# foxBMS

## *Release Tue Sep 3 14:53:10 2019 UTC*

For the online version of the documentation, see https://docs.foxbms.org/en/latest/index.html

A software tool called PCAN-View can be used to control the CAN communications when PCAN-USB is used.

**Sep 03, 2019**

# General Information

Table of contents

Welcome to the foxBMS documentation of the first modular open source BMS development platform from the Fraunhofer IISB. foxBMS is a free, open and flexible development environment to design state-of-the-art complex battery management systems.

**Note:** The present version of the foxBMS Sphinx documentation is `1.6.3`. It is available in HTML, PDF and EPUB formats through the bottom left menu on Read the Docs. More detailed information on the version history can be found in the *Releases* section. This documentation was generated on 2019-09-03 14:53:17.

**Warning:** The foxBMS hardware and software is under permanent development. The free and open research and development platform foxBMS as presented in the documentation hereafter is not provided to be used without any adaptions (e.g., to fulfill mandatory regulations) in consumer products, electric vehicles, production environments or any similar usages: it is only intended to be used by skilled professionals trained in designing battery system prototypes.

The documentation is divided in 4 parts:

1. *General Information*
2. *Getting Started*
3. *Software Documentation*
4. *Hardware Documentation*

In the first part of the documentation, general information about foxBMS can be found (*General Information*):

- Information about the versions and the related release notes of foxBMS
- Licenses of the hardware, software and document parts of foxBMS
- Motivation behind the foxBMS initiative
- Overview of the hardware and software of foxBMS
- Roadmap of the upcoming hardware and software components
- Team involved in the development of foxBMS

In the second part of the documentation, the *Getting Started* guide can be followed to commission foxBMS rapidly. For getting foxBMS desktop software, obtaining and compiling the sources, and setting up a short test of hard- and software the minium sections to be read are sections 1 to 4 of *Getting Started*

- The location of the electronic schematic and layout files in the HTML documentation is indicated
- The hardware guide describes how the hardware is structured and works
- The software guide explains how to configure and flash the software
- The CAN documentation shows how to communicate with the system via the CAN bus

To study and modify the software, the *Software Documentation* presents the structure of the software and of the most important modules, how the basic tasks are running and how to call user-defined functions. Further, a FAQ is provided to answers the most common questions. A detailed description of the software functions and variables can be found in the Doxygen documentation.

Finally, the *Hardware Documentation* gives the details needed to understand the foxBMS hardware.

# CHAPTER 1

# Releases

The current **software version** of foxBMS is `1.6.3`.

The latest **hardware release** versions are:

- BMS-Master Board: `1.0.6`
- BMS-Extension Board: `1.0.5`
- BMS-Interface Board: `1.9.4`
- Slave 12-cell board: `2.1.7`
- Slave 18-cell board: `1.1.5`

**Note:** EEPROM module from **v1.5.0** is **NOT** compatible with previous versions due to the changed memory layout because of added timestamps! `Data written by a privious software cannot be read by the newer version.`

**Note:** foxBMS uses now python3. The foxconda3 installer is found at https://iisb-foxbms.iisb.fraunhofer.de/foxbms/. foxconda3 must be installed to C:\foxconda3!

**Note:** Until version **v1.1.0**, the foxBMS software was split into several repositories, with an independent version number for each repository. Starting from version **v1.5.0**, the foxBMS software is contained in only one repository, with one version number!

The following tables describe the different versions of foxBMS that were released. The first line is the most recent one, the last one the oldest one.

**Versions 1.5.0 and higher:**

| foxbms | foxconda | Release date |
|---|---|---|
| v1.6.3 | 3.0.3 | 2019-09-03 |
| v1.6.2 | 3.0.3 | 2019-07-05 |
| v1.6.1 | 3.0.3 | 2019-06-19 |
| v1.6.0 | 3.0.3 | 2019-06-19 |
| v1.5.5 | 3.0.3 | 2019-01-25 |
| v1.5.4 | 3.0.3 | 2018-12-14 |
| v1.5.3 | 3.0.3 | 2018-12-07 |
| v1.5.2 | 3.0.3 | 2018-10-26 |
| v1.5.1 | 3.0.3 | 2018-10-09 |
| v1.5.0 | 3.0.3 | 2018-09-25 |

**Versions before 1.5.0:**

**Note:** The following table is only presented as information on previous foxBMS versions. This software structure with different software repositories is not used anymore.

| Documentation | Embedded Software | | | | | foxBMS Hardware | | | | foxConda |
|---|---|---|---|---|---|---|---|---|---|---|
| | Primary | Secondary | Common | HAL | FreeRTOS | Master Board | Interface Board | Slave Board | Extension Board | |
| 1.0.3 | 1.1.0 | 1.1.0 | 1.1.0 | 1.0.1<br>1.0.0 | 1.1.0 | 1.0.2 | 1.0.1 | 2.1.0<br>2.0.3<br>1.0.1 | 1.0.2 | 1.0.0 |
| 1.0.2 | 1.0.2<br>1.0.1<br>1.0.0 | 1.0.1<br>1.0.0 | 1.0.2<br>1.0.1<br>1.0.0 | 1.0.1<br>1.0.0 | 1.0.0 | 1.0.2 | 1.0.1 | 2.1.0<br>2.0.3<br>1.0.1 | 1.0.2 | 1.0.0 |
| 0.5.2 | 0.5.2 | 0.5.2 | | | | 1.0.2 | 1.0.1 | 2.1.0<br>2.0.3<br>1.0.1 | 1.0.2 | 0.5.0 |
| 0.5.1 | 0.5.1 | 0.5.1 | | | | 1.0.2 | 1.0.1 | 1.0.1 | 1.0.2 | 0.5.0 |
| 0.5.0 | 0.5.0 | 0.5.0 | | | | 1.0.2 | 1.0.1 | 1.0.1 | 1.0.2 | 0.5.0 |
| 0.4.4 | 0.4.4 | 0.4.4 | | | | 1.0.1 | 1.0.1 | 1.0.1 | 1.0.1 | 0.4.3 |
| 0.4.3 | 0.4.3 | 0.4.3 | | | | 1.0.0 | 1.0.0 | 1.0.0 | 1.0.0 | 0.4.3 |
| 0.4.2 | 0.4.2 | 0.4.2 | | | | 1.0.0 | 1.0.0 | 1.0.0 | 1.0.0 | 0.4.0 |
| 0.4.1 | 0.4.1 | 0.4.1 | | | | 1.0.0 | 1.0.0 | 1.0.0 | 1.0.0 | 0.4.0 |
| 0.4.0 | 0.4.0 | 0.4.0 | | | | 1.0.0 | 1.0.0 | 1.0.0 | 1.0.0 | 0.4.0 |

# CHAPTER 2

# Changelog

**Release 1.6.3**

Software:

- Toolchain:
  - the flash process sometimes did not work correctly for the secondary MCU. The robustness of the *_ack* function was generally improved and flashing both, primary and secondary, MCU should no longer fail.
  - if foxBMS was not developed inside a git repository, building the binaries failed. For these cases building now succeeds and the binary is still runnable. This requires the definition `BUILD_ALLOW_DIRTY_STARTUP` to be set to `1` (in `general.h` for both, primary and secondary MCU). Defining `BUILD_ALLOW_DIRTY_STARTUP` to `1` is the default setting.
- Bugfixes:
  - minimum, maximum and average SOC were identically if Coulomb counting feature of current sensors was used (`sox.c/h`)
  - recommended safety limit flag for undertemperature in charge direction was never set. (`diag_cfg.c`)
  - diagnosis system monitoring error occured if `ISOGUARD`-module was disabled (`isoguard.c`)
  - current sensor detection failed always if triggered mode was selected, as CAN TX messages were activated after check for current sensor (`sys.c`)
- Enhancements:
  - changed file structure to allow type definitions to be used as return values for static function prototypes (`cansignal_cfg.c`)

Hardware:

- none

Documentation:

- none

**Release 1.6.2**

Software:

- Toolchain:
  - added cpplint configuration file
  - added cppcheck configuration file
  - added flake8 configuration file
  - added busmaster project file
- Bugfixes:
  - cell voltages for module 4 for cells 12 and upwards were not transmitted via CAN (`cansignal_cfg.c`)
  - ISO_MeasureInsulation() did not correctly utilize DIAG_SysMonNotify(), which could lead to initialization errors (`isoguard.c`)
  - cell voltages and temperatures are now depicted in foxBMS-GUI if more than eight modules are selected (`foxbms_interface.py`)
  - compiling primary or secondary MCU binaries without COM module enabled led to a compile error
  - SPI chipselect pin for FPGA extension board was erroneously set in EEPROM module (`eepr.c`)
- Enhancements:
  - source code cleanup of interlock module (`interlock.c/h`)

Hardware:

- none

Documentation:

- fixed pinout of temperature sensor connectors X201 and X202 for 18-cell Slave `v1.1.3 and above`

---

**Release 1.6.1**

Software:

- Toolchain:
  - none
- Bugfixes:
  - fixed the extension of the startup scripts (`*.S` to `*.s`)
- Enhancements:
  - none

Hardware:

- none

Documentation:

- none

**Release 1.6.0**

Software:

- Toolchain:
  - Updated to `waf-2.0.15` (from `waf-2.0.14`)
  - `size` is now implemented as a waf-feature to speed up build times
  - foxBMS custom waf tasks displayed wrong information in the terminal about the running processes (e.g., `Compiling` instead of `Creating hex file`)
  - fixed a task order constraint when building the elf file. For details see the updated build documentation (see section *Build Process*).
  - added a simple test, that verifies that low level drivers do not relay on higher level modules (e.g., FreeRTOS, database, etc.). A project can be tested by running `python tools\waf build_primary_bare` or `python tools\waf build_secondary_bare` (see section *Build Process*).
  - removed unnecessary `run_always` attribute to reduce build time
  - rewrote the build of libraries. Now libraries can be build independent from the project configuration. Libraries can then later be added to the project as needed with the `configure --libs=...` command (see section *Build Process*).
  - fixed `clean_libs` command as it did not remove all `build_libs` artifacts
  - if an include directory does not exist, an error is raised
  - if a directory is included more than once, an error is raised
  - some build attributes of `bld.env` inside the `wscript` s haven been renamed (`__inc_hal` to `hal_dirs`, `__inc_FreeRTOS` to `FreeRTOS_dirs`, `__bld_common` to `common_dir`, `__sw_dir` to `es_dir` and `__bld_project` to `mcu_dir`
  - an error is raised if the path to the foxBMS project directory contains whitespace
  - an error is raised if a comparison between signed and unsigned integers is used
  - ensured compatibility with PyYAML v5.1 by supplying a Loader-parameter to calls of yaml.load()
  - fixed generation of .hex file to only contain flash content
  - raise the `switch`-warning to error in GCC
  - add a configuration for cppcheck to the repository
  - raise the `type-limits`-warning to error in GCC
  - raise the `double-promotion`-warning to error in GCC
- Bugfixes:
  - in the function `LTC_RX_PECCheck()`, the LTC PEC (packet error code) check of the last module in the daisy-chain was overriding the PEC check of all preceding modules. If the PEC of the last module was correct, all other PECs were detected as correct, even if some errors were present (`ltc.c`)
  - fixed compile error, when setting define `MEAS_TEST_CELL_SOF_LIMITS` to `TRUE` (`bms.c`)
  - some variables used to store database content at the beginning of the `BMS_Trigger()` function were defined as local variables. This could lead to a stack overflow if a high number of modules was configured. These variables have been defined as static to solve this issue (`bms.c`)

  - the `DIAG_GeneralHandler` has been removed. The `DIAG_Handler` must be used for all diagnosis instead, except for the diagnosis of the contactors, which is handled by `DIAG_ContHandler`.
  - `DIAG_Handler` returned wrong value if it was called when an error has already been detected (`diag.c`)
  - If more cell voltages or temperatures were defined for CAN transmission than for measurement in the battery system, during transmission, the array boundaries of the local variables storing database entries were violated. now boundary violations are checked and default values are sent for CAN signals not corresponding to existing measurements. (`cansignal_cfg.c/h`)
  - fixed incorrect array offset mapping CAN1 RX messages to respective CAN1 RX signals (`cansignal.c`)
  - moved fuse state error handling from `CONT` module to `BMS` module to avoid `CONT` state machine switching into error state without `BMS` state machine transitioning into error state
  - fixed an implicit cast to uint that prevented working protection of the battery cells against overdischarge and -charge below zero degrees (`bms.c`)
  - moved checksum feature to main `wscript`
  - added initialization state flags to contactor-, BMS- and balancing-statemachine in order to fix race-conditions between these statemachines and the sys-statemachine (`sys.c/h`, `bal.c/h`, `bms.c/h`, `contactor.c/h`)
  - fixed overlapping signals in dbc file for CAN message CAN_Cell_voltage_M2_0
  - fixed function `BMS_CheckCurrent()`. If contactors opened in case of over-current, error flags remained set in spite of the current being back to zero.
  - allow for current thresholds in `BMS_CheckCurrent()` differing from the cell-limits and adaptable to both charge and normal powerline (`bms.c`, `contactor.c/h`)

- Enhancements:
  - reimplemented UART COM Decoder as a non-realtime background task for easier use
  - removed direct register access in UART module to improve portability (`uart.c/h`)
  - added plausibility module to check pack voltage (`plausibility.c/h`)
  - added plausibility module to check cell voltage and cell temperature (`slaveplausibility.c/h`)
  - the Python wrapper and DLL needed to run the graphical user interface with a CAN-adapter from the company Peak are now redistributed with foxBMS. Before using them, the conditions in the file `readme.txt` in `tools\gui` must be read and accepted.
  - updated STM32 HAL to version 1.7.4 and CMSIS to version 2.6.2
  - rewrote CAN driver to work with new HAL CAN module introduced in HAL version 1.7.0 (`can.c/h`, `can_cfg.c/h`)
  - updated FreeRTOS to version 10.2.0 and adapted FreeRTOSConfig.h accordingly
  - added dedicated datasheet files for EPCOS B57251V5103J060, EPCOS B57861S0103F045 and Vishay NTCALUG01A103G NTC sensors to calculate cell temperatures using either a look-up table or polynomial approximation
  - the diagnosis modules of primary and secondary were unified and moved to mcu-common
  - error and safe operating area flags are now written periodically (1ms) to prevent erroneous database operations due to concurrency effects
  - moved #define to configure current sensor response timeout from `cansignal_cfg.h` to `batterysystem_cfg.h`

  - moved #define to select if current sensor is in cyclic or triggered mode from `can_cfg.h` to `batterysystem_cfg.h`
  - added simple diag-function that allows to track the call period of system tasks
  - moved linker scripts from `src` to `general\config\STM32F4xx`
  - moved FreeRTOS configuration headers from `src\general\config` to `src\general\config\FreeRTOS`
  - removed `MCU_0_` and `MCU_1_` from the pin defines in `io_mcu_cfg` to increase the readability of the drivers
  - added deep-discharge flag that gets set if the deep-discharge voltage limit is violated. Flag is stored in non-volatile backup SRAM and can only be reset with CAN debug message. This prevents closing the contactors before the affected cell has been replaced
  - added support for FreeRTOS runtime stats. The stats can be accessed by the new 'printstats' command in the COM module.
  - added state transition functions for ltc-statemachine to reduce code size (`ltc.c`)
  - added stack overflow-handler that can be used for debugging and detecting stack overflows during development
  - disabled dynamic allocation for operating system, removed heap-implementation and switched to static allocation for operating system components
  - information about the git repository from which the binaries are built is included in the binaries. The define `BUILD_ALLOW_DIRTY_STARTUP` has been added in `general.h` to allow or disallow the startup of the BMS in case of a non clean repository.
  - set error flag if current flows in spite of all contactors being open (`bms.c`)
  - added support for TCA6408A port expander in `LTC` module (write output pins and read input pins) (from Texas Instrument)
  - added decoding for up to 18 cell voltages per module in foxBMS interface

Hardware:

- removed version number from hardware file names
- Slave 12-cell v2.1.7
  - EMI layout improvements (targeting UN ECE R10 Revision 5)
  - added RC filters on NTC sensor inputs
  - replaced linear regulators for LTC6811 5V supply with DC/DC converters
  - added circuit for switching off 5V DC/DC converters in LTC sleep mode, thus reducing the current consumption to less than 20µA
- Interface LTC6820 v1.9.4
  - replaced connectors J500 and J501 with TE 534206-4 due to clearance issues in component placement

Documentation:

- added missing unit information for some CAN signals in section `Communicating with foxBMS`
- added a section on how to configure `conda` to work behind a proxy.

---

**Release 1.5.5**

Software:

- Toolchain:
  - Updated to `waf-2.0.14` (from `waf-2.0.13`)
- Bugfixes:
  - fixed UART frame error due to floating RX pin by enabling pull-up in the MCU
  - fixed reading wrong entry from database when checking battery system current against SOF limits (`bms.c`)
  - the flag SPI transmit_ongoing was reset incorrectly after SPI dummy byte was transmitted. This lead to invalid measured cell voltages if the daisy-chain was too long (i.e., more than 10 BMS-Slaves in the daisy-chain)
  - enabled simultaneous measurement of lithium-coin-cell V_bat and MCU temperature in ADC module
  - fixed error calculating MCU temperature in ADC module
  - balancing threshold for voltage-based balancing was set in the wrong place: it is now set in the function BAL_Activate_Balancing_Voltage (`bal.c`)
- Enhancements:
  - database entries are initialized with 0 to prevent undefined data if entries are read before valid values are written into the database (`database.c`)

Hardware:

- BMS-Slave 18-cell v1.1.5
  - EMI layout improvements (targeting UN ECE R10 Revision 5)
  - adapted component variations to simplify the management of component variations in Altium Designer
  - replaced DC/DC converter power inductor to comply with AEC-Q
  - added circuit for switching off DC/DC converters in LTC sleep mode, thus reducing the current consumption to less than 20µA
  - added pull-ups on GPIOs 6-9 of the LTCs (open-drain outputs) to enable them to be used as digital I/O

Documentation:

- BMS-Interface: fixed pin 11 in the pinout of the connectors for version 1.2.0 and above
- Updated BMS-Slave 18-cell hardware documentation for version 1.1.5
- Updated year in copyright
- Fixed some wrong @file attributes in doxygen comments

---

**Release 1.5.4**

Software:

- Toolchain:
  - Added a Python script that implements a graphical user interface to communicate with foxBMS. The instructions in the README.md file supplied with the script must be followed.
  - Removed obsolete `build.py` wrapper
  - Updated to `waf-2.0.13` (from `waf-2.0.12`)
  - Fixed a build error when using the `build_all` option

- Bugfixes:
  - In BMS module, wait time between error request to contactors and open request to interlock was increased. Otherwise, interlock opened before contactors were open. If this is the case, both contactors open at the same time without any delay between first and second contactor.
  - Fixed error in reading of interlock feedback.
- Enhancements:
  - none

Hardware:

- none

Documentation:

- none

---

**Release 1.5.3**

Software:

- Toolchain:
  - raised compiler warning `[-Werror=comment]` to error level
  - write compiler macros to header file for improved eclipse support
- Bugfixes:
  - fixed a bug, that caused the mcu temperature for primary and secondary mcu to be never updated.
  - fixed a bug, that caused the coin cell voltage of the primary mcu to be never updated.
  - rewrite of struct `DIAG_RETURNTYPE_e`. The enumeration had non-consecutive numbering and potentially dangerous typo in duplicate enum (`DIAG_HANDLER_RETURN_ERR_OCCURED = 2` and `DIAG_HANDLER_RETURN_ERR_OCCURRED = 4`).
  - fixed a bug, that diagnosis entry for a voltage violation of the maximimum safety limit wrote to wrong database entry.
  - NVRAM module was compiled twice for primary mcu. Once it was compiled by mcu-common module and once again in mcu-primary module). Now compiled only once by mcu-common module as on mcu-secondary.
- Enhancements:
  - debug printing is replaced by `printf` for easier and more versatile usage
  - added additional basic math macros (e.g., LN10, PI etc.) in `foxmath.h`
  - Fuse state is now monitored. Fuse can be placed in NORMAL and/or CHARGE path. Added flag to `CAN0_MSG_SystemState_2` message
  - added support to build and link multiple libraries
  - added warning flag if MCU die temperature is outside of operating range to `CAN0_MSG_SystemState_2` message
  - added warning flag to replace coin cell if measured coin cell voltage is low to `CAN0_MSG_SystemState_2` message
  - added daisy-chain communication error flags to `CAN0_MSG_SystemState_2` message

  - added error flag if an open voltage sense wire is detected

Hardware:

- none

Documentation:

- updated library build documentation
- updated .dbc file

---

**Release 1.5.2**

Software:

- Toolchain:
  - updated to `waf-2.0.12` (from `waf-2.0.11`)
- Bugfixes:
  - fixed bug that delay after SPI wake-up byte was not long enough
- Enhancements:
  - increased CPU clock frequency from 168MHz to 180MHz
  - increased SPI bitrate from 656.25kHz to 703.125kHz
  - added CAN boot message with SW-version and flash checksum (0x101)
  - CAN messages are now always sent, even if system error was detected
  - foxBMS SW-version requestable via CAN (request ID: 0x777, response ID: 0x101)
  - added insulation error flag to `DATA_BLOCK_ERRORSTATE_s`
  - configurable behavior if contactors should be open on insulation error or not
  - separate configurable precharging for charge/discharge path possible

Hardware:

- Master v1.0.6
  - adapted CAN filter circuit for improved fault tolerance at short of CAN_L to GND or CAN_H to supply
- Extension v1.0.5
  - adapted CAN filter circuit for improved fault tolerance at short of CAN_L to GND or CAN_H to supply

Documentation:

- updated instruction for flashing primary MCU
- updated FAQ section

---

**Release 1.5.1**

Software:

- Toolchain:
  - toolchain compatible with POSIX operating systems
  - updated to `waf-2.0.11` (from `waf-2.0.10`)

  - fixed missing files in eclipse workspace (CHANGELOG.rst and compiler-flags.yml)
- Bugfixes:
  - fixed bug updating BKPSRAM values to EEPROM: BKPSRAM checksum was calculated wrong
- Enhancements:
  - modules CONTACTOR, INTERLOCK and ISOGUARD can be disabled if not needed
  - selected new EEPROM M95M02 as default EEPROM (equipped on foxBMS-Master since v1.0.5)

Hardware:

- none

Documentation:

- added a section on how to build and include a library
- removed references to directory `foxbms-setup`, as it is now simply called `foxbms`
- removed references to script `bootstrap.py`, as this script is no longer used

---

**Release 1.5.0**

- **foxBMS has been migrated from Python 2.7 to Python 3.6. The foxconda3 installer is found at https://iisb-foxbms.iisb.fraunhofer.de/foxbms/. foxconda3 must be installed to C:foxconda3.**
- **EEPROM addresses on the BMS-Master were changed. Previous saved EEPROM data will be lost with new update.**
- **introduction of an improved software structure to differentiate between hardware-dependent and hardware-independent software layers**

Software:

- Toolchain:
  - switched to monolithic repository structure to simplify the versioning
  - raised compiler warning `[-Wimplicit-function-declaration]` to error level
  - avoid `shell=True` in python subprocess
  - updated python checksum script
  - updated to `waf-2.0.10` and renamed the waf binary to simply `waf`
- Bugfixes:
  - fixed bug passing *mV* instead of *V* to function `LTC_Convert_MuxVoltages_to_Temperatures()`
  - typedef `DATA_BLOCK_ID_TYPE_e` starts at 0x00 instead of 0x01 (renamed DATA_BLOCK_1 to DATA_BLOCK_00) for consistency
  - fixed bug in ltc module: wrote wrong values to database when using filtered mode for measuring cell voltages and temperatures
  - #define `CONT_PRECHARGE_VOLTAGE_THRESHOLD` used *V* instead of *mV*
  - fixed bug in function `CAN_WakeUp()`: wrong HAL function call was corrected
  - fixed bug in diag module: did not evaluated diagnostic errors with `DIAG_ERROR_SENSITIVITY_HIGH`
- Enhancements:
  - adapted wscripts to new restructured software architecture

  - added timestamp to MCU backup SRAM and external EEPROM entries
  - added three alarm levels (maximum operating limit, recommended safety limit, maximum safety limit)
  - enhanced voltage based balancing algorithm
  - updated .dbc file
  - added measure AllGPIO state to ltc module
  - added CAN message for pack voltage (CAN-ID: 0x1F0)
  - added algorithm module to enable future advanced algorithms
  - increased FreeRTOS heap size from 15kByte to 20kByte
  - increased stack size of 100ms application task from 512bytes to 1024bytes
  - increased size of CAN TX message buffer from 16 to 24 messages
  - added calculation of moving average values (1s, 5s, 10s, 30s and 60s) for current and power
  - database timestamp are now automatically written on DB_Write - no need to manually update timestamps anymore
  - added native matlab datatypes support
  - cleanup of ASCII conversion functions (uint to ASCII, hex to ASCII, int to ASCII)
  - added nvramhandler to automatically update non-volatile memory (i.e., external EEPROM on BMS-Master)
  - renamed various structs, variables and functions for an improved code understanding and increased readability

Hardware:

- added hardware changelogs
- ported hardware PCB design files to Altium Designer format (AutoDesk Eagle files no longer supported)
- updated hardware PCB designs: Master V1.0.5, Extension V1.0.4, Interface 1.9.3, Slave 12-cell (LTC6811-1) V2.1.5, Slave 18-cell (LTC6813-1) V1.1.3

Documentation:

- added foxbms styleguide
- fixed spelling errors
- added documentation of software architecture
- added documentation of algorithm module
- added documentation of nvramhandler
- updated isoguard documentation

---

**Release 1.1.0**

foxbms-setup(v1.0.1):

- updated build scripts
- updated waf script
- updated README.md

mcu-common(v1.1.0):

- updated license header
- seperated database entries to prevent concurrent read/write requests to the database
- updated wscripts to build specific files only for primary/secondary
- moved sdram from common repository to primary repository
- renamed database functions to `DB_WriteBlock()` and `DB_ReadBlock()`
- There was a compile error when CAN0 and CAN1 are deactivated
- updated README.md

mcu-freertos(v1.1.0):

- updated license header
- updated wscripts to build specific files only for primary/secondary
- moved sdram from common repository to primary repository
- updated README.md

mcu-hal(v1.0.1):

- updated license header
- updated README.md

mcu-primary(v1.1.0):

- uses now wafs feature of variant builds
- baudrate of CAN0 and CAN1 can now be set independently
- the setup of the tasks in engine and application layer is now consistent
- updated license header
- fixed a bug in contactor module to write unnecessary often into the database which caused a high cpuload
- seperated database entries to prevent concurrent read/write requests to the database
- added support of external SDRAM using keyword `MEM_EXT_SDRAM`
- moved sdram from common repository to primary repository
- fixed a bug that closed the interlock for a short period of time after restart even if no CAN message was received to switch to STANDBY state
- renamed database functions to `DB_WriteBlock()` and `DB_ReadBlock()`
- updated README.md

mcu-secondary(v1.1.0):

- uses now wafs feature of variant builds
- the setup of the tasks in engine and application layer is now consistent
- updated license header
- seperated database entries to prevent concurrent read/write requests to the database
- renamed database functions to `DB_WriteBlock()` and `DB_ReadBlock()`
- deleted unused code

- updated README.md

tools(v1.0.2):

- Updated waf
- Updated copyright
- Updated the Eclipse Project
- Updated checksum tool from gdb-based to object-copy-based toolchain
- updated README.md

documentation(v1.0.2):

- updated documentation for the build process
- updated FAQ section
- updated copyright
- updated README.md

---

**Release 1.0.1**

- updated build scripts
- updated waf script

---

**Release 1.0.0**

- renamed repository from `foxBMS-setup` to `foxbms-setup`.
- Removed update functionallity
- Moved arm-none-eabi-size call as post function in build process
- added a `.config.yaml` file which includes a list of repositories which are boostrapped and their bootstrap location.

---

**Release 0.5.2**

Release notes: We fixed a bug in the ltc driver, leading to a non-functional temperature sensing for foxBMS Slave Hardware version 1.xx. The slave version is configuration for the primary MCU in foxBMS-primarysrcmoduleconfigltc_cfg.h by the define SLAVE_BOARD_VERSION and for the secondary MCU in foxBMS-secondarysrcmoduleconfigltc_cfg.h by the define SLAVE_BOARD_VERSION.

- Set SLAVE_BOARD_VERSION to "1" if you are using version 1.xx of the foxBMS Slave.
- Set SLAVE_BOARD_VERSION to "2" if you are using version 2.xx of the foxBMS Slave. Version 2.xx is the default configuration.

Changelog:

- foxBMS-primary
  - fixed LTC temperature sensing bug
- foxBMS-secondary
  - fixed LTC temperature sensing bug

**Release 0.5.1**

- foxBMS-setup
  - added parameter '-u', '–update' to bootstrap.py for updating the setup repository.
- foxBMS-primary
  - updates for waf 1.9.13 support
  - updated module/EEPROM and migrated to module/nvmram
  - minor code adaptations and cleanup
- foxBMS-secondary
  - support for waf 1.9.13
  - minor code adaptations and cleanup
- foxbMS-tools
  - updated waf from version 1.8.12 to version 1.9.13

---

**Release 0.5.0**

A new project structure is now used by foxBMS. The documentation is no more contained in the embedded software sources and has its own repository. FreeRTOS and hal have their own repository, too. A central repository called foxBMS-setup is now used. It contains several scripts:

- bootstrap.py gets all the repositories needed to work with foxBMS
- build.py is used to compile binaries and to generate the documentation
- clean.py is used to removed the generated binaries and documentation

Release notes:

- New project structure
- Added support for external (SPI) EEPROM on the BMS-Master
- Redesign of can and cansignal module to simplify the usage
- Added support for triggered and cyclic current measurement of Isabellenhütte current sensor (IVT)
- Current sensor now functions by default in non-triggered modus (no reprogramming needed for the sensor)
- Updated and restructured complete documentation
- Restructured file and folder structure for the documentation
- Added safety and risk analysis section
- Cleaning up of non-used files in the documentation
- Consistency check and correction of the naming and wording used
- Addition of the source files (e.g., Microsoft Visio diagrams) used to generate the figures in the documentation
- Reformatted the licenses text formatting (no changes in the licenses content)
- Updated the battery junction box (BJB) section with up-to-date components and parameters

---

**Release 0.4.4**

The checksum tool is now automatically called when building binaries. Therefore the command `python tools/waf-1.8.12 configure build chksum` is NOT longer supported. The command to build binaries with checksum support is `python tools/waf-1.8.12 configure build` This is the build command used in foxBMS FrontDesk, that is, FrontDesk software is compatible with this change and now supports automatic checksum builds.

Release notes:

- Improved checksum-feature
- Updated copyright 2010 - 2017

---

**Release 0.4.3**

Starting from this version, a checksum mechanism was implemented in foxBMS. If the checksum is active and it is not computed correctly, it will prevent the flashed program from running. Details on deactivating the checksum can be found in the Software FAQ, in How to use and deactivate the checksum.

Release notes:

- Important: Changed contactor configuration order in the software to match the labels on the front
  - Contactor 0: CONT_PLUS_MAIN
  - Contactor 1: CONT_PLUS_PRECHARGE
  - Contactor 2: CONT_MINUS_MAIN
- Fixed an bug which could cause an unintended closing of the contactors after recovering from error mode
- Increased stack size for the engine tasks to avoid stack overflow in some special conditions
- Added a note in the documentation to indicate the necessity to send a periodic CAN message to the BMS
- Fixed DLC of CAN message for the current sensor measurement
- Added checksum verification for the flashed binaries
- Updated linker script to allow integration of the checksum tool
- Activated debug without JTAG interface via USB

---

**Release 0.4.2**

Release notes:

- Removed schematic files from documentation, registration needed to obtain the files
- Added entries to the software FAQ

---

**Release 0.4.1**

Release notes:

- Corrected daisy chain connector pinout in quickstart guide
- Corrected code for contactors, to allow using contactors without feedback

- Corrected LTC code for reading balancing feedback
- Quickstart restructured, with mention of the necessity to generate the HTML documentation

---

**Release 0.4.0**

Beta version of foxBS that was supplied to selected partners for evaluation.

Release notes:

---

**foxBMS Hardware Change Log (deprecated)**

The hardware changelog is now included in the regular changelog (since version 1.5.0).

*foxBMS Master*

| | |
|---|---|
| V1.0.6 | adapted CAN filter circuit for improved fault tolerance at short of CAN_L to GND or CAN_H to supply |
| V1.0.5 | schematic cleanup, improved fonts and sizes on PCB |
| V1.0.4 | introduced minor improvements to design<br>replaced EEPROM with M95M02-DRMN6TP |
| V1.0.3 | ported schematics and layout to Altium Designer<br>created hierarchical design<br>introduced minor improvements to design |
| V1.0.2 | replaced ADuM14XX isolators by ADuM34XX |
| V1.0.1 | added fuse protection on power supply input |
| V1.0.0 | initial release |

*foxBMS Extension*

| | |
|---|---|
| V1.0.5 | adapted CAN filter circuit for improved fault tolerance at short of CAN_L to GND or CAN_H to supply |
| V1.0.4 | schematic cleanup, improved fonts and sizes on PCB |
| V1.0.3 | ported schematics and layout to Altium Designer<br>created hierarchical design<br>introduced minor improvements to design |
| V1.0.2 | replaced ADuM14XX isolators by ADuM34XX |
| V1.0.1 | swapped input protection of isolated GPIOs |
| V1.0.0 | initial release |

*foxBMS Interface*

| | |
|---|---|
| V1.9.4 | replaced connectors J500 and J501 with TE 534206-4 due to clearance issues in component placement |
| V1.9.3 | replace NAND-gate with SN74LVC00AQPWRQ1 |
| V1.9.2 | replace OR-gate with NAND-gate and add direction pin |
| V1.9.1 | rotate pinout of Daisy-Chain-Connectors in order to mirror Slave-Connectors<br>add labels to Daisy-Chain-Connectors<br>update with new layermarker<br>replace OR-gate with AEC-Q100 qualified COTS |
| V1.9.0 | update design for reverse isoSPI with second channel<br>port to Altium Designer |
| V1.1.0 | replaced isoSPI transformer HX1188 by HM2102 |
| V1.0.2 | modified connection of isoSPI transformer HX1188 |
| V1.0.1 | added fiducials |
| V1.0.0 | initial release |

*foxBMS Slave 12-cell (LTC6811-1)*

| | |
|---|---|
| V2.1.7 | modified component designators to be compatible with 18-cell versions |
| V2.1.6 | EMI improvements (layout)<br>added RC filters on NTC sensor inputs<br>added DC/DC converters for 5V LTC supplies |
| V2.1.5 | Replaced Opamps, Port Expanders and Optocouplers with AEC-Q100 compliant ones<br>Modified silkscreen texts |
| V2.1.4 | Primary software timer is now switched on by default<br>added layermarker on PCB |
| V2.1.3 | replaced EOL port expander with PCF8574 |
| V2.1.2 | ported schematics and layout to Altium Designer<br>created hierarchical design<br>introduced minor improvements to design |
| V2.1.1 | improved isolation distances between external DC/DC converter supply and battery module signals |
| V2.1.0 | added DC/DC converter for external power supply |
| V2.0.3 | fixed isoSPI transformer CMC issue |
| V2.0.2 | replaced LTC1380 MUXs with ADG728 (400 kHz I2C)<br>adjusted connection of 100 ohm resistors for V+/V_REG supply<br>reduced value of I2C pullups to 1k2 |
| V2.0.1 | added missing cooling areas on bottom side, adjusted silk screen<br>enlarged PCB tracks, R201/202/301/302 other package<br>enlarged T201/301 cooling area<br>Replaced PCF8574 with PCA8574 (400 kHz I2C) |
| V2.0.0 | initial release |

*foxBMS Slave 18-cell (LTC6813-1)*

| | |
|---|---|
| V1.1.5 | EMI layout improvements<br>adapted component variants to other changes |
| V1.1.4 | replaced DC/DC converter power inductor with AEC-Q compliant one<br>added circuit for switching off DC/DC converters in LTC sleep mode<br>added pull-ups on all GPIOs of the LTCs |
| V1.1.3 | schematic cleanup, improved fonts and sizes on PCB |
| V1.1.2 | replaced ACPL-247 with ACPL-217 optocoupler in order to be able to use automotive components |
| V1.1.1 | replaced port expander with TCA6408APWR (automotive)<br>replaced analog buffer opamp with AD8628ARTZ-R2 (automotive)<br>replaced DC/DC buck controller with LM5161QPWPRQ1 (automotive) |
| V1.1.0 | ported schematics and layout to Altium Designer<br>created hierarchical design<br>introduced minor improvements to design<br>replaced linear regulation (PNP transistor) for LTC power supply with DC/DC converters<br>improved isolation distances between external DC/DC converter supply and battery module signals<br>added 8-24 V isolated external power supply<br>replaced I2C EEPROM 24AA02UID with M24M02-DR (ECC)<br>replaced isoSPI transformers HX1188NL with HM2102NL<br>reduced balancing resistors from 2x 68 Ohm to 2x 130 Ohm due to shrinked cooling areas<br>added layermarker on PCB<br>Primary discharge timer is now switched on by default |
| V1.0.1 | replaced all LTC1380 MUXs with ADG728 MUXs |
| V1.0.0 | initial release |

CHAPTER 3

# Overview

Hereafter a general overview of the content of foxBMS can be found.

The foxBMS Master Unit consists of 3 boards: BMS-Master Board, BMS-Interface Board, BMS-Extension Board. Two ARM-based microcontroller units (Cortex-M4) are used on BMS-Master Board: MCU0 (also called primary MCU) and MCU1 (also called secondary MCU). The BMS software runs on MCU0, while MCU1 is used for redundant safety.

MCU0 communicates with the outside world via a CAN bus (CAN0). The current flowing through the battery system is measured via a current sensor connected via the CAN bus. The sensor is controlled via CAN by MCU0 and sends the resulting measurement via CAN.

The foxBMS Slave Unit (BMS-Slave Board) are used to measure cell voltages and cell temperatures in the battery modules. The foxBMS Slave Units are linked via a proprietary daisy chain from the company Linear Technology (i.e., isoSPI). , isoSPI,

In order for the foxBMS Master Unit to communicate with the foxBMS Slave Units, an interface board (i.e., BMS-Interface Board) is needed. It converts the SPI signals from the BMS-Master Board into differential signals used by the daisy chain and vice versa.

Three power contactors are used to connect and disconnect the battery modules (or pack) from the load:

- Main Contactor Plus
- Main Contactor Minus
- Pre-charge contactor

These contactors are driven by MCU0. Requests are made via CAN to the system to open and close the contactors. Based on the measurements and the algorithms implemented in the software, MCU0 decides if the contactors should be closed or opened. It sends information via CAN so that the user knows the state of the system.

An interlock line is also present. If it is opened, either by MCU0, MCU1 or somewhere else (e.g., emergency stop), all contactors will immediately open. This interlock cannot be used in the aviation applications.

A secondary ARM-based microcontroller is present on the master, called MCU1 (or secondary MCU). It monitors the slaves via a second and separate daisy chain. Like the MCU1, it can open the interlock in case something goes wrong with the system. 0

In case more inputs and outputs and further functions are required, an BMS-Interface Extension Board is available and can be easily adapted to specific needs.

This description reflects the current state of foxBMS. Due to the open nature of the system, many other possibilities can be implemented, like for example:

- Use of other types of current sensors (e.g., shunt-based or Hall-effect based)
- No foxBMS Slave Unit needs to be used: a direct measurement of the cell voltages and cell temperatures can be performed by the foxBMS Master Unit
- A higher number of contactors can be controlled (e.g., up to 9)
- etc. . .

CHAPTER 4

# Motivation

## 4.1 What is foxBMS?

With foxBMS, Fraunhofer IISB delivers the first generation of its open source battery management system (BMS) research and development platform. The foxBMS platform is completely free and open, designed for a maximum of flexibility, and comprehensively documented. It includes all necessary hardware and software for potentially any kind of mobile and stationary application that uses modern rechargeable electrochemical energy storage systems (e.g., lithium-ion batteries, redox-flow batteries, supercapacitors):

- foxBMS Hardware: the schematics and the layout of all electronic boards are available for download and the design is based on commonly available components and devices, that do not require NDAs or confidentiality agreements.
- foxBMS Software: the software toolchain uses only own developed open source and free of charge components or free of charge third-party software, and the entire BMS source code is provided online with its own development environment and configuration files, thus enabling immediate use on Windows, Mac, and Linux operating systems.

## 4.2 What is foxBMS designed for?

Experience gained from international research and development projects over the last 15 years in the field of electrochemical energy storage systems at Fraunhofer IISB has been implemented in the hardware and software of the foxBMS platform. It is designed to manage high-performance prototypes of advanced and innovative lithium-ion battery systems of any size (i.e., from a few cells up to several hundreds of kWh and kW), especially for systems requiring the highest availability and safety levels.

The free and open source version of foxBMS is not intended for immediate use in commercial products as they have to meet specific standards and require application-dependent certifications. In fact, foxBMS is a safe research, development, and test platform providing all functions for managing the complexity and size of state-of-the-art electrochemical energy storage systems.

Specific adaptions of foxBMS can be ordered directly from Fraunhofer IISB or can be jointly developed, for example for automotive, aviation, space, submarine, industrial, and renewable energy storage applications.

## 4.3 Who is the intended audience?

foxBMS is a free and open BMS platform that can be used for developing and testing products. The foxBMS hardware and documentation are licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. The foxBMS software is licensed under the BSD 3-Clause license. This means, foxBMS parts can be utilized unrestrictedly, including commercial use. The foxBMS platform addresses especially:

- R&D and test engineers requiring a smart and powerful, well documented BMS platform
- Engineering companies requiring a maintained and supported BMS for their developments
- Small enterprises requiring a flexible and future proof BMS for developing their products
- Large enterprises requiring a reliable and safe BMS for testing their prototypes
- Research organizations requiring a simple and universal BMS development platform
- Students looking for a free and open BMS software development toolchain

# CHAPTER 5

# Safety

This document describes the hazard risk analysis and assessment done for the foxBMS platform. The content of this document must be read and understood before any tests are done with the foxBMS hardware. All physical quantities and units used in the document are SI based.

## 5.1 Hazard Risk Analysis and Assessment

The risk analysis and assessment was last performed in May 2017. Following persons were involved:

| Name | Responsibility |
|---|---|
| Vincent Lorentz | Global project Management |
| Stéphane Koffel | Technical Project Management |
| Martin Wenger | Hardware Development (BMS-Extension) |
| Radu Schwarz | Hardware Development (BMS-Master) |
| Sebastian Wacker | Hardware Development (BMS-Slave) |
| Stefan Waldhör | Software Development |
| Müsfik Akdere | Software Development |
| Johannes Wachtler | Validation and Tests |
| Markus Freund | Moderation |

## 5.2 General Description of the foxBMS Platform

foxBMS is a research and development platform aimed to be used to develop battery management systems (BMS) for rechargeable energy storage systems based on lithium-ion batteries (LIB) or comparable electrochemical rechargeable accumulator cells (e.g., other chemistries like lithium-sulfur, sodium-ion or even all-solid-state batteries), lithium-ion capacitors (LIC), electric double-layer capacitors (EDLC or supercapacitors). The lithium-ion battery packs or battery modules are a major source of hazards and are not part of the foxBMS platform (see fig. 5.1). The main purposes of the battery management system are charge monitoring and keeping the battery cells in their safe operating area to ensure optimal safety and the longest battery lifetime.

For those tasks, the foxBMS platform is equipped with measurement and control circuits (i.e., BMS-Slave Boards) to monitor the battery cells and to get information about their states. With this, the cell voltages, temperatures and overall battery pack current are measured, and kept in the safe operating area. The battery charge equalization (i.e., balancing) is controlled and fed back to the foxBMS Master Unit. The setup of the system is shown in fig. 5.2.

The foxBMS platform uses a passive balancing circuit based on the LTC6811-1 multicell battery monitoring integrated circuit. The unique selling point of the foxBMS platform is its virtually unlimited possibilities to be adapted the needs of the user.

## 5.3 Limits of the foxBMS Platform

The foxBMS platform consists only of the foxBMS Master Unit with the foxBMS Slave Units and the provided computer software. It does not include battery cells, battery cell models, power contactors, current sensors, fuses, chargers, power supplies, ground fault detector.

The foxBMS Master Unit and the foxBMS Slave Units have a primary and a secondary side which are separated and galvanically isolated from each other.

The area of the use of foxBMS must be free of hazardous materials like flammable gases, combustible dust or ignitable dust and fibers. The temperature should be around 20°C and may not exceed the battery cells restrictions given by the battery cell manufacturer (e.g., around 45°C to 60°C, depending on the battery cell chemistry). The operating and storage environment needs to be dry and free of static electricity with air humidity between 35% and 55%.

The lifetime of the electronics of foxBMS is supposed to be limited to three years since the target application is research and development. During this time, hardware and software updates and bug fixes are to be expected. Besides that, the research or work on the prototypes should be finished after three years.

The interfaces of the foxBMS Master Unit are:

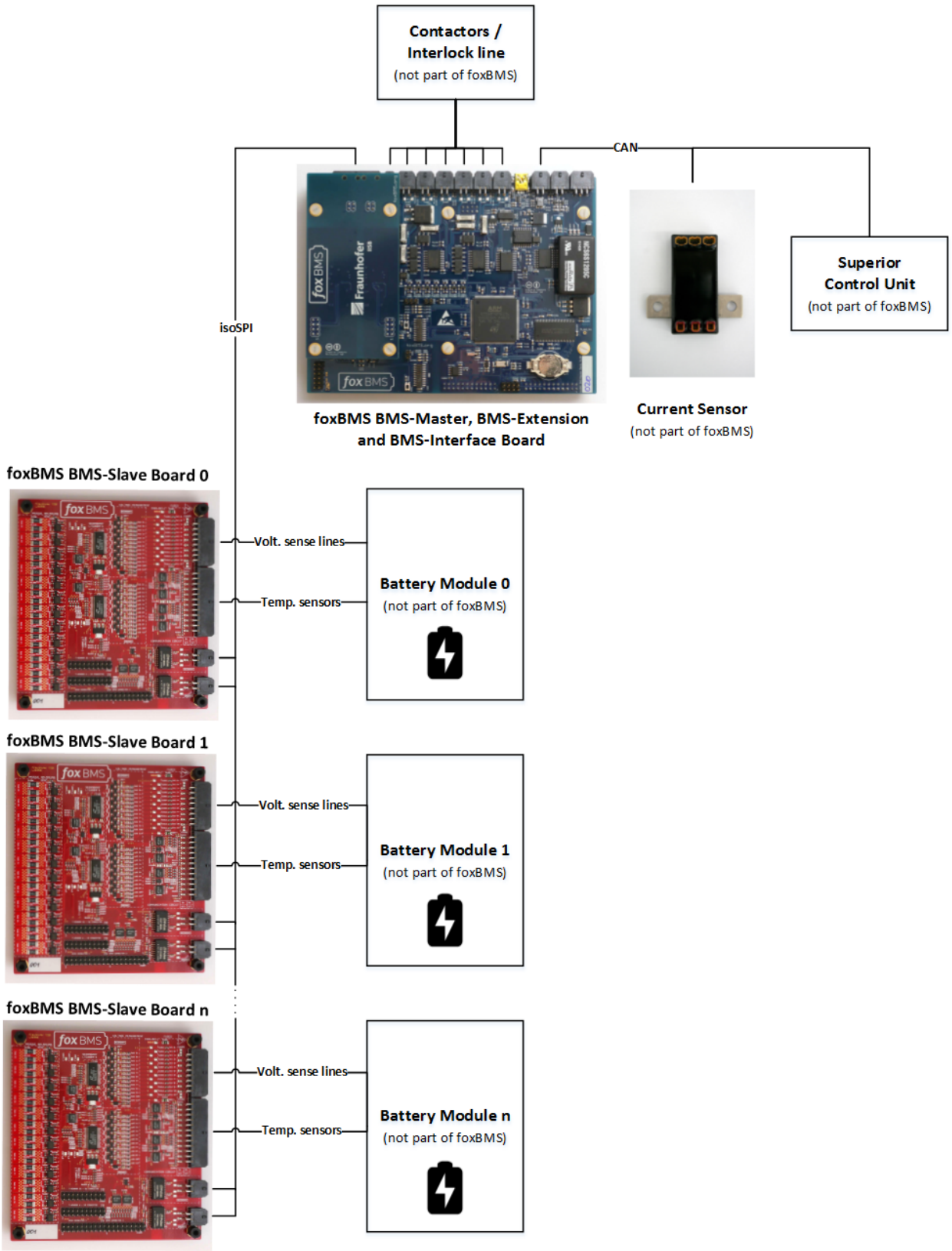



Fig. 5.1: foxBMS in the battery system

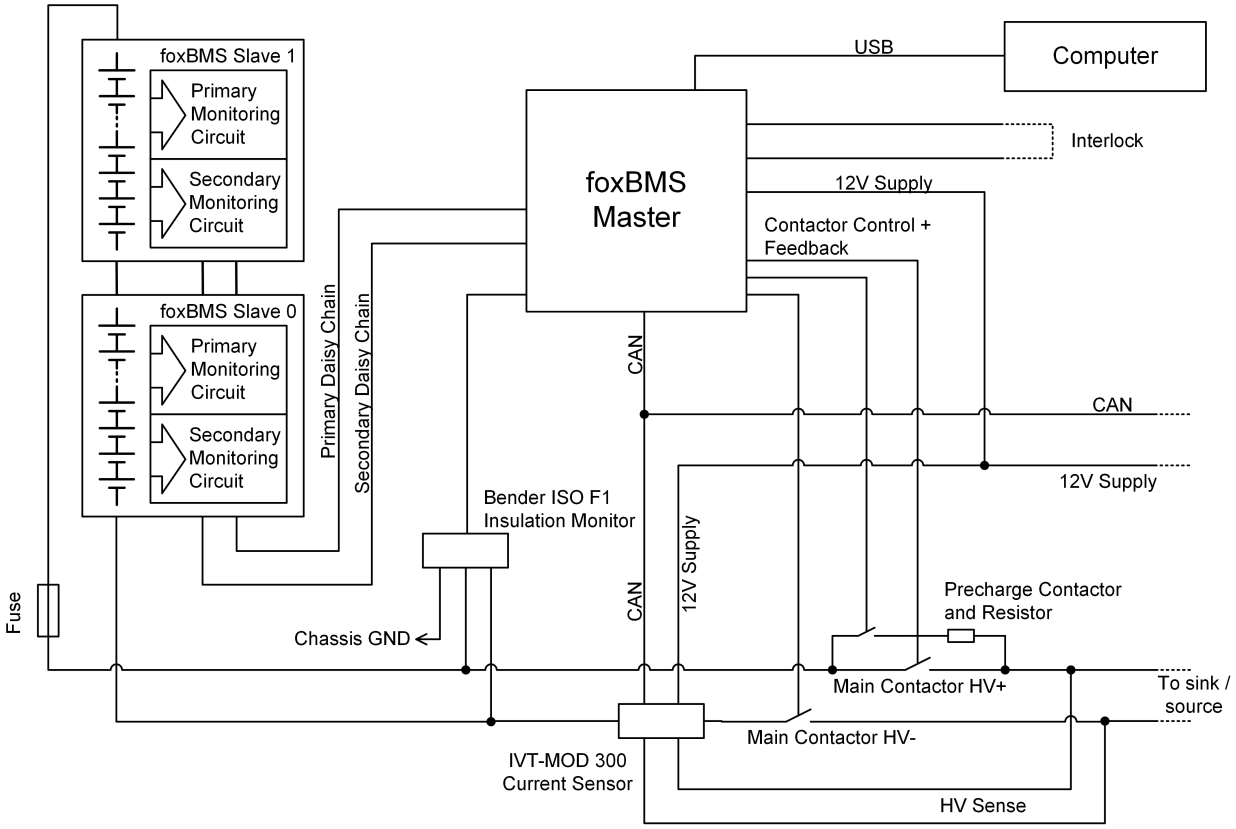



Fig. 5.2: Block diagram showing the typical topology of a battery system

| Connection | Description |
|---|---|
| Supply Voltage DC | 12VDC-24VDC supply input |
| Ground Fault Detector | Connection for the insulation monitor to detect an error (Bender ISOMETER IR155-3203) |
| CAN0 | Galvanically isolated connection on foxBMS Master Unit for additional sensors (IVT-MOD-300) |
| Primary USB | Galvanically isolated USB connection to microcontroller on the primary side for flashing and communication |
| Contactors 0-8 | Power contactors between battery and the supplied load |
| Interlock | Galvanically isolated connection between the microcontrollers and connector control |
| Daisy Chain (Primary/Secondary) | Connection for the next BMS-Slave Board in the battery system |
| Secondary USB | Galvanically isolated USB connection to microcontroller on the secondary side for flashing and communication |
| CAN1 | Galvanically isolated connection on BMS-Extension Board |
| RS485 | Galvanically isolated RS485 interface as alternative for CAN or USB |
| Isolated GPIO | Galvanically isolated general purpose IO for user specific needs |
| Isolated NOC | Galvanically isolated Normally Open Contact interface for any purpose left to the user |
| Analog Inputs | Analog inputs for any purpose left to the user |
| Memory Card | Data storage for the primary microcontroller |

The interfaces of the foxBMS Slave Unit are:

| Connection | Description |
|---|---|
| 12 Cells | 13 voltage sense connections for 12 cells |
| 8+16 Temperatures | Temperature sensor connection (one for each cell) |
| 4 Daisy Chains | Primary and secondary daisy chain connections to the next and to the previous BMS-Slave Board |

The maximum amount of BMS-Slave units depends on the number of battery cells connected to the slaves and the voltage of each cell. The total voltage of the battery pack must never exceed 1500Vdc (continuous and peak). Battery types to be used with the system are lithium-ion batteries or comparable electrochemical rechargeable accumulator cells, lithium-ion capacitors or supercapacitors.

## 5.4 Targetted Users of foxBMS

The persons who use the foxBMS R&D platform must be electrically skilled according to the IEC. They must also be educated in rechargeable batteries and have knowledge in electrochemistry. With the sum of battery cells exceeding 120VDC, these persons also need to be skilled in live working. It is not enough to be an electrically instructed person. Targeted users of the foxBMS R&D platform are R&D and test engineers working in a well-defined, safe and controlled environment, like a test bay or a test bench. Since these persons assemble the whole system by themselves, they are the only persons who can do the service and the maintenance. The R&D and test engineers are both users and servicers.

The foxBMS R&D platform is not intended to be used by ordinary persons as they are not well and not enough educated in batteries, or in electrical subjects. This also includes students, handymen, and do-it-yourselfer or DYI enthusiasts who do not fulfil the above requirements.

## 5.5 Intended Use of foxBMS

The foxBMS R&D platform is only for lithium-ion batteries or comparable energy storage systems. It is an evaluation kit intended only to professionals and to be used at research and development facilities for such purposes. It is designed for research, development and tests purposes to manage prototypes of lithium-ion-battery systems when developing new products.

## 5.6 Reasonably Foreseeable Misuse of foxBMS

The foxBMS R&D platform is not intended to be used as a voltage monitoring system of any other electrical systems than battery systems build with battery cells providing a maximum single cell voltage of 5V.

## 5.7 Phases Throughout the System Lifecycle of foxBMS

1. Development
2. Manufacturing
3. Packaging
4. Transportation
5. Assembling
6. Initial operation
7. Usage (operation)
8. Maintenance
9. Repair
10. Further operation
11. Shutting down
12. Storage
13. Disassembling
14. Disposal

## 5.8 Hazard Zones of foxBMS

Only few hazard zones are obvious. One is the foxBMS and the other its environment with the battery system including bus bars, battery cells and other parts. Since the battery system is defined by the user, only a general consideration can be done.

## 5.9 Risk Assessment

Depending on the countries of the target application, there are different standards to regard. In the European Union, the applicable directives lead to different safety requirements. These have a variety of risk graphs for the risk assessment.

Although they are only slightly different they are not adaptable to each other. Since the target country and use of foxBMS is unknown, the users of the system need to do a risk assessment on their own according to their concerns.

Some European directives that can be relevant for the foxBMS users:

- General product safety
- Low voltage
- Machinery
- etc…

Some example of safety standards that might fit the application of the foxBMS users:

- IEC 61508 (Functional Safety)
- ISO 25119 (Agriculture)
- ISO 26262 (Road Vehicles)
- EN 13849 (Machinery)
- EN 61511 (Process Industry)
- EN 50156 (Furnaces)
- etc…

A list of standards that might fit to the target application can be found here:

- http://www.mpoweruk.com/standards.htm

## 5.10 Standards

In order to use the foxBMS safely, at least the following standards or similar ones should be regarded:

- DIN VDE 0100-410 (IEC 60364-4-41)
- DIN VDE 0100-600 (IEC 60361-6)
- EN 60529
- EN 50272
- etc…

Recommended readings:

- http://www.mpoweruk.com
- http://batteryuniversity.com

## 5.11 Safety Instructions Before Using foxBMS

**Danger:**

Risk of electric shock. You need to be an electrically skilled person in order to work with batteries and assemble battery systems. Regard IEC 60364-4-41, IEC 60364-6, IEC 60529 (DIN VDE 0100-410, DIN VDE 0100-600, VDE 0470-1).

**Danger:**

Risk of electric shock while assembling, repairing, maintaining, servicing or disassembling the battery system. You need skills in live working in order to work with the battery system and assemble it. Wear personal insulating protective equipment.

**Danger:**

Risk of fire, explosion and chemical hazards through the battery cells. You need to be a battery skilled person in order to work with the battery system and assemble it. Use safe cells with CID, PTC and OPD. Use the battery system in a confined area. Keep sand, water and fire extinguisher close to the system to fight fire. Regard local fire safety regulations.

**Danger:**

With an electrical short, battery cells will heat up and explode. The short might create an electric arc and cause fire. Do not short-circuit battery cells or batteries. Always cover at least one terminal of the cell or battery and keep the poles away from each other. Watch out while working on the cells or batteries. Do not wear necklaces or jewelry to prevent shorts. Wear personal arc protective equipment (e.g., protection clothes, face protection, protection glasses, protection gloves). Keep other persons in a safe distance.

**Danger:**

Battery cells and batteries expand and shrink through thermal differences and through charging and discharging during usage. If the battery cells are too tight and squeezed together, they can get damaged. The movement of the cells will loosen screws that might fall down and create an electrical shortage. A fire might start or other hazards can occur. Leave enough room between the cells and use locknuts.

**Danger:**

Electric short and hazards through reversed polarity or wrong connection can occur. Be cautious and prevent battery cells and batteries from wrong connection.

**Danger:**

Electrolyte may cause irritation or intoxication and lead to death or threaten your health. Hydrofluoric acid (HF) or phosphane ($PH_3$) might develop. Wear eye protection and gloves while working with electrolyte. Regard material safety datasheet (MSDS) from the battery cell manufacturer.

**Danger:**

Old and new battery cells, different technologies or capacities of cells may vary very much in both voltage and current. This can lead to an overcurrent or overvoltage same as undervoltage and result in cell or battery damage with dramatic consequences. Do not use old cells or batteries with the system and do not mix cells or batteries of different chemistries or technologies. Change all cells of a battery at the same time. Do not mix cells within the battery. Use only one type of cell throughout the whole system.

**Warning:**

Developing gas from the battery cells or battery systems may cause fire. Use the battery cells and the battery system only in a good ventilated environment to ensure flammable and toxic gases will be removed in case of degassing of a battery cell.

**Warning:**

Overvoltage or reverse polarity at the foxBMS Slave Units can cause an electrical short, fire and following hazards. Use fuses on each cell to prevent overcurrent through the electrical short. The foxBMS Slave Units can take a maximum amount of 12 cells with a voltage sum between 11V and 55V. Beware of the amount of cells and the cell voltage.

**Warning:**

Chemicals from the battery might threaten your health. Do not touch chemicals and wear chemical protective gloves and safety goggles. Regard material safety datasheet (MSDS) from the battery cell manufacturer.

**Warning:**

The soldering heat can damage safety parts of the cells or battery. Do not solder anything directly to the cells or battery. Follow the mounting instructions of the manufacturer.

**Warning:**

Damaged batteries or cells can cause fire. Use flame retardant materials in your system and keep burnable materials away. Apply a temperature sensor to detect over temperature and keep a fire extinguisher close to the battery system. Do not use damaged cells or batteries.

**Warning:**

Parallel cells with an electrical short can cause a large over current and over temperature that bring other hazards with them. Assemble breaking elements between parallel cells to avoid the short current.

**Warning:**

Overvoltage or overcurrent through charging or discharging can lead to fire, leakage or explosion. Always use a proper charger for the cells and the batteries and avoid heavy loads and rapid charges and discharges. Remove fully charged battery packs from the charger.

**Warning:**

Risk of electric shock through an error in the insulation. Use a ground fault detector or an insulation monitor and regard IEC 60364-4-41 and IEC 60364-6 (DIN VDE 0100-410, DIN VDE 0100-600).

**Warning:**

Risk of electric shock while working on live parts. Use only insulated tools while assembling, disassembling, maintaining, servicing or dismantling the battery system. Never open battery cells.

**Warning:**

Explosive or flammable environment around the battery cells or the battery system can start burning or explode. Do not use the battery system in an explosive environment.

**Warning:**

Toxic fumes through evaporating electrolyte or other substances through an electrical short may lead to intoxication, chocking or breathing problems. Assemble and disassemble the system carefully and keep gas mask or breathing aid in close environment.

**Caution:**

Heavy battery parts can fall on your feet and sharp edges might cut or hurt you. Use solid housing and add handles. Wear personal protective equipment, gloves, shoes and other clothes for working.

**Caution:**

An electric arc or exploding cell may cause loud noise. Wear ear plugs muffs with other personal protective equipment. Stay out and keep other persons out of the testing area.

**Caution:**

Soldering material may cause irritation or intoxication and lead to death or threaten your health. Wear gloves while working with soldering materials and wash hands properly afterwards.

**Caution:**

Corrosive materials, humidity and gas can lead to corrosion of any part of the system. Failure of electrical or electronic parts especially safety responsible parts may lead to other hazards. Keep the battery system in a dry and clean environment and away from corrosive materials.

**Note:**

Damaged battery cells or batteries may leak electrolyte, especially when pouch battery cells are used and the pouch bag has been damaged. Put a basin underneath for dripping the electrolyte.

---

---

**Note:**

Undervoltage leads to a damaged cell or battery system. Prevent deep discharge. Never charge and use again battery cells or batteries after deep discharge has occurred.

---

---

**Note:**

Cell or battery damage through too low or too high temperature during storage, transportation or usage. Always keep cells or batteries in their valid temperature range. Keep the cells and batteries out of the sun and use heating or cooling to keep the safe temperature range.

---

---

**Note:**

Battery cells, battery systems, electrical and electronic parts must be stored and disposed properly.

## 5.12 Measures Through Control with foxBMS

- Detection of cells with over temperature and safe switch off or disconnect.
- Detection of cells with over voltage and safe shut off or disconnect. the charger
- Detection of cells with deep discharge and safe shut off or disconnect charger. the load
- Detection of ground short (insulation error) with safe switch off or disconnect.

The recommended structure for failure tolerant systems (1oo2) in EN 13849 is shown in fig. 5.3:

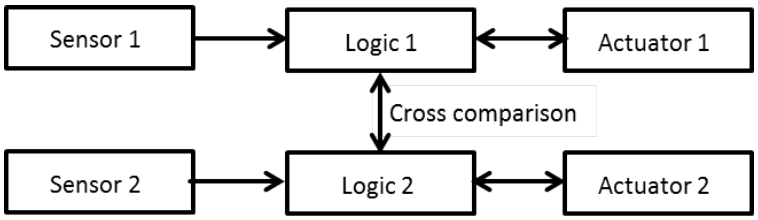



Fig. 5.3: Recommended structure for failure tolerant systems (1oo2) in EN 13849

With this structure, there are requirements to the reliability of the used hardware. Refer to the standard used. The implemented safety systems might have a common cause failure. Refer to the required standard to check how to minimize this failure.

CHAPTER 6

# Licenses

## 6.1 foxBMS Software License



The foxBMS embedded software and computer software are licensed under the BSD 3-Clause License.

**Important:** We kindly request you to use one or more of the following phrases to refer to foxBMS in your hardware, software, documentation or advertising materials:

```
This product uses parts of foxBMS ®
This product includes parts of foxBMS ®
This product is derived from foxBMS ®
```

## 6.2 foxBMS Hardware and Documentation License



The foxBMS hardware and documentation are licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) License. To view a copy of this license, visit CC BY 4.0.

**Important:** We kindly request you to use one or more of the following phrases to refer to foxBMS in your hardware, software, documentation or advertising materials:

```
This product uses parts of foxBMS ®
This product includes parts of foxBMS ®
This product is derived from foxBMS ®
```

CHAPTER 7

# Team

## 7.1 foxBMS Management Team Fraunhofer IISB

A team of 5 management personals:

| Firstname Lastname | Main Role |
|---|---|
| Vincent Lorentz | Project Initiator and Technical Marketing |
| Stéphane Koffel | Project Management and Coordination |
| Martin März | Technical Consulting |
| Jürgen Lorenz | Licenses and Legal Consulting |
| Lothar Frey | Technical Marketing and Legal Consulting |

## 7.2 foxBMS Development Team Fraunhofer IISB

A team of 14 engineers:

| Firstname Lastname | Main Role |
|---|---|
| Martin Wenger | Hardware Team Coordination, Hardware Development |
| Stéphane Koffel | Software Team Coordination, Algorithm Development |
| Müsfik Akdere | Technical Lead Embedded Software |
| Tim Fühner | Technical Lead Software, Algorithm Development |
| Martin Giegerich | Software Development |
| Stefan Waldhör | Software Development |
| Johannes Wachtler | Software Development |
| Christian Freund | Software Configuration Management |
| Radu Schwarz | Hardware Development |
| Sebastian Wacker | Hardware Development |
| Patrick Kanzler | Hardware Development |
| Joshua Grosch | Hardware Development |
| Markus Gepp | Mechanical Construction |
| Reinhold Waller | Validation and Tests |

## 7.3 foxBMS Student Team Fraunhofer IISB

A team of 5 students:

| Firstname Lastname | Main Role |
|---|---|
| Robin Heim | Hardware and Software Development |
| Christian Thomas | Software Development |
| Lucas Grunenberg | Software Development |
| Georg Kordowich | Software Development |
| Pauline Thierauf | Documentation |

For every kind of information, do not hesitate to contact us: info@foxbms.org

# CHAPTER 8

# Getting Started with foxBMS

## 8.1 How to use this documentation

The complete foxBMS documentation is available and maintained in HTML, PDF and EPUB formats.

**Note:** A search function is available under the foxBMS logo in the HTML version of the documentation.

## 8.2 How to Generate the HTML Documentation and Set up foxBMS rapidly

First, the *Installing foxconda3* chapter (Ch 9) part of the documentation explains how to install and configure the foxconda3 software used to compile the embedded software and generate the HTML documentation. Then, the *Building the foxBMS Software and Documentation* chapter (Ch 10) part explains how to get and build the sources. Finally, the *Eclipse Workspace Setup and Flashing* chapter (Ch 11) guide explains how to setup an Eclipse workspace to browse the foxBMS sources conveniently, how to supply the hardware with power and how to flash the foxBMS firmware on MCU0 and MCU1 located on the BMS-Master Board in the foxBMS Master Unit.

If these steps are done, foxBMS already runs. The last step is longer because connectors must be made. ~~In this last step,~~ it is explained how to prepare the other parts of the foxBMS hardware. is ready to run. In chapters 12, 13, and 14,

## 8.3 Where Can the Layout and Schematic Files be Found?

The layout and schematic are available online on GitHub. Refer to the ~~section~~ *Design Resources* chapter (Ch 19).

# CHAPTER 9

# Installing foxconda3

The foxBMS embedded software consists of a program written in C. Before it can be flashed and run on the microcontroller units (i.e., MCU0 and MCU1 located on the BMS-Master Board), it must be compiled to generate a binary file called firmware.

Different softwares (i.e., the software toolchain) are needed for this compilation step. They are based on a Python environment. All the needed software, including the Python environment, are contained in a Python distribution called foxconda3. It will be installed in the next steps.

This section shows how to set up the development environment, foxconda3 (the software toolchain).
Ch 11,
In the section *Eclipse Workspace Setup and Flashing*, it is shown how the compiled sources are flashed on the foxBMS Master Unit.

## 9.1 Installation of the Needed Software via the foxconda3 Distribution

The first step is to install the software environment needed to compile and flash the foxBMS sources. First, the foxconda3 installer must be downloaded from the server containing the foxconda3 installers. For Windows, the installer is run by executing `foxconda-3.0.3-Windows-x86_64.exe`. The installer version with the highest version and build number must always be used.

**Warning:**

- Setting the installation directory from `C:\foxconda3` to something else other will create a lot extra effort when setting up the Eclipse workspace. It is highly advised against changing the default installation directory.
- When installing foxconda3 to another directory, foxconda3 must not be installed into a directory containing whitespace (e.g. `C:\Program Files`)

The following figures from fig. 9.1 to fig. 9.8 will show the installation process.
Fig. 9.1 shows
1. The startpage of the installer. Click "Next".
2. The license terms of the installer. Click "I Agree".
Fig. 9.2 displays


Fig. 9.1: Installer start page

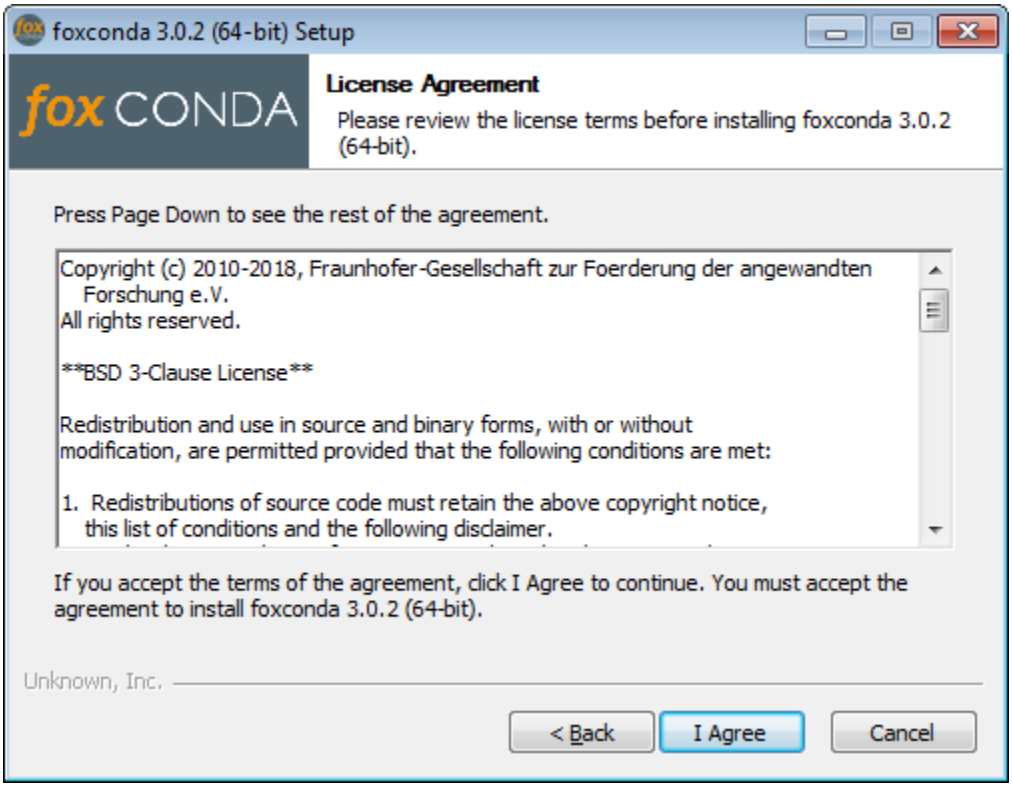



Fig. 9.2: Installer license terms

Fig. 9.3 provides the choice of

3. The installation type foxconda3 binaries. Select `Just Me (recommended)`.

of the

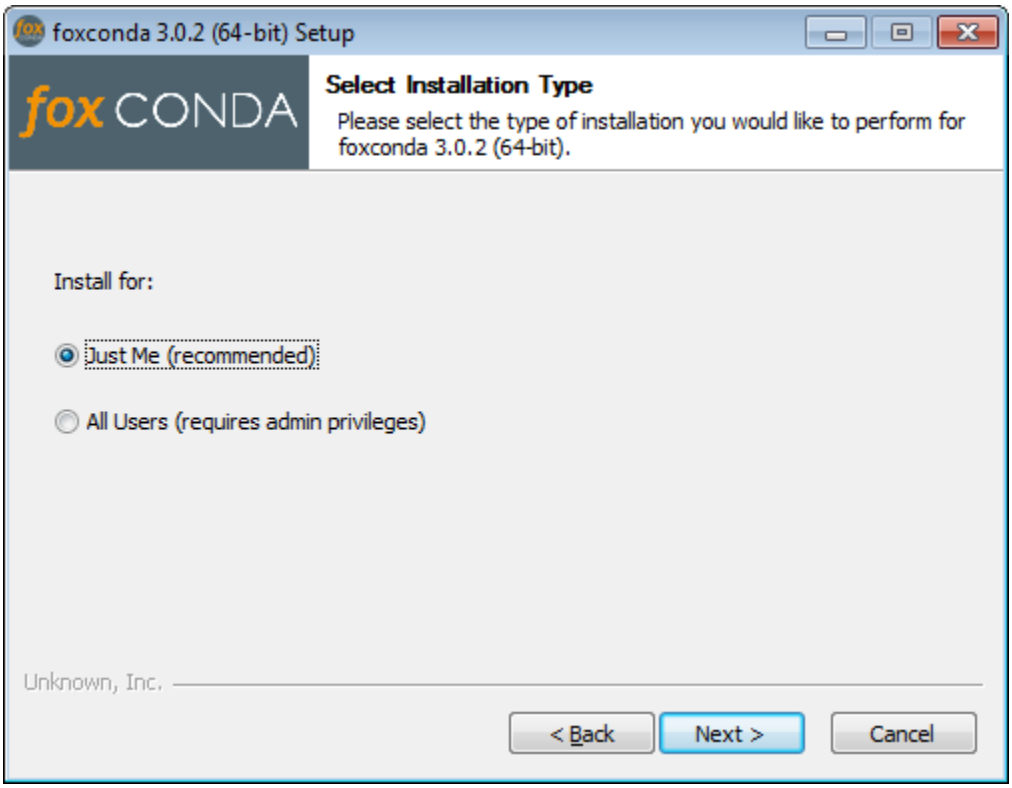



Fig. 9.3: Installation type of foxconda3

Fig. 9.4 provides the choice of

4. The installation path for foxconda3. Set the path to `C:\foxconda3`.

> **Warning:** Changing the default installation directory from `C:\foxconda3` to something else other will create a lot extra effort when setting up the Eclipse workspace. This is not advised.

Fig. 9.5 provides the choices for

5. Advanced options of the installation. **Unselect** `Register Anaconda as my default Python 3.6`

Click "Install" to proceed.

Fig. 9.6 shows

6. The installation process which tall take a few minutes.

Fig. 9.7 shows

7. The installation process is completed. Click `Next >`.

Fig. 9.8 confirms

8. The finish of the installation process. Click `Finish`.

To run

From the command line, the convenience terminal environment can be used by executing the `Anaconda Prompt`:

- Go the the start menu
- Type `Anaconda Prompt` or click the "Anaconda Prompt"
- Enter

It will open a CMD window with a `PATH` environment ready to work with foxBMS.

The prompt can also be started manually by opening a CMD window followed by following command:

```
C:\foxconda3\Scripts\activate
```

Note that if you have multiple versions of conda installed, you can use "conda info --envs"
to see which version is being used. Make sure to use the "C:\foxconda3" version.

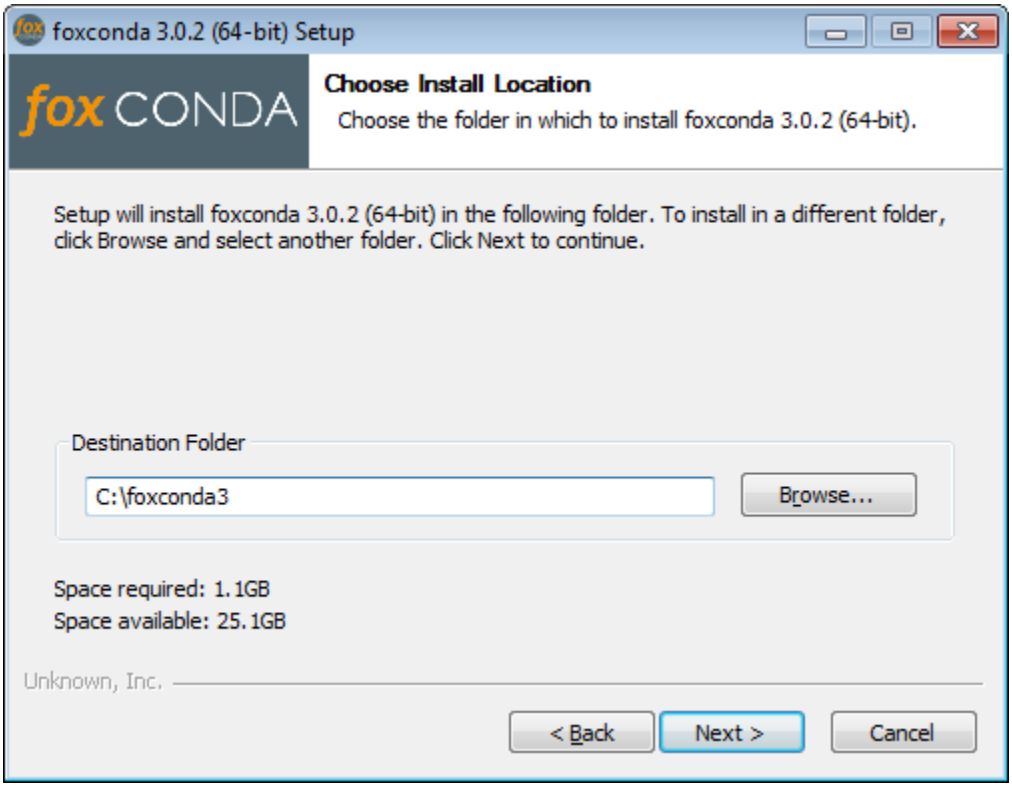



Fig. 9.4: Installation directory

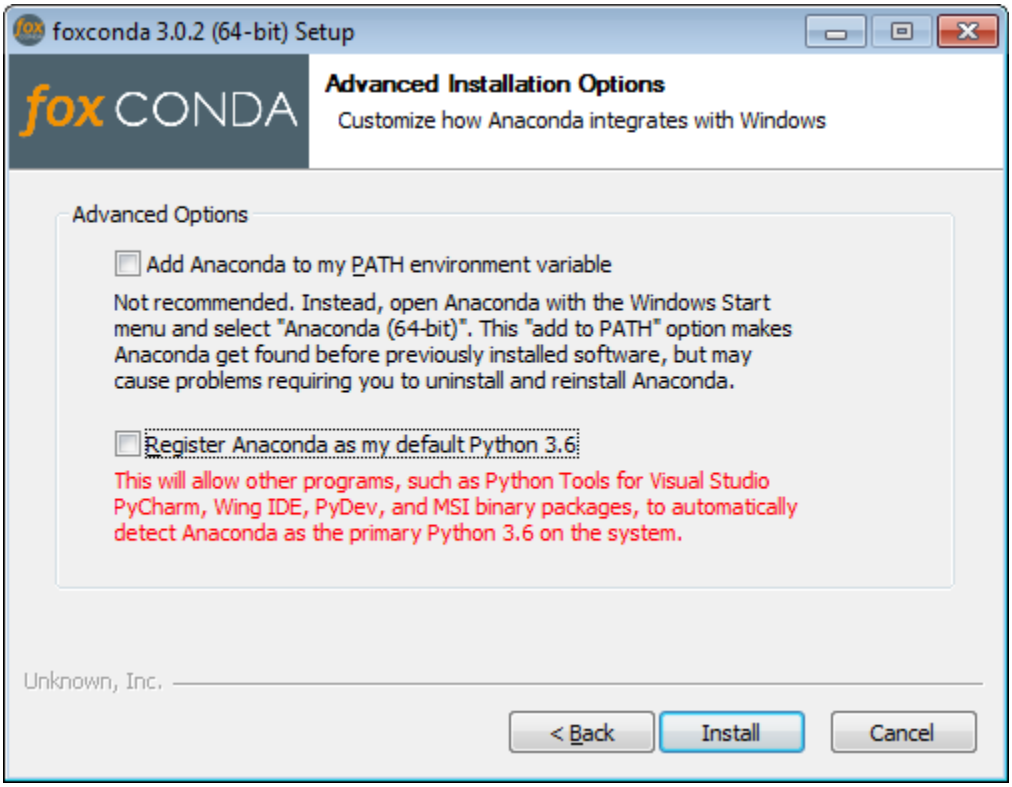



Fig. 9.5: Advanced Options

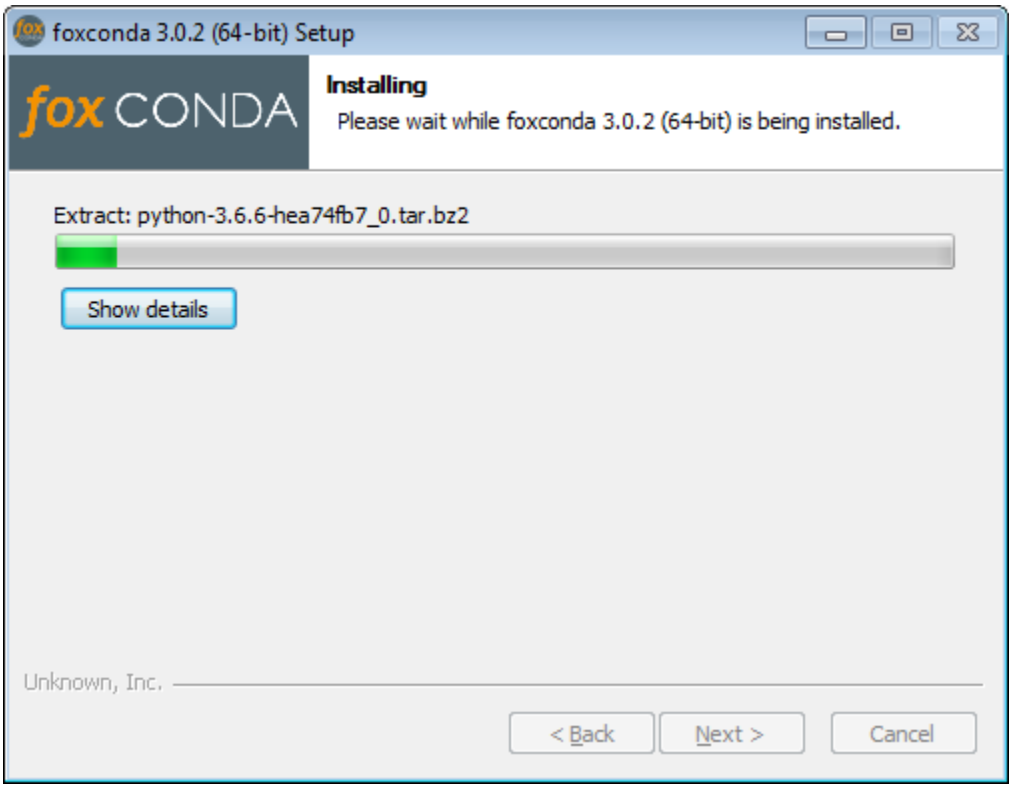



Fig. 9.6: Installation process

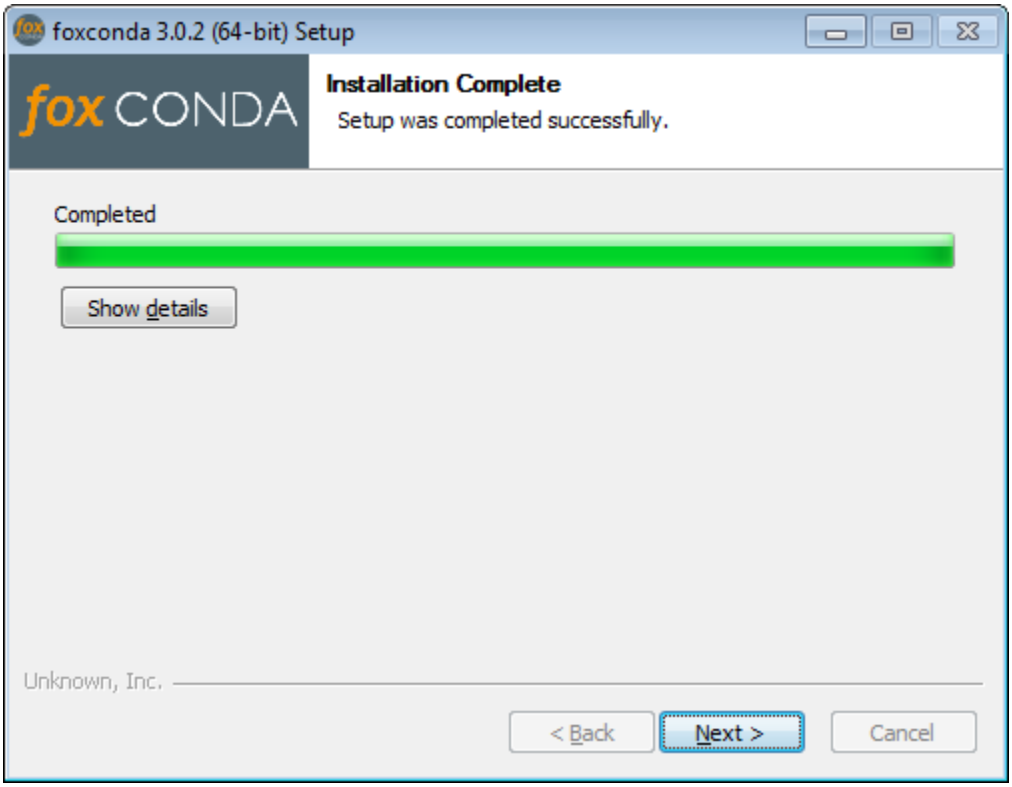



Fig. 9.7: Completed installation step

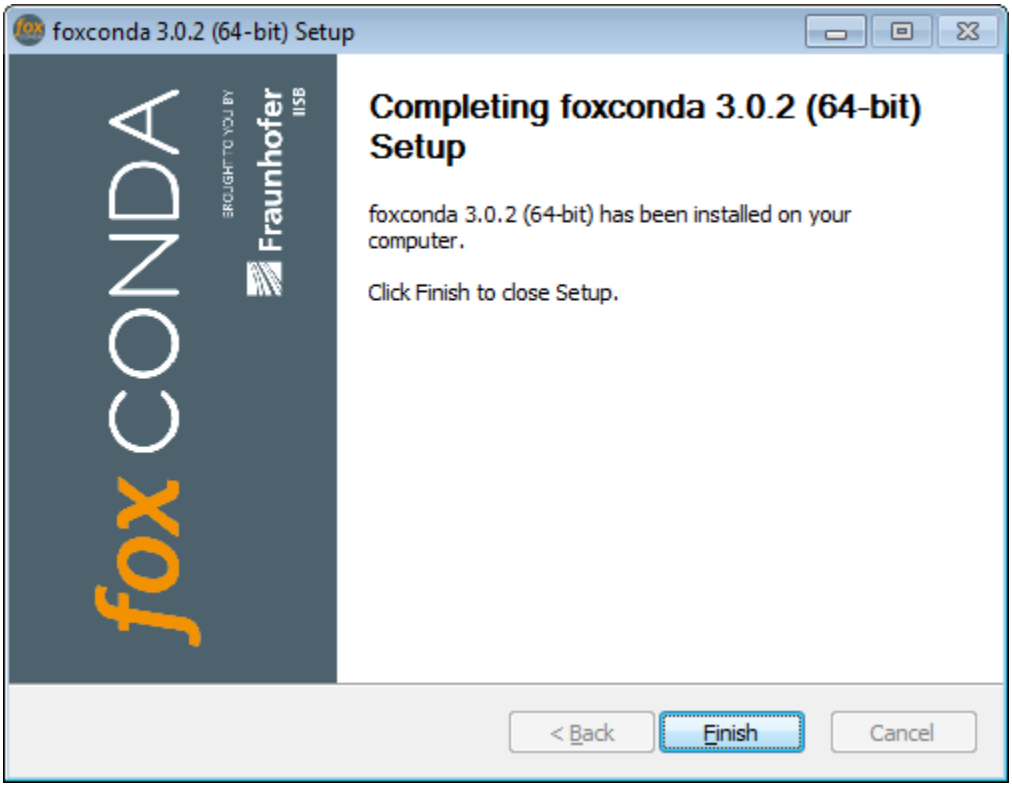



Fig. 9.8: Installation finish

## 9.2 Proxy

Some `conda` commands (e.g., `conda install`) will not work behind a proxy by default. The following steps configure `conda` to work behind a proxy:

1. The `.condarc` is created by running the following command in the `Anaconda Prompt`

Listing 9.1: conda config command

```
conda config
```

   The `.condarc` file is created in the user home directory (e.g., `C:\users\username\.condarc`)

2. Open the `.condarc` with a text editor and add the proxy information:

Listing 9.2: conda proxy configuration

```
proxy_servers:
  http: http://my-http-proxy-server:Port
  https: https://my-https-proxy-server:Port
```

Details on proxy configuration are found in the official `conda` documentation at Configure conda for use behind a proxy server (proxy_servers).

More detailed information on conda configuration is also found in the conda configuration documentation.

## 9.3 Software Related Frequently Asked Questions

### 9.3.1 Where are the Sources?

The sources are available on GitHub.

### 9.3.2 What Libraries and Programs Must be Installed?

None, because they are installed with the foxconda3 installer.

CHAPTER 10

# Building the foxBMS Software and Documentation

## 10.1 Requirements

The build process of foxBMS heavily depends on Python, for example, for code generation purposes. foxBMS hence comes with its own Python distribution, called foxconda3, powered by Anaconda. These build instructions assume that foxconda3 was successfully installed and that the `PATH` environment has been adjusted accordingly. For further information refer to the foxconda3 documentation (*Installing foxconda3*).

On p. 56, it was suggested not to set PATH.

## 10.2 Obtaining the Sources

Download or clone the foxBMS repository from github.com/foxBMS/.

**Warning:** Do not change directory names or the structure inside `foxbms`. If this is changed most, if not all, `wscripts` have to be heavily adapted and this can get very complex extremely fast.

## 10.3 Building the Binaries and Documentation

**Note:** All commands described here must be run in the `Anaconda Prompt` provided by foxconda3.

foxBMS targets can be build using the command line or using the foxBMS Eclipse workspace. This section describes the build from command line. Details on building the binaries using the Eclipse Workspace can be found in "*Eclipse Workspace Setup and Flashing*".

The tool for building targets is found in `foxbms\tools` and is called `waf`. A help is displayed by running `python tools\waf -h`.

In the foxBMS project, several targets can be built. The output is stored in a subdirectory of `\build\`.

Waf is a Python-based modern build tool. It works in a similar way to "make" but uses two stages to speed up the building---configuration and bilding.

- To be able to build binaries and documentation, the project needs to be configured once:

```
python tools\waf configure
```

This will create a directory `build` including the configuration files.

- Primary MCU:
  - Doxygen documentation
  - This target is built with `python tools\waf doxygen_primary`.
  - The output directory is `build\primary\doxygen`.
  - The main document of the software documentation is found at `build\primary\doxygen\html\index.html`.
  - Binaries

    **Note:** The output files where the filenames end with `.unpatched` are intermediate build results which can not run on the hardware.

    - This target is built with `python tools\waf build_primary`.
    - The output directory is `build\primary\embedded-software`.
    - The files generated in the directory `build\primary\embedded-software\mcu-primary\src\general` are:
      - `foxbms_primary.elf`,
      - `foxbms_primary_flash.bin`,
      - `foxbms_primary_flashheader.bin` and
      - `foxbms_primary.hex`.
    - The primary mcu binaries and documentation are cleaned by running `python tools\waf clean_primary`.
- Secondary MCU:
  - Doxygen documentation
  - This target is built with `python tools\waf doxygen_secondary`.
  - The output directory is `build\secondary\doxygen`.
  - The main document of the software documentation is found at `build\secondary\doxygen\html\index.html`.
  - Binaries
    - This target is built with `python tools\waf build_secondary`.
    - The output directory is `build\secondary\embedded-software`.
    - The files generated in the directory `build\secondary\embedded-software\mcu-secondary\src\gene` are:
      * `foxbms_secondary.elf`,
      * `foxbms_secondary_flash.bin`,
      * `foxbms_secondary_flashheader.bin` and

        * foxbms_secondary.hex.

    - The secondary mcu binaries and documentation are cleaned by running `python tools\waf clean_secondary`.

- General documentation (`sphinx` documentation):
  - This target is built with `python tools\waf sphinx`.
  - The output directory is `build\sphinx`.
  - The main document of the software documentation is found in `build\documentation\index.html`.

# CHAPTER 11

# Eclipse Workspace Setup and Flashing

chapter
In this section, it is shown how to set up an Eclipse workspace to work with the foxBMS sources. Within the workspace, it is possible to flash the compiled sources on the foxBMS Master Unit.

## 11.1 Setting an Eclipse Workspace

**Warning:** Every part of this setup instruction must be read carefully and it must be followed step by step, including details. If this is not the case, the Eclipse Workspace will not work.

**Warning:** Only `foxconda3` (see *Installing foxconda3*) is supported. The support for older versions of `foxconda` has been dropped.

In the following, it is described which requirements have to be fulfilled and what steps have to be made to setup an Eclipse workspace for working with the foxBMS sources.

---

**Note:** The following setup instructions use two variables which have to be adapted to the user's settings. The example below shows how to adapt these two variables.

- **Variable 1**: Path to the project directory
  - The project directory is `foxbms`. The path to the project is therefore `path\to\foxbms` (e.g., `C:\Users\username\Documents\foxbms`, `C:\projects\foxbms`, etc.). This means, during this setup procedure where ever the variable `path\to\foxbms` occurs it needs to be replaced with actual project directory on the machine (e.g., `C:\Users\username\Documents\foxbms` or `C:\projects\foxbms`, etc.). Note that we can just use "C:\foxbms" for conveninece.
  - The path to the project directory must not contain spaces.
- **Variable 2**: Path to `foxconda3` installation

- The preferred installation path of `foxconda3` is `C:\foxconda3`. This path will be used during the setup. If this is changed, it is up to the reader to change it at time to the appropriate path. It is strongly recommended to install foxconda to `C:\foxconda3`.
- If `foxconda3` is installed into `C:\foxconda3` some steps of these instructions can be skipped. These steps are indicated.

### 11.1.1 Requirements

The following requirements have to be fulfilled to be able to set up an Eclipse Workspace for working with the foxBMS sources.

- `foxconda3` is installed.
- foxBMS source files are downloaded to `path\to\foxbms`. Getting the source files is described in section "*Building the foxBMS Software and Documentation*".
- Eclipse CDT , in version Oxygen.1.Release (4.7.1a). Eclipse CDT Oxygen.1.Release (4.7.1a) can be downloaded form eclipse.org. After downloading the `.zip file` and extracting it, `eclipse.exe` must be started to launch `Eclipse CDT`.
- The following plugins must be installed into `Eclipse CDT`. `PyDev` (PyDev project website) is mandatory, while `LiClipseText` (LiClipseText project website) is optional. However it is recommended to install both plugins. The plugins can be installed from the `Eclipse Marketplace`,

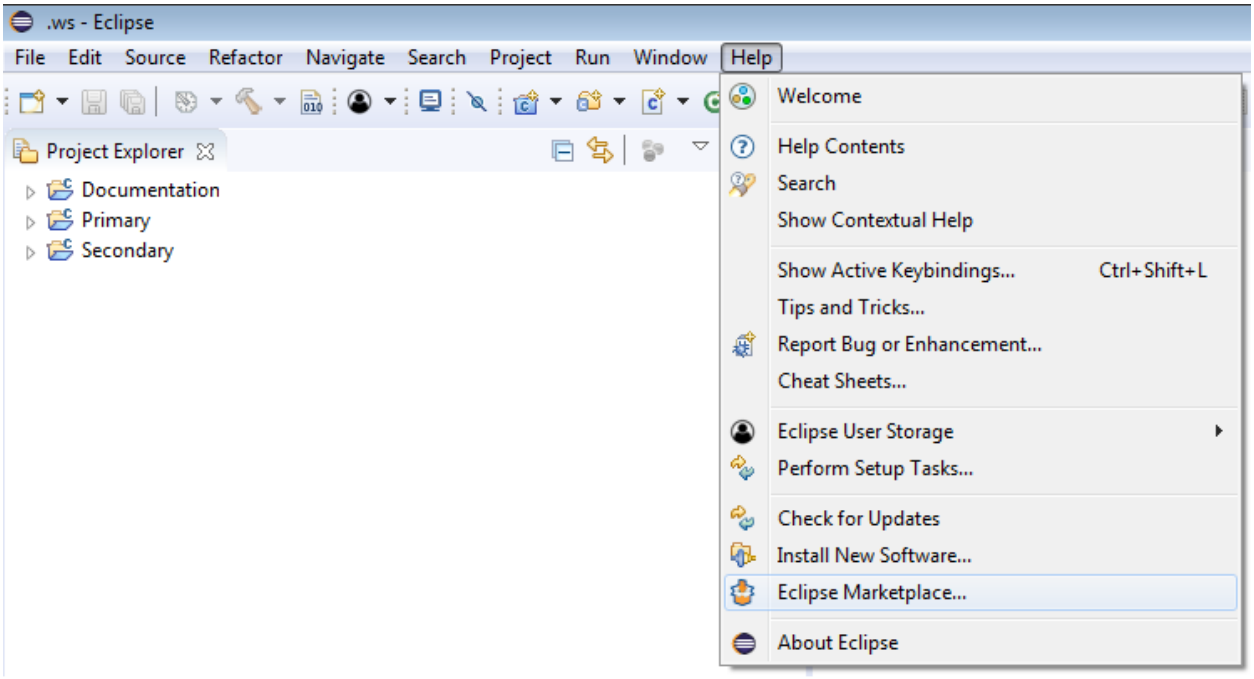



Fig. 11.1: `Eclipse Marketplace`

- Installing `PyDev`
  - Click **Install** to install the plugin.
- Installing `LiClipseText`

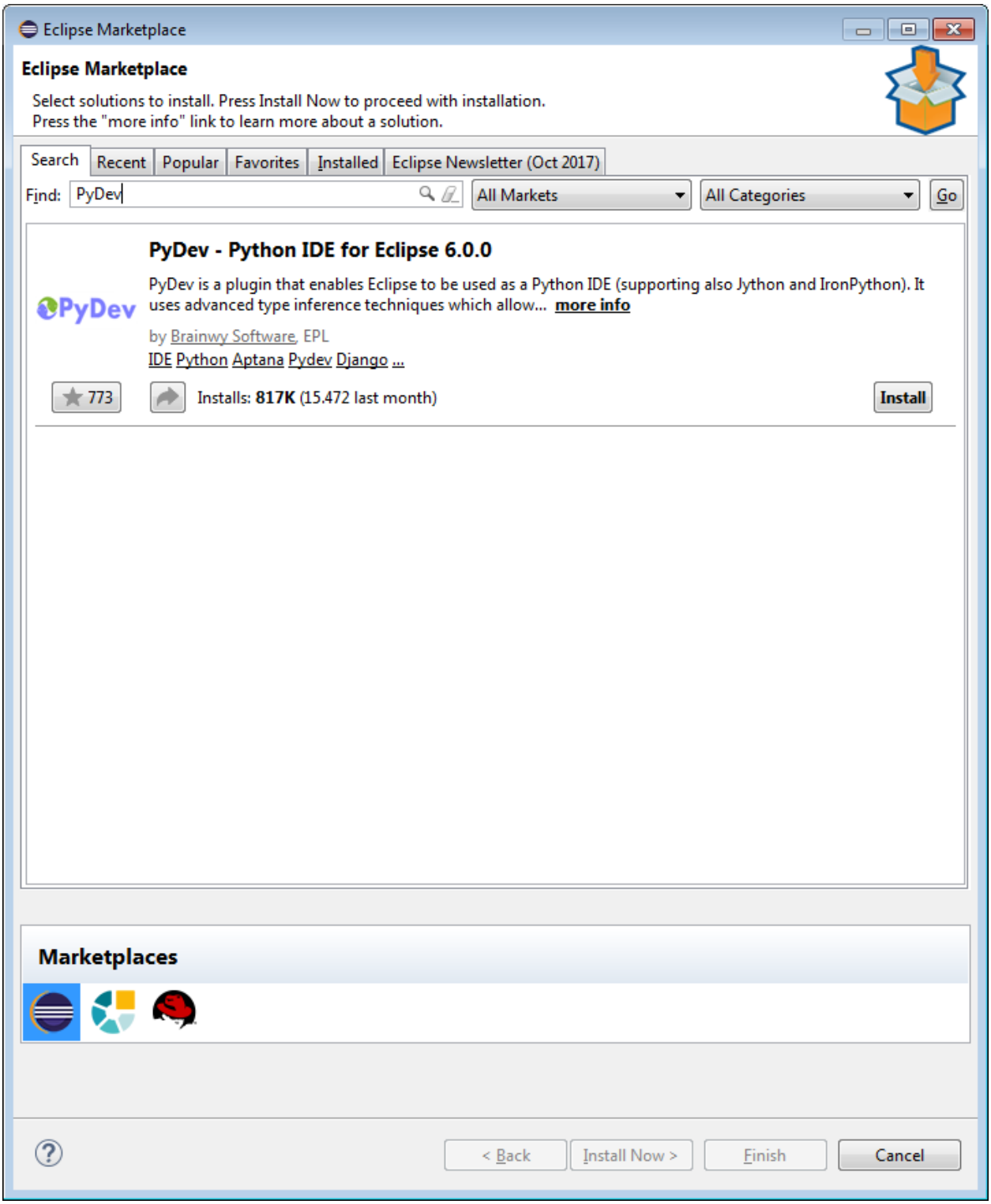



Fig. 11.2: `PyDev` Eclipse plugin

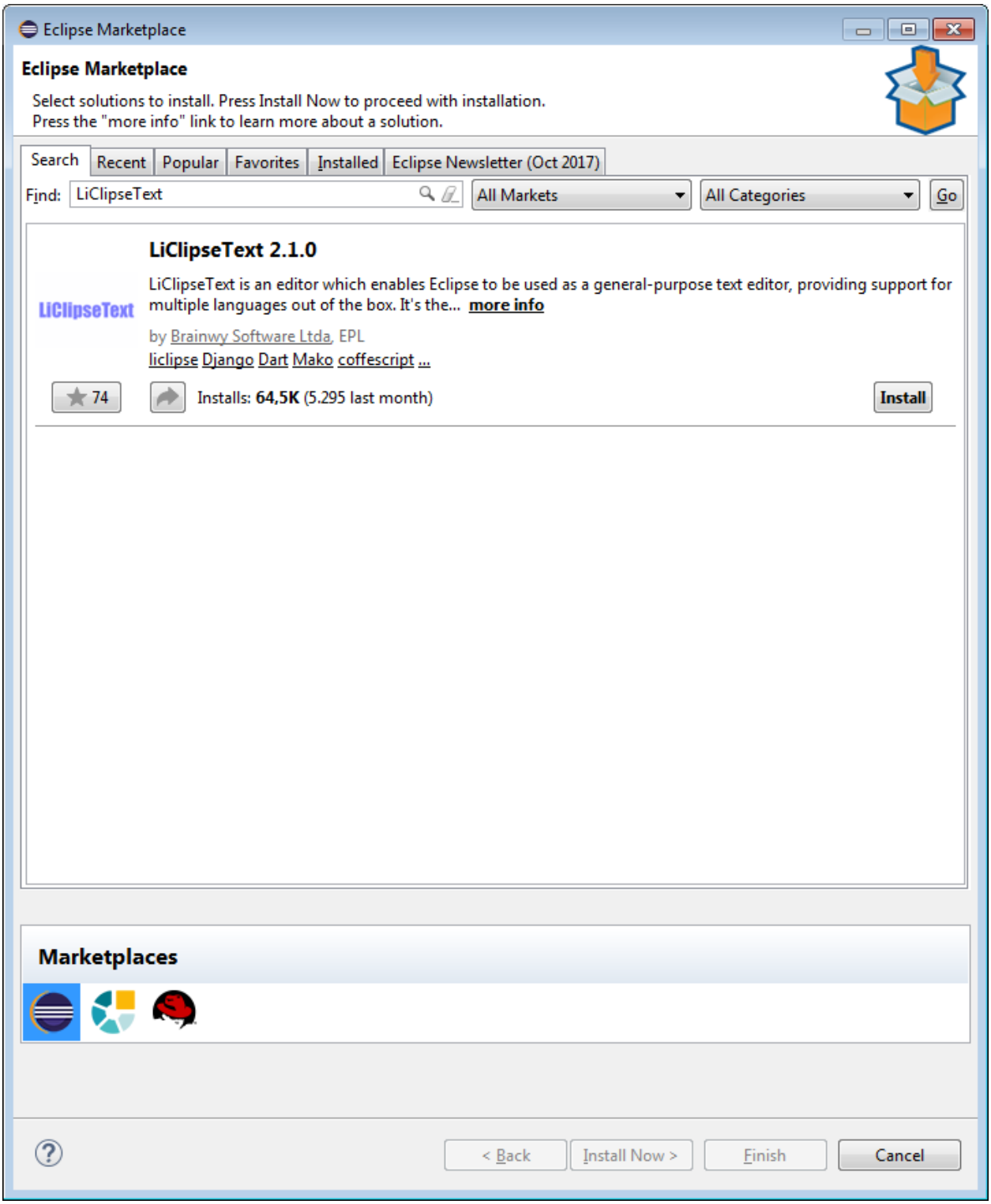



Fig. 11.3: `LiClipseText` Eclipse plugin

* Search the "LiClipseText" plugin, as shown in Fig. 11.3,

* Click Install to install the plugin.

### 11.1.2 How to Setup the Eclipse Workspace

The setup process is divided into following parts:

1. *Creating the Workspace*
2. *Configuring the Python Interpreter*
3. *Importing the Project*
4. *Configuring the Project PATHs*
5. *Configuring the Project Includes*
6. *Testing the Project Setup*

The order of these steps must **not** be changed.

#### Creating the Workspace

The Eclipse Workspace will be called `.ws`.

1. Start `Eclipse` and click `File` -> `Switch Workspace` -> `other` and configure `path\to\foxbms\.ws` as workspace, see Fig. 11.4.

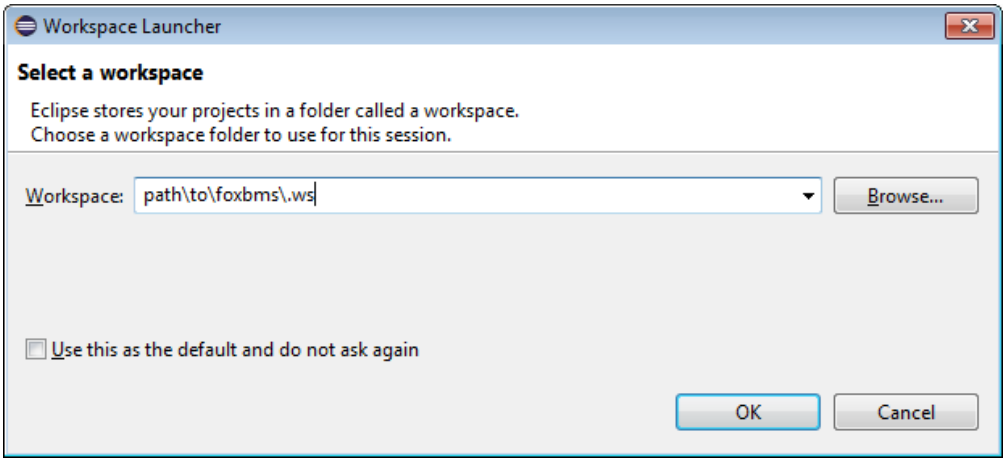



Fig. 11.4: Selecting the correct workspace location

2. Click Launch

The foxBMS Eclipse Workspace is now successfully created.

#### Configuring Default Encoding and Newline Delimiter

1. Open `Window` -> `Preferences` -> `General` -> `Workspace`, as shown in Fig. 11.5.

2. Set `Text file encoding` to `Other: UTF-8` and `New text file line delimiter` to `Other: Windows`

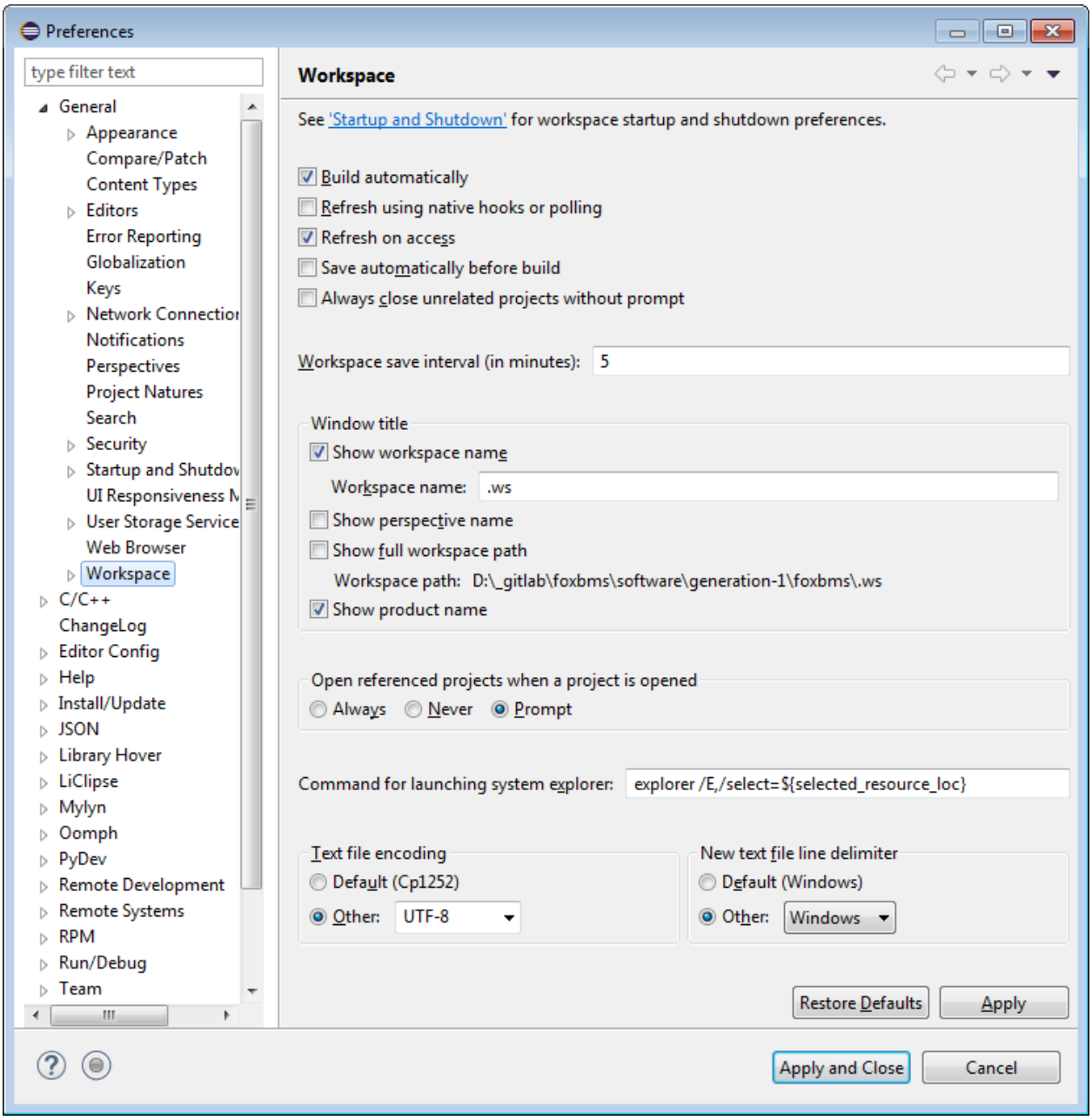



Fig. 11.5: Setting the default file encoding and the line delimiter

3. Click Apply and Close

## Configuring the Python Interpreter

1. Open `Window` -> `Preferences` -> `PyDev` -> `Interpreters` -> `Python Interpreter`, as shown in Fig. 11.6.
2. Click New... . In the window ~~`Select Interpreter`~~ use

2. In the pop-up "Selectt Interpreter" window, shown in Fig. 11.7, use

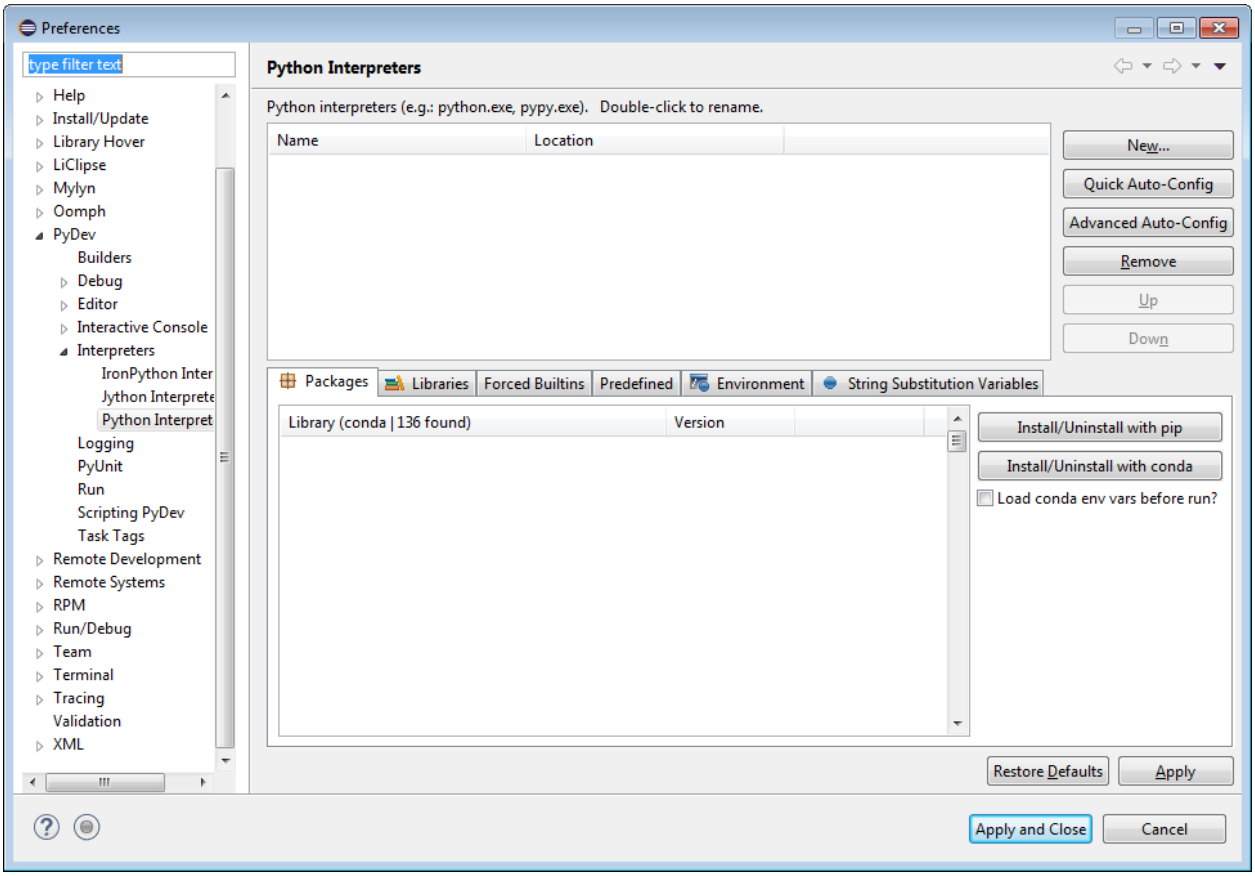



Fig. 11.6: Selecting the python interpreter

1. `foxconda3` for the `Interpreter Name`
2. `C:\foxconda3\python.exe` for `Interpreter Executable`.

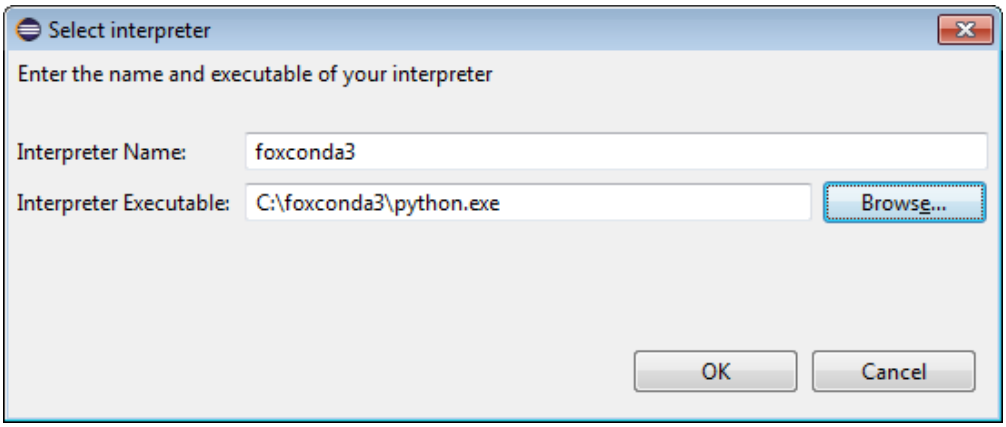



Fig. 11.7: Choosing the python interpreter name and executable

3. Click OK
4. In menu `Selection needed` do not change anything and click OK , shown in Fig. 11.8,
5. Click Apply and Close in the next window, as shown in Fig. 11.9.

`foxconda3` is now successfully selected as Python interpreter.

### Importing the Project

1. Import the project via `File` -> `Import` , as shown in Fig. 11.10.
2. Click `General` -> `Existing Projects into Workspace`, as shown in Fig. 11.11.
3. Click Next >
4. Chose `Select archive file:` and use `path\to\foxbms\tools\eclipse\foxbms-eclipse-project.zip` This will list the foxBMS projects, as shown in Fig. 11.12.

---

**Note:** Always import all projects that appear, as the number of projects might increase with future updates and this help might not be up-to-date.

---

5. Click Finish to complete the project import process.

The foxBMS projects are now successfully imported.

### ~~Configuring the Project PATHs~~

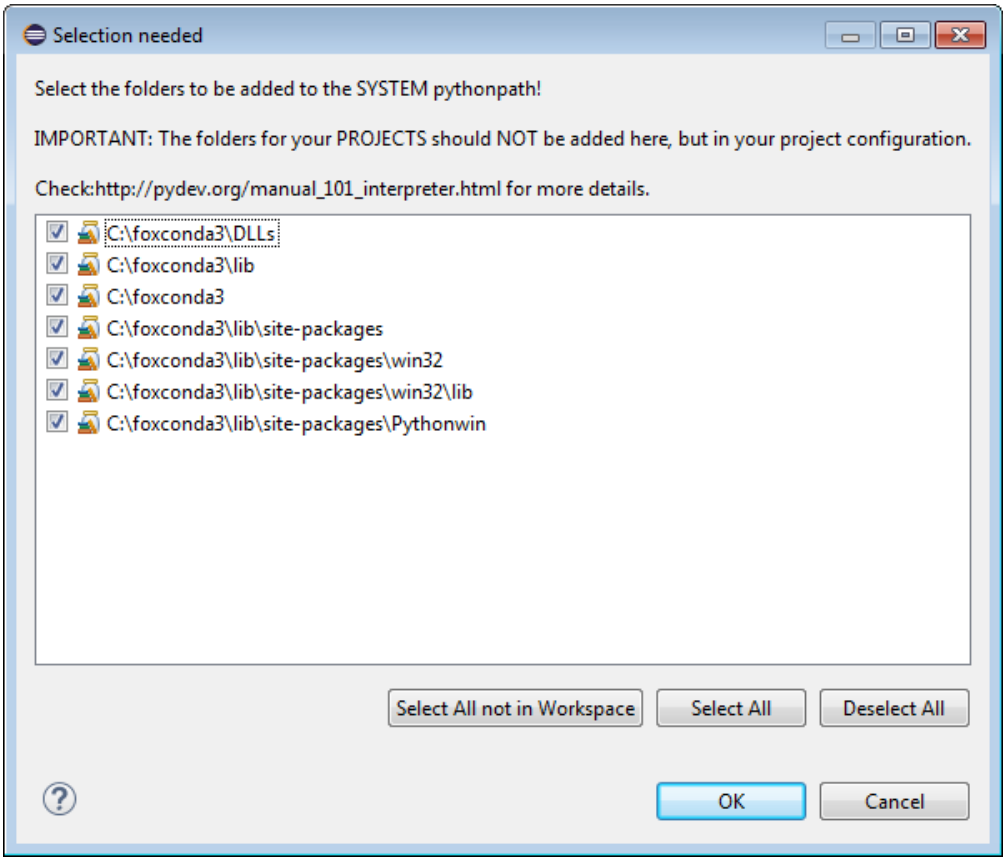



Fig. 11.8: Adding the `PYTHONPATH`

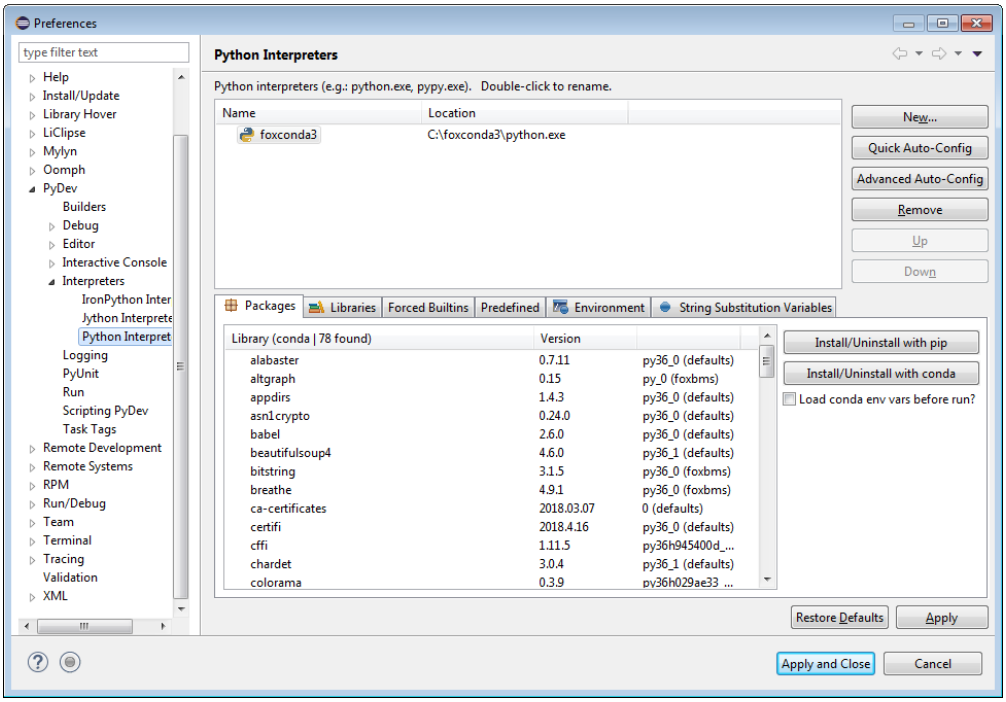



Fig. 11.9: Configuration done

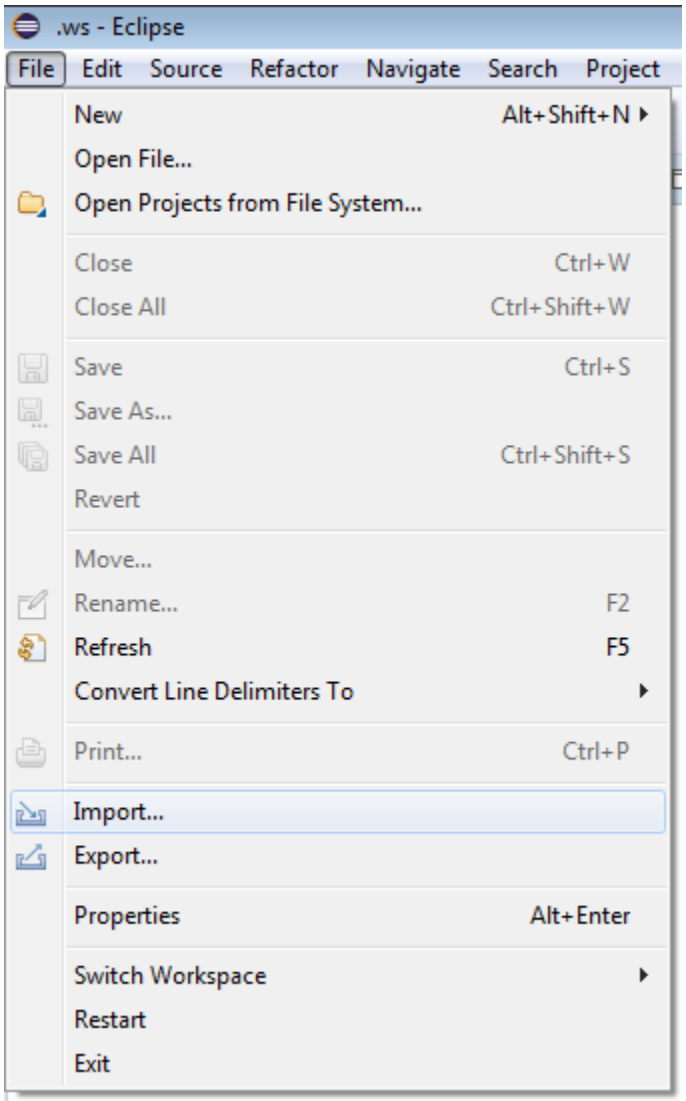



Fig. 11.10: Select `Import`

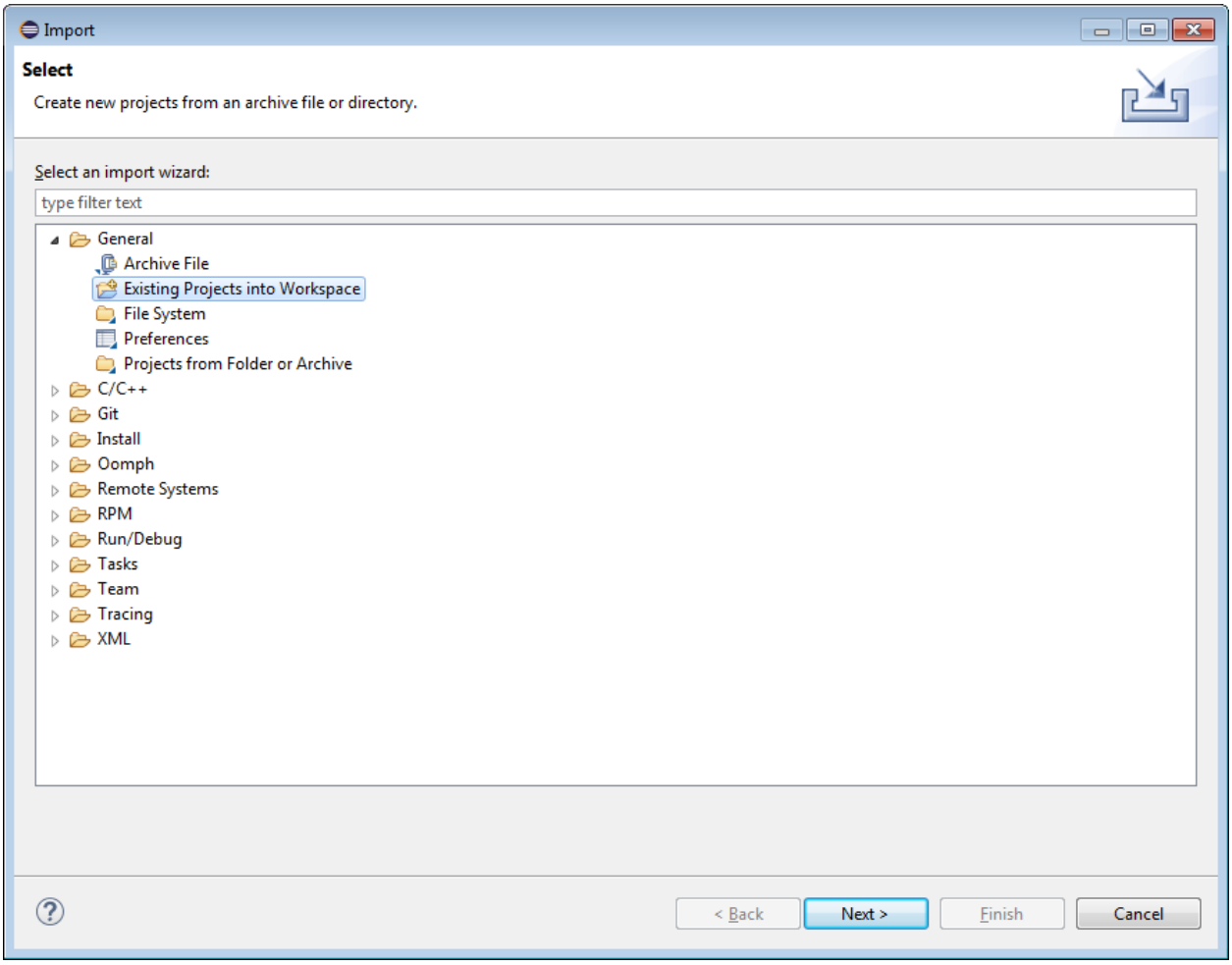



Fig. 11.11: Select projects import by `Existing Projects into Workspace`

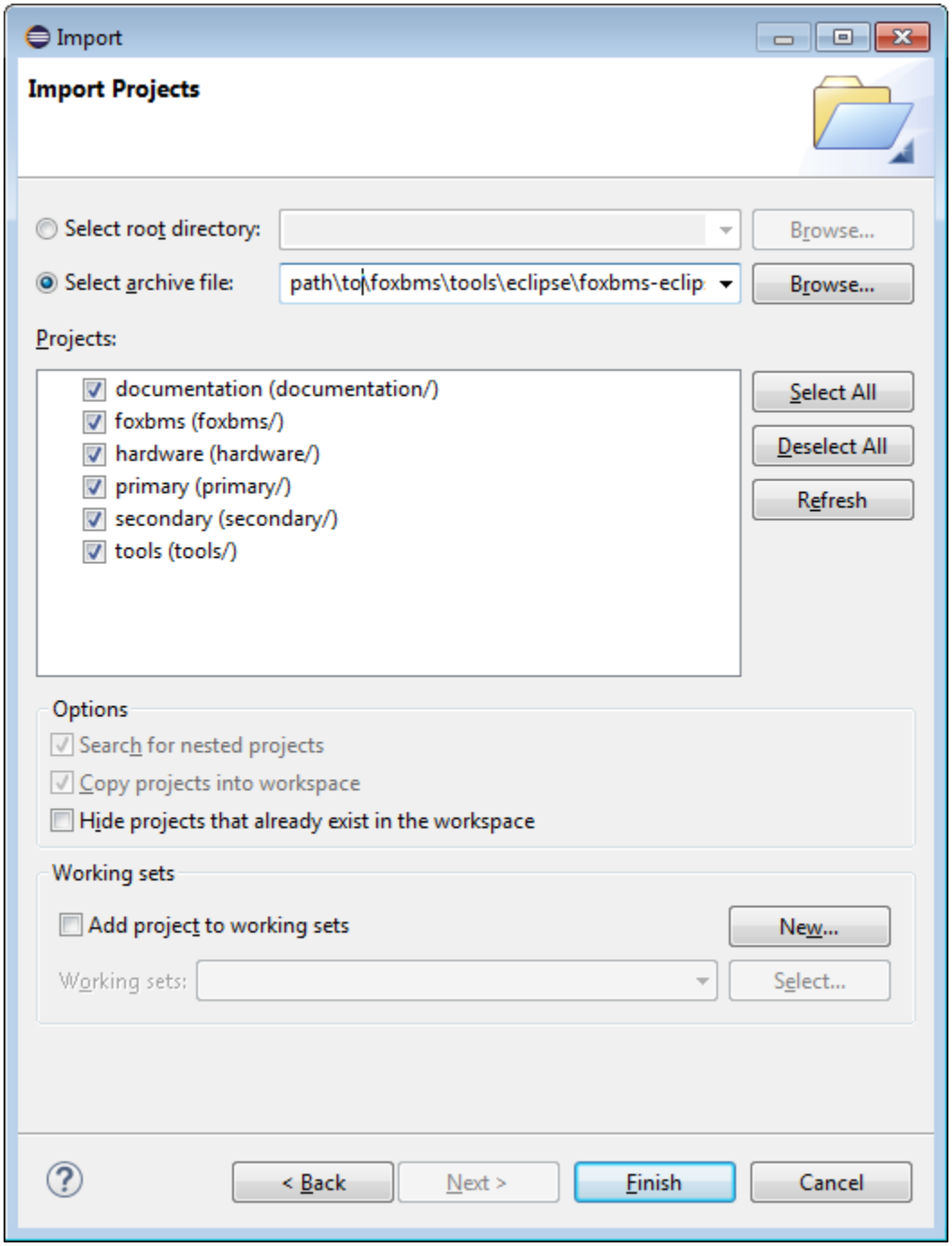



Fig. 11.12: Select all projects to be imported

### Configuring the Project PATHs

**Note:** If `foxconda3` was installed in the default installation path (`C:\foxconda3`), this section can be skipped.

For this setup: Change the path `C:\foxconda3` to the path where `foxconda3` was installed to.

1. Select the `Primary` project and click right and select `Properties`, as shown in Fig. 11.13.
2. Goto `C/C++ Build` -> `Environment`. Select `[All configurations]`

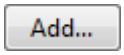
   The result should look like this: the one shown in Fig. 11.14. Use the steps on p. 79 after Fig. 11.14 to do the configurations.
3. Repeat these steps for the following projects:
   - `documentation`
   - `foxbms`
   - `secondary`

The foxBMS projects now include the correct environment settings.

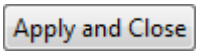
### Configuring the Project Includes

**Note:** If `foxconda3` was installed in the default installation path (`C:\foxconda3`), this section can be skipped.

For this setup: Change the path `C:\foxconda3` to the path where `foxconda3` was installed to.

1. Select the `primary` project and click right. Goto `C/C++ General` -> `Paths and Symbols` and switch to Tab `Includes`, as shown in Fig. 11.16.
2. Click 

for `Assembly`, `GNU C` and `GNU C++` and configure the following two paths as `Includes`
   1. `C:\foxconda3\Library\arm-none-eabi\include`
   2. `C:\foxconda3\Library\lib\gcc\arm-none-eabi\4.9.3\include`

   1. Click 

3. Repeat this procedure for the `secondary` project.
4. Restart Eclipse

The foxBMS projects are now successfully configured.

### Testing the Project Setup

The system may complain "cannot find tools/waf." The default folder for building foxbms is c:\foxbms\.ws\foxbms. There are two approaches:
1. Copy tools/waf to this folder.
2. Change the directory to c:\foxbms in the terminal.
The second approach works better.

1. `foxbms` project

   1. Select the `foxbms` project and click  button and select `1 configure`. The project is now configured and all options (building documentation and binaries) can now be used.

      ```
      'configure' finished successfully (0.450s)
      ```

(Continue on P. 81.)

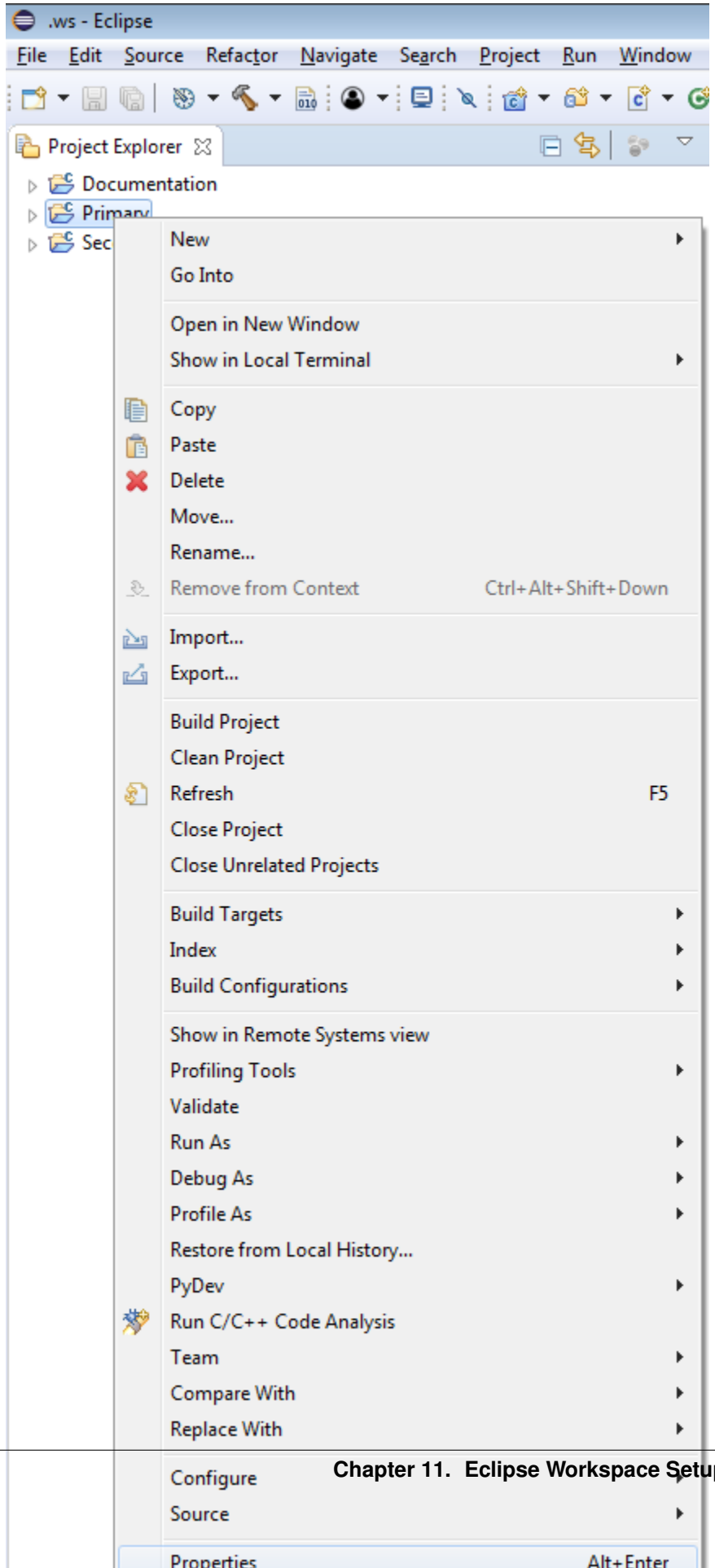

.ws - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Project Explorer
Documentation
New
Go Into
Open in New Window
Show in Local Terminal
Copy
Paste
Delete
Move...
Rename...
Remove from Context Ctrl+Alt+Shift+Down
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Targets
Index
Build Configurations
Show in Remote Systems view
Profiling Tools
Validate
Run As
Debug As
Profile As
Restore from Local History...
PyDev
Run C/C++ Code Analysis
Team
Compare With
Replace With
Configure
Source
Properties Alt+Enter

Fig. 11.13.

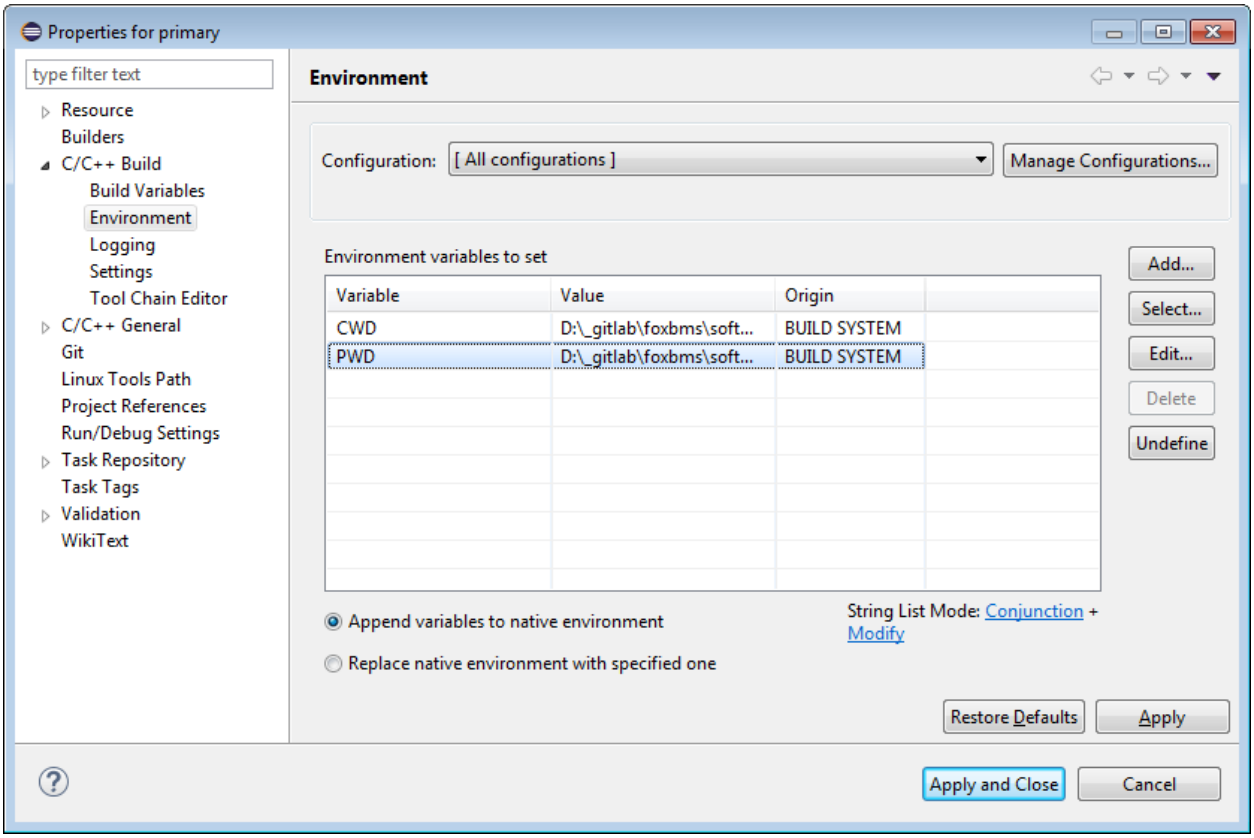



Fig. 11.14: `C/C++ Build` project properties

1. Click Add... and

   Note: The text here are steps used to go from Fig. 11.14 to Fig 11.15.

   1. use `FOXCONDA3` for `Name`
   2. use `C:\foxconda3;` `C:\foxconda3\bin;` `C:\foxconda3\Scripts;` `C:\foxconda3\Library\bin;` `C:\foxconda3\Lib;` for `Value`.

      **Warning:**

      - The `Value` property must not include spaces.
      - Do not add a backslash (`\`) at the end of the paths.
      - These paths need to be in a single line.

   3. Check `Add to all configurations`
   4. Click OK
2. Click Add... and
   1. Use `PATH` for `Name`
   2. Leave `Value` empty
   3. Click OK
3. Select the `PATH` entry and click Edit...
   1. Delete all entries in `Value`
   2. Add `${FOXCONDA3};` to `Value`
   3. Click OK

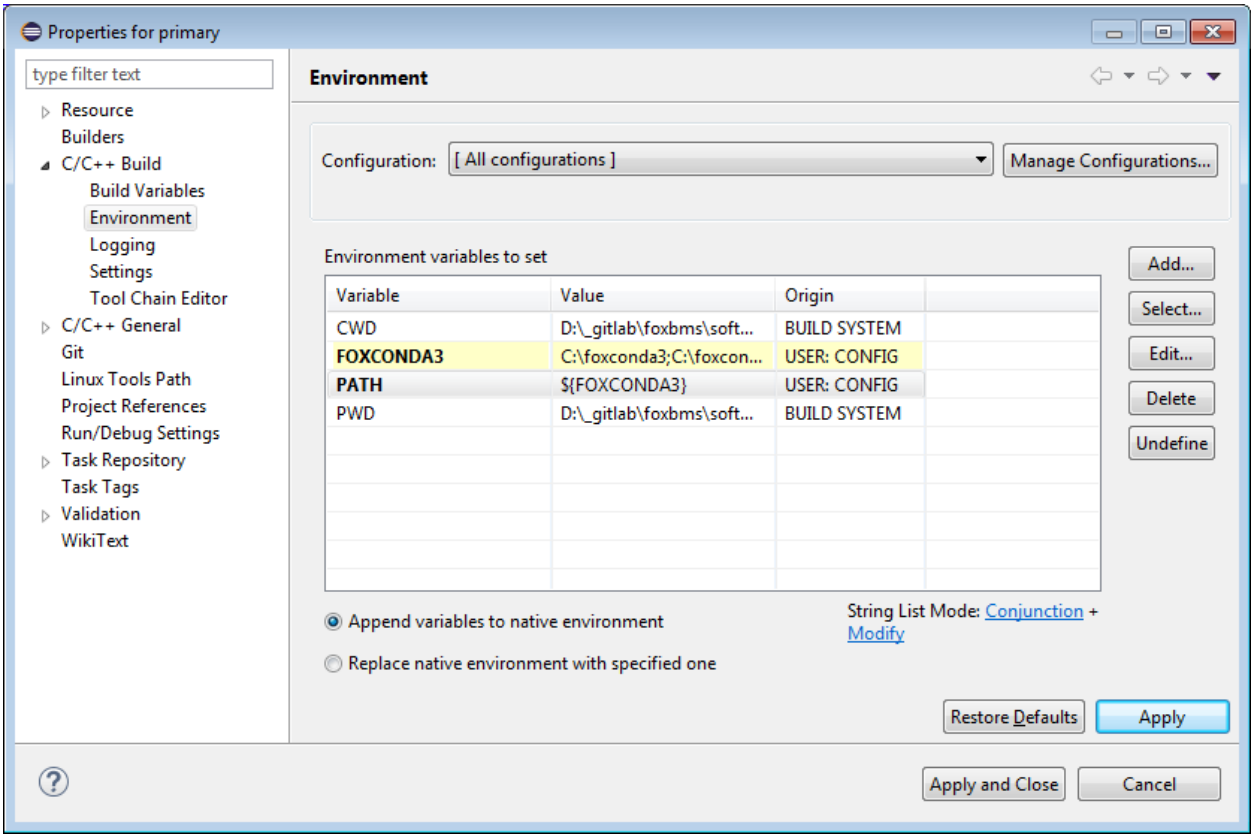



Fig. 11.15: `Environment` settings for the project.

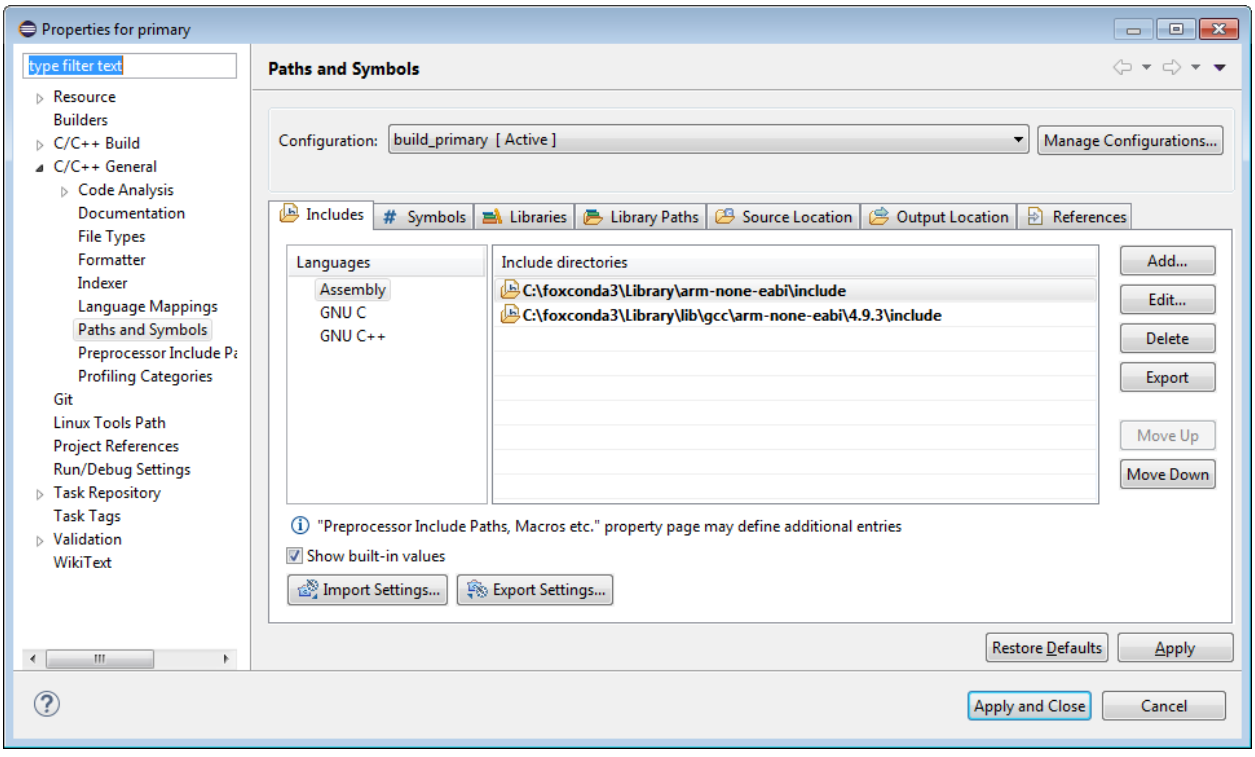



Fig. 11.16: Adding the gcc-includes to a project

(Cont'd from P. 77.)

2. Select the `foxbms` project and click  button and select `2 help`. The project help for building is printed.

2. `documentation` project

   1. Select the `documentation` project and click on the dropdown menu of the  button and select `1 sphinx` to build the general foxBMS documentation.

      ```
      'sphinx' finished successfully (0.105s)
      ```

   2. The `Documentation` project is now successfully tested.

3. `primary` project

   1. Select the `primary` project and click on the dropdown menu of the  button and select `1 build_primary` to build the foxBMS binaries of the primary mcu.

      The binary generation was successful, if the Eclipse console puts the following line at the bottom:

      ```
      'build_primary' finished successfully (1.340s)
      ```

   2. Select the `primary` project and click on the dropdown menu of the  button and select `2 doxygen_primary` to build the foxBMS embedded documentation of the primary mcu.

      The primary mcu dopxygen documentation generation was successful, if the Eclipse console puts the following line at the bottom:

      ```
      'doxygen_primary' finished successfully (46.906s)
      ```

   3. Select the `primary` project, right click the project and select `Clean Project`

      The cleaning of the primary mcu binaries and doxygen documentation was successful, if the Eclipse console puts the following line at the bottom:

      ```
      'clean_primary' finished successfully (1.065s)
      ```

   4. The `primary` project is now successfully tested.

4. `secondary` project

   1. Select the `secondary` project and repeat all steps of the `primary` project.
   2. The `secondary` project is now successfully tested.

The foxBMS Eclipse Workspace is now successfully tested.

**Warning:** To test flashing in the next step, binaries of the primary and secondary mcu need to be build again.

### 11.1.3 Optional Settings ---Associating wscript files with Python Editor

When working with the build scripts (`wscript`) it is more convenient to have python syntax highlighting. The following has to be configured to achieve it:

1. Click `Window` -> `Preferences`

2. Select `General` -> `Editors` -> `File Associations`, as shown in Fig. 11.17.
3. At the entry `File types` click Add...

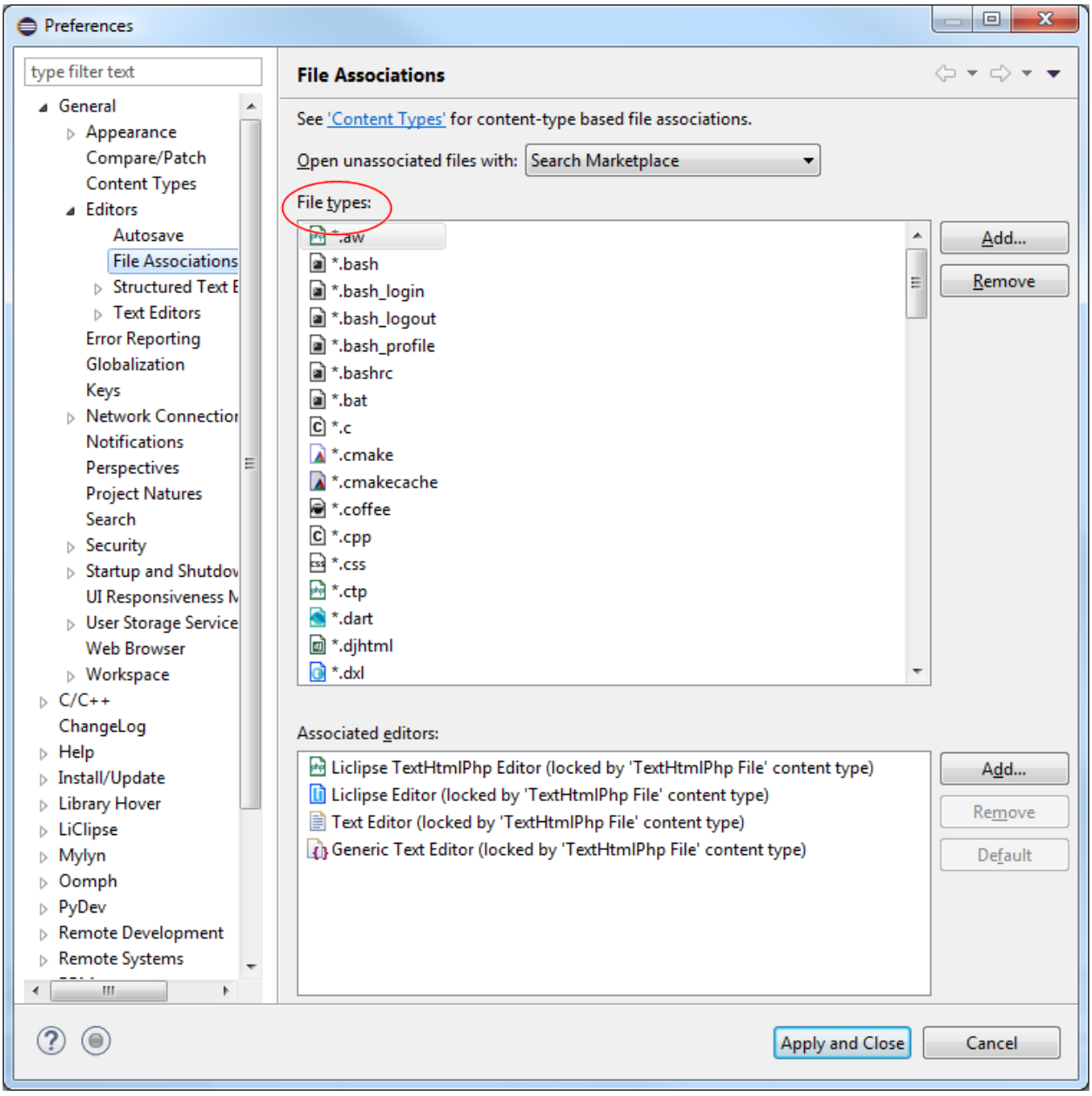



Fig. 11.17: `File Associations` settings

4. Define the file type `wscript` and click OK, as shown in Fig. 11.18.
5. Select the `Python Editor` as and click OK, as shown in Fig. 11.18a.

~~1. `Editor Selection` setting~~

6. Finish the configuration by clicking Apply and Close, as shown in Fig. 11.19.

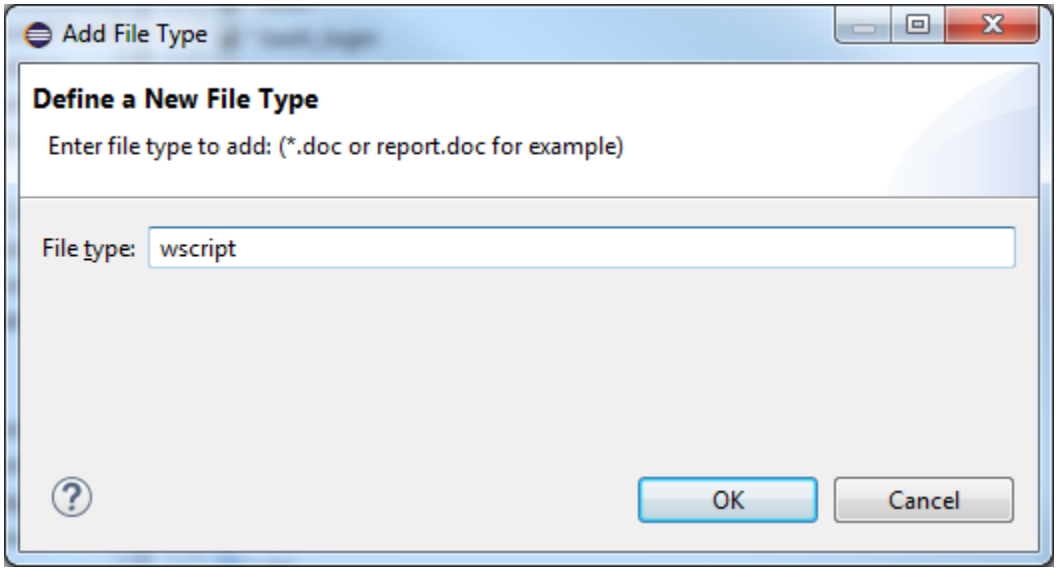



Fig. 11.18: `File type definition`

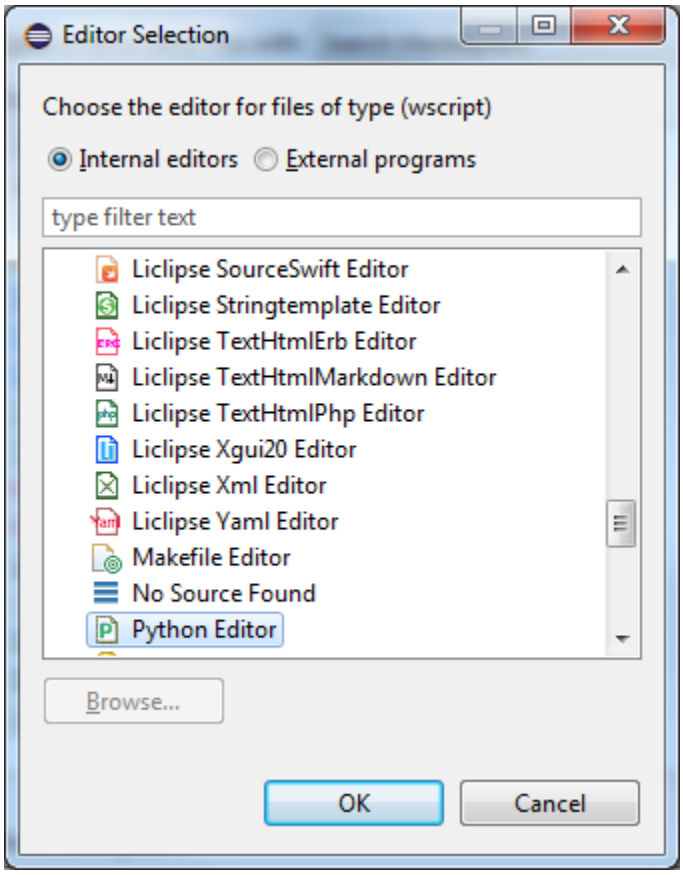



Fig. 11.18a.

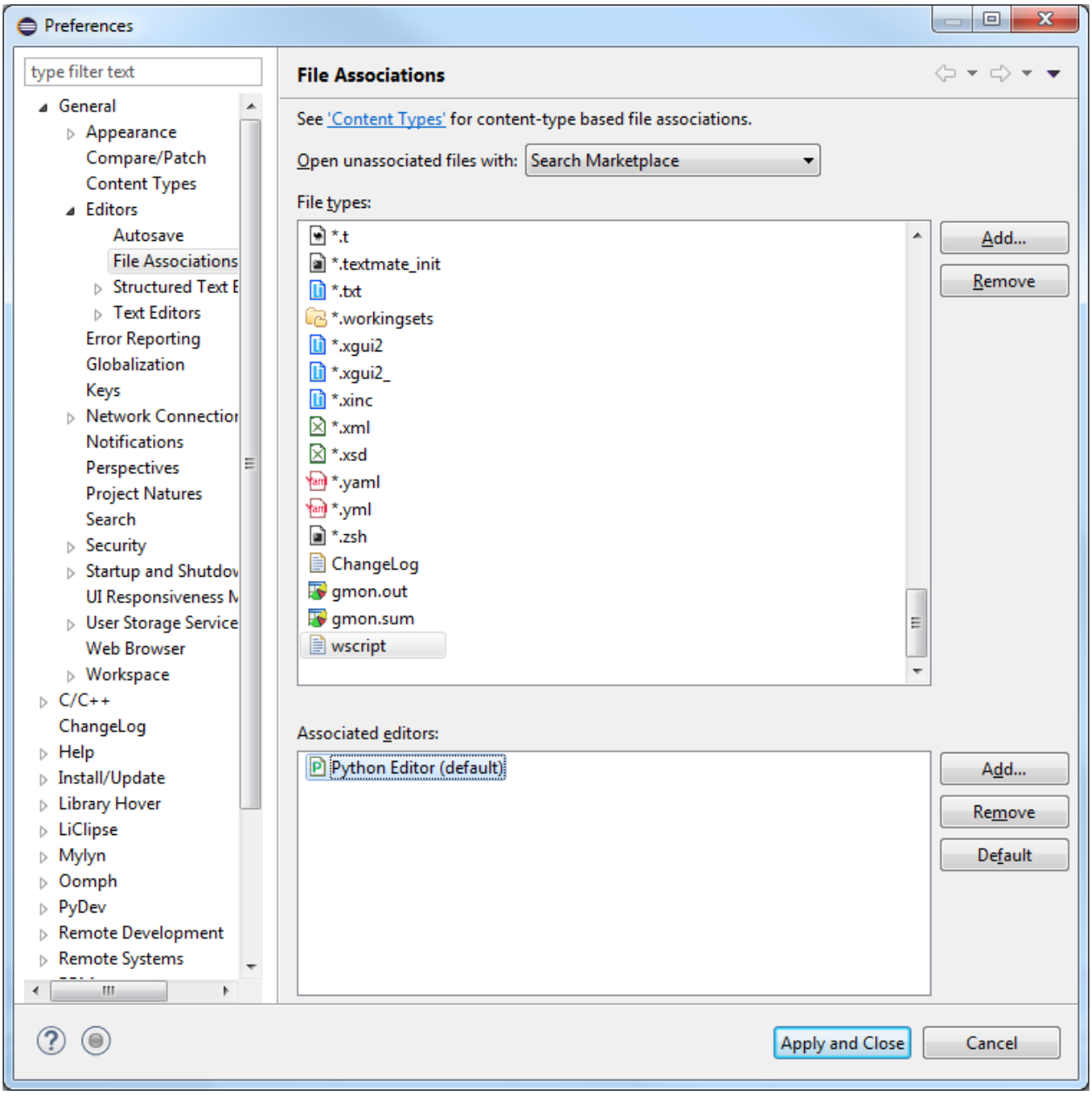



Fig. 11.19: Final file association setting for `wscript`

## 11.2 Flashing foxBMS

This section describes how to supply the foxBMS Master Unit in order to flash both MCU0 and MCU1 with the firmware produced by compiling the C-code source files.

This section shows how to connect the different parts of the foxBMS platform.

### 11.2.1 Convention for Connector Numbering

Fig. 11.20 presents the convention for the connector numbering. It is used throughout the documentation.

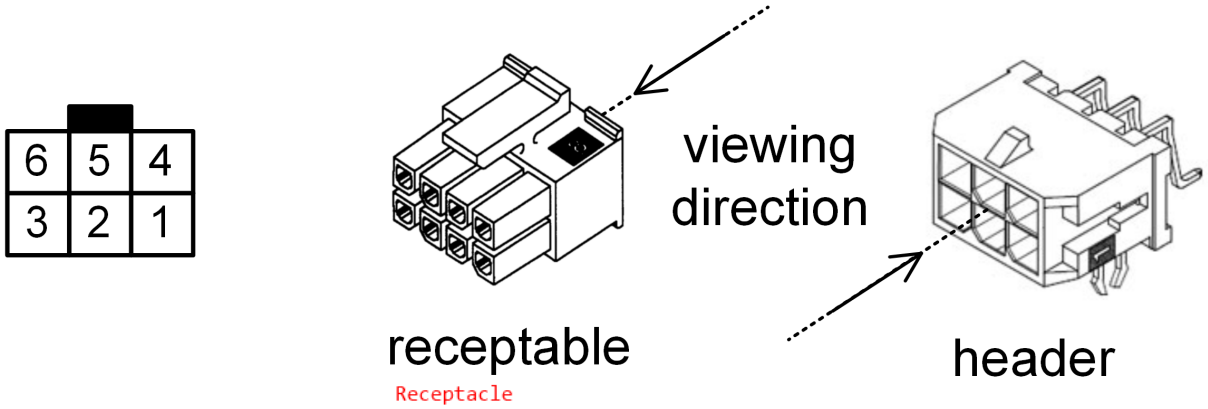



Fig. 11.20: Supply connector pin out, receptable - rear view, header - front view (image source: MOLEX)

There are two types of connectors:

- Header
  Receptacle
- Receptable, plugged into the header

The numbering shown on the left in fig. 11.20 is always valid when viewing in the direction indicated by the arrow with the indication `viewing direction`. This must be taken into account when crimping the receptables.

### 11.2.2 Hardware Setup of foxBMS Master Unit and foxBMS Slave Units

The foxBMS system can be mounted in a metal housing, as shown in fig. 11.21.

Connectors are available, shown in Fig. 11.22, which presents all the connectors of the foxBMS Master Unit.

For this section on flashing, only the connector `Supply` is needed.

### 11.2.3 Supply of the foxBMS Master Unit

The first step is to supply the foxBMS Master Unit, which works with supply voltages between 12V and 24V DC. To supply the foxBMS Master Unit, a connector must be prepared for the `Supply` connector as shown in table 11.1, which describes the different pins used.


Fig. 11.21: foxBMS Master Unit housing

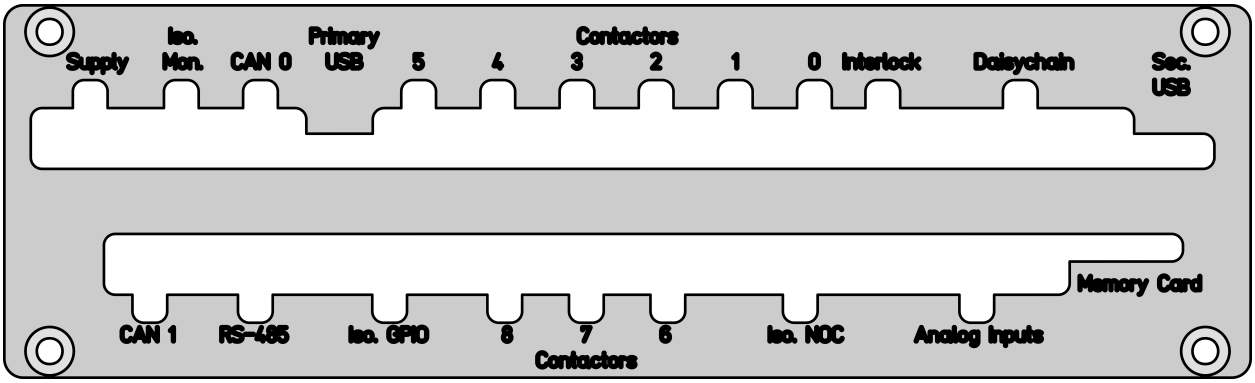



Fig. 11.22: Front view of the foxBMS Master Unit indicating the location of each header

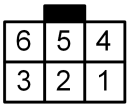


Table 11.1: BMS-Master Board Supply Connector

| Pin | Signal | Input/Output | Description |
|---|---|---|---|
| 1 | SUPPLY_EXT_2 | Input | 12 - 24V |
| 2 | SUPPLY_EXT_2 | Input | 12 - 24V |
| 3 | GND_EXT_2 | Input | GND |
| 4 | SUPPLY_EXT_0 | Input | 12 - 24V |
| 5 | GND_EXT_0 | Input | GND |
| 6 | GND_EXT_2 | Input | GND |

The supply is separated as follows:

- The microcontrollers and the isolation devices are supplied through the pins `SUPPLY_EXT_0` and `GND_EXT_0`
- The contactors and the interlock are supplied through the pins `SUPPLY_EXT_2` and `GND_EXT_2`

To power up the foxBMS Master Unit, plug in the supply connector and apply a voltage between 12V and 24V. `SUPPLY_EXT_0` / `GND_EXT_0` and `SUPPLY_EXT_2` / `GND_EXT_2` may be connected to the same source for this initial test. At this point, the foxBMS Master Unit should draw approximately 150mA at 12V or 110mA at 24V.

### 11.2.4 Primary and Secondary MCU

The BMS-Master Board has two MCUs: primary (MCU0) and secondary (MCU1). ~~The~~ MCU1 is present to ensure redundant safety especially when used in research and development prototyping.

First, the primary MCU will be flashed.

In order to program the primary MCU, the mini USB jack indicated as `Prim. USB` in fig. 11.22 must be used to connect the foxBMS Master Unit to a PC. ~~The foxBMS Master Unit can be connected to a PC immediately.~~ When connecting foxBMS Master Unit for the first time, the required drivers will install automatically.

---

**Note:** Before the connection is made between the foxBMS Master Unit and the computer for the first time, the computer must be connected to the internet, because the operating system might look for drivers on the internet. It this fails, administrator rights are needed to install the driver.

---

In case of problems by the installation of the drivers, administrator rights might be needed. Once the hardware is supplied with the appropriate voltage, the foxBMS binaries can be flashed.

The same procedure must be made for secondary: the mini USB jack indicated as `Sec. USB` in fig. 11.22 must be used to connect the foxBMS Master Unit to a PC. As for the primary MCU, the PC must be connected to the internet before the connection is made with the foxBMS Master Unit for the first time.

### 11.2.5 Flashing the Compiled Sources for the Primary MCU

In the chapter *Building the foxBMS Software and Documentation*, the foxBMS sources have been compiled and the generated binary is ready to be flashed. The MCU0 will be flashed first.

These binaries are needed:

1. a binary file with the header
2. a binary file with the main code

Unless the build process has been altered or different binaries should be flashed, there is no need to change the default file locations. For this guide, the default file locations are used.

The device must be connected to the computer with the USB cable. The MCU0 USB-connector must be used. The `primary` project must be select.ed Clicking on the dropdown menu of the  button and selecting `3 flash_primary` allows flashing the foxBMS binaries of the primary mcu.

---

**Note:** Make sure that CAN0 connector is either disconnected or that there is no traffic on CAN0 bus while flashing `primary` MCU. If this is not the case, the internal bootloader of the controller will select a wrong boot source and the flashing fails.

---

The connection state can also be checked by watching the LEDs on the foxBMS Master Unit hardware: for a running board, the green power LED is on, and the two indicator LEDs (green and red) are blinking alternately. If the device is connected and being flashed, the power-on LED is on but the two indicator LEDs are off.

~~The LEDs for the MCU0 must be used.~~ In the *Cabling foxBMS* chapter, it is explained how to find the MCU0 indicator LEDs.

When the flashing is complete, the flashing windows disappears and the indicator LEDs start to blink again.

Under Windows 7 (64 bit), an issue it can happen --- that the COMPORT number is increased by Windows, leading to problems during flashing. This can be reset in registry, by setting `ComDB` to zero in the entry `HKEY_LOCAL_MACHINE::SYSTEM::CURRENT_CONTROL_SET::CONTROL::COM NAME ARBITER`. This way, all COM ports used are reset.

### 11.2.6 Flashing the Compiled Sources for the Secondary MCU

The same procedure as explained above must be followed for flashing the MCU1. The difference is that the MCU1 USB-connector must be used. The `secondary` project must be selected before clicking on the dropdown menu of the  button. `3 flash_secondary` must then be selected to flash the foxBMS binaries of the secondary MCU mcu.

With the supplied code, ~~the~~ MCU1 checks the temperatures and voltages. If one or more of the values are outside the limits, the MCU1 opens the interlock line (thus opening the contactors). The interlock line remains open until ~~the~~ MCU1 is reset.

---

**Note:** Setting the limits is safety relevant and must be done with care.

---

---

**Note: Working without configuring the right battery cell voltage limits is dangerous and should never be done when real batteries are connected, since they may burn and explode when overcharged or shorted.**

---

The next step shown in *Cabling foxBMS* is to connect a foxBMS Slave Unit to perform voltage and temperature measurement.

### 11.2.7 Debugging the Primary and Secondary MCU

The following two debuggers can be used for debugging and have been tested with foxBMS:

- Segger J-Link Plus, with the 19-Pin Cortex-M adapter (needed to connect to foxBMS)
- Lauterbach ÂµTrace Debugger for Cortex-M

# CHAPTER 12

# Cabling foxBMS

Assume, as guided by

In the preceding chapter sections, the foxBMS Master Unit has been connected to the power supply and the foxBMS firmware has been flashed on the MCU0 and MCU1. This chapter section describes the foxBMS hardware in more details and how to connect the foxBMS Master Unit to the foxBMS Slave Units that perform the cell voltage and temperature measurements. The CAN communication is also described.

**Note:** When the connection is made between the foxBMS Master Unit and the foxBMS Slave Units, both primary and secondary isoSPI daisy chains have to be connected.

## 12.1 Convention for Connector Numbering

Fig. 12.1 presents the convention for the connector numbering. It is used throughout the documentation.

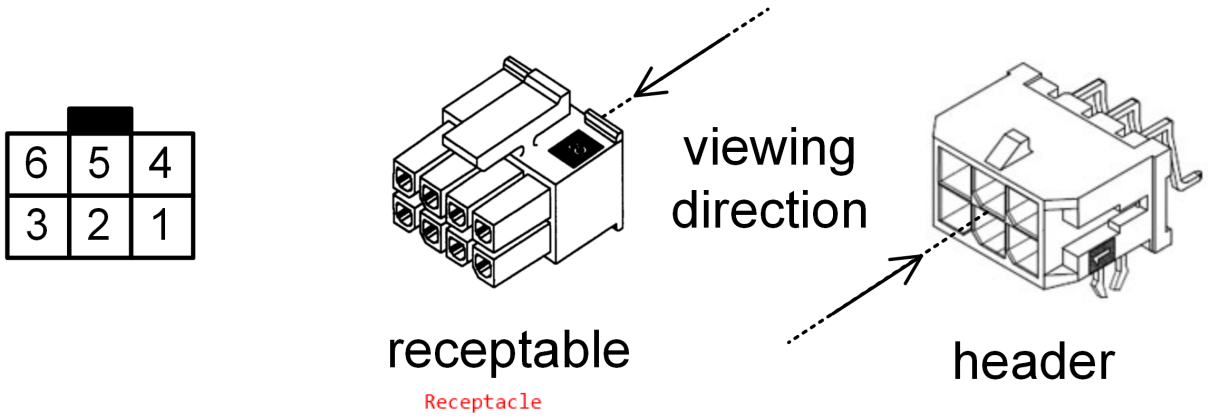



Fig. 12.1: Supply connector pin out, receptable - rear view, header - front view (image source: MOLEX)

There are two types of connectors:

- Header
  Receptacle
- Receptable, plugged into the header

The numbering shown on the left in fig. 12.1 is always valid when viewing in the direction indicated by the arrow with the indication `viewing direction`. This must be taken into account when crimping the receptables.

## 12.2 Global Description of the foxBMS Hardware

The foxBMS system can be mounted in a metal housing, shown in fig. 12.2.


Fig. 12.2: foxBMS Master Unit housing

In this configuration, the top plate can be removed to have access to the foxBMS electronic boards. This is done by unscrewing the four screws holding the top plate.

The open housing is shown in fig. 12.3.

The boards can be removed from the housing. The boards without housing are shown in fig. 12.4. To start, it is not necessary to remove the boards from the housing, but it is helpful to be able to look at the LEDs located on the BMS-Master Board.

Fig. 12.5 shows how to put the boards back in the housing.

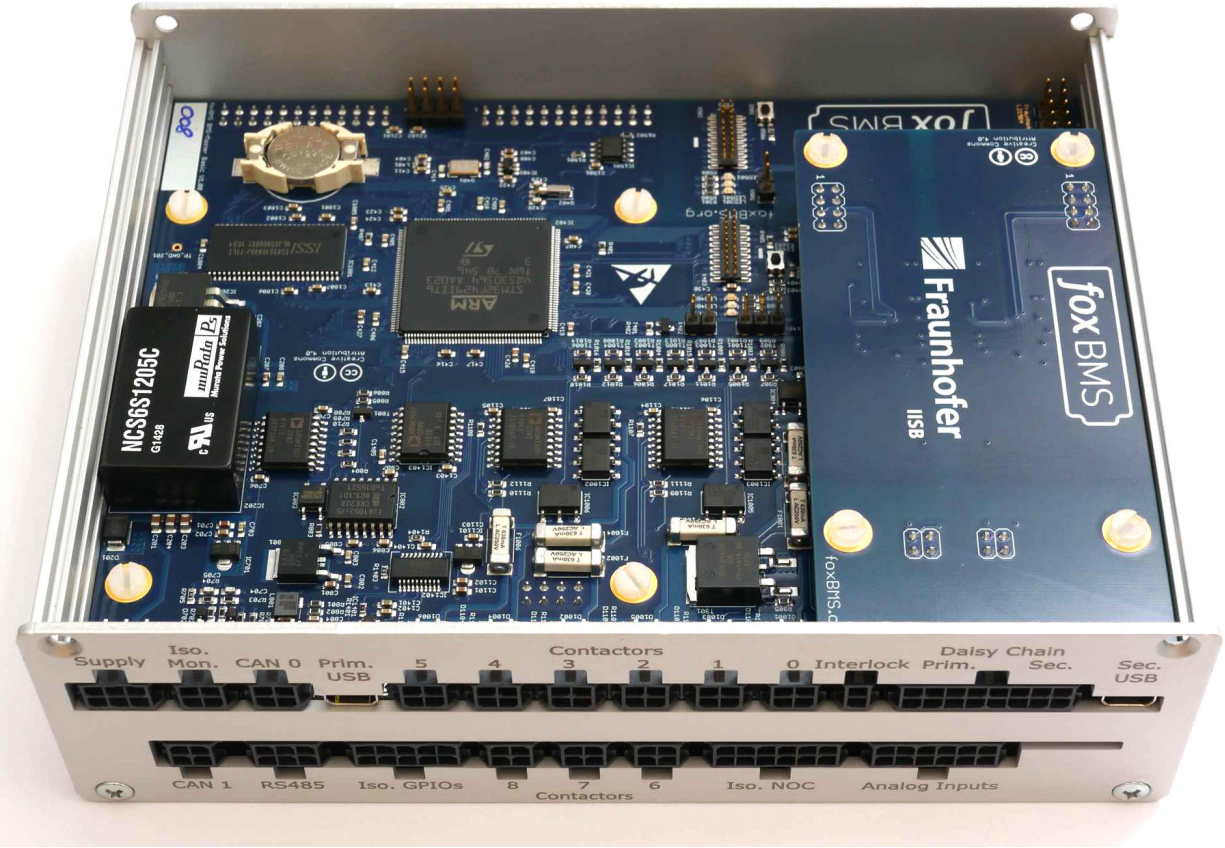



Fig. 12.3: foxBMS Master Unit housing, top plate removed

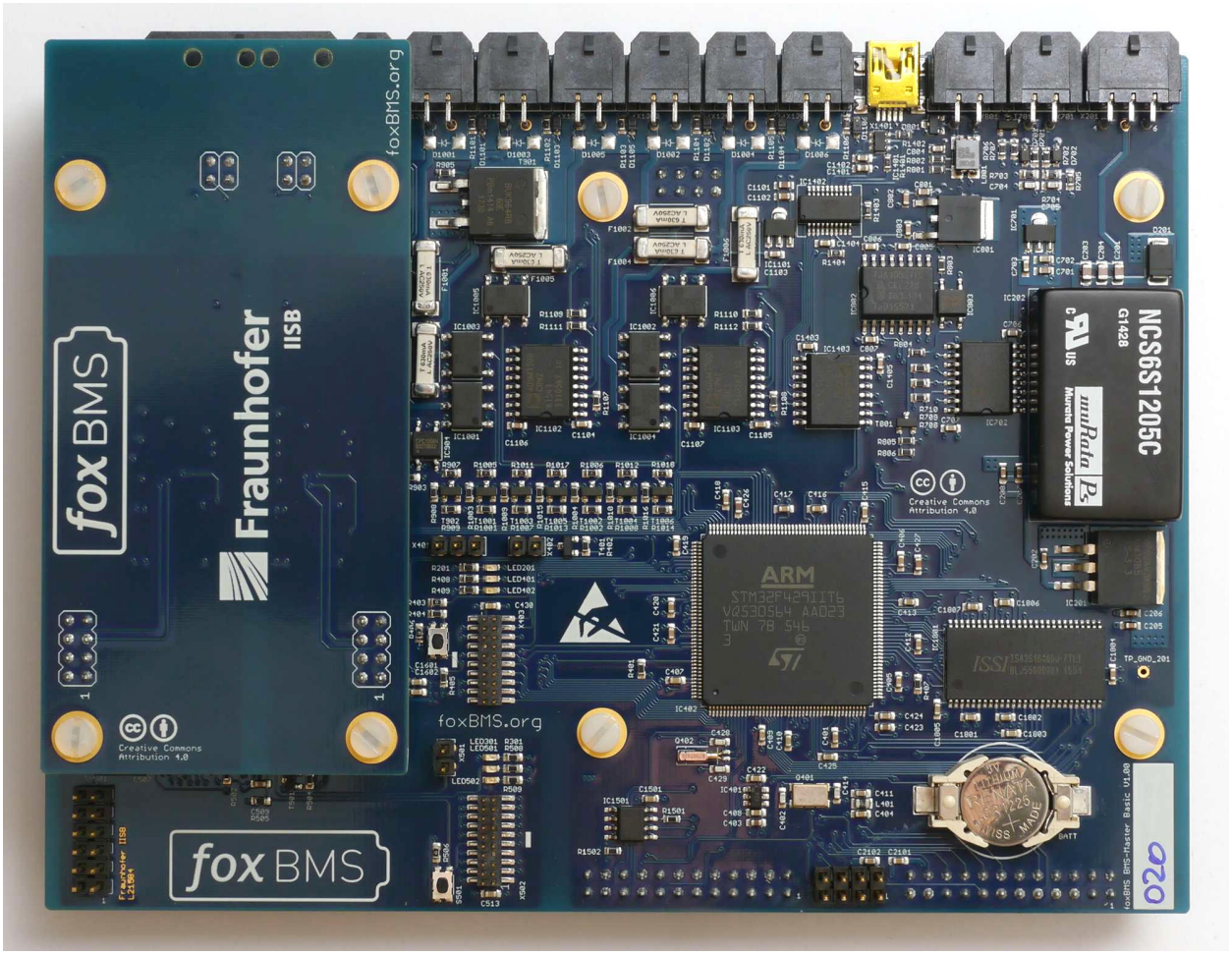



Fig. 12.4: foxBMS board stack removed from the housing (both BMS-Master Board and BMS-Interface Board are shown)

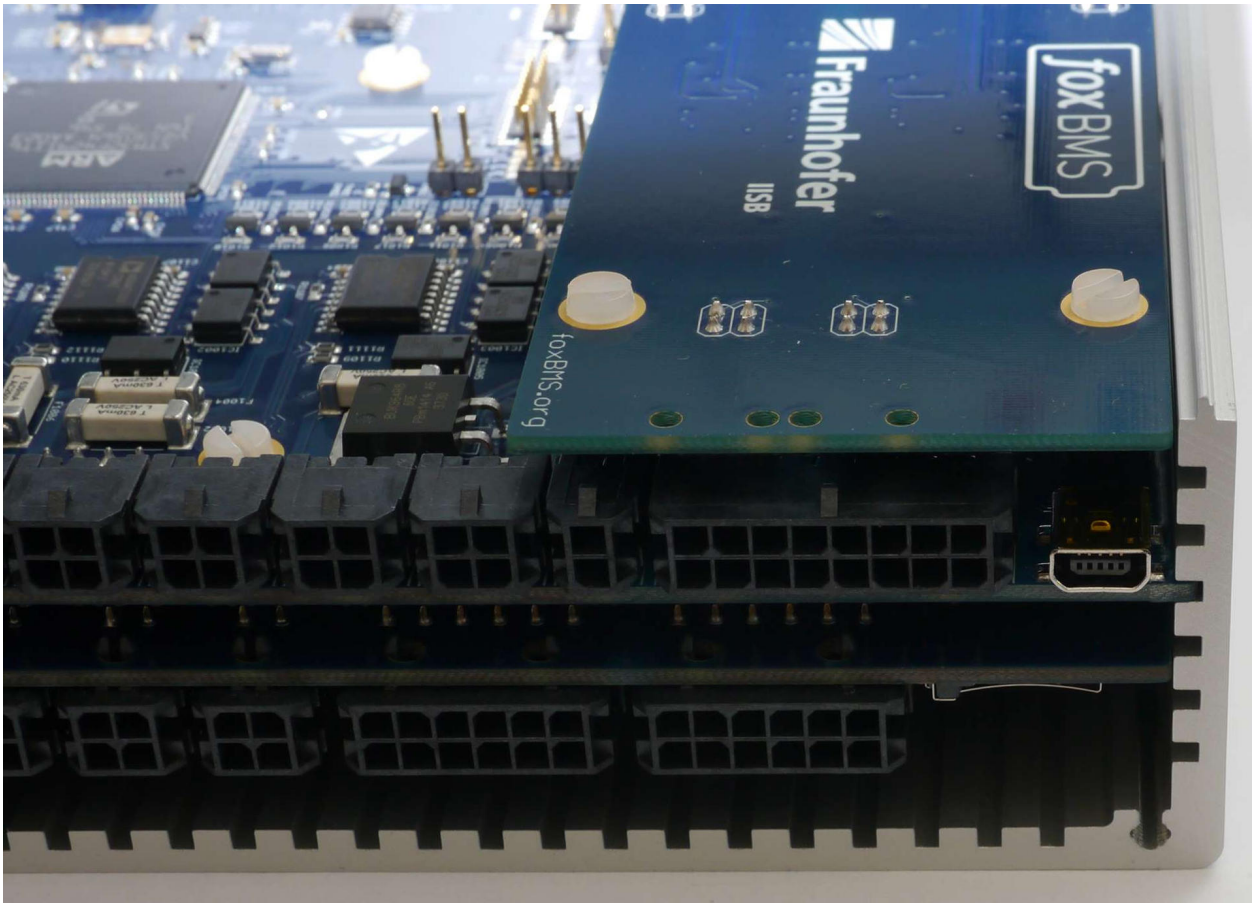



Fig. 12.5: Introduction of the board stack in the foxBMS Master Unit housing

## 12.3 Detailed Description of the Hardware Parts

The heart of foxBMS is the BMS-Master Board, shown in fig. 12.6.

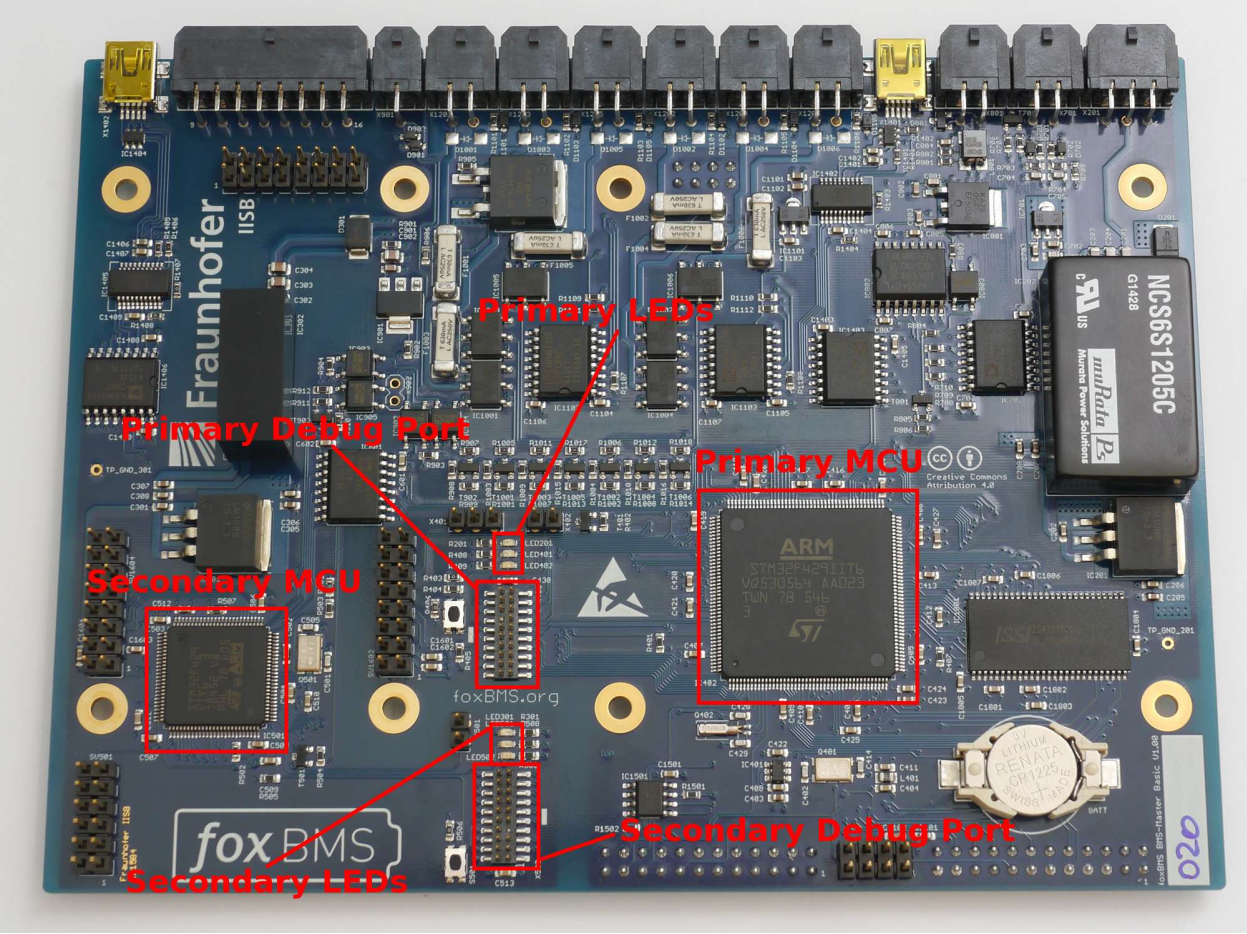



Fig. 12.6: foxBMS BMS-Master Board

As shown in fig. 12.6, the BMS-Master Board has two microcontroller units (MCU):

- Primary (also called MCU0)
- Secondary (also called MCU1)

Primary is the MCU where the foxBMS software is run. The secondary MCU is present for redundant safety when developing software code on the primary MCU.

Each MCU has a set of LEDs, as shown in fig. 12.6:

- Power LED
- Red indicator LED
- Green indicator LED

The power LED must lit when power is supplied to the BMS-Master Board, and the indicator LEDs should blink, except during flashing of software on the MCU.

If a debugger is used, it must be connected to the debug port (i.e., JTAG-interface) corresponding to the MCU being used.

An extension board named BMS-Extension Board is present under the BMS-Master Board and is shown in fig. 12.7.

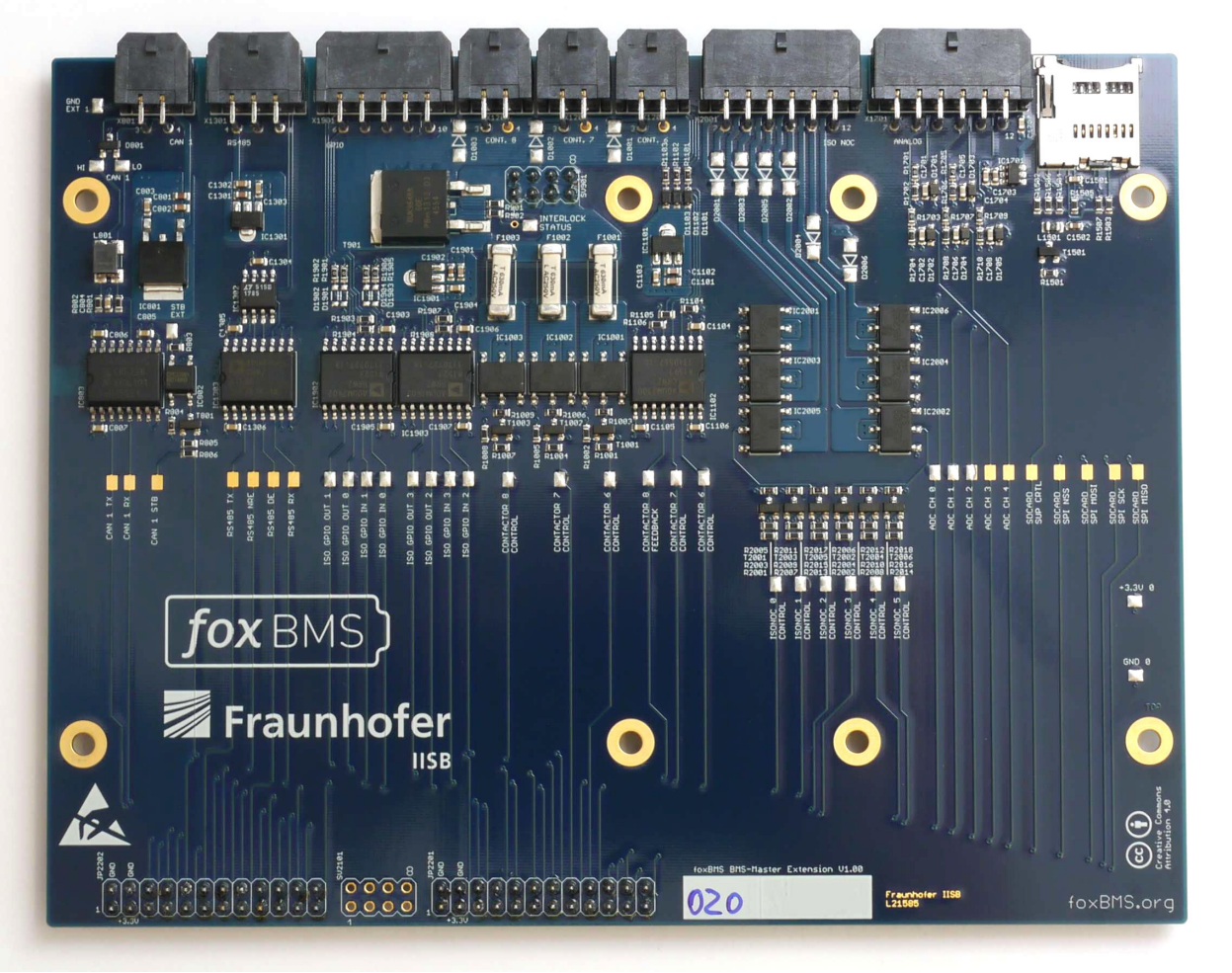



Fig. 12.7: foxBMS BMS-Extension Board

It is used to provide more I/O and interfaces than with the BMS-Master Board alone.

The BMS-Interface Board is located on top of the BMS-Master Board (shown in fig. 12.8).

Its purpose is to convert the signals sent by the Serial Peripheral Interface (SPI) of the BMS-Master Board to the first BMS-Slave Board in the daisy by using a proprietary isoSPI interface from Linear Technology.

An example of a BMS-Slave Board is shown in fig. 12.9.

The BMS-Slave Board is based on the LTC6811-1 battery cell monitoring chip. More information on the LTC6811-1 integrated circuit can be found in the datasheet ([ltc_datasheet6804] and [ltc_datasheet6811]). It supervises up to 12 battery cells connected in series. It performs voltage measurements, temperature measurements and passive cell balancing. In the daisy chain, the BMS-Slave Boards are connected via differential pair cables.

The BMS-Slave Board is not designed to be used in a specific housing.

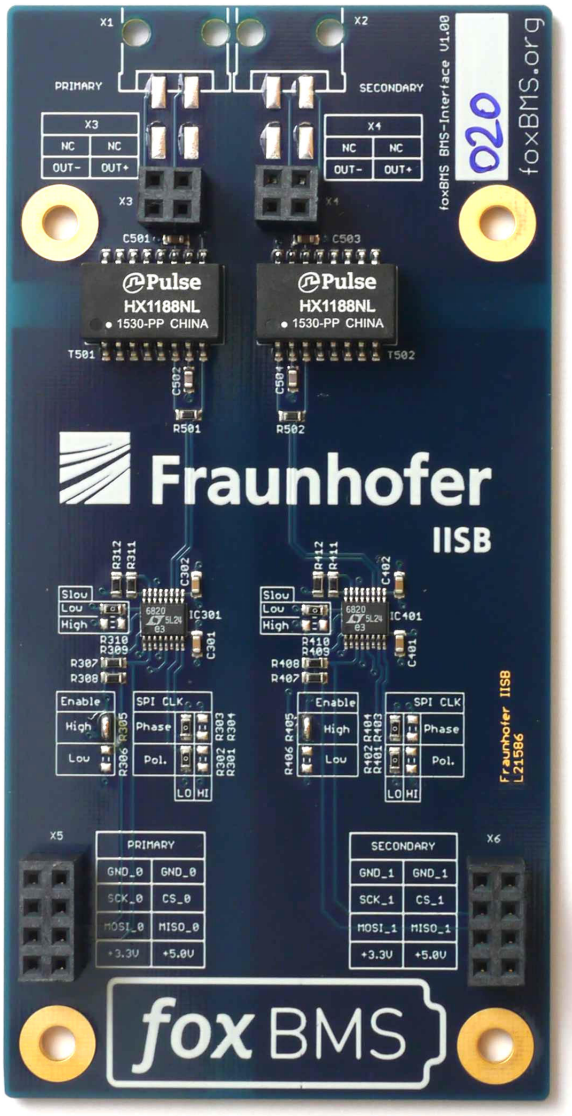



Fig. 12.8: foxBMS BMS-Interface Board

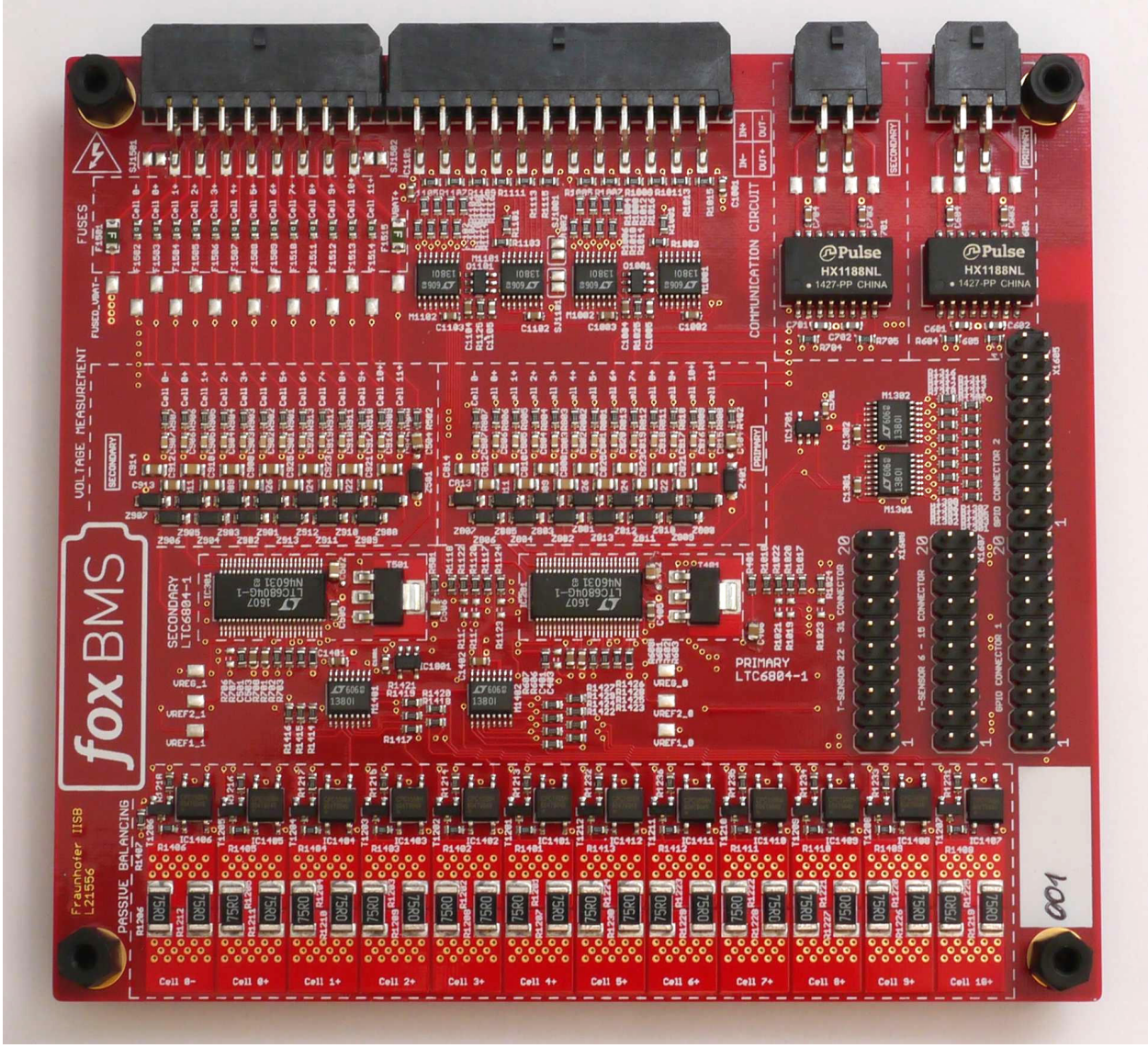



Fig. 12.9: foxBMS BMS-Slave Board

This is a very early version. The 5 V source is the simple transistor type, not the new DC/DC converter type.

## 12.4 Use of foxBMS in a Battery System

Fig. 12.10 presents the organization of the hardware. The system consists of $n$ battery modules and $m$ BMS-Slave Boards. Each BMS-Slave Board is connected to a battery module, where it measures cell voltages and cell temperatures. The BMS-Slave Boards are connected in a daisy chain configuration: when a data package is sent to the daisy chain, it is first received by slave 1, which transmits it to slave 2 and so on until the data package is received by the last BMS-Slave Board.

The BMS-Interface Board converts the messages sent by the BMS-Master Board so that they can be transmitted to the daisy chain and vice versa.

The foxBMS Master Unit communicates with the current sensor (for instance, Isabellenhuette IVT-MOD or IVT-S) via CAN.

Communication with the control unit (for instance, a personal computer), is also made via CAN.

## 12.5 Hardware Setup of the BMS-Master Board and the BMS-Slave Board

Fig. 12.11 presents all the connectors of the BMS-Master Board.

The connectors needed for this quickstart guide are indicated in the following parts.

### 12.5.1 Connecting the BMS-Slave Board to the BMS-Master Board

**Note:** When the connection is made between the BMS-Master Board and the BMS-Slave Board, two communication lines have to be connected: the primary and the secondary daisy chain.

The pin assignments and cabling instructions for all different BMS-Slave Boards are documented in section *Slaves*.

### 12.5.2 CAN Connector

The connector indicated as `CAN 0` in fig. 12.11 must be used on the BMS-Master Board. Its layout is described in table 12.1.

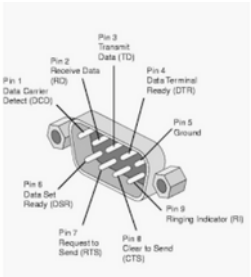
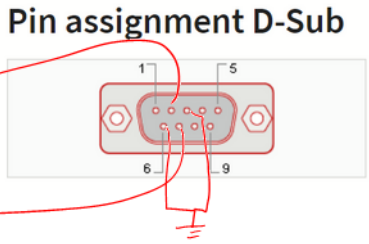
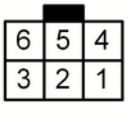


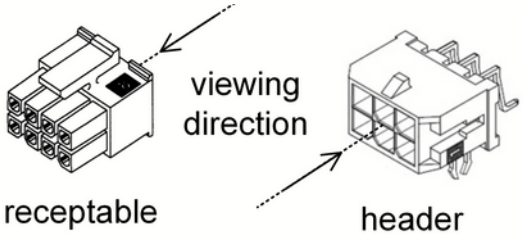
Table 12.1: Master CAN connection

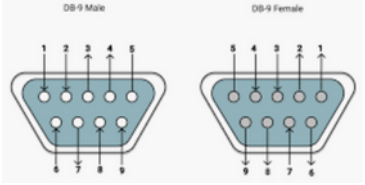
| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT0 | Output | |
| 2 | NC | – | |
| 3 | CAN0_L | Input/Output | |
| 4 | CAN0_H | Input/Output | |



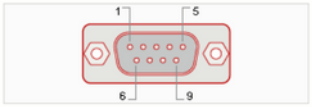

Ground of CAN0 is shared with supply ground `GND_EXT0`. CAN0 is isolated from the MCU0 via the isolated CAN transceiver TJA1052. The CAN transceiver may be put into standby mode by MCU0.



Pin assignment D-Sub




| Pin | Pin assignment |
|---|---|
| 1 | Not connected / optional +5V |
| 2 | CAN-L |
| 3 | GND |
| 4 | Not connected |
| 5 | Not connected |
| 6 | GND |
| 7 | CAN-H |
| 8 | Not connected |
| 9 | Not connected / optional +5V |

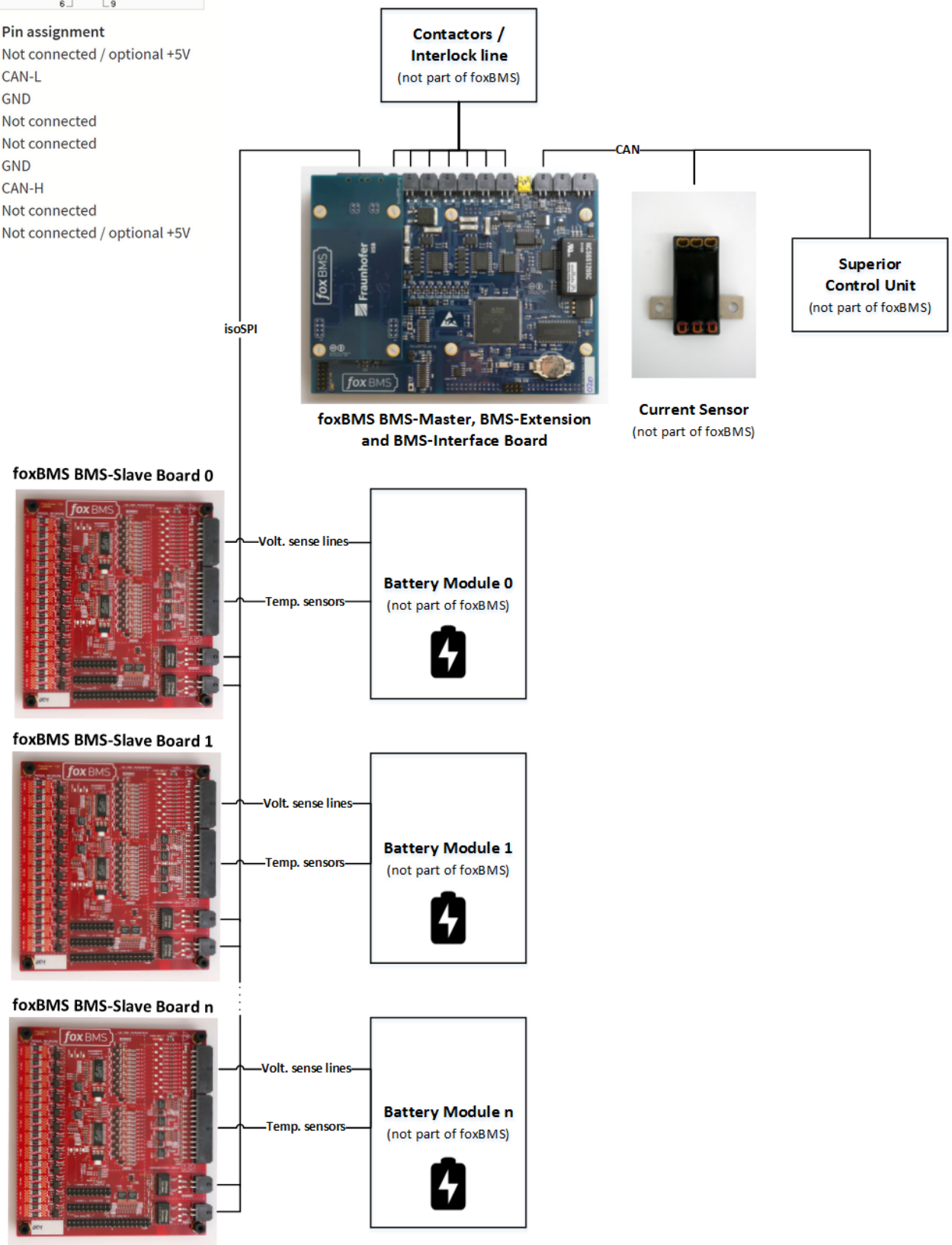



Fig. 12.10: foxBMS in the battery system

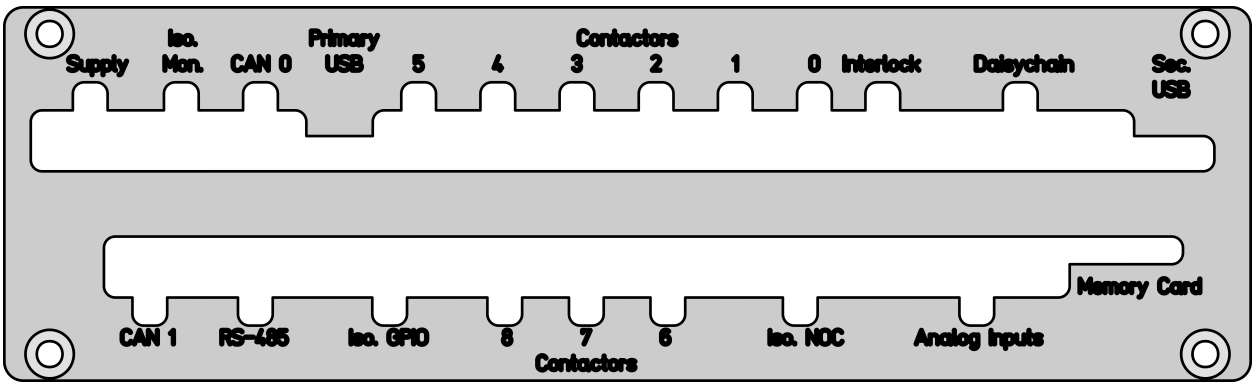
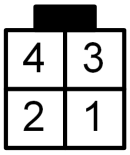



Fig. 12.11: Front view of the foxBMS Master, indicating the location of each header




## 12.6 Communication with the BMS-Master Board

Once foxBMS is running, CAN messages should be sent.

---

**Note: A periodic state request must be made to the system by sending a message with ID 0x152 on the ~~bus~~ bus CAN periodically.** If not, the system will go into an error state. The period must be 100ms. More information on state requests can be found in ~~the section~~ Section 14.1 *Communicating over CAN*. Typically the request *Standby state* can be made.

---

If voltages have been applied to the voltage connector, they are sent with the message IDs 0x550, 0x551, 0x552 and 0x553. Details can be found in ~~the section~~ Section 14.1 *Communicating over CAN*.

If temperature sensors have been connected and the conversion function changed in the software, temperature are sent with the messages with IDs 0x353 and 0x354. Details can be found in the section *Communicating over CAN*.

## 12.7 Current Sensor Configuration

Further information on the current sensor tested with foxBMS can be found in the datasheet of the current sensor. By default, foxBMS uses the factory defaults of the current sensor, which works in cyclic (non-triggered) mode. Another possibility tested with foxBMS is to use the current sensor in triggered mode. For this, the current sensor IVT-MOD from Isabellenhuette has to be reprogrammed. The changes compared to factory default are then:

- The CAN Message IDs was changed
- The triggered measurement mode was activated

IVT series current/voltage sensor: https://www.isabellenhuette.de/en/precision-measurement/standard-products/ivt-series/

The ~~two~~ following parts sum up the differences between factory setup and triggered setup tested with foxBMS.

### 12.7.1 Factory Default

- CAN IDs

  - 0x411 (dec: 1041) Command
  - 0x521 (dec: 1313) Current Measurement
  - 0x522 (dec: 1314) Voltage Measurement 1
  - 0x523 (dec: 1315) Voltage Measurement 2
  - 0x524 (dec: 1316) Voltage Measurement 3
  - 0x511 (dec: 1297) Response
- Measurement Mode: Cyclic (Disabled and Triggered are other possible modes)

### 12.7.2 After Configuration

- CAN IDs
  - 0x35B (dec: 859) Command
  - 0x35C (dec: 860) Current Measurement
  - 0x35D (dec: 861) Voltage Measurement 1
  - 0x35E (dec: 862) Voltage Measurement 2
  - 0x35F (dec: 863) Voltage Measurement 3
  - 0x7FF (dec: 2047) Response
- Measurement Mode: Triggered

## 12.8 Hardware Related Frequently Asked Questions

### 12.8.1 Are the schematic and layout source files available?

Yes, the complete foxBMS schematic and layout files are available in EAGLE format on GitHub. Refer to the HTML documentation section *Design Resources*.

### 12.8.2 Have both MCU0 and MCU1 to be connected?

Yes, they should always be connected and configured to provide redundant monitoring and improved safety during software development in prototyping applications.

# CHAPTER 13

# Connectors Pinout

This section describes the pinout of all connectors on the foxBMS Master Unit.

## 13.1 Overview of the Connectors on the foxBMS Master Unit

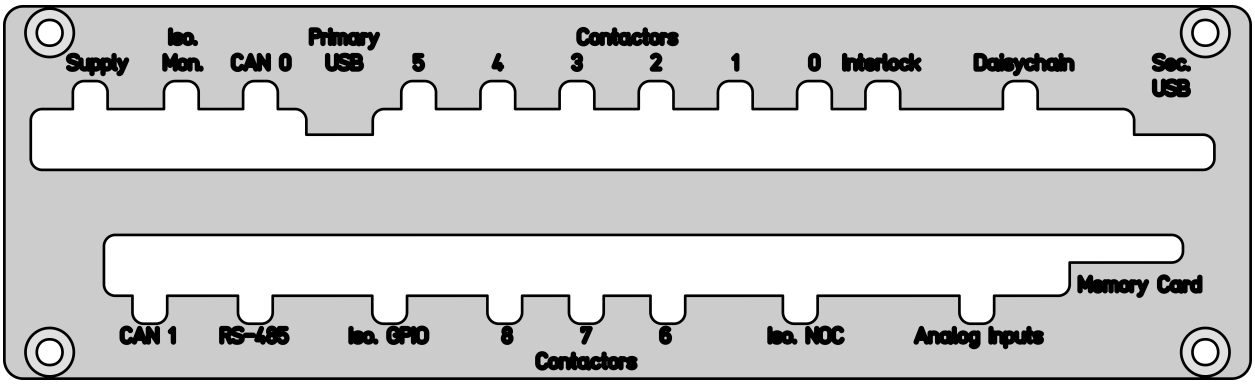


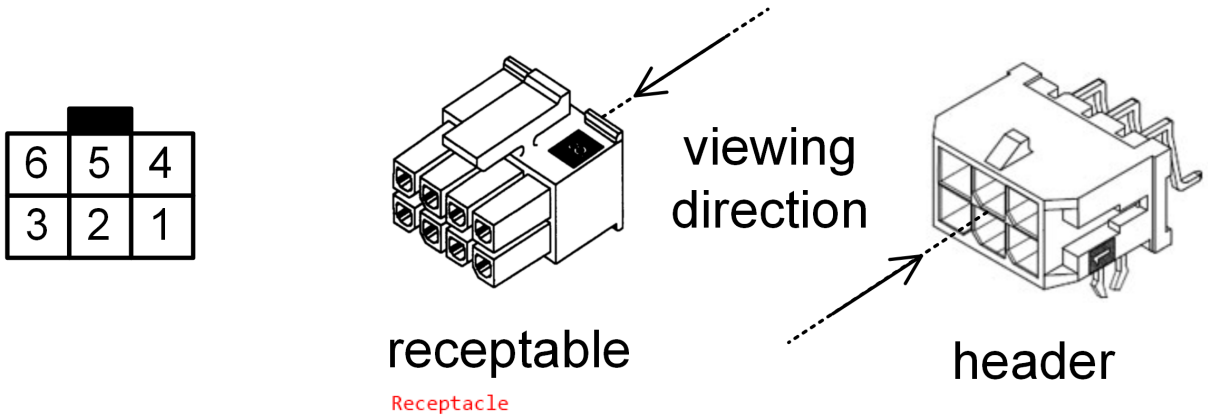

Fig. 13.1: Front view of the foxBMS Master Unit indicating the location of each pin header

foxBMS uses only Molex Micro-Fit 3.0 type connectors, except for USB. A comprehensive set of connectors and crimps is supplied with foxBMS to start connecting immediately. In case crimps or housings are missing, they are commonly available at major distributors.

Molex Micro-Fit 3.0 Crimps Part Number: 46235-0001 (Farnell 2284551)

Molex Micro-Fit 3.0 Connectors:

| Pin Count | Molex Housing Part Number | Farnell Order Number |
|---|---|---|
| 2 | 43025-0200 | 672889 |
| 4 | 43025-0400 | 672890 |
| 6 | 43025-0600 | 672907 |
| 10 | 43025-1000 | 672920 |
| 12 | 43025-1200 | 629285 |
| 16 | 43025-1600 | 9961321 |



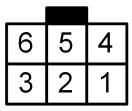

Receptacle

Fig. 13.2: Defined viewing direction for the connector pin out; receptable - rear view; header - front view (image source: MOLEX)

receptacle

Note that the direction below is always with respect to the board of interest.

## 13.2 Supply (X201 on BMS-Master Board)




Slave board

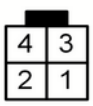
| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | SUPPLY_EXT2 | Input | 12 - 24V |
| 2 | SUPPLY_EXT2 | Input | 12 - 24V |
| 3 | GND_EXT2 | Input | GND |
| 4 | SUPPLY_EXT0 | Input | 12 - 24V |
| 5 | GND_EXT0 | Input | GND |
| 6 | GND_EXT2 | Input | GND |


Table 18.9: External DC supply connector

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | DC+ | Input | positive supply terminal |
| 2 | DC- | Input | negative supply terminal |

- `SUPPLY_EXT0` / `GND_EXT0`: Microcontroller supply and insulation monitoring devices supply
- `SUPPLY_EXT2` / `GND_EXT2`: Contactor supply and interlock supply

## 13.3 CAN0 (X801 on BMS-Master Board)




See pages 98 and 99.

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT0 | Output | |
| 2 | NC | – | |
| 3 | CAN0_L | Input/Output | |
| 4 | CAN0_H | Input/Output | |

Ground of CAN0 is shared with supply ground `GND_EXT0`. CAN0 is isolated from the MCU0 via the isolated CAN transceiver TJA1052. The CAN transceiver may be put into standby mode by MCU0.

## 13.4 CAN1 (X801 on BMS-Extension Board)

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT1 | Input | |
| 2 | SUPPLY_EXT1 | Input | |
| 3 | CAN1_L | Input/Output | |
| 4 | CAN1_H | Input/Output | |

CAN1 has to be supplied externally (`GND_EXT1` / `SUPPLY_EXT1`) with 12 - 24V. CAN1 is isolated from the MCU0 via the isolated CAN transceiver TJA1052. The CAN transceiver may be put into standby mode by MCU0.

## 13.5 Insulation Monitor (Bender ISOMETER) (X701 on BMS-Master Board)


| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | BENDER_NEGATIVE_SUPPLY | Output | Supply to insulation monitoring device |
| 2 | SUPPLY_EXT0 | Output | Supply to insulation monitoring device |
| 3 | BENDER_OK_EXT | Input | Status signal of insulation monitoring device |
| 4 | BENDER_PWM_EXT | Input | Insulation monitoring device diagnostic signal |

This interface is intended to be used with a Bender insulation monitoring device. Bender ISOMETER IR155-3203/-3204/-3210 are supported. The Bender ISOMETER is supplied and may be switched on or off (lowside) by the foxBMS Master Unit. By factory, the foxBMS Master Unit is configured to operate with the Bender ISOMETER IR155-3204/-3210. In order to operate with ~~the~~ Bender ISOMETER IR155-3203, Jumper R705 must be removed on BMS-Master Board. For more details, check the schematic of the BMS-Master Board in section *Design Resources*.

(by pin 1, the negative supply.)

default

## 13.6 Contactors (X1201 - X1206 on BMS-Master Board and X1201 - X1203 on BMS-Extension Board)

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT0 | – | Contactor auxiliary contact |
| 2 | CONTACTOR_X_FEEDBACK_EXT | – | Contactor auxiliary contact |
| 3 | CONTACTOR_X_COIL_POS | Output | Positive contactor coil supply |
| 4 | CONTACTORS_COMMON_NEG | Output | Negative contactor coil supply |

All contactor connectors share one common ground. This common ground is switched lowside by the interlock circuit. Opening the interlock loop deactivates the contactor supply and opens all contactors. A contactor auxiliary contact may be read by connecting the auxiliary contact to the corresponding pins of the contactor connector.

Freewheeling diodes are not populated on the PCB, since some contactors like the Gigavac GX16 have built-in diodes and should not be used with additional external freewheeling diodes in parallel. If the used contactors do not have built-in freewheeling diodes, freewheeling diodes must be added to protect the contactor control circuitry. The load current is limited by the optically isolated power switch AQV25G2S (6A continuous load). Every contactor connector is fused with an onboard slow blow fuse type Schurter UMT-250 630mA (3403.0164.xx).

## 13.7 Interlock (X901 on BMS-Master Board)



| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | INTERLOCK_IN | Input | – |
| 2 | INTERLOCK_OUT | Output | – |

The interlock circuit has a built-in current source, adjusted to 10mA constant current. In fault conditions, all contactors are opened by opening the interlock circuit. If the interlock circuit is externally opened, the contactor supply is deactivated immediately. This circuit has no effect on the foxBMS supply or communication interfaces.

## 13.8 Daisy Chain - Primary and Secondary (X1601 on BMS-Master Board)

**Hint:** The pinout for the BMS-Interface Board hardware versions `1.1.0 and below` is shown in table 13.1. For newer versions `1.2.0 and above` see table 13.2.

Table 13.1: Daisy Chain Connector (Interface version `1.1.0 and below`)

| Pin | Signal |
|---|---|
| 1 | NC |
| 2 | OUT+ (Secondary BMS-Slave Board) |
| 3 | OUT- (Secondary BMS-Slave Board) |
| 4 | NC |
| 5 | NC |
| 6 | OUT+ (Primary BMS-Slave Board) |
| 7 | OUT- (Primary BMS-Slave Board) |
| 8 | NC |
| 9 | NC |
| 10 | NC |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | NC |
| 15 | NC |
| 16 | NC |

Table 13.2: Daisy Chain Connector (Interface version `1.2.0 and above`)

| Pin | Signal |
|---|---|
| 1 | OUT+ (Secondary BMS-Slave Board forward) |
| 2 | NC |
| 3 | OUT+ (Secondary BMS-Slave Board reverse) |
| 4 | NC |
| 5 | NC |
| 6 | OUT+ (Primary BMS-Slave Board forward) |
| 7 | NC |
| 8 | OUT+ (Primary BMS-Slave Board reverse) |
| 9 | OUT- (Secondary BMS-Slave Board forward) |
| 10 | NC |
| 11 | OUT- (Secondary BMS-Slave Board reverse) |
| 12 | NC |
| 13 | NC |
| 14 | OUT- (Primary BMS-Slave Board forward) |
| 15 | NC |
| 16 | OUT- (Primary BMS-Slave Board reverse) |


Table 18.5: isoSPI Daisy Chain Input Connectors

| Connector Pin | Daisy Chain |
|---|---|
| 1 | IN+ (Primary/Secondary LTC6811-1) |
| 2 | IN- (Primary/Secondary LTC6811-1) |

Table 18.6: isoSPI Daisy Chain Output Connectors

| Connector Pin | Daisy Chain |
|---|---|
| 1 | OUT+ (Primary/Secondary LTC6811-1) |
| 2 | OUT- (Primary/Secondary LTC6811-1) |

Please note that this connector pin out is only valid when the BMS-Interface Board uses the LTC6820 interfacing IC.

## 13.9 RS485 (X1301 on BMS-Extension Board)

Will not be used.

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT2 | Output | |
| 2 | RS485_A | Input/Output | |
| 3 | RS485_B | Input/Output | |
| 4 | SUPPLY_EXT2 | Output | |
| 5 | SUPPLY_EXT2 | Input | |
| 6 | GND_EXT2 | Input | |

The RS485 interface uses the ESD rugged transceiver LT1785. Moreover, the interface is galvanically isolated. An external supply has to be provided (12 - 24V).

## 13.10 Isolated GPIO (X1901 on BMS-Extension Board)

Will not be used.

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | ISOGPIO_IN0 | Input | |
| 2 | ISOGPIO_IN1 | Input | |
| 3 | ISOGPIO_IN2 | Input | |
| 4 | ISOGPIO_IN3 | Input | |
| 5 | ISOGPIO_OUT0 | Output | |
| 6 | ISOGPIO_OUT1 | Output | |
| 7 | ISOGPIO_OUT2 | Output | |
| 8 | ISOGPIO_OUT3 | Output | |
| 9 | GND_EXT0 | Output | |
| 10 | GND_EXT0 | Output | |

The BMS-Extension Board provides 4 isolated general purpose inputs and 4 isolated general purpose outputs. The GPIOs are isolated by an ADUM3402 (i.e., ESD rugged version of the ADUM1402). The inputs are equipped with 10kOhm pull down resistors and are intended for a maximum input voltage of 5V. The output voltage is also 5V. An external supply is not needed.

## 13.11 Isolated Normally Open Contacts - isoNOC (X2001 on BMS-Extension Board)

Will not be used.

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | ISONOC0_POSITIVE | | |
| 2 | ISONOC1_POSITIVE | | |
| 3 | ISONOC2_POSITIVE | | |
| 4 | ISONOC3_POSITIVE | | |
| 5 | ISONOC4_POSITIVE | | |
| 6 | ISONOC5_POSITIVE | | |
| 7 | ISONOC0_NEGATIVE | | |
| 8 | ISONOC1_NEGATIVE | | |
| 9 | ISONOC2_NEGATIVE | | |
| 10 | ISONOC3_NEGATIVE | | |
| 11 | ISONOC4_NEGATIVE | | |
| 12 | ISONOC5_NEGATIVE | | |

The BMS-Extension Board features 6 isolated normally open contacts. The load current of each channel is limited by the optically isolated power switch AQV25G2S. The channels are not fused, however freewheeling diodes type GF1B are installed on board.

## 13.12 Analog Inputs (X1701 on BMS-Master Board)

Will not be used.

| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | V_REF | Output | |
| 2 | ANALOG_IN_CH0 | Input | |
| 3 | V_REF | Output | |
| 4 | ANALOG_IN_CH1 | Input | |
| 5 | V_REF | Output | |
| 6 | ANALOG_IN_CH2 | Input | |
| 7 | V_REF | Output | |
| 8 | ANALOG_IN_CH3 | Input | |
| 9 | V_REF | Output | |
| 10 | ANALOG_IN_CH4 | Input | |
| 11 | GND0 | Output | |
| 12 | GND0 | Output | |

On the BMS-Extension Board 4 nonisolated analog inputs to MCU0 are available. For applications using NTCs as temperature sensors, also a reference voltage of 2.5V is provided. The maximum input voltage is limited to 3.3V and is Zener protected. For further information on the input circuit, please refer to the foxBMS *Master-Unit* and to the foxBMS *Design Resources*.

# CHAPTER 14

# Communicating with foxBMS

(This is the only section of this chapter.)

Mesage vs signal:
* A message is a formated data sequence over CAN bus.
* A signal is the data or interpretation of the data sent in the CAN message.

## 14.1 Communicating over CAN

Table 14.1, CAN Messareg,

The table *CAN Messages* shows the CAN messages sent and received by the BMS with their respective names and IDs, the direction and a short comment.

Table 14.2, CAN RX Signals,

The table *CAN RX Signals* gives the CAN signals sent and received in these messages.

To add or change a signal ~~it must be refered~~ please refer to the software FAQ: *How to manually add CAN entries (transmit and receive) and to change the transmit time period?*.

---

**Note:** 18 cell voltages, 12 cell temperatures and 8 modules are configured in the default CAN configuration of the foxBMS Software.

---

If less then 8 modules, 18 cell voltages or 12 temperature sensors are configured in the software (e.g., in `batterysystem_cfg.h`), default values are transmitted on these messages/signals. These values from the unused CAN messages must simply be ignored.

In case it is really wanted to not transmit these values, the configuration of the CAN module must be changed as shown in the software FAQ in *How to adapt the CAN module when less or more than 8 battery modules are used?* and in *How to adapt the CAN module when less than 18 battery cells or 12 temperature sensors per module are used?*. The configuration must also be changed if more than 8 modules, 18 cell voltages or 12 temperature sensors are used.

### 14.1.1 CAN Messages

Message names in the DBC file can vary from the following tables in order to be compliant with the 32 character limit of DBC files.

DBC = DataBase Container

**RX Messages:**

Table 14.1.

DLC = data length code, the length of the CAN message in bytes.

| Message Name | Message ID | DLC | Direction | Description |
|---|---|---|---|---|
| CAN0_Software_Reset | 0x95 0x095 | 8 | RX | Perform software reset |
| CAN0_Debug_Message | 0x100 | 8 | RX | Debug message |
| CAN0_State_Request | 0x120 | 8 | RX | State request |
| CAN0_IVT_Current | 0x35C/0x521 | 6 | RX | Current Sensing Current Value |
| CAN0_IVT_Voltage_1 | 0x35D/0x522 | 6 | RX | Current Sensing Voltage 1 |
| CAN0_IVT_Voltage_2 | 0x35E/0x523 | 6 | RX | Current Sensing Voltage 2 |
| CAN0_IVT_Voltage_3 | 0x35F/0x524 | 6 | RX | Current Sensing Voltage 3 |
| CAN0_IVT_Temperature | 0x525 | 6 | RX | Current Sensing Temperature |
| CAN0_IVT_Power | 0x526 | 6 | RX | Current Sensing Power |
| CAN0_IVT_CoulombCount | 0x527 | 6 | RX | Current Sensing Coulomb Counting Value |
| CAN0_IVT_EnergyCount | 0x528 | 6 | RX | Current Sensing Energy Counting Value |
| CAN0_GetReleaseVersion | 0x777 | 8 | RX | Request software version |

See P. 101.

0x521 to 0x524 are used by default.

**TX Messages:**

Table 14.1.

| Message Name | Message ID | DLC | Direction | Description |
|---|---|---|---|---|
| CAN0_MSG_Boot | 0x101 | 8 | TX | BMS booting successful |
| CAN0_MSG_SystemState_0 | 0x110 | 8 | TX | BMS system state 0 |
| CAN0_MSG_SystemState_1 | 0x111 | 8 | TX | BMS system state 1 |
| CAN0_MSG_SystemState_2 | 0x112 | 8 | TX | BMS system state 2 |
| CAN0_MSG_SlaveState_0 | 0x115 | 8 | TX | BMS slave state 0 |
| CAN0_MSG_SlaveState_1 | 0x116 | 8 | TX | BMS slave state 1 |
| CAN0_RecOperatingCurrent | 0x130 | 8 | TX | Recommended operating current |
| CAN0_SOP | 0x131 | 8 | TX | State-of-power |
| CAN0_SOC | 0x140 | 8 | TX | State-of-charge |
| CAN0_SOH | 0x150 | 8 | TX | State-of-health |
| CAN0_SOE | 0x160 | 8 | TX | State-of-energy |
| CAN0_MinMaxCellVoltages | 0x170 | 8 | TX | Cell voltages Max Min Mean |
| CAN0_SOV | 0x171 | 8 | TX | State-of-voltage |
| CAN0_MinMaxCellTemperatures | 0x180 | 8 | TX | Cell temperatures Max Min Mean |
| CAN0_Tempering | 0x190 | 8 | TX | Cooling/Heating |
| CAN0_Insulation | 0x1A0 | 8 | TX | Insulation |
| CAN0_MovAveragePower_0 | 0x1D0 | 8 | TX | Moving average power 0 |
| CAN0_MovAveragePower_1 | 0x1D1 | 8 | TX | Moving average power 1 |
| CAN0_MovAveragePower_2 | 0x1D2 | 8 | TX | Moving average power 2 |
| CAN0_MovAverageCurrent_0 | 0x1E0 | 8 | TX | Moving average current 0 |
| CAN0_MovAverageCurrent_1 | 0x1E1 | 8 | TX | Moving average current 1 |
| CAN0_MovAverageCurrent_2 | 0x1E2 | 8 | TX | Moving average current 2 |
| CAN0_PackVoltage | 0x1F0 | 8 | TX | Battery pack voltage |
| CAN0_Cell_voltage_M0_0 | 0x200 | 8 | TX | Cell voltages module 0 cell 0 1 2 |
| CAN0_Cell_voltage_M0_1 | 0x201 | 8 | TX | Cell voltages module 0 cell 3 4 5 |
| CAN0_Cell_voltage_M0_2 | 0x202 | 8 | TX | Cell voltages module 0 cell 6 7 8 |
| CAN0_Cell_voltage_M0_3 | 0x203 | 8 | TX | Cell voltages module 0 cell 9 10 11 |
| CAN0_Cell_voltage_M0_4 | 0x204 | 8 | TX | Cell voltages module 0 cell 12 13 14 |
| CAN0_Cell_voltage_M0_5 | 0x205 | 8 | TX | Cell voltages module 0 cell 15 16 17 |
| CAN0_Cell_temperature_M0_0 | 0x210 | 8 | TX | Cell temperatures module 0 - 0 1 2 |
| CAN0_Cell_temperature_M0_1 | 0x211 | 8 | TX | Cell temperatures module 0 - 3 4 5 |
| CAN0_Cell_temperature_M0_2 | 0x212 | 8 | TX | Cell temperatures module 0 - 6 7 8 |
| CAN0_Cell_temperature_M0_3 | 0x213 | 8 | TX | Cell temperatures module 0 - 9 10 11 |

Continued on next page

Table 14.1 – continued from previous page

| | | | | |
|---|---|---|---|---|
| CAN0_Cell_voltage_M1_0 | 0x220 | 8 | TX | Cell voltages module 1 cell 0 1 2 |
| CAN0_Cell_voltage_M1_1 | 0x221 | 8 | TX | Cell voltages module 1 cell 3 4 5 |
| CAN0_Cell_voltage_M1_2 | 0x222 | 8 | TX | Cell voltages module 1 cell 6 7 8 |
| CAN0_Cell_voltage_M1_3 | 0x223 | 8 | TX | Cell voltages module 1 cell 9 10 11 |
| CAN0_Cell_voltage_M1_4 | 0x224 | 8 | TX | Cell voltages module 1 cell 12 13 14 |
| CAN0_Cell_voltage_M1_5 | 0x225 | 8 | TX | Cell voltages module 1 cell 15 16 17 |
| CAN0_Cell_temperature_M1_0 | 0x230 | 8 | TX | Cell temperatures module 1 - 0 1 2 |
| CAN0_Cell_temperature_M1_1 | 0x231 | 8 | TX | Cell temperatures module 1 - 3 4 5 |
| CAN0_Cell_temperature_M1_2 | 0x232 | 8 | TX | Cell temperatures module 1 - 6 7 8 |
| CAN0_Cell_temperature_M1_3 | 0x233 | 8 | TX | Cell temperatures module 1 - 9 10 11 |
| CAN0_Cell_voltage_M2_0 | 0x240 | 8 | TX | Cell voltages module 2 cell 0 1 2 |
| CAN0_Cell_voltage_M2_1 | 0x241 | 8 | TX | Cell voltages module 2 cell 3 4 5 |
| CAN0_Cell_voltage_M2_2 | 0x242 | 8 | TX | Cell voltages module 2 cell 6 7 8 |
| CAN0_Cell_voltage_M2_3 | 0x243 | 8 | TX | Cell voltages module 2 cell 9 10 11 |
| CAN0_Cell_voltage_M2_4 | 0x244 | 8 | TX | Cell voltages module 2 cell 12 13 14 |
| CAN0_Cell_voltage_M2_5 | 0x245 | 8 | TX | Cell voltages module 2 cell 15 16 17 |
| CAN0_Cell_temperature_M2_0 | 0x250 | 8 | TX | Cell temperatures module 2 - 0 1 2 |
| CAN0_Cell_temperature_M2_1 | 0x251 | 8 | TX | Cell temperatures module 2 - 3 4 5 |
| CAN0_Cell_temperature_M2_2 | 0x252 | 8 | TX | Cell temperatures module 2 - 6 7 8 |
| CAN0_Cell_temperature_M2_3 | 0x253 | 8 | TX | Cell temperatures module 2 - 9 10 11 |
| CAN0_Cell_voltage_M3_0 | 0x260 | 8 | TX | Cell voltages module 3 cell 0 1 2 |
| CAN0_Cell_voltage_M3_1 | 0x261 | 8 | TX | Cell voltages module 3 cell 3 4 5 |
| CAN0_Cell_voltage_M3_2 | 0x262 | 8 | TX | Cell voltages module 3 cell 6 7 8 |
| CAN0_Cell_voltage_M3_3 | 0x263 | 8 | TX | Cell voltages module 3 cell 9 10 11 |
| CAN0_Cell_voltage_M3_4 | 0x264 | 8 | TX | Cell voltages module 3 cell 12 13 14 |
| CAN0_Cell_voltage_M3_5 | 0x265 | 8 | TX | Cell voltages module 3 cell 15 16 17 |
| CAN0_Cell_temperature_M3_0 | 0x270 | 8 | TX | Cell temperatures module 3 - 0 1 2 |
| CAN0_Cell_temperature_M3_1 | 0x271 | 8 | TX | Cell temperatures module 3 - 3 4 5 |
| CAN0_Cell_temperature_M3_2 | 0x272 | 8 | TX | Cell temperatures module 3 - 6 7 8 |
| CAN0_Cell_temperature_M3_3 | 0x273 | 8 | TX | Cell temperatures module 3 - 9 10 11 |
| CAN0_Cell_voltage_M4_0 | 0x280 | 8 | TX | Cell voltages module 4 cell 0 1 2 |
| CAN0_Cell_voltage_M4_1 | 0x281 | 8 | TX | Cell voltages module 4 cell 3 4 5 |
| CAN0_Cell_voltage_M4_2 | 0x282 | 8 | TX | Cell voltages module 4 cell 6 7 8 |
| CAN0_Cell_voltage_M4_3 | 0x283 | 8 | TX | Cell voltages module 4 cell 9 10 11 |
| CAN0_Cell_voltage_M4_4 | 0x284 | 8 | TX | Cell voltages module 4 cell 12 13 14 |
| CAN0_Cell_voltage_M4_5 | 0x285 | 8 | TX | Cell voltages module 4 cell 15 16 17 |
| CAN0_Cell_temperature_M4_0 | 0x290 | 8 | TX | Cell temperatures module 4 - 0 1 2 |
| CAN0_Cell_temperature_M4_1 | 0x291 | 8 | TX | Cell temperatures module 4 - 3 4 5 |
| CAN0_Cell_temperature_M4_2 | 0x292 | 8 | TX | Cell temperatures module 4 - 6 7 8 |
| CAN0_Cell_temperature_M4_3 | 0x293 | 8 | TX | Cell temperatures module 4 - 9 10 11 |
| CAN0_Cell_voltage_M5_0 | 0x2A0 | 8 | TX | Cell voltages module 5 cell 0 1 2 |
| CAN0_Cell_voltage_M5_1 | 0x2A1 | 8 | TX | Cell voltages module 5 cell 3 4 5 |
| CAN0_Cell_voltage_M5_2 | 0x2A2 | 8 | TX | Cell voltages module 5 cell 6 7 8 |
| CAN0_Cell_voltage_M5_3 | 0x2A3 | 8 | TX | Cell voltages module 5 cell 9 10 11 |
| CAN0_Cell_voltage_M5_4 | 0x2A4 | 8 | TX | Cell voltages module 5 cell 12 13 14 |
| CAN0_Cell_voltage_M5_5 | 0x2A5 | 8 | TX | Cell voltages module 5 cell 15 16 17 |
| CAN0_Cell_temperature_M5_0 | 0x2B0 | 8 | TX | Cell temperatures module 5 - 0 1 2 |
| CAN0_Cell_temperature_M5_1 | 0x2B1 | 8 | TX | Cell temperatures module 5 - 3 4 5 |
| CAN0_Cell_temperature_M5_2 | 0x2B2 | 8 | TX | Cell temperatures module 5 - 6 7 8 |
| CAN0_Cell_temperature_M5_3 | 0x2B3 | 8 | TX | Cell temperatures module 5 - 9 10 11 |

Continued on next page

Table 14.1 – continued from previous page

| | | | | |
|---|---|---|---|---|
| CAN0_Cell_voltage_M6_0 | 0x2C0 | 8 | TX | Cell voltages module 6 cell 0 1 2 |
| CAN0_Cell_voltage_M6_1 | 0x2C1 | 8 | TX | Cell voltages module 6 cell 3 4 5 |
| CAN0_Cell_voltage_M6_2 | 0x2C2 | 8 | TX | Cell voltages module 6 cell 6 7 8 |
| CAN0_Cell_voltage_M6_3 | 0x2C3 | 8 | TX | Cell voltages module 6 cell 9 10 11 |
| CAN0_Cell_voltage_M6_4 | 0x2C4 | 8 | TX | Cell voltages module 6 cell 12 13 14 |
| CAN0_Cell_voltage_M6_5 | 0x2C5 | 8 | TX | Cell voltages module 6 cell 15 16 17 |
| CAN0_Cell_temperature_M6_0 | 0x2D0 | 8 | TX | Cell temperatures module 6 - 0 1 2 |
| CAN0_Cell_temperature_M6_1 | 0x2D1 | 8 | TX | Cell temperatures module 6 - 3 4 5 |
| CAN0_Cell_temperature_M6_2 | 0x2D2 | 8 | TX | Cell temperatures module 6 - 6 7 8 |
| CAN0_Cell_temperature_M6_3 | 0x2D3 | 8 | TX | Cell temperatures module 6 - 9 10 11 |
| CAN0_Cell_voltage_M7_0 | 0x2E0 | 8 | TX | Cell voltages module 7 cell 0 1 2 |
| CAN0_Cell_voltage_M7_1 | 0x2E1 | 8 | TX | Cell voltages module 7 cell 3 4 5 |
| CAN0_Cell_voltage_M7_2 | 0x2E2 | 8 | TX | Cell voltages module 7 cell 6 7 8 |
| CAN0_Cell_voltage_M7_3 | 0x2E3 | 8 | TX | Cell voltages module 7 cell 9 10 11 |
| CAN0_Cell_voltage_M7_4 | 0x2E4 | 8 | TX | Cell voltages module 7 cell 12 13 14 |
| CAN0_Cell_voltage_M7_5 | 0x2E5 | 8 | TX | Cell voltages module 7 cell 15 16 17 |
| CAN0_Cell_temperature_M7_0 | 0x2F0 | 8 | TX | Cell temperatures module 7 - 0 1 2 |
| CAN0_Cell_temperature_M7_1 | 0x2F1 | 8 | TX | Cell temperatures module 7 - 3 4 5 |
| CAN0_Cell_temperature_M7_2 | 0x2F2 | 8 | TX | Cell temperatures module 7 - 6 7 8 |
| CAN0_Cell_temperature_M7_3 | 0x2F3 | 8 | TX | Cell temperatures module 7 - 9 10 11 |
| CAN0_MSG_ISENS_TRIG | 0x35B | 8 | TX | Trigger for CurrentSensor |

Need to add the messages Name/IDs for Module 8. IDs can start from 0x400, repeating the pattern of Modules 0 to 7.

### 14.1.2 CAN RX Signals

| **Message: CAN0_Software_Reset** | **Message ID: 0x95** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |

When this message is received, a software reset is performed.

---

| **Message: CAN0_Debug_Message** | **Message ID: 0x100** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_DEBUG_Data | 0 | 64 | 0.0 | 0xFFFFFFFF | 1.0 | 0.0 | none | Intel |

This message contains only one single signal. The debug data is interpreted by the embedded software differently depending on its actual databits. In the software FAQ (*How to simply trigger events via CAN?*), further details are given on the handling of debug messages.

---

| **Message: CAN0_State_Request** | **Message ID: 0x120** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_ReceiveStateRequest | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

CAN0_SIG_ReceiveStateRequest 8

In the messages described in the previous table, the signal `CAN0_SIG_ReceiveStateRequest` is used to change the state of the foxBMS Master Unit. This is one of the most important signals: it allows initiating the procedure to close the power contactors. Possible request numbers in default configuration are 3 (Normal state), 4 (Charge state) and 8 (Standby state), but can be configured (0 is No Request). More details can be found in the documentation of the module *BMS*. (Section 28.3)

A state request has to be sent every 100ms, otherwise the BMS enters into an error state. The requests must come in the window 95-105ms to be valid.

---

| **Message: CAN0_MSG_IVT_Current** | **Message ID: 0x35C/0x521** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_Current_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Current_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Current_Measurement | 16 | 32 | -2147483648 | 2147483647 | 1.0 | 0.0 | mA | Intel |

`CAN0_IVT_Current_MuxID` indicates the measurement type and `CAN0_IVT_Current_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_Current_Measurement` is the current value measured by the sensor. It is a 32 bit signed value with 1mA resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_Voltage_1** | **Message ID: 0x35D/0x522** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **unit** | **Byte-order** |
| CAN0_SIG_IVT_Voltage_1_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_1_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_1_Measurement | 16 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | mV | Intel |

`CAN0_IVT_Voltage_1_MuxID` indicates the measurement type and `CAN0_IVT_Voltage_1_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_Voltage_1_Measurement` is the voltage value measured by the sensor at the voltage input channel 1. It is a 32 bit unsigned value with 1mV resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_Voltage_2** | **Message ID: 0x35E/0x523** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **unit** | **Byte-order** |
| CAN0_SIG_IVT_Voltage_2_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_2_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_2_Measurement | 16 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | mV | Intel |

`CAN0_IVT_Voltage_2_MuxID` indicates the measurement type and `CAN0_IVT_Voltage_2_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_Voltage_2_Measurement` is the voltage value measured by the sensor at the

voltage input channel 2. It is a 32 bit unsigned value with 1mV resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_Voltage_3** | **Message ID: 0x35F/0x524** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_Voltage_3_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_3_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Voltage_3_Measurement | 16 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | mV | Intel |

`CAN0_IVT_Voltage_3_MuxID` indicates the measurement type and `CAN0_IVT_Voltage_3_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_Voltage_3_Measurement` is the voltage value measured by the sensor at the voltage input channel 3. It is a 32 bit unsigned value with 1mV resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| Message: CAN0_MSG_IVT_Temperature | Message ID: 0x525 | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Signal name | Start bit | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_Temperature_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Temperature_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Temperature_Measurement | 16 | 32 | -2147483648 | 2147483647 | 1.0 | 0.0 | 0.1°C | Intel |

`CAN0_IVT_Temperature_MuxID` indicates the measurement type and `CAN0_IVT_Temperature_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_Temperature_Measurement` is the temperature value measured by the sensor. It is a 32 bit signed value with 0.1°C resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_Power** | **Message ID: 0x526** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_Power_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Power_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_Power_Measurement | 16 | 32 | -2147483648 | 2147483647 | 1.0 | 0.0 | W | Intel |

`CAN0_IVT_Power_MuxID` indicates the measurement type and `CAN0_IVT_Power_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN_IVT0_Power_Measurement` is the power value measured by the sensor (referring to current and voltage U1). It is a 32 bit signed value with 1W resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_CoulombCount** | **Message ID: 0x527** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_CC_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_CC_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_CC_Measurement | 16 | 32 | -2147483648 | 2147483647 | 1.0 | 0.0 | As | Intel |

`CAN0_IVT_CoulombCount_MuxID` indicates the measurement type and `CAN0_IVT_CoulombCount_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_CoulombCount_Measurement` is the current counted by the sensor. It is a 32 bit signed value with 1As resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_IVT_EnergyCount** | **Message ID: 0x528** | **DLC: 6** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_IVT_EC_MuxID | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_EC_Status | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_IVT_EC_Measurement | 16 | 32 | -2147483648 | 2147483647 | 1.0 | 0.0 | Wh | Intel |

`CAN0_IVT_EnergyCount_MuxID` indicates the measurement type and `CAN_IVT0_EnergyCount_Status` indicates status values and counters of the current sensor measurement. For details the datasheet of the current sensor must be consulted. `CAN0_IVT_EnergyCount_Measurement` is the energy counted by the sensor (referring to current and voltage U1). It is a 32 bit signed value with 1Wh resolution. It must be noted that resolution in CAN transmission is not the same as the measurement accuracy.

---

| **Message: CAN0_MSG_GetReleaseVersion** | **Message ID: 0x777** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_GetReleaseVersion | 0 | 64 | 0.0 | 0 | 1.0 | 0.0 | none | Intel |

When message `CAN0_MSG_GetReleaseVersion` is received, CAN message `CAN0_MSG_BOOT` is transmitted as response message.

---

### 14.1.3 CAN TX Signals

| Message: CAN0_MSG_BOOT | Message ID: 0x101 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_VersionNumberMajor | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_VersionNumberMinor | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_VersionNumberBugfix | 16 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Checksum | 32 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | none | Intel |

`CAN0_SIG_VersionNumberMajor`, `CAN0_SIG_VersionNumberMinor` and signal `CAN0_SIG_VersionNumberMinor` add up to the current flashed SW-version. The latest Releaseversion is `1.6.3`. `CAN0_SIG_Checksum` holds the calculated CRC32 checksum of the flashed software. This message is transmitted once after startup or can be requested via CAN.

---

GS = genreal status?

| Message: CAN0_MSG_SystemState_0 | Message ID: 0x110 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_GS0_general_error | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_current_state | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_overtemp_charge | 16 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_undertemp_charge | 24 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_overtemp_discharge | 32 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_undertemp_discharge | 40 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_overcurrent_charge | 48 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS0_error_overcurrent_discharge | 56 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

| Message: CAN0_MSG_SystemState_1 | Message ID: 0x111 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_GS1_error_overvoltage | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_error_undervoltage | 8 | 3 | 0.0 | 7 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_error_deep_discharge | 11 | 5 | 0.0 | 31 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_error_temperature_MCU0 | 16 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_error_contactor | 24 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_error_selftest | 32 | 8 | for future use (not implemented yet) | | | | | |
| CAN0_SIG_GS1_error_cantiming | 40 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_current_sensor | 48 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS1_balancing_active | 56 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

| Message: CAN0_MSG_SystemState_2 | Message ID: 0x112 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_GS2_state_cont_interlock | 0 | 16 | 0.0 | 65535 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_error_insulation | 16 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_fuse_state | 24 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_lowCoinCellVolt | 32 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_error_openWire | 40 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_daisyChain | 48 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_GS2_plausibilityCheck | 56 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

The signals of the messages `CAN0_MSG_SystemState_0` , `CAN0_MSG_SystemState_1` and `CAN0_MSG_SystemState_2` display information about the system status of the battery system.

* `CAN0_SIG_GS0_general_error` indicates if and an error is detected from the BMS:
  - Bit[0]: maximum safety limit violated, CAN timing, current sensor or contactor error detected
  - Bit[1]: recommended safety limit violated
  - Bit[2]: maximum operating limit violated
* The current state of the battery system is represented by the a bitfield in signal `CAN0_SIG_GS0_current_state`:
  - 0x00: Uninitialized
  - 0x03: Idle
  - 0x04: Standby
  - 0x05: Precharge (normal)
  - 0x06: Normal
  - 0x07: Precharge (charge)
  - 0x08: Charge
  - 0xF0: Error

| | |
|---|---|
| CAN0_SIG_GS0_general_error | 0 |
| CAN0_SIG_GS0_current_state | 8 |
| CAN0_SIG_GS0_error_overtemp_charge | 16 |
| CAN0_SIG_GS0_error_undertemp_charge | 24 |
| CAN0_SIG_GS0_error_overtemp_discharge | 32 |
| CAN0_SIG_GS0_error_undertemp_discharge | 40 |
| CAN0_SIG_GS0_error_overcurrent_charge | 48 |
| CAN0_SIG_GS0_error_overcurrent_discharge | 56 |

| | |
|---|---|
| CAN0_SIG_GS1_error_overvoltage | 0 |
| CAN0_SIG_GS1_error_undervoltage | 8 |
| CAN0_SIG_GS1_error_deep_discharge | 11 |
| CAN0_SIG_GS1_error_temperature_MCU0 | 16 |
| CAN0_SIG_GS1_error_contactor | 24 |
| CAN0_SIG_GS1_error_selftest | 32 |
| CAN0_SIG_GS1_error_cantiming | 40 |
| CAN0_SIG_GS1_current_sensor | 48 |
| CAN0_SIG_GS1_balancing_active | 56 |

* The various error signals `CAN0_SIG_GSx_error_xxxx` give information about limit violations and internal errors. In general, the following bitfield is used:
  - Bit[0]: maximum safety limit violated
  - Bit[1]: recommended safety limit violated
  - Bit[2]: maximum operating limit violated

Flag `CAN0_SIG_GS1_error_deep_discharge` is a non-volatile flag saved in backup SRAM that indicates if a deep-discharge of a cell has been detected. This means that a power-cycle of the BMS won't reset the flag provided the coin cell is not empty. The affected cell has to be replaced and the BMS won't reset the flag until the corresponding CAN debug message was sent (see *How to reset deep-discharge flag via CAN?*).

`CAN0_SIG_GS1_error_temperature_MCU0` indicates if the operating range of the MCU0 junction temperature is violated. `CAN0_SIG_GS1_error_cantiming` is set to 1 if the can timing is violated (see #define CHECK_CAN_TIMING for more information). `CAN0_SIG_GS1_current_sensor` is 1 if no current sensor is detected and otherwise 0. `CAN0_SIG_GS1_error_contactor` is set to 1 if contactor error is detected. This means the feedback signal of the contactor doesn't match the controlled state. `CAN0_SIG_GS1_error_cantiming` occurs if no state request message is received within 100ms. This feature can be enabled/disabled (see *Important Switches in Code*). The signal `CAN0_SIG_GS1_balancing_active` is

a simple flag to show if balancing is ON (1) or OFF (0). `CAN0_SIG_GS2_error_insulation` indicates an insulation error. `CAN0_SIG_GS2_dieTempMCU` indicates a violation of the primary MCU die temperature.

Cannot find it in the table.

The status of contactors and the interlock is represented as a bitfield in `BMS1_status_contactors` (0: open, 1: closed):

CAN0_SIG_GS2_state_cont_interlock

| | |
|---|---|
| CAN0_SIG_GS2_state_cont_interlock | 0 |
| CAN0_SIG_GS2_error_insulation | 16 |
| CAN0_SIG_GS2_fuse_state | 24 |
| CAN0_SIG_GS2_lowCoinCellVolt | 32 |
| CAN0_SIG_GS2_error_openWire | 40 |
| CAN0_SIG_GS2_daisyChain | 48 |
| CAN0_SIG_GS2_plausibilityCheck | 56 |

- Bit[0]: Plus (normal)
- Bit[1]: Precharge (normal)
- Bit[2]: Minus (normal)
- Bit[3]: Plus (charge)
- Bit[4]: Precharge (charge)
- Bit[5]: Minus (charge)
- Bit[6-8]: reserved for future use
- Bit[9]: Interlock status
- Bit[10-15]: reserved for future use

Bitfield `CAN0_SIG_GS2_lowCoinCellVolt` displays the coin cell status.

- Bit[0]: Coin cell voltage low: swap battery in the near future
- Bit[1]: Coin cell voltage critically low: swap battery as soon as possible

`CAN0_SIG_GS2_error_openWire` is set to 1 if an open voltage sense wire is detected. Bitfield `CAN0_SIG_GS2_fuse_state` indicates the fuse state (0: fuse intact, 1: fuse tripped):

- Bit[0]: Fuse state normal path
- Bit[1]: Fuse/Contactor state normal path
- Bit[2]: Fuse state charge path
- Bit[3]: Fuse/Contactor state charge path

A problem with the daisy-chain communication is indicated with bitfield `CAN0_SIG_GS2_daisyChain` (0: no error, 1: error)

- Bit[0]: spi error
- Bit[1]: ltc PEC error (LTC)
- Bit[2]: multiplexer error

Plausibility errors detected by one of the checks are indicated with bitfield `CAN0_SIG_GS2_plausibilityCheck` (0: no error, 1: error)

- Bit[0]: cell voltage plausibility error
- Bit[1]: temperature sensor plausibility error
- Bit[2]: pack voltage plausibility error

---

| **Message: CAN_MSG_SlaveState_0** | **Message ID: 0x115** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_SS0_states | 0 | 64 | for future use (not implemented yet) | | | | | |

| Message: CAN_MSG_SlaveState_1 | Message ID: 0x116 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_SS1_states | 0 | 64 | for future use (not implemented yet) | | | | | |

This messages indicate any errors with the foxBMS Slave Units but this functionality is currently not supported.

| Message: CAN0_RecOperatingCurrent | Message ID: 0x130 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_RecChargeCurrent | 0 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | A | Intel |
| CAN0_SIG_RecChargeCurrent_Peak | 16 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | A | Intel |
| CAN0_SIG_RecDischargeCurrent | 32 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | A | Intel |
| CAN0_SIG_RecDischargeCurrent_Peak | 48 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | A | Intel |

The recommended current capability of the battery system is indicated by the signals of the previous table. For both discharge and charge directions, two values are stated, one for the continuous current capability and one for the peak current capability. Currently, the peak value is simply the same as the continuous value. The resolution of the values is 0.1A, which should be of sufficient granularity for most applications. It is a tradeoff that enables a range up to 6553.5A in charge and discharge direction with reasonable resolution.

**Note:** This feature is currently not supported!

| Message: CAN0_MSG_SOP | Message ID: 0x131 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MaxChargePower | 0 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | kW | Intel |
| CAN0_SIG_MaxChargePower_Peak | 16 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | kW | Intel |
| CAN0_SIG_MaxDischargePower | 32 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | kW | Intel |
| CAN0_SIG_MaxDischargePower_Peak | 48 | 16 | 0.0 | 6553.5 | 0.1 | 0.0 | kW | Intel |

The power capability of the battery system is indicated by the signals of the previous table. For both discharge and charge directions, two values are stated, one for the continuous power capability and one for the peak power capability. Currently, the peak value is simply the same as the continuous value. The resolution of the values is 100W, which should be of sufficient granularity for most applications. It is a tradeoff that enables a range up to 6553.5kW in charge and discharge direction with reasonable resolution.

**Note:** This feature is currently not supported!

| Message: CAN0_MSG_SOC | Message ID: 0x140 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_SOC_mean | 0 | 16 | 0.0 | 100.0 | 0.01 | 0.0 | % | Intel |
| CAN0_SIG_SOC_min | 16 | 16 | 0.0 | 100.0 | 0.01 | 0.0 | % | Intel |
| CAN0_SIG_SOC_max | 32 | 16 | 0.0 | 100.0 | 0.01 | 0.0 | % | Intel |
| CAN0_SIG_SOC_max | 48 | 16 | currently not used | | | | | |

Once initialized, `CAN0_SIG_SOC_mean`, `CAN0_SIG_SOC_min`, `CAN0_SIG_SOC_max` the values are updated via Coulomb counting. The values can be initialized either with a look-up table or by setting the values directly.

---

| Message: CAN0_MSG_SOH | Message ID: 0x150 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |

This message is currently not backed up with information and provided for forward-compatibility only.

---

| Message: CAN0_MSG_SOE | Message ID: 0x160 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |

This message is currently not backed up with information and provided for forward-compatibility only.

---

| Message: CAN_MinMaxCellVoltages | Message ID: 0x170 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Cellvolt_mean | 0 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Cellvolt_min | 16 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Cellvolt_max | 32 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_ModNumber_volt_min | 48 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_ModNumber_volt_max | 56 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

The minimum, maximum and mean voltage are transmitted in the signals `CAN0_SIG_Cellvolt_min`, `CAN0_SIG_Cellvolt_max` and `CAN0_SIG_Cellvolt_mean`. Additionally, the number of the module with minimum and maximum voltage is indicated in the signals `CAN0_SIG_ModNumber_volt_min` and `CAN0_SIG_ModNumber_volt_max`. Indexing of module numbers starts from zero.

---

| Message: CAN0_MSG_SOV | Message ID: 0x171 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |

This message is currently not backed up with information and provided for forward-compatibility only.

| Message: CAN0_MinMaxCellTemperatures | Message ID: 0x180 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Celltemp_mean | 0 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Celltemp_min | 16 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Celltemp_max | 32 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_ModNumber_temp_min | 48 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_ModNumber_temp_max | 56 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |

The minimum, maximum and mean temperature are transmitted in the signals `CAN0_SIG_Celltemp_min`, `CAN0_SIG_Celltemp_max` and `CAN0_SIG_Celltemp_mean`, respectively. Additionally, the number of the module with minimum and maximum temperature is indicated in the signals `CAN0_SIG_ModNumber_temp_min` and `CAN0_SIG_ModNumber_temp_max`, respectively. Indexing of module numbers starts from zero.

| Message: CAN0_Tempering | Message ID: 0x190 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_CoolingNeeded | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_HeatingNeeded | 8 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_TemperingDemand | 16 | 32 | 0.0 | 0xFFFFFFFF | 1.0 | 0.0 | none | Intel |

The signals `CAN0_SIG_CoolingNeeded`, `CAN0_SIG_HeatingNeeded` and `CAN0_SIG_TemperingDemand` are left free for use for application specific development. Here, `CAN0_SIG_CoolingNeeded` and `CAN0_SIG_HeatingNeeded` could be simple ON/OFF flags and `CAN0_SIG_TemperingDemand` could be a 'continuous' value indicating the needed cooling/heating power.

| Message: CAN0_Insulation | Message ID: 0x1A0 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_InsulationStatus | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_InsulationValue | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | kOhm | Intel |

The signal `CAN0_SIG_InsulationStatus` indicates if the insulation value is greater than the threshold value and `CAN0_SIG_InsulationValue` holds the measured insulation value by the insulation monitoring device.

| Message: CAN0_MovAveragePower_0 | Message ID: 0x1D0 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Power_1s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |
| CAN0_SIG_MovAverage_Power_5s | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |

| Message: CAN0_MovAveragePower_1 | Message ID: 0x1D1 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Power_10s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |
| CAN0_SIG_MovAverage_Power_30s | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |

| Message: CAN0_MovAveragePower_2 | Message ID: 0x1D2 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Power_60s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |
| CAN0_SIG_MovAverage_Power_config | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | W | Intel |

The messages `CAN0_MovAveragePower_0`, `CAN0_MovAveragePower_1` and `CAN0_MovAveragePower_2` provide moving average values over the different time frames 1s, 5s, 10s, 30s and 60s. The signal `CAN0_SIG_MovAverage_Power_config` can be configured to an arbitrary duration. The default configuration is 3 seconds.

---

| Message: CAN0_MovAverageCurrent_0 | Message ID: 0x1E0 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Current_1s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |
| CAN0_SIG_MovAverage_Current_5s | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |

| Message: CAN0_MovAverageCurrent_1 | Message ID: 0x1E1 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Current_10s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |
| CAN0_SIG_MovAverage_Current_30s | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |

| Message: CAN0_MovAverageCurrent_2 | Message ID: 0x1E2 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_MovAverage_Current_60s | 0 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |
| CAN0_SIG_MovAverage_Current_config | 32 | 32 | -2500000 | 4292467295 | 1.0 | -2500000 | mA | Intel |

The messages `CAN0_MovAverageCurrent_0`, `CAN0_MovAverageCurrent_1` and `CAN0_MovAverageCurrent_2` provide moving mean values over the different time frames 1s, 5s, 10s, 30s and 60s. The signal `CAN0_SIG_MovAverage_Current_config` can be configured to an arbitrary duration. The default configuration is 3 seconds.

---

| Message: CAN0_PackVoltage | Message ID: 0x1F0 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_PackVolt_Battery | 0 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_PackVolt_PowerNet | 32 | 32 | 0.0 | 4294967295 | 1.0 | 0.0 | mV | Intel |

The signal `CAN0_SIG_PackVolt_Battery` provides the measured battery pack voltage (V0 Sense) from the current sensor between `Battery MINUS` and `Battery PLUS`. `CAN0_SIG_PackVolt_PowerNet` contains the power net voltage measured between `Battery MINUS` and `Load PLUS` (V2 Sense). See Fig. 21.2 for a detailed BJB block diagram.

---

| Message: CAN0_Cell_voltage_M0_0 | Message ID: 0x200 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_0_2 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_0 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_1 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_2 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

| Message: CAN0_Cell_voltage_M0_1 | Message ID: 0x201 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_3_5 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_3 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_4 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_5 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

Important changes to make:
1. CAN0_SIG_Mod0_xxx to CAN0_SIG_Modx_xxx so that we can specify Module x in the signal.
2. We use Bits 56 to 63 to represent Modx, starting from 0 for Module 0.

Advantages:
1. We can have up to 255 modules in the system.
2. Significantly reduced CAN IDs.
3. Easy programming.

| Message: CAN0_Cell_voltage_M0_2 | Message ID: 0x202 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_6_8 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_6 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_7 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_8 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

| Message: CAN0_Cell_voltage_M0_3 | Message ID: 0x203 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_9_10 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_9 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_10 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_11 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

| Message: CAN0_Cell_voltage_M0_4 | Message ID: 0x204 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_12_0 | 4 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_12 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_13 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_14 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

| Message: CAN0_Cell_voltage_M0_5 | Message ID: 0x205 | DLC: 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_volt_valid_15_0 | 7 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_volt_15 | 8 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_16 | 24 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |
| CAN0_SIG_Mod0_volt_17 | 40 | 16 | 0.0 | 65535 | 1.0 | 0.0 | mV | Intel |

Three cell voltages are transmitted within one CAN message. The signal `CAN0_SIG_Modx_volt_valid_x_x` contains three valid flags (bit 0, 1 and 2) for the voltages transmitted in the same messages. 0 –> valid cell voltage, 1 –> invalid cell voltage. Signal `CAN0_SIG_Mod0_volt_valid_6_8` for example indicates valid cell voltages with

- bit[0]: Module 0 cell voltage 6
- bit[1]: Module 0 cell voltage 7
- bit[2]: Module 0 cell voltage 8

| **Message: CAN0_Cell_temperature_M0_0** | **Message ID: 0x210** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_temp_valid_0_2 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_temp_0 | 8 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_1 | 24 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_2 | 40 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |

| **Message: CAN0_Cell_temperature_M0_1** | **Message ID: 0x211** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_temp_valid_3_5 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_temp_3 | 8 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_4 | 24 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_5 | 40 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |

| **Message: CAN0_Cell_temperature_M0_2** | **Message ID: 0x212** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_temp_valid_6_8 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_temp_6 | 8 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_7 | 24 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_8 | 40 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |

| **Message: CAN0_Cell_temperature_M0_3** | **Message ID: 0x213** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_Mod0_temp_valid_9_11 | 0 | 8 | 0.0 | 255 | 1.0 | 0.0 | none | Intel |
| CAN0_SIG_Mod0_temp_9 | 8 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_10 | 24 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |
| CAN0_SIG_Mod0_temp_11 | 40 | 16 | -128.0 | 527.35 | 0.01 | -128.0 | °C | Intel |

Three cell temperatures are transmitted within one CAN message. The signal `CAN0_SIG_Modx_temp_valid_x_x` contains three valid flags (bit 0, 1 and 2) for the tempera-

tures transmitted in the same messages. 0 –> valid temperature, 1 –> invalid temperature. Signal `CAN0_SIG_Mod0_temp_valid_6_8` for example indicates valid cell temperatures with

- bit[0]: Module 0 cell temperature 6
- bit[0]: Module 0 cell temperature 7
- bit[0]: Module 0 cell temperature 8

Currently, the valid signals are not backed up with information and provided for forward-compatibility only. Therefore the flags always indicate valid cell voltages.

---

| **Message: CAN0_MSG_ISENS_TRIG** | **Message ID: 0x35B** | **DLC: 8** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal name** | **Start bit** | **Bit length** | **Min** | **Max** | **Factor** | **Offset** | **Unit** | **Byte-order** |
| CAN0_SIG_ISA_Trigger | 0 | 32 | 0.0 | 0xFFFFFFFF | 1.0 | 0.0 | none | Intel |

`ISENS_Trigger` is set to 0x31FFFF and sent via CAN bus to trigger a current sensor measurement cycle. This is the command of the Isabellenhütte sensor to trigger a measurement of all internal measurement channels when the sensor is not configured in cyclic mode.

This is not needed since we use the sensor in cycle mode.

# CHAPTER 15

# Checking-up foxBMS

This section describes how to test the BMS-Master Board with one BMS-Slave Board connected. These tests show if the hardware and software components of the BMS-Master Board and BMS-Slave Board are working correctly:

## 15.1 Required Hardware

- 1 * BMS-Master Board including a supply cable
- 1 * BMS-Slave Board
- 1 * Battery voltage simulation at the BMS-Slave Board (e.g., voltage divider)
- 1 * CAN-Bus to PC adapter
- 1 * Normally closed switch for opening the `interlock` (hereafter referred to as `emergency off`)
- 3 * Contactors with normally open feedback (hereafter referred to as `plus main contactor`, `plus precharge contactor`, `minus main contactor`)
- 1 * Debugger (optionally, but recommended)

### 15.1.1 Testing Without Debugger

It is also possible to get a little information if the system is running correctly without variable checking on the debugger. When requesting the ~~later described~~ states to the BMS, the contactors opening and closing can be heard.

### 15.1.2 Testing With Debugger

This test procedure gives more detailed test information when a debugger is used. The following variables should be checked during the test:

- `os_timer`
- `sys_state`

- `bms_state`
- `ltc_cellvoltage`
- `cont_contactor_states`
- `ilck_interlock_state`

## 15.2 Required Software

- foxBMS binaries for MCU0 (latest release on GitHub for primary)
- foxBMS binaries for MCU1 (latest release on GitHub for secondary)
- Debugger-Software
- Software to send CAN messages from the PC on the CAN bus.
- CAN message for Normal and Standby

## 15.3 Test Procedure

The test procedure consists of three steps:

1. Preparing the hardware
2. Building the software from source
3. Requesting bms-states to BMS-Master Board and checking variable values on the debugger

If no debugger is available, only a partially test if all parts of foxBMS are running correctly can be performed. As mentioned above, it can be checked accoustically if the contactors are opening and closing. This is marked in the *Test* with **(H)** and the corresponding explanation if needed.

### 15.3.1 Hardware Setup

1. Apply voltages to the BMS-Slave Board cell measurement inputs (e.g., voltage divider)
2. Connect the daisy chain connector from the BMS-Slave Board to the BMS-Master Board.
3. Connect the `emergency off` to the `interlock` of the BMS-Master Board.
4. Connect the following contactors to the BMS-Master Board:
    1. `contactor 0` the `plus main contactor` in the positive current path
    2. `contactor 1` the `plus precharge contactor` in the positive current path
    3. `contactor 2` the `minus main contactor` in the negative current path
5. Connect the CAN-interface to the PC
6. The debugger is connected from the PC to the JTAG-interface of MCU0 of BMS-Master Board (optional).

### 15.3.2 Software Setup

1. Build the foxBMS binaries for both, MCU0 and MCU1. (There should be a link for how to do it.)

### 15.3.3 Test

1. Power BMS-Master Board
2. Flash foxBMS MCU0 binaries on MCU0 How?
3. Flash foxBMS MCU1 binaries on MCU1
4. Start CAN-communication viewer (e.g., PCAN-View when using PCAN-USB)
5. Restart BMS-Master Board by pressing S401, as described on p. 138.
6. Send CAN message for Request *Standby*-state via CAN How?
7. Check on the debugger if the system timer is running; variable: `os_timer` (Expected value?)
8. Check if BMS-Slave Board reads voltages; variable: `ltc_cellvoltage`
9. Request *Standby*-state via CAN
    1. Check on the debugger if `interlock` is closed (variable: `ilck_interlock_state` with `ilck_interlock_state.feedback=ILCK_SWITCH_ON`)
10. Request *Normal*-state via CAN
    1. Check on the debugger if contactors are closed in the correct order (variable: `cont_contactor_states`)
        1. close `minus main contactor`
        2. close `plus precharge contactor`
        3. close `plus main contactor`
        4. open `plus precharge contactor`

    **(H)**: If this test is performed with no debugger, contactors can be clicking four times.
11. Request *Standby*-state via CAN
    1. Check on the debugger if `interlock` is closed (variable: `ilck_interlock_state`)
    2. Check on the debugger if contactors are opened (variable: `cont_contactor_states`)

    **(H)**: If this test is performed with no debugger, each contactor can be heard clicking one time.
12. Request *Normal*-state via CAN
    1. Check on the debugger if contactors are closed in the correct order (variable: `cont_contactor_states`)
        1. close `minus main contactor`
        2. close `plus precharge contactor`
        3. close `plus main contactor`
        4. open `plus precharge contactor`

    **(H)**: If this test is performed with no debugger, contactors can be heard clicking four times.
13. Open `interlock` by pressing `emergency off`
    1. Check on the debugger if `interlock` is opened (variable: `ilck_interlock_state`)
    2. Check on the debugger if contactors are opened (variable: `cont_contactor_states`)

    **(H)**: If this test is performed with no debugger, each contactor can be heard clicking one time.
14. Request *Normal*-state via CAN

    1. BMS should switch to `bms_state.state=BMS_STATEMACH_ERROR` as the `interlock` is still open (variable `bms_state`)

15. Close `interlock` by releasing `emergency off`
    1. Check on the debugger if `interlock` is still open (variable: `ilck_interlock_state`)
    2. BMS has to stay in error state (check variable `bms_state`)
16. Request *Standby*-state via CAN
    1. Check on the debugger if `interlock` is closed (variable: `ilck_interlock_state`)
    2. Check on the debugger if contactors are still open (variable: `cont_contactor_states`)
17. Request *Normal*-state via CAN
    1. Check on the debugger if contactors are closed in the correct order (variable: `cont_contactor_states`)
        1. close `minus main contactor`
        2. close `plus precharge contactor`
        3. close `plus main contactor`
        4. open `plus precharge contactor`

    **(H)**: If this test is performed with no debugger, contactors can be heard clicking four times.
18. Request *Standby*-state via CAN
    1. Check on the debugger if `interlock` is closed (variable: `ilck_interlock_state`)
    2. Check on the debugger if contactors are still open (variable: `cont_contactor_states`)
19. done

CHAPTER 16

# Specifications

The following specifications must be met to ensure a safe and optimal work with the foxBMS hardware.

## 16.1 BMS-Master

### 16.1.1 Electrical Ratings

| Description | Minimum | Typical | Maximum | Unit |
|---|---|---|---|---|
| Supply Voltage DC | 10 | – | 26 | V |
| Contactor Continuous Current | – | – | 4 | A |
| Contactor Feedback Supply Voltage | – | 5.0 | – | V |
| Analog Input | – | – | 3.3 | V |
| Isolated Contacts Continuous Current | – | – | 4 | A |
| Interlock Circuit Sink Current | – | 10 | – | mA |
| Idle Supply Current at 12V supply | – | 150 | – | mA |
| Idle Supply Current at 24V supply | – | 110 | – | mA |

### 16.1.2 Mechanical Dimensions (BMS-Master PCB only)

| Description | Value | Unit |
|---|---|---|
| Width | 120 | mm |
| Length | 160 | mm |
| Height | 15 | mm |
| Weight | 122 | g |

### 16.1.3 Mechanical Dimensions (BMS-Master Unit in housing)

The BMS-Master Unit consists of BMS-Master board, BMS-Interface board and BMS-Extension board and is mounted in an aluminum housing.

| Description | Value | Unit |
|---|---|---|
| Width | 124 | mm |
| Length | 164 | mm |
| Height | 49.2 | mm |
| Weight | 785 | g |

## 16.2 BMS-Slave 12-Cell

### 16.2.1 Electrical Ratings

| Description | Minimum | Typical | Maximum | Unit |
|---|---|---|---|---|
| Battery Module Voltage | 11 | – | 60 | V |
| Single Battery Cell Voltage | 0 | – | 5 | V |
| Temperature Sensor Inputs | – | 10k | – | Ω |
| Analog Inputs (pin headers) | 0 | – | 5 | V |
| Digital Inputs/Outputs (pin headers) | 0 | – | 5 | V |
| External DC Supply (HW version 2.1.0 and above) | 8 | 12 | 24 | V |

### 16.2.2 Mechanical Dimensions (PCB only)

| Description | Value | Unit |
|---|---|---|
| Width | 100 | mm |
| Length | 160 | mm |
| Height | 15 | mm |
| Weight | 83 | g |

## 16.3 BMS-Slave 18-Cell

### 16.3.1 Electrical Ratings

| Description | Minimum | Typical | Maximum | Unit |
|---|---|---|---|---|
| Battery Module Voltage | 16 | – | 90 | V |
| Single Battery Cell Voltage | 0 | – | 5 | V |
| Temperature Sensor Inputs | – | 10k | – | Ω |
| Analog Inputs (pin headers) | 0 | – | 5 | V |
| Digital Inputs/Outputs (pin headers) | 0 | – | 5 | V |
| External DC Supply | 8 | 12 | 24 | V |

### 16.3.2 Mechanical Dimensions (PCB only)

| Description | Value | Unit |
|---|---|---|
| Width | 100 | mm |
| Length | 160 | mm |
| Height | 15 | mm |
| Weight | 88 | g |

# CHAPTER 17

# Master-Unit

This section describes the hardware modules used in foxBMS. Fig. 17.1 shows a block diagram of the foxBMS Master Unit with all its components.

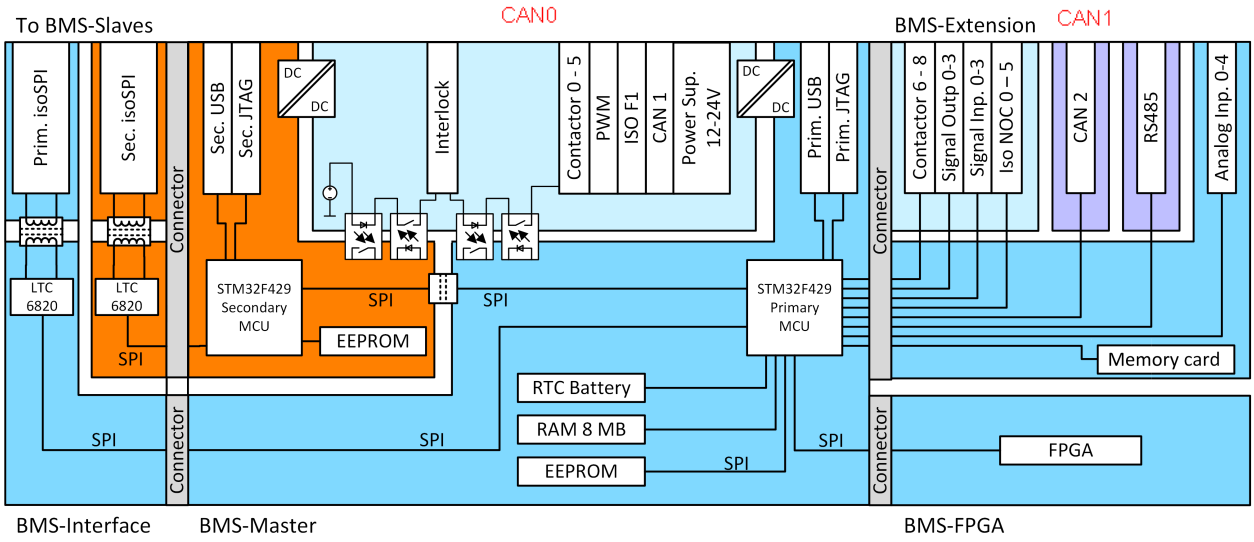



Fig. 17.1: foxBMS Master Unit hardware block diagram

## 17.1 Supply 0 (primary)

The external supply for the BMS-Master Board ~~is divided in a part supplying~~ powers the interlock circuit, and the contactors using (`SUPPLY_EXT2`), and the rest of the circuit (`SUPPLY_EXT0`) as shown in Fig. 17.2 . The foxBMS Master Unit can be supplied with a voltage between 12V and 24V DC. The primary part of the foxBMS is isolated from `SUPPLY_EXT0` by an isolating DC/DC converter (IC202). The 5V output of the DC/DC converter is stepped down to 3.3V by an

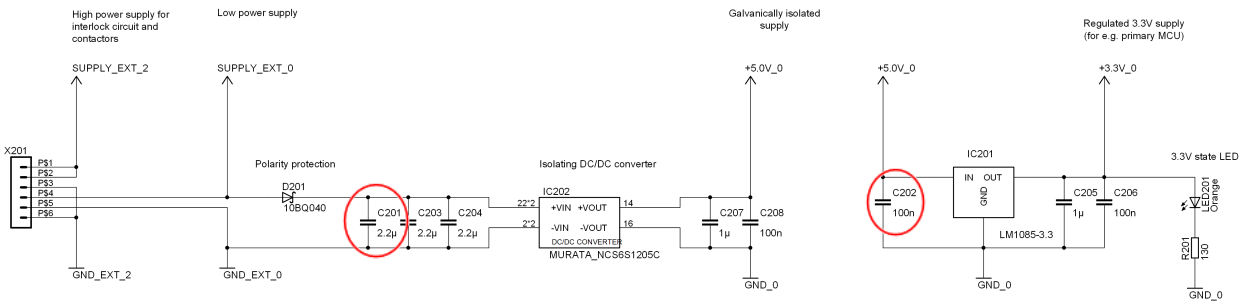



Fig. 17.2: Supply for all circuit parts related to the primary part of foxBMS

LM1085 LDO (IC201). The output of IC201 (+3.3V_0) supplies the primary microcontroller MCU0 and the related circuits.

## 17.2 Supply 1 (secondary)

Note: the naming of the capacitors is not consistent.

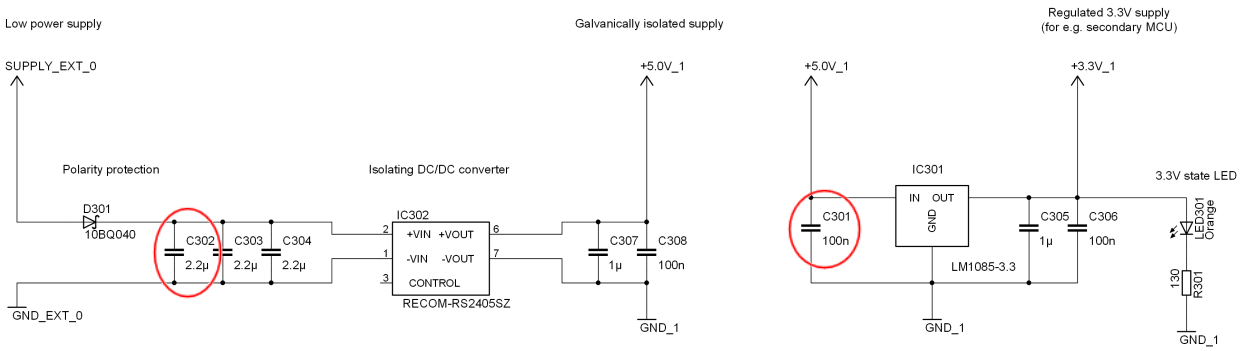



Fig. 17.3: Supply for all circuit parts related to the secondary part of foxBMS

The secondary part of foxBMS is also supplied by `SUPPLY_EXT0` as shown in Fig. 17.3. Also the secondary part of the BMS-Master Board is isolated from `SUPPLY_EXT0` by an isolating DC/DC converter (IC302). The 5V output of the DC/DC converter is stepped down to 3.3V by an LM1085 LDO (IC301). The output of IC301 (+3.3V_1) supplies the secondary microcontroller MCU1 and the related circuits.

## 17.3 MCU0 (primary)

Fig. 17.4 shows the boot and reset related circuits of the primary microcontroller MCU0. MCU0 can be manually reset by push button S401. Please note that the housing has to be opened to reach S401, therefore resetting MCU0 by means of S401 is intended for use in a laboratory setting/debugging situation.

The ADCs of the primary microcontroller MCU0 are supplied with a 2.5V reference voltage provided by an ADR3425 (IC401) as shown in Fig. 17.5.

The primary microcontroller MCU0 is clocked by an 8MHz oscillator as shown in Fig. 17.6. An separate oscillator (Q402) is used to clock the RTC (real time clock) integrated in MCU0.

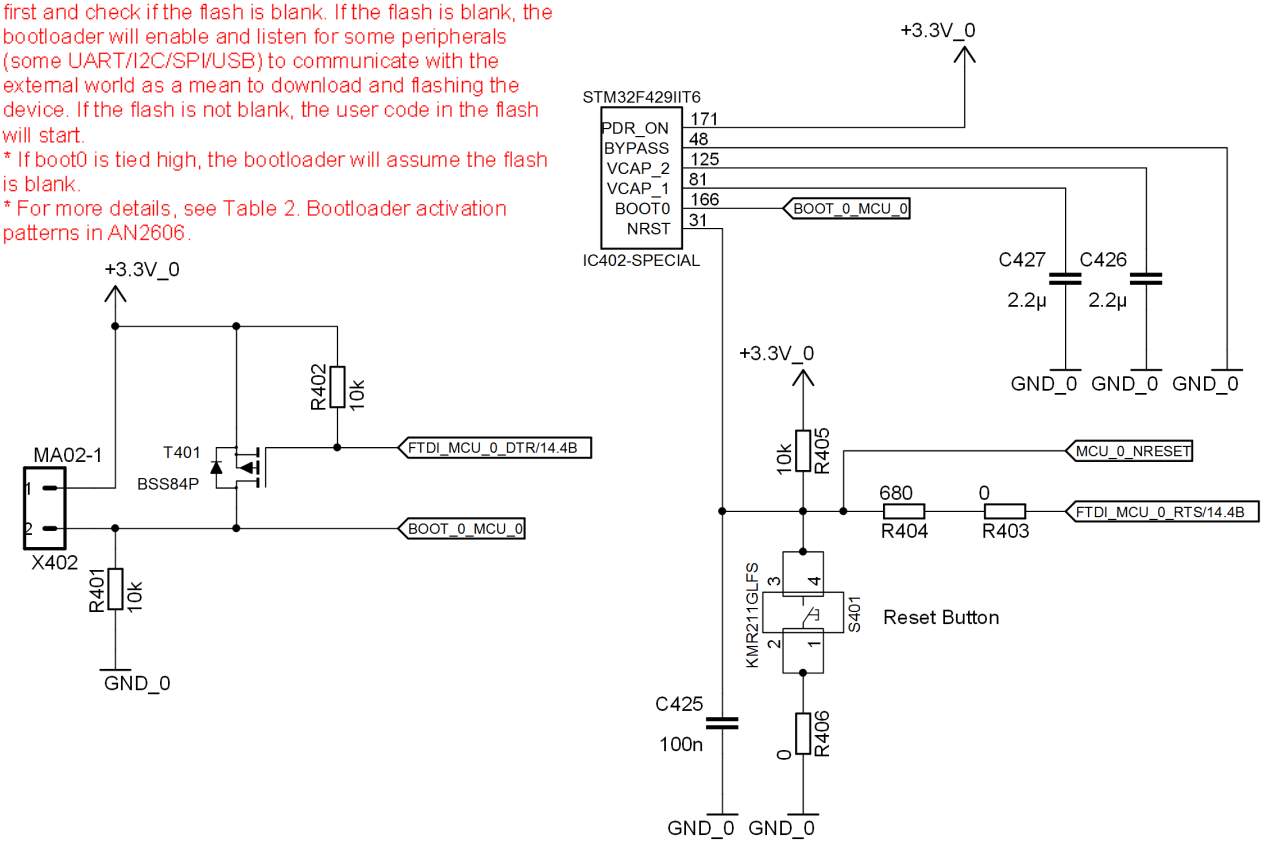



Fig. 17.4: Boot and reset circuit for the primary microcontroller

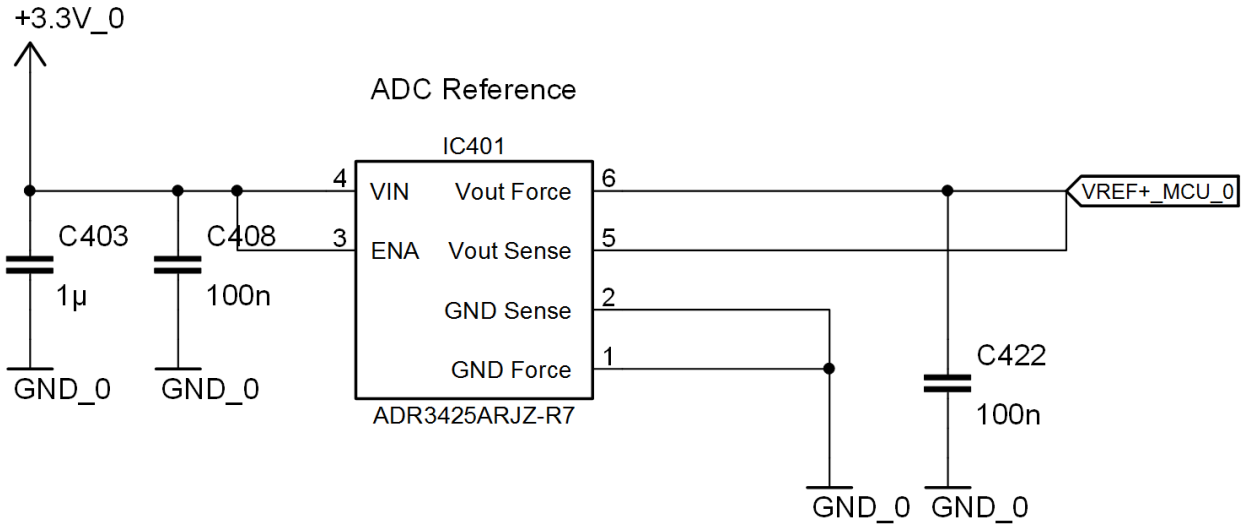



Fig. 17.5: ADC reference voltage for the primary microcontroller

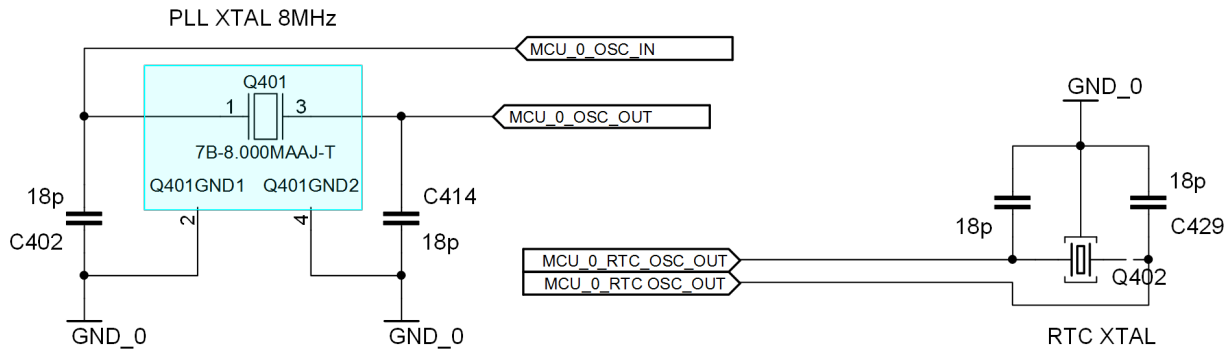



Fig. 17.6: Clock circuits of the primary microcontroller

## 17.4 Interface between MCU0 and MCU1

Besides being linked over the common interlock line, the primary and secondary microcontroller also have a common SPI data interface. The secondary microcontroller MCU1 acts as the master in the SPI communication. The interface is isolated using an ADUM3401 as shown in Fig. 17.7.

## 17.5 Interface to Bender ISOMETER

The BMS-Master Board supports Bender ISOMETER IR155-3203/-3204/-3210. Two inputs are provided: one for a PWM coded diagnostic signal and one for a simple status signal (OK or NOK) as shown in Fig. 17.8. The Bender ISOMETER is supplied by `SUPPLY_EXT0` and may be switched on or off (lowside) by the BMS-Master Board. The input signals are limited to level of 5V with Zener diodes D701 and D702. In order to adapt the interface for use with a IR155-3204/-3203 device, the solder jumper R705 has to be removed. The input signals are isolated from the microcontoller by an ADUM3301 (IC702).

## 17.6 CAN0

The CAN0 interface is intended to connect the foxBMS Master Unit to additional sensors, such as the Isabellenhütte IVT-MOD-300 current sensor ~~to the foxBMS Master Unit~~ and ~~the foxBMS Master Unit to~~ other devices such as a test bench control unit or an HMI unit. The circuit in Fig. 17.9 shown the input circuit consisting of protection diode D801, common mode choke L801, C804, and termination resistors R801 and R802. The CAN transceiver TJA1052 provides isolation and can be put to sleep by the primary microcontroller MCU0 via an CPC1008N optocoupler (IC803). The external part of the CAN0 interface is supplied by `SUPPLY_EXT0`.

## 17.7 Interlock

The primary and secondary microcontroller share a common interlock line as shown in Fig. 17.10. The interlock line is isolated from both microcontrollers MCU0 and MCU1 by optocouplers. The interlock line is supplied with 10mA by a current source (LM317 - IC901). It can be interrupted by the primary microcontroller MCU0 via optocoupler IC902 and can be read back by MCU0 via optocoupler IC903. The secondary microcontroller MCU1 can interrupt the interlock line via IC904 and read the interlock status via IC905. The 10mA cause a voltage drop on R906, which turns on MOSFET T901. T901 switches the common ground of all contactors (connected to the BMS-Master Board

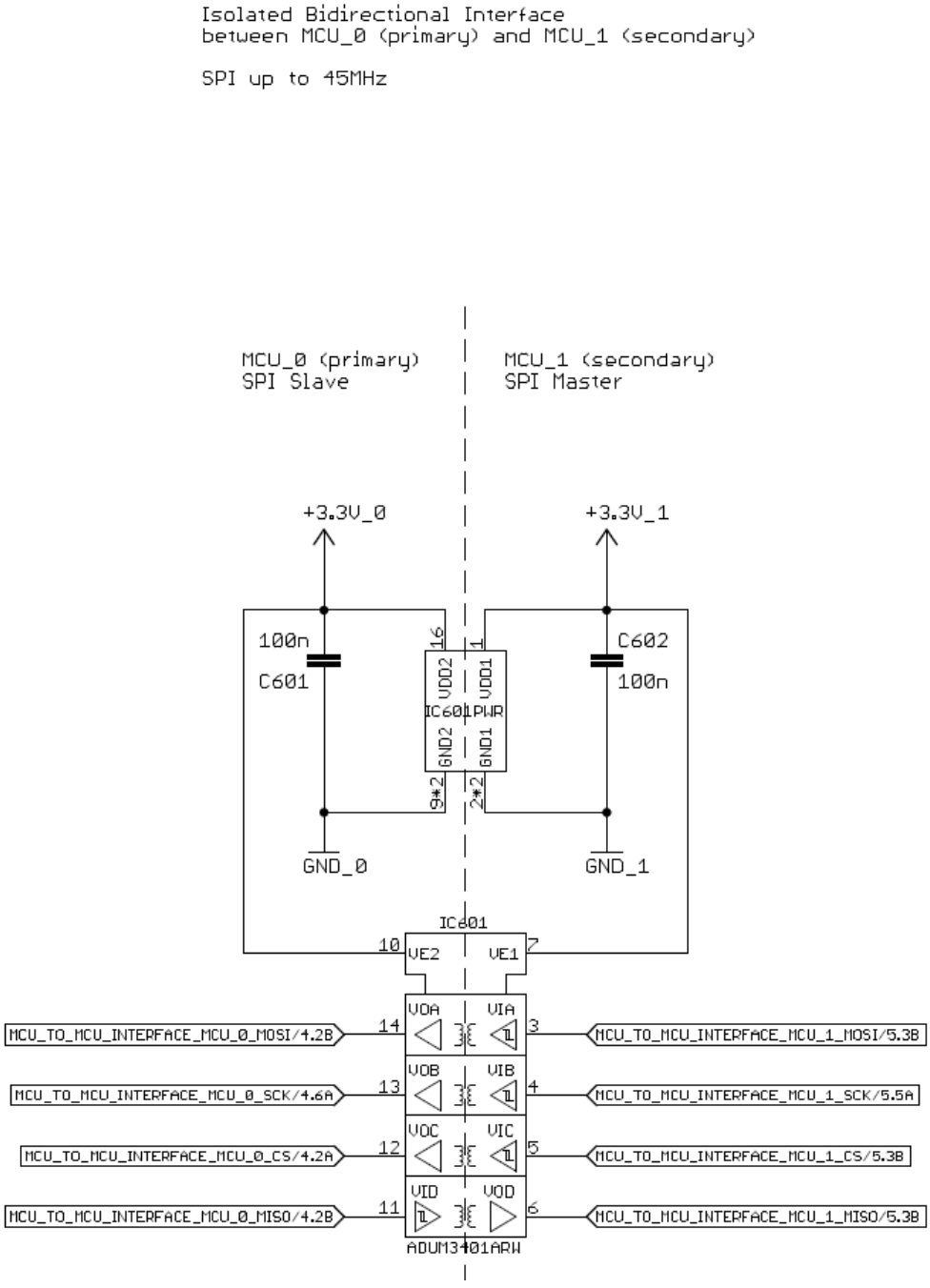



Fig. 17.7: Interface between primary and secondary microcontroller

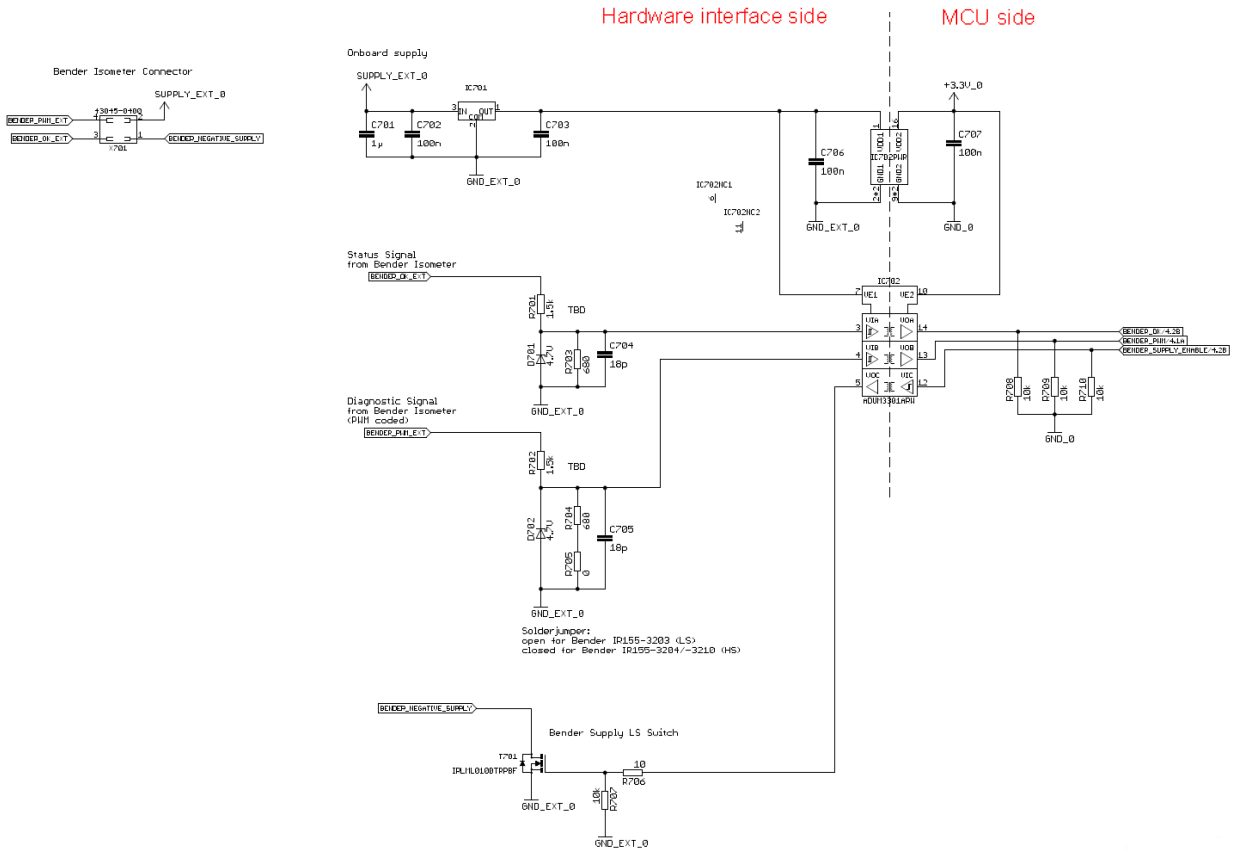



Fig. 17.8: Interface to the Bender ISOMETER

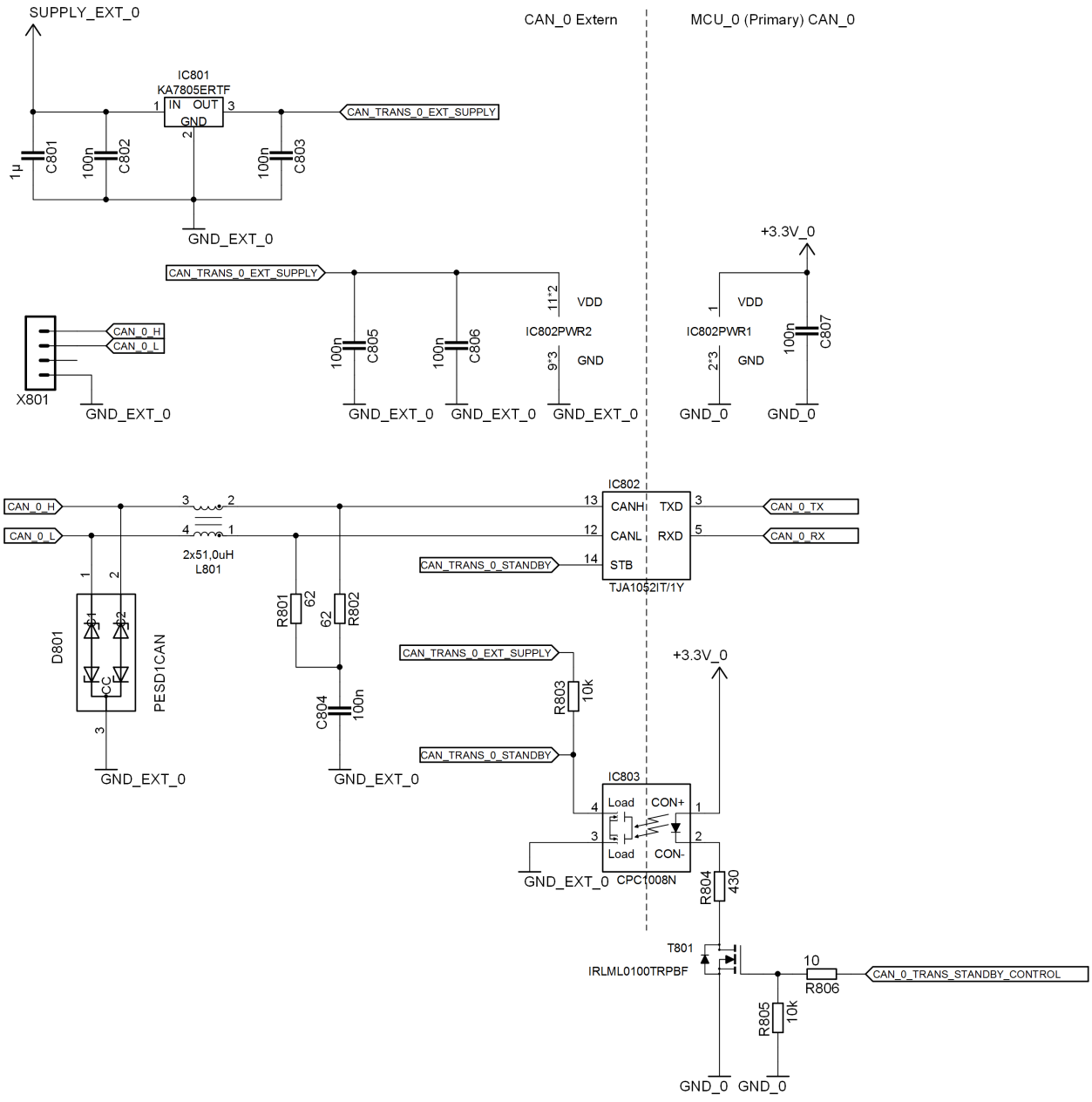



Fig. 17.9: Circuit of the CAN interface (CAN0)

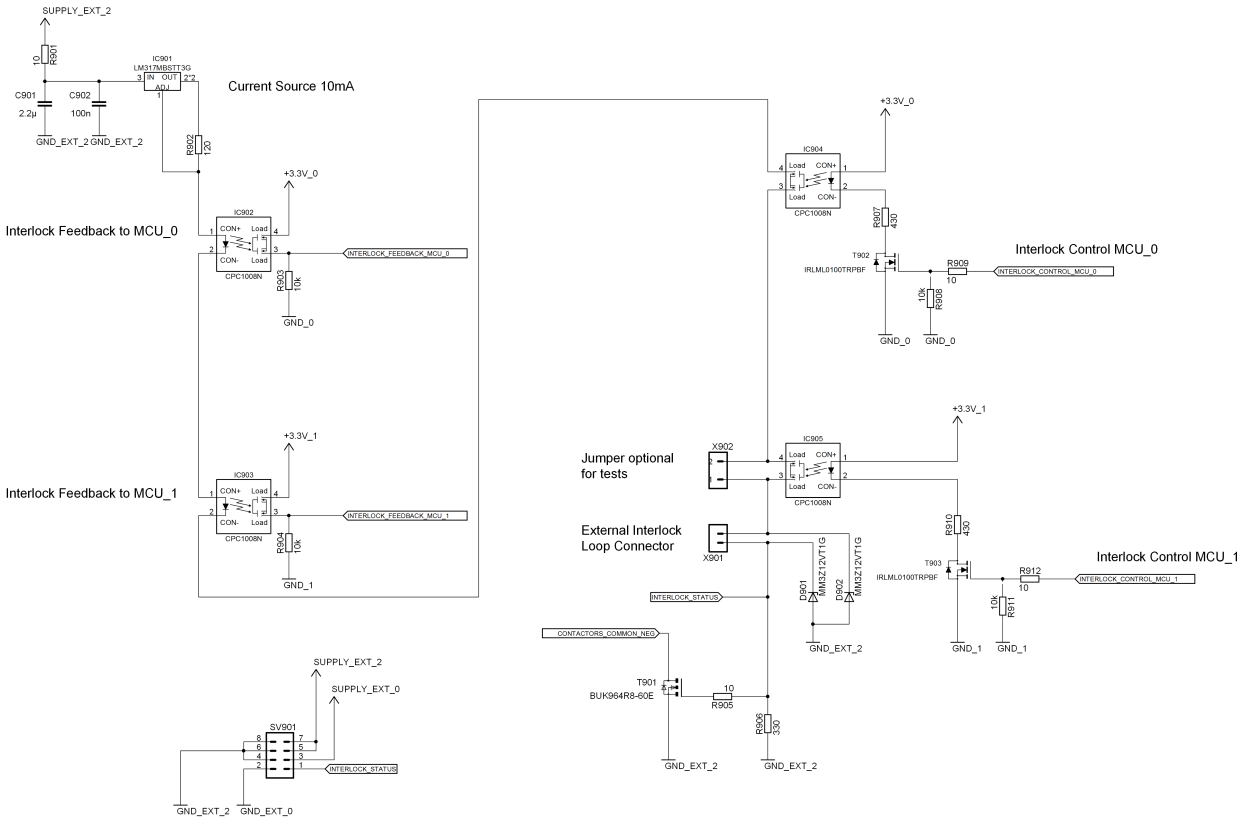
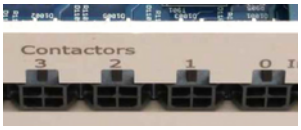



Fig. 17.10: Interlock circuit

and BMS-Extension Board). Therefore, when the interlock line is interrupted, the contactors are no longer supplied and open.

## 17.8 Contactors




| Pin | Signal | Direction | Description |
|---|---|---|---|
| 1 | GND_EXT0 | – | Contactor auxiliary contact |
| 2 | CONTACTOR_X_FEEDBACK_EXT | – | Contactor auxiliary contact |
| 3 | CONTACTOR_X_COIL_POS | Output | Positive contactor coil supply |
| 4 | CONTACTORS_COMMON_NEG | Output | Negative contactor coil supply |

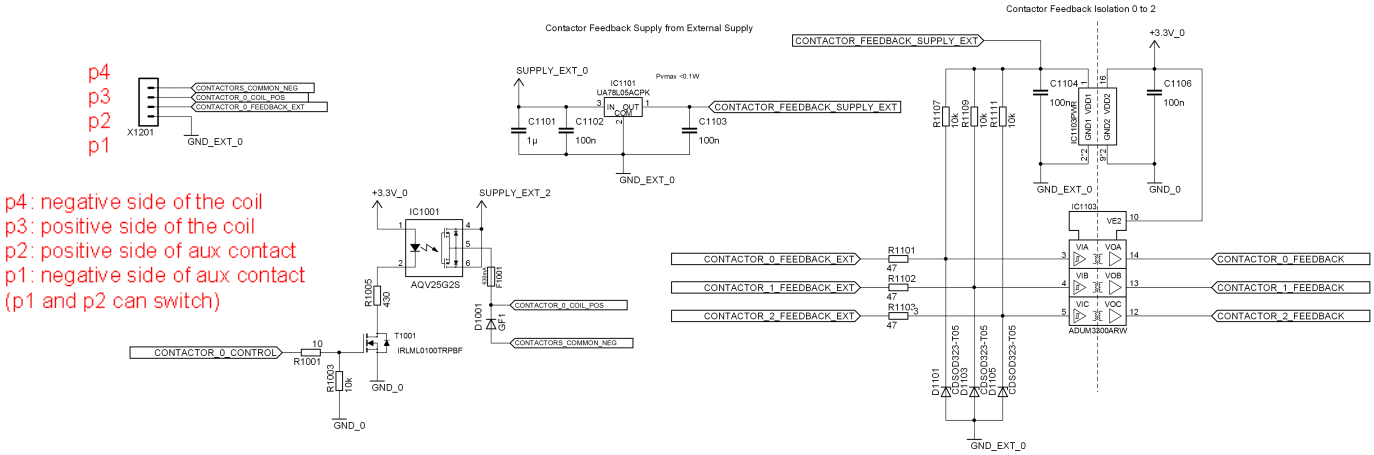



Fig. 17.11: Contactor circuit, exemplarily shown for contactor0

The foxBMS Master Unit can control up to 9 contactors: 6 on the BMS-Master Board and 3 on the BMS-Extension Board. The ~~according~~ control and feedback circuit is exemplarily shown for contactor 0 in Fig. 17.11. The contactor is switched on and off by an AQV25G2S photoMOS (IC1001) by the primary microcontroller MCU0. Every contactor channel is protected with slow blowing fuse (F1001) type Schurter UMT-250 630mA (3403.0164.xx). The free wheeling diode D1001 is not populated. It has to be inserted when using contactors ~~are used,~~ that do not provide an internal free wheeling diode. The contactor interface also supports a feedback functionality for contactors with auxiliary contacts. The contactor status can be read back by MCU0 via an ADUM3300 (IC1103).

## 17.9 Isolated USB interface (primary and secondary)

Both mircocontrollers MCU0 and MCU1 provide an isolated USB interface, as exemplarily shown for MCU0 in Fig. 17.12. A FT231XS-R interface IC (IC1402) converts the USB signal to UART, which can easily be interfaced by the microcontroller. The UART signals are isolated by an ADUM3401 isolation IC (IC1403). The USB interface can be used to flash the microcontroller and for communication.

## 17.10 EEPROM

The BMS-Master Board provides an 2MB EEPROM for data storage for the primary and secondary MCU (see Fig. 17.13). It uses an SPI interface, which is shared with the memory card, which is also connected to MCU0.

## 17.11 Isolated RS485 Interface

Will not be used.

On the BMS-Extension Board an isolated RS485 interface is provided. It can be used to communicate with the foxBMS Master Unit as an alternative to the CAN interface or the UART over USB interface. Moreover, via this interface, monitoring circuits (slaves) using RS485 instead of CAN or another proprietary communication protocol

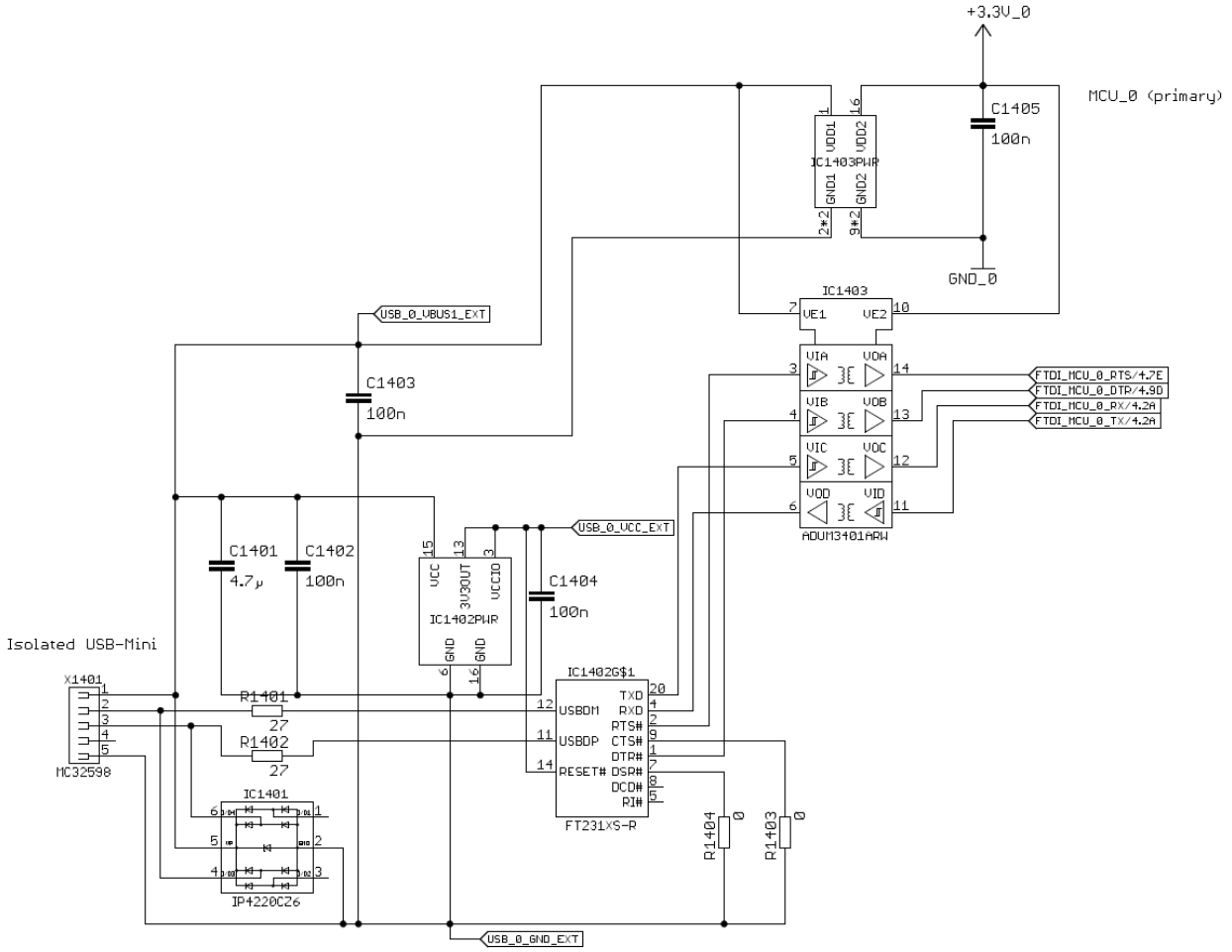



Fig. 17.12: USB interface circuit

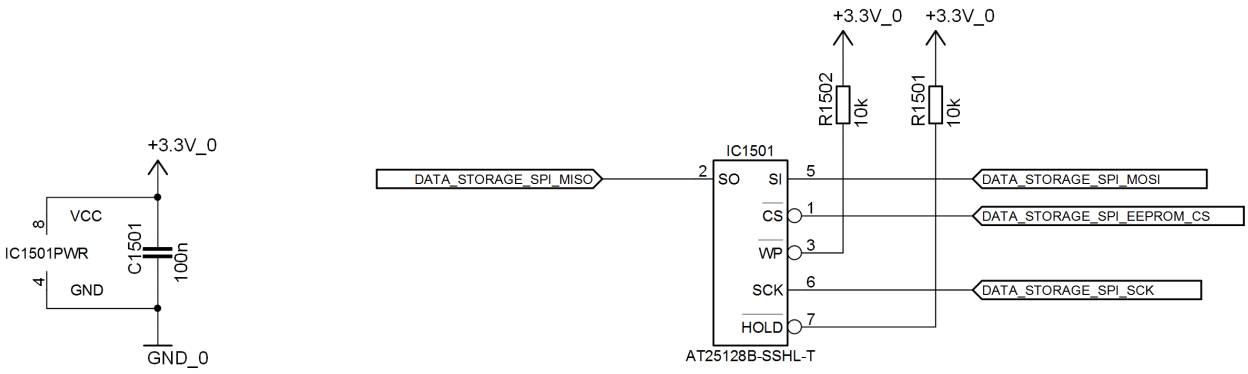



Fig. 17.13: EEPROM, exemplarily shown for the MCU0

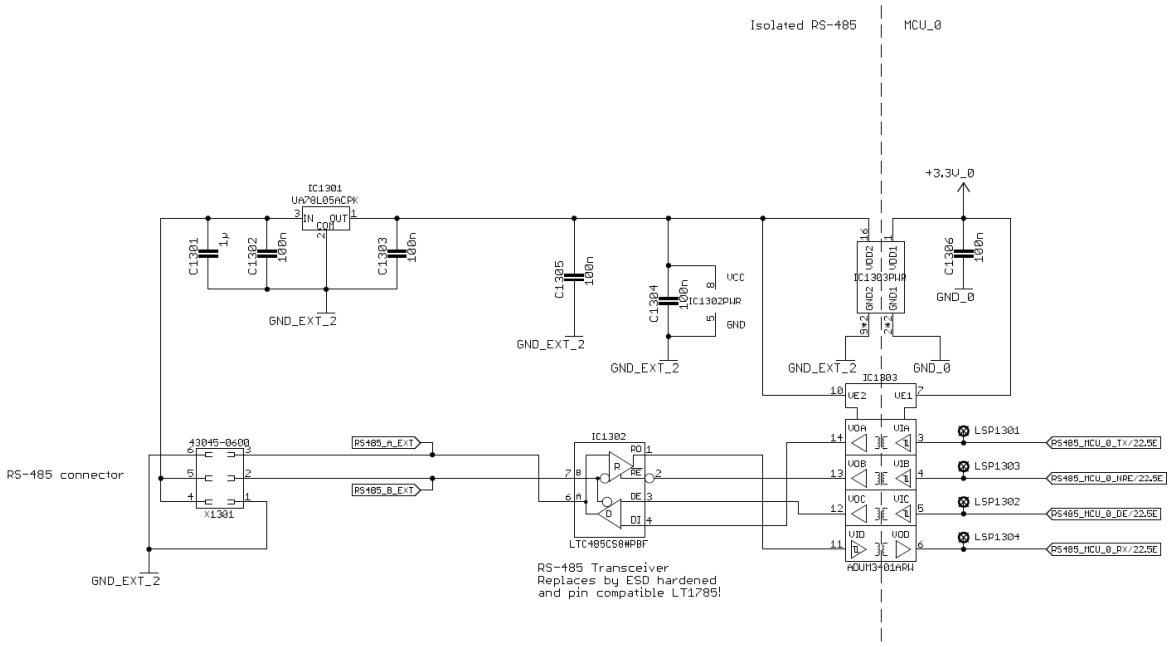



Fig. 17.14: Isolated RS485 interface circuit

can be connected to the foxBMS Master Unit. Fig. 17.14 shows the RS485 interface schematic. The external part of the circuit is supplied via a voltage applied to pins 5 (7V - 20V) and 6 (GND) of connector X1301. The external supply voltage is also available on pin 1 (GND) and pin 4. IC1301 provides 5V supply voltage for the transceiver (IC1302) and the external side of the isolator (IC1303). The transceiver (LT1785) features a receiver enable (!RE) and a driver enable (DE) functionality, which can be controlled by the primary microcontroller via the signals `RS485_MCU0_NRE` and `RS485_MCU0_DE` respectively. For data transmission the signals `RS485_MCU0_TX` and `RS485_MCU0_RX` are used. The data signals are available on connector X1301 pins 1 and 2. The data signals and the enable signals are galvanically isolated from the BMS-Master Board by an ADUM3401 isolator IC.

## 17.12 Isolated Normally Open Contacts (isoNOC)

Will not be used. The difference between these NOC and the contactors is that the latter have feedback.

The BMS-Extension Board provides 6 normally open contacts (`ISONOC0` to `ISONOC5`) for multi-purpose use. Their function is exemplarily described for ISONOC channel0 ( shown in Fig. 17.15). Isolation and switching functionality are realized by AQV25G2S photoMOS (IC2001). The photoMOS are controlled by a MOSFET (T2001), which again is switched by the primary microcontroller (`ISONOC0_CONTROL`). The photoMOS is configured for a maximum load current of 6A at 50V. Diode D1002 is optionally and not populated by default. Both power terminals of the photoMOS are available on connector X2001 as `ISONOC0_POSITIVE` and `ISONOC0_POSITIVE` on consecutive pins 1 and 2.

## 17.13 Analog Inputs

Will not be used.

For the acquisition of analog data, there are 5 ADC channels (`ANALOG_IN_CH0` - `ANALOG_IN_CH4`) available on BMS-Extension Board board. Fig. 17.16 shows the input circuit for channel 0. The analog input of the microcontroller (`ADC_MCU0_CH0`) is protected by diode D1701, which clamps the input voltage to 3.3V. By default R1701 is shorted with jumper, while R1702 is 7.75kOhm and C1701 is 100nF. The analog input channels are available on connector X1701. A reference voltage of 2.5V is provided by IC1701 (ADR3425), which can supply a total load current up to

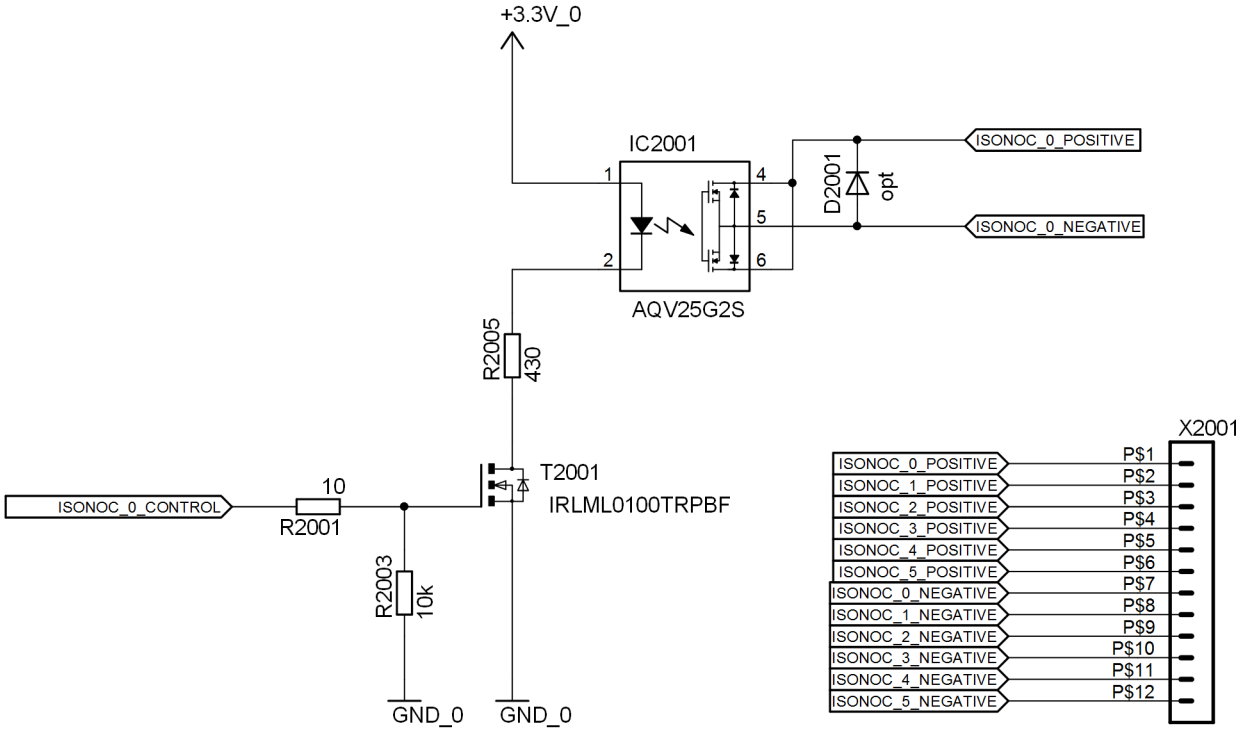



Fig. 17.15: Isolated normally open contacts (`ISONOC0` examplarily)

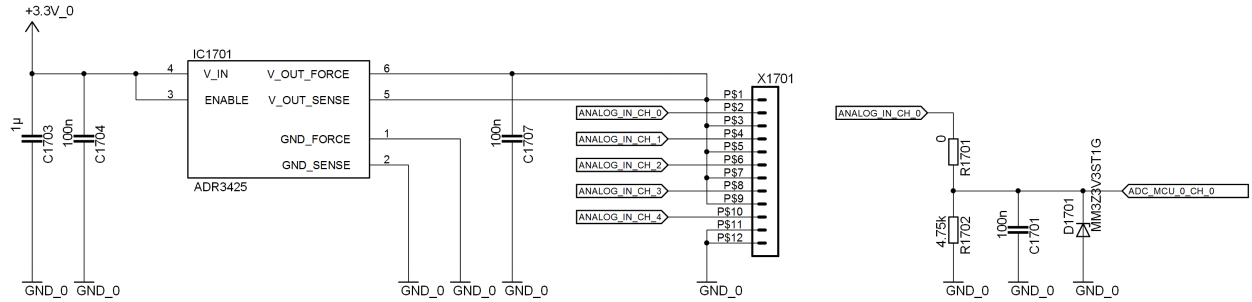



Fig. 17.16: Non isolated analog inputs (analog channel 0 exemplarily)

+10mA and sink up 3mA. The reference voltage is available in X1701 next to every analog input pin. Pin 11 and 12 are connected to GND.

**Note:** The analog inputs are not isolated. They are referenced to the same potential as the primary microcontroller.

## 17.14 Isolated GPIO

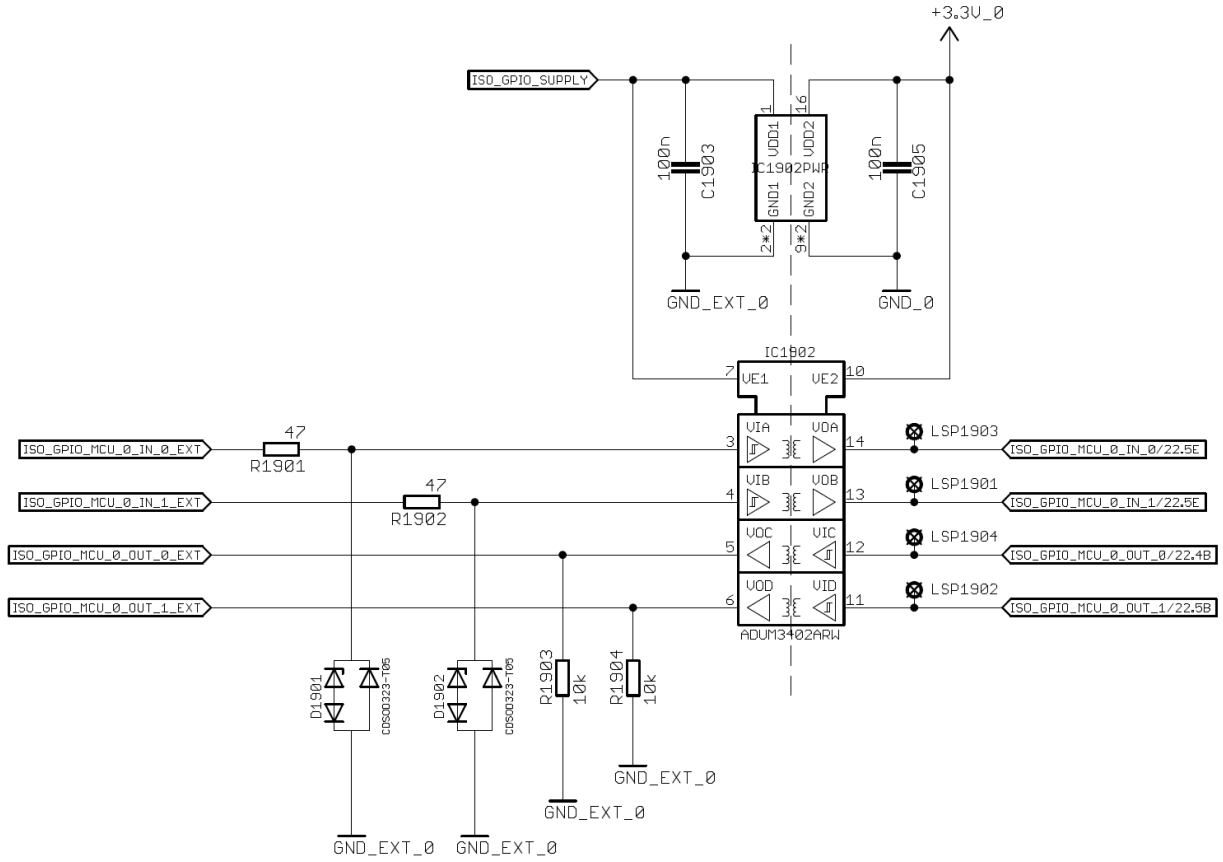



Fig. 17.17: Isolated GPIOs (Input 0 and 1; Output 0 and 1 shown exemplarily)

The BMS-Extension Board provides 4 isolated inputs and 4 isolated outputs for general purpose (shown in Fig. 17.17 ). Two ADUM3402 (IC1902 and IC1903) are used for isolation. Their external side of is supplied by `SUPPLX_EXT0` via a 78L05F linear voltage regulator (IC1901). The inputs are equipped with a 10kOhm pull down resistor. All isolated GPIOs are available on the connector X1901 pins 1 to 8. Pins 9 and 10 of X1901 are connected to `GND_EXT0`.

(They are drawn in the output pins.)

## 17.15 Memory Card

On the BMS-Extension Board also a memory card slot can be found. It is directly connected to the Data Storage SPI of the primary microcontroller. Fig. 17.18 shows the schematic. Via the signal `CARD_SUPPLY_CONTROL` (primary microcontroller) the supply voltage of the memory card can be switched on and off.

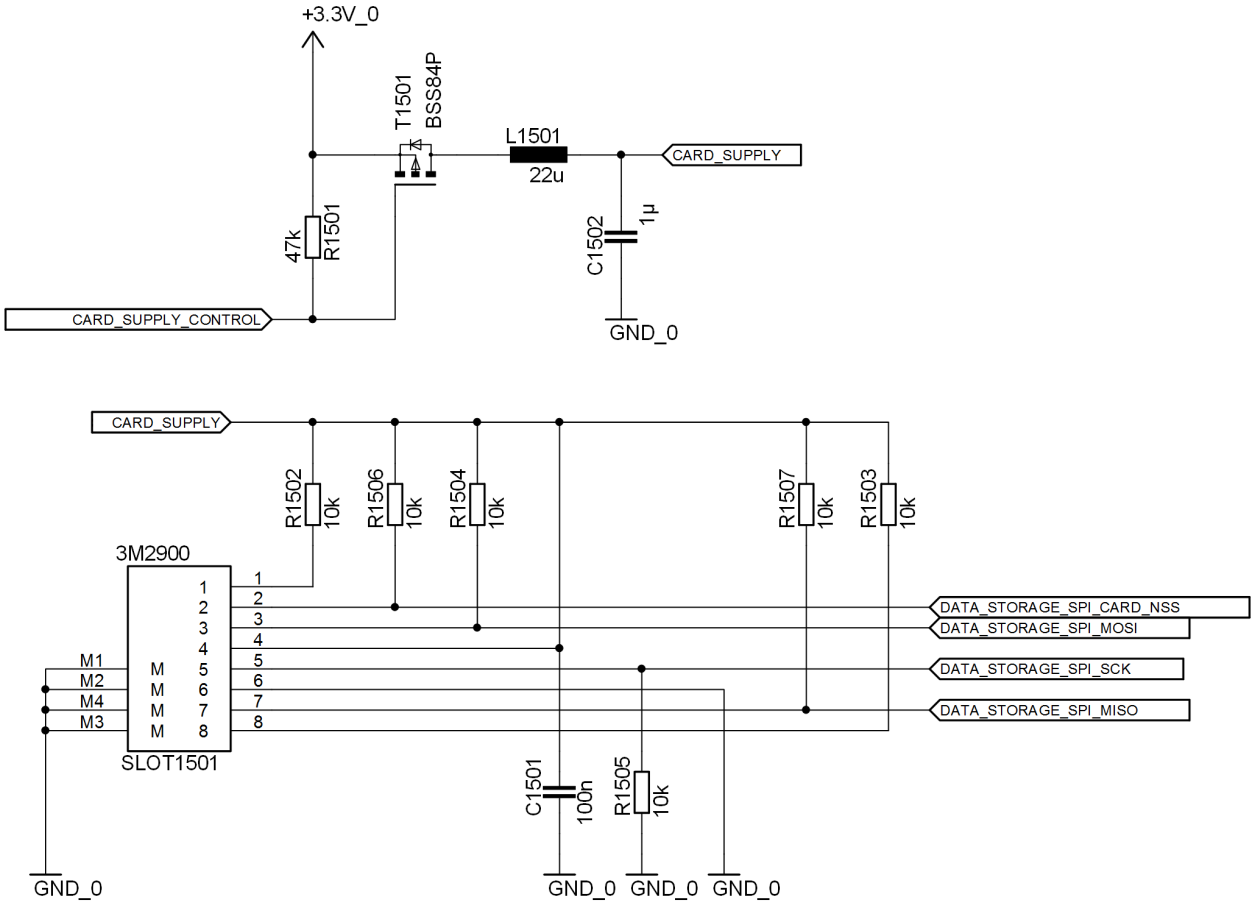



Fig. 17.18: Memory card