**Documentation of MAL**

**Introduction**

MAL is a software component which was created to abstract the MAVLink interface and to integrate it into the APOLI project. Various abstractions, so-called Vehicles, are provided. These are the CTHVehicle, the EXSVehicle and the TestVehicle. They can be defined using runtime arguments. The MAL is buildable and fully functional under Windows as well as Linux. Further vehicles can be added in the future.

**Code Structure**

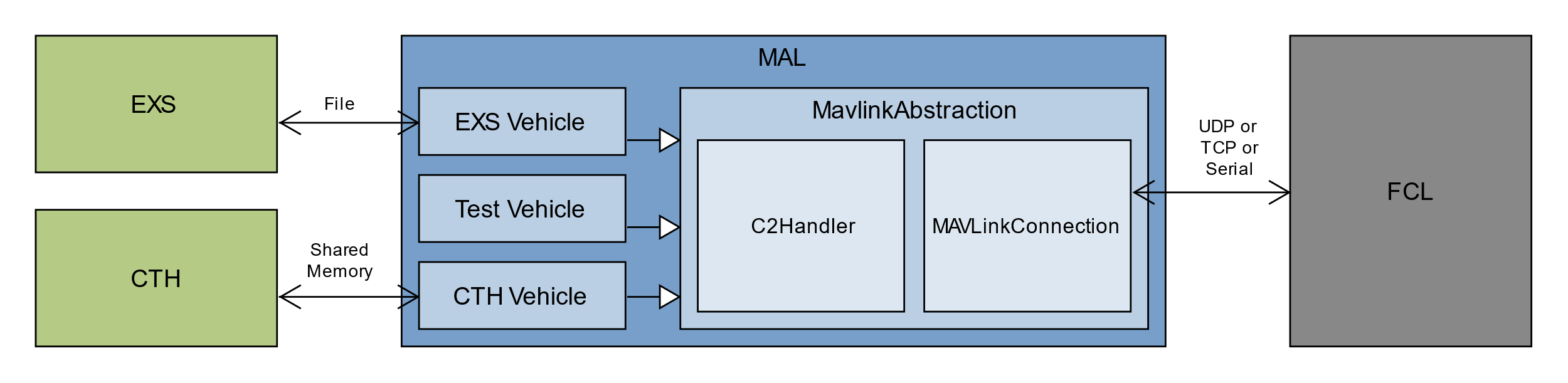


Fig 1: MAL architecture (created by Marco Stephan)

The MAL consists of the actual MAVLink abstraction and the vehicles for the individual use cases. As shown above, the abstraction consists of the MAVLinkConnection, which handles the data transfer with the MAVLink device. For this, a UDP, TCP or serial connection can be used. For this purpose, the respective IPC abstractions are inherent. The C2Handler is used to handle the requests to the MAVLink device and incoming responses from it and to translate the response the correct format.

The EXS, CTH and the Test Vehicle serve different purposes. The EXS Vehicle is used to communicate with the Expert System directly. The CTH Vehicle to communicate with the CTH. Note that currently only a raw idea of this communication is implemented. The Test Vehicle is used for testing and receives its input directly via the terminal. The vehicle abstractions consist of the IPC or UI and the corresponding commands. These are then executed by MAVLink abstraction.

**User Manual**

**Start and use**

As soon as the binary is available no further dependencies have to be considered. You can start the program without runtime arguments to get a list of all runtime arguments. Each runtime argument has a long form represented by *"--longValue"* and a short form represented by *"-shortValue"*.

The necessary argument *--mavconntype / -m* is used to set the type of the connection to the MAVLink device. A following argument has to be given. *"udp"* establishes a UDP connection. For this, the argument *--port/ -p* is needed, which can take a value between 1 and 65535. *"tcp"* is used to establish a TCP connection, for this the argument *--port/ -p* is required which takes a value between 1 and 65535. *"serial"* establishes a connection via a serial interface to the MAVLink device. For this, the arguments *--port / -p* and *--baudrate/ -b* are needed. The port provides the name of the serial interface and baud rate the baud rate of the port.

Furthermore, there is the optional parameter *--systemid / -s*. This parameter specifies the own MAVLink System ID, which must be between 1 and 255. The default value is 17.

Additionally, *--targetid / -t* specifies the target MAVLink System ID, which must be between 1 and 255. The default value is 1.

Moreover, there is the optional parameter *--log / -l*. This parameter specifies if the system should create a log file. The following parameters are available for selection. *"none"* to not create a log file. *"error"* to write error messages to the log file. *"status"* to write error and status messages to the log file. *“info”* to write everything into the log file. The default value is *"none"*.

Furthermore, there is the optional parameter *--outdoor/-o*. This parameter scales the missions’ movement by a constant factor.

*--vehicle / -v* specifies the vehicle to launch. The value given can either be *“test”*, *“exs”* or *“cth”*. The default value is “test”.

**TestVehicle**

The Test Vehicle is used to test single MAVLink commands or to perform a predefined sequence of commands. The input is done via the terminal. If a wrong input is registered, all possible options are displayed with description for help. The following commands can be executed. Some commands require additional input for parameters. These are requested in the terminal as well. The Test Vehicle is started using the vehicle value *test*.

|  |  |
| --- | --- |
| Parameters | Description |
| altHold | Changes the flight mode to ALTHOLD |
| arm | Arms the MAVLink device |
| beep | Plays a sound |
| disarm | Disarms the MAVLink device |
| guided | Changes the flight mode to GUIDED |
| kill | Disarms the MAVLink device by force |
| land | Changes the flight mode to LAND |
| mission1 | Performs a take-off and lands |
| mission2 | Performs a takeoff, descends, ascends and lands |
| mission3 | Performs a take-off, moves forward and lands |
| mission4 | Performs a take-off, moves up, forward and lands |
| mission5 | Performs a take-off, rotates right, rotates left, rotates right, ascends, rotates right, rotates left, rotates right and lands |
| mission6 | Performs a take-off, moves right, moves left, moves right, ascends, moves right, moves left, moves right and lands. |
| mission7 | Performs a take-off, moves right, rotates, moves left, rotates, moves right, ascends moves right, rotates, moves left, rotates, moves right and lands. |
| mission8 | Performs a takeoff, moves in a triangle and lands |
| moveBy | Moves using an offset |
| moveTo | Moves to a GPS position |
| posHold | Changes the flight mode to POSHOLD |
| printDeviceData | Outputs current data of the MAVLink device |
| rtl | Changes the flight mode to RTL (return to launch) |
| rebootFlightController | Reboots the flight controller |
| stabilize | Changes the flight mode to STABILIZE |
| stop | Terminates the Test Vehicle |
| takeOff | Performs a take-off |

**EXSVehicle**

The EXS vehicle communicates with the Expert System by reading and writing "facts". The currently implemented facts are given in the following table. The EXS vehicle is started using the vehicle value *exs*.

|  |  |
| --- | --- |
| Commands | Description |
| arm | Arms the MAVLink device |
| disarm | Disarms the MAVLink device |
| guided | Changes the flight mode to GUIDED |
| land | Changes the flight mode to LAND |
| takeOff | Performs a take-off |

**CTHVehicle**

The CTH vehicle communicates with the CTH using shared memory, and currently has the following commands. The CTH Vehicle can be started using the vehicle value *cth*.

|  |  |
| --- | --- |
| Commands | Description |
| arm | Arms the MAVLink device |
| disarm | Disarms the MAVLink device |
| guided | Changes the flight mode to GUIDED |
| land | Changes the flight mode to LAND |
| takeOff | Performs a take-off |

**Dependencies**

**Dependencies for executing the binary**

The binary file has no dependencies, it can be executed on Windows and Linux

**Dependencies for building**

A Visual Studio solution is available. It was created using Visual Studio 16.5.1. The solution contains the build structures for the MAL and the unit tests for both Windows and Linux. For Windows no further steps are necessary. For Linux compilation, the package for cross-platform C++ development must be installed. With this, an SSH connection is established to the target system on which the compilation is to be carried out. Under Linux, linking with the "rt" and "pthread" libraries is also necessary.

**Bug report**

No known bugs