OK TIME 15 : 55 : 58

- Command ID: 0x32 (Set Real Time to OBC Real time clock module)

DATA: ASCII format from ‘0’ to ‘9’. DATA1 and DATA2 are for hour format, DATA3 and DATA4 are for minutes, DATA5 and DATA6 are for the seconds.

Return: OK TIME HH : MM : SS, where HH is hours 0..24, MM are minutes 0..59, SS are seconds 00..59.

Example: Let’s set time 16:32:10
ESTTC command: ES+W113206313633323130<CR> (11- OBC address, 32-current command, 06 - six bytes are sent, 3136 – “16” hours, 3332 – “32” minutes, 3130 – “10” seconds)

Return: OK TIME 16 : 32 : 10

If there is unsuccessful operation, returned message is “ERR exe“.

- Command ID: 0x33 (Get Date from OBC Calendar)

ESTTC command: ES+R1133<CR> (No additional ”DATAn” is required)

Return: OK DATE YY / MM / DD, where YY is year 0..50, MM is the month 1..12, DD is date 0..31.

Example:
ES+R1133<CR>
Return: OK DATE YY/MM/DD 17 / 11 / 22

- Command ID: 0x34 (Set Date to OBC Calendar)

DATA: ASCII format from ‘0’ to ‘9’. DATA1 and DATA2 are for year, DATA3 and DATA4 are for month, DATA5 and DATA6 are for the date.

Return: OK DATE YY/MM/DD, where YY is year 0..50, MM is the month 1..12, DD is date 0..31.

Example: Let’s set date 7-Jan-2018
ESTTC command: ES+W113406313830313037<CR> (11- OBC address, 34-current command, 06- four bytes are sent, 3138 – “18” year, 3031 – “01” month, 3037 - “07” date)

Return: OK DATE YY/MM/DD 18 / 1 / 7

If there is unsuccessful operation, returned message is “ERR exe“.

9 PROGRAM CONTROL

OBC support two working modes – bootloader and application. Commands that are used to activate them are:

ES+W117F010B <CR> Return: OK+BOOT	Bootloader mode
ES+W117F010A <CR> Return: OK+APPL	Application mode

Additionally, user has an option to receive current mode and software version of OBC via the command:

ES+R117F <CR>	Current mode, software version

In order to update OBC Application version using specified file command below can be used:

ES+D11FA[filename]<CR>, see section seven.
Answer: ERR+INAPP<CR> - the command has been executed within an active application.
Answer: ERR+FNF(code)=filename<CR> - fail to open file and error code.
Answer: OK+[size]<CR>, where size is 0-padded 8-digit size of the file in HEX format.
Answer: ERR+FU<CR> - fail to unlock the flash memory for erasure.
Answer: ERR+FBSEcode@snum<CR> - fail to erase sector number and error code.
Answer: ERR+FBcode addr=beg size=fsize<CR> - memory blank check failure
Answer: ERR+FIR(code)=num<CR> - fail to read from file error code and read bytes number.
Answer: ERR+FRS=num<CR> – invalid number of read bytes
Answer: ERR+FWcode<CR> - fail to write in flash memory error code
Answer: OK+[checksum] [size]<CR>, where checksum is 0-padded 8-digit sum mod 2^32 of the flash memory content by bytes in HEX format, and size is 0-padded 8-digit size of the file in HEX format.

NOTE 1: Do not use this command without ENDUROSAT approval !
NOTE 2: Any answer including “ERR” terminates the execution of the command
NOTE 3: Any answer excluding “ERR” and “OK” may not be parsed and is for debug purposes
Command: ES+D11FA[filename]<CR>, where filename is according the above format.