

# OEM

ArduPilot OEM, , :

- , gimbals, , .  
Mission Planner MAVProxy" ,  
/IMU/ RC .
- ROM Lua , SD . , ArduPilot  
Lua .
- 
- 
- APJ .( 4.5 ,  
)
- [LUA](#) MAVLink

## Customization Steps

This section assumes that the OEM has set up the build environment (**building-the-code**) and cloned the ArduPilot GitHub repo locally (**where-to-get-the-code**), to build its customized version of the firmware.

1. Create a branch with the version of firmware you wish to base the customization upon. This will usually be the current stable version. To do this for ArduPlane Stable, for example, assuming you are already in the ArduPilot directory on your PC:

“

```
git fetch https://github.com/ArduPilot/ardupilot.git <version>
      where <version> is the tag for the stable version for the
      desired vehicle: ArduPlane-stable, ArduCopter-stable,
      APMrover2-stable,etc.

git checkout -b <your branch name> FETCH_HEAD
git submodule update --init --recursive
```

”

2. In the `libraries/AP_HAL_ChibiOS/hwdef` directory, create a new subdirectory for your customized board definitions. In this example, the directory will be named `OEM_CubeOrange` to create a derivative for that board.

3. Create a new `hwdef.dat` file in this format. In this case, our ready-to-fly plane will be using a CubeOrange autopilot and only requires a single line:

```
“  
include ../CubeOrange/hwdef.dat
```

4. The firmware name can be customized by adding one line to the `hwdef.dat` file.

```
“  
define AP_CUSTOM_FIRMWARE_STRING "MyMagicFrame"
```

□ □ □

Custom frame type strings can be created through Lua the scripting method `motors:set_frame_string("Custom frame name")`.

5. Now in that same directory, copy the base board's `hwdef-bl.dat` bootloader file, and then include a file named `defaults.parm`. This file will be the parameter overrides of the standard defaults to match your system's configuration. Things like output function assignments, auxiliary RC switches, flight and tuning parameters, etc.

□ □ □

a parameter can be made readonly (ie user cannot change its value) by marking it `@READONLY` in the file

□ □ □

The `defaults.parm` file should be as small as possible. Some boards only allow 1024 bytes total for this file. Every ASCII byte in the file counts against this limit (except for comment lines). Use integer values where possible. Below is a simple example. Serial port protocols, baud rate, and options defaults can be set directly in the hwdef, as well as NTF\_LED\_TYPES, and battery monitor defaults, and should be done there instead of a defaults file.

```
# setup for NTF LEDs on output5  
SERVO5_FUNCTION 120  
NTF_LED_TYPES 256
```

1. You can also embed **Lua scripts** in the ROM of the chip that will automatically run. Since Lua is currently only run on autopilots with a lot of flash space, they are only restricted in

.lua.

Alternatively, you can specify the LUA applet or driver at compile by using the "--embed-<filename>" command, for example:

```
./waf plane --embed-LTE_modem
```



The user may also run Lua scripts off the SD card, so care should be taken in naming the embedded script file names to not conflict with potential user file names. It is recommended that the file names of embedded Lua scripts be provided in the product documentation for the user.

1. You can also embed small pieces of documentation in the ROM of the chip that are readable when examining the @ROMFS folder via MAVFtp. These can be pictures or small informational documents. These must fit within the free flash space of the autopilot. These files can be located in sub-directories in `libraries/AP_HAL_ChibiOS/hwdef/OEM_CubeOrange` (e.g. `libraries/AP_HAL_ChibiOS/hwdef/OEM_CubeOrange/AircraftManual` ).
2. Now build as normal with OEM-CubeOrange as the board name in the configuration. The default parameters, Lua scripts, and the custom firmware name will be embedded appropriately.

# Alternative To Customizing hwdef.dat

Instead of creating a separate branch and modifying the hwdef file, you can also insert Lua scripts or even informational files into the ROMFS of the build. Simply go to the top level of your ArduPilot git checkout and create a sub-folder named "ROMFS\_custom" in the root of the source tree. Place your LUA scripts in a sub-folder in this directory, named "scripts" (i.e. the path `ardupilot/ROMFS_custom/scripts`). You can have other sub-folders for informational files and these will be included and viewable when examining the @ROMFS folder with MAVFtp.

```
[copywiki destination="plane,copter,rover,sub,dev"]
```

- ArduPilot #2
- Admin

2026-10-09 14:41:15 UTC
- Admin

2026-10-10 01:46:01 UTC