



Experiment MOTORE++ D1.1 new Device Design

MOTORE++: A new Rehabilitation Robot for the upper limb: refinement and experimental trials

Version 1
Submission date: 17.09.2015

Date	Name	Changes/Comments
17.09.2015		

1 Publishable Summary

The goal is to develop a rehabilitation robot named MOTORE++ aimed to restore upper limb functionality in patients with neurological diseases and to assess his performance. This is a new haptic portable device, the first suitable for home based rehabilitation. Starting from a prototype developed in the last years, the project aims at delivering a small omnidirectional robot moving on *transwheel*, interacting with a patient providing assistance and force feedback during rehabilitation sessions. The software will allow to select among several exercises. Biomechanical studies of the interaction with the robot and on the arm impedance during exercises will be part of the Echord experiment.

During the first six months the device has been completely redesigned in order to achieve better performance and be ready for a CE certification process and an easy maintenance.

2 Redesign of the robot

As mentioned above, during the first six months the robot underwent through a redesign process in order to. endow the robot with: (1) a new motherboard and a new CPU to achieve better performance, (2) a SD card and an USB port for data transfer, firmware upgrade and debugging, (3) a Wi Fi module replacing the Bluetooth module for a more stable connection to the PC, (4) a coulomb counter in the battery pack, (5) a new sump suitable for CE the safety issues related to the CE certification and an easy maintenance of the device.

From a mechanical point of view the changes can be seen in the pictures below



Fig.1 the old design of the robot

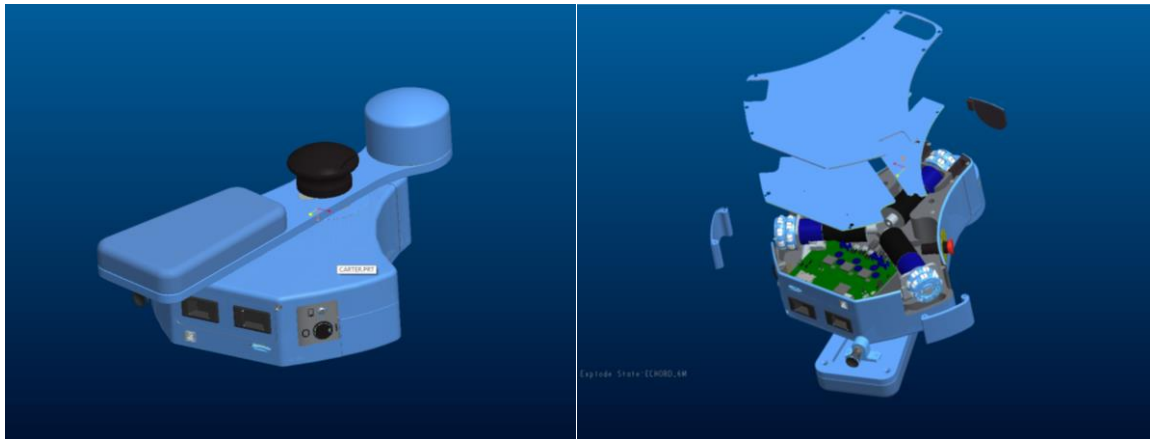


Fig.2 the new design of the robot

The new electronic board has been moved and it is now protected by means of an inner cover, as well as all the cabling. The wheels can be easier changed removing the external cover and the wheel doors. Also the battery pack has been redesigned and it is has more safe spring loaded contacts. The interior of the sump has been modified in order to support all the new component mentioned above.

The electronics of the system has been completely redesigned in order to add new sensors and IO devices, improve performances, reduce electrical noise, and rationalize the wiring. The new electronics consists of two boards: the main board or motherboard and the Wi Fi board. In the reporting period the two boards have been designed, the components selected and the schematics completed: the routing and the realization of PCBs is ongoing.

The new motherboard is based on the more powerful STM32F746ZGT6 microcontroller (ARM Cortex-M7, 216Mhz, RAM 320KB, Flash 1024KB, 144 pin) by ST Microelectronics and includes, as additional devices with respect to the old motherboard, a SD card for saving exercise data and USB for data transfer. Motor drivers (now VNH5019A) and accelerometer (now LIS302DL) have been updated. Additional current sensor for measuring motor currents and resistors on the motor control signals to avoid ringing and reduce electrical noise have been added. The Anoto digital pen, which was Bluetooth connected in the previous version, is now directly connected on a serial port. Furthermore, the electronics related to robot sensors and devices such as encoders, load cell, led, emergency buttons, etc. has been revised and updated in order improve robustness. Finally, the Wi Fi module has been separated from the motherboard in order to be mounted out of the robot metallic case and to make the system modular keeping the possibility of using Bluetooth for wireless communication. The schematic of new motherboard and Wi Fi board is shown in the following pictures.

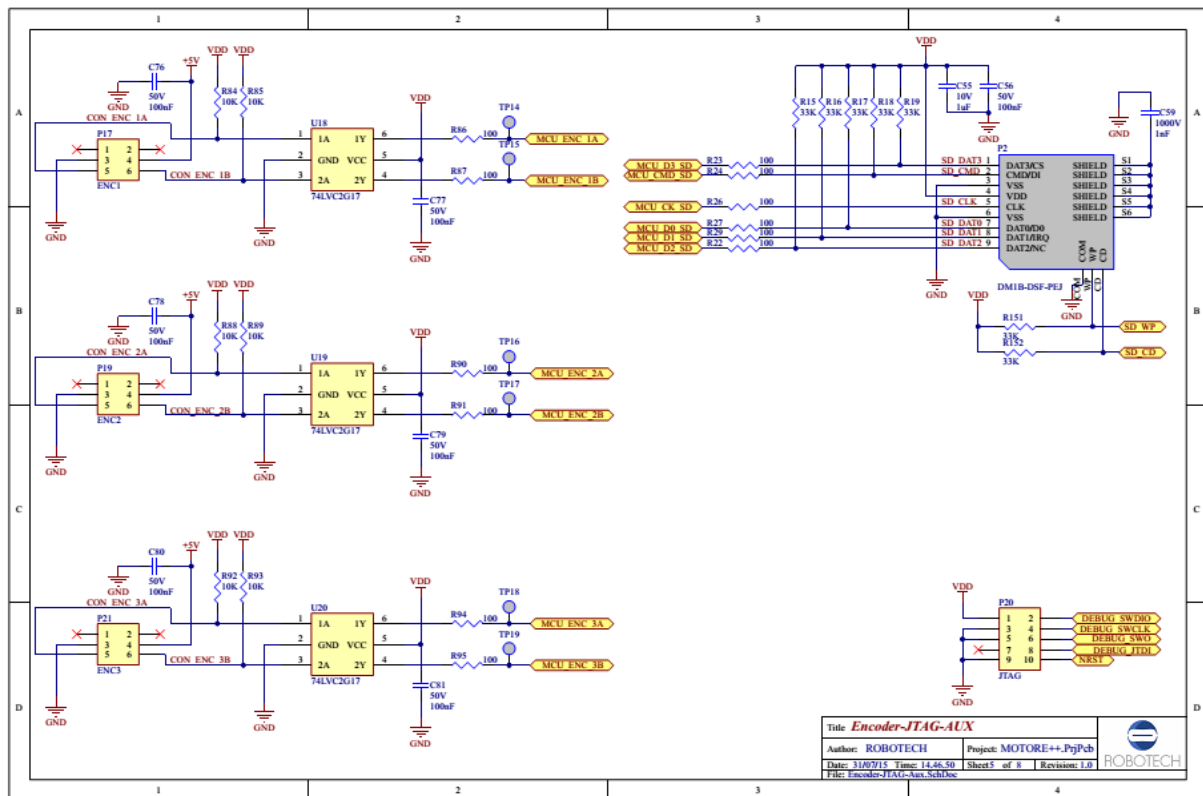


Fig.3 the new electronic motherboard

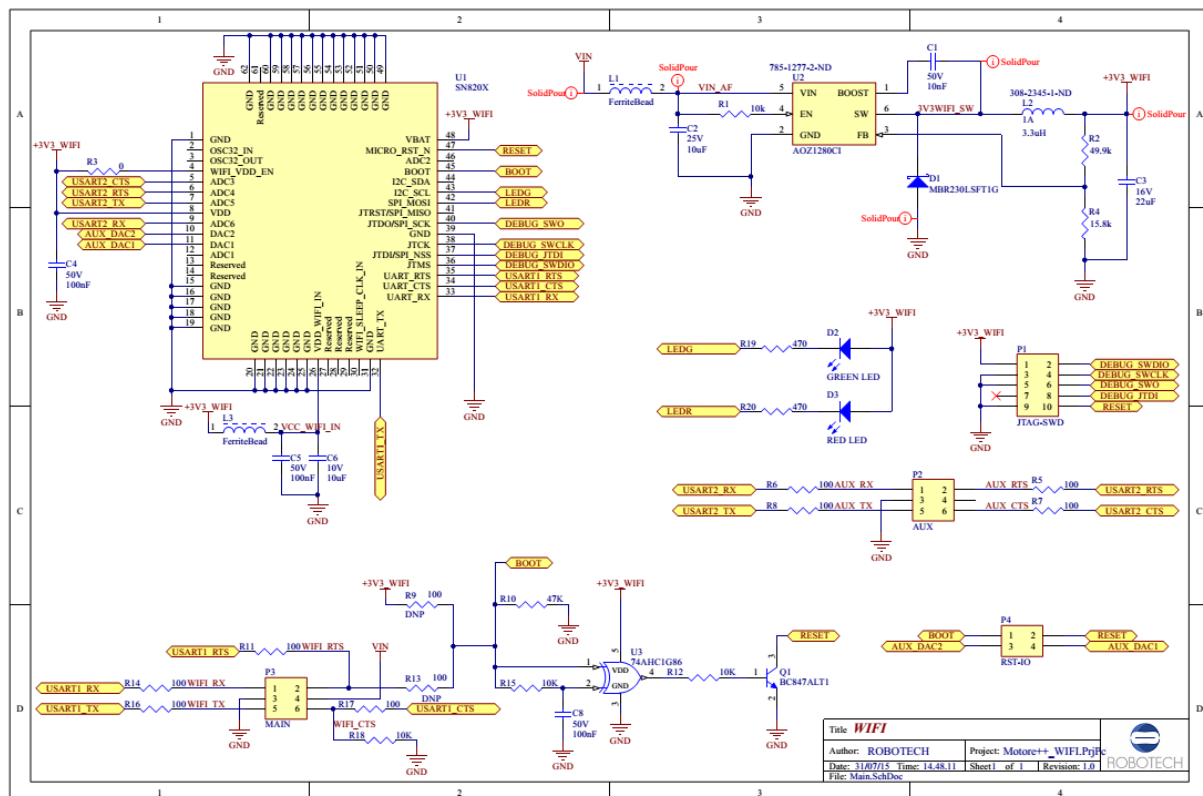


Fig.4 the new wi-fi PCB

A new architecture based on the emerging Cortex M4F M7F has been defined. These microcontrollers offer on the side of a standard ARM architecture, better performances, full implemented floating point, portability and free development tool-chains, and access to byte, half-word and word in a natural fashion.

The implementation of a driver strategy based on DMA support allow the present architecture to overcome the existing message limitation based on the natural FIFO of the standard TMS320 architecture used for motore 1.

The pinout and functionalities of the novel architecture has been defined. MOTORE++ will use a 144 pin device and the electronic will be designed by Robotech s.r.l while SSSA/FABRICA will develop all the onboard firmware.

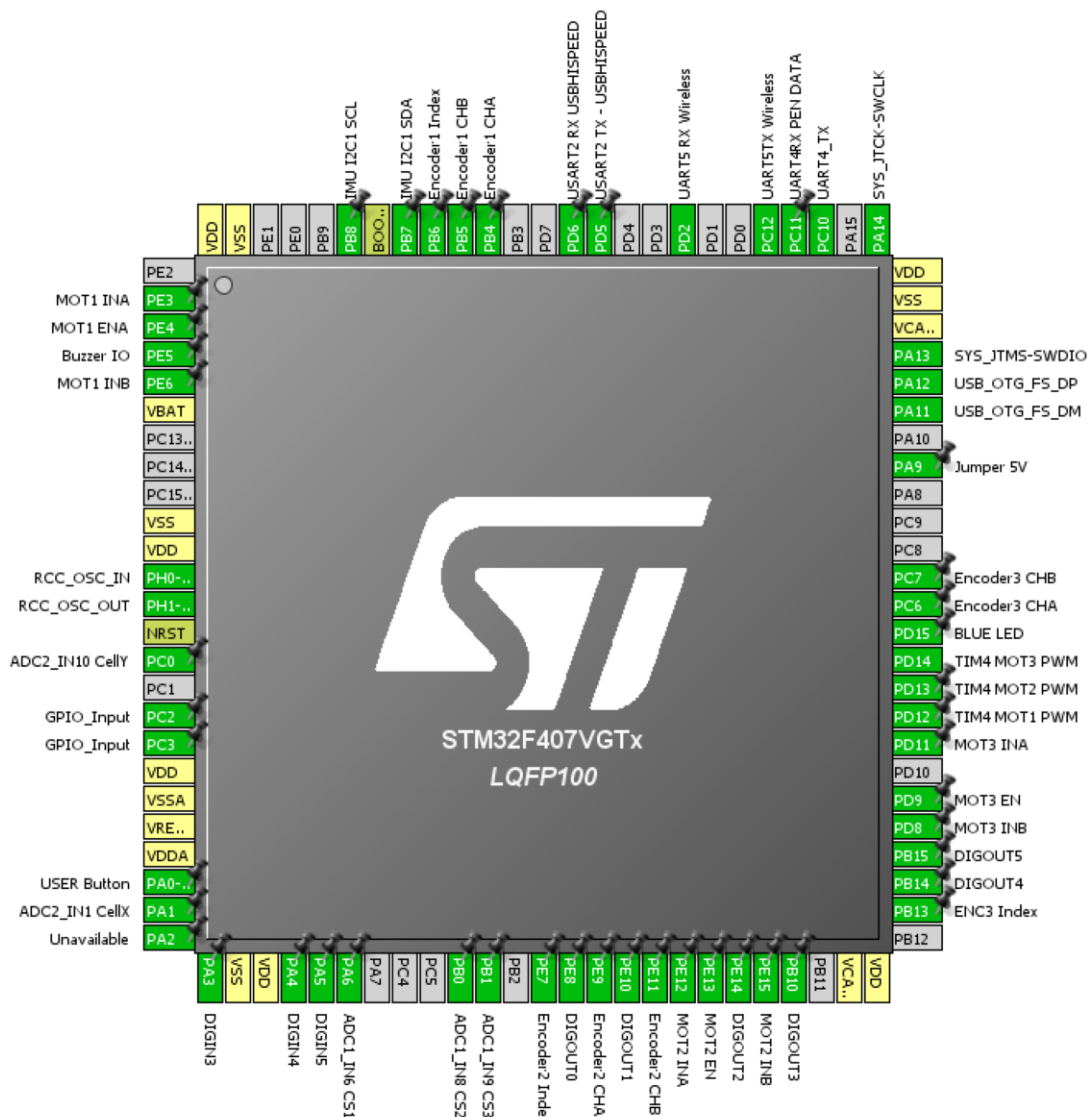


Figure 5: Pinout configuration

The above picture summarizes the connections that will be implemented and agreed for the target microcontroller. These implementation so far have been tested on an indoor device (cortex M4F STM32F407 who is available on an internal board at PERCRO).

The following devices have been programmed:

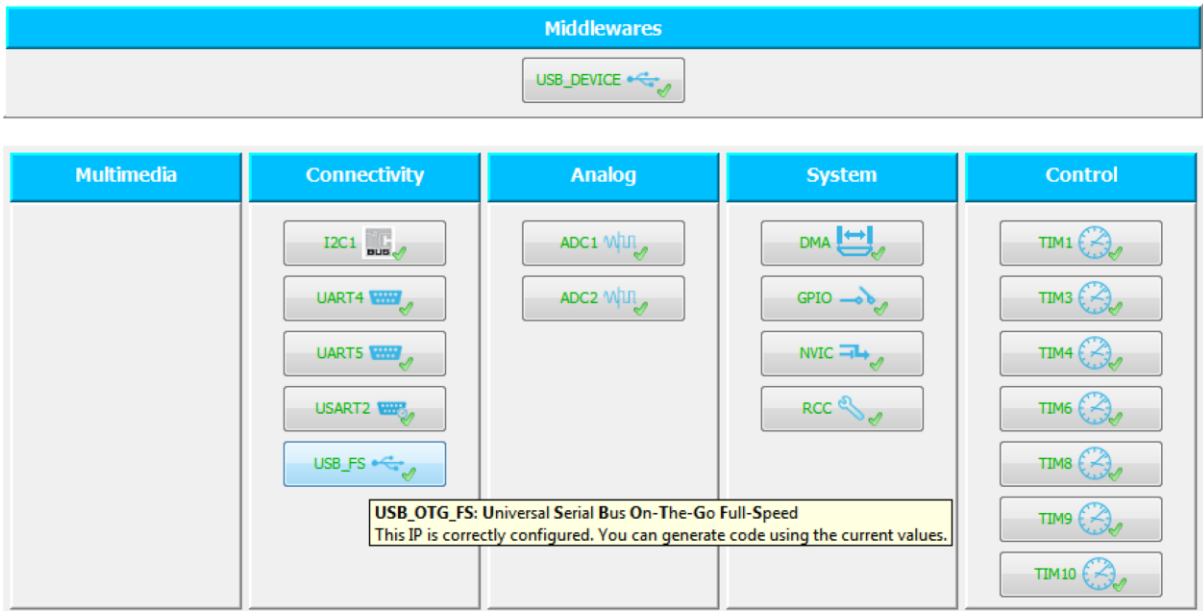


Figure 5: Programmed Devices

Three timers will be used for ENCODER detection (MOTORE++ will have three hardware encoder decoders); one timer for PWM generation; one timer for scheduler procedures; one timer for high frequency current control; One timer for ADC synchronization; I2C1 will be used to decode accelerometer signal; three (two on the first prototype) usart port will be connected as wireless link (wireless backup link) and PEN link; USB FS will be used as a backup channel for debug; two ADC converter allow to have separate conversion frequencies for high speed motors and force detection and low speed battery monitors.

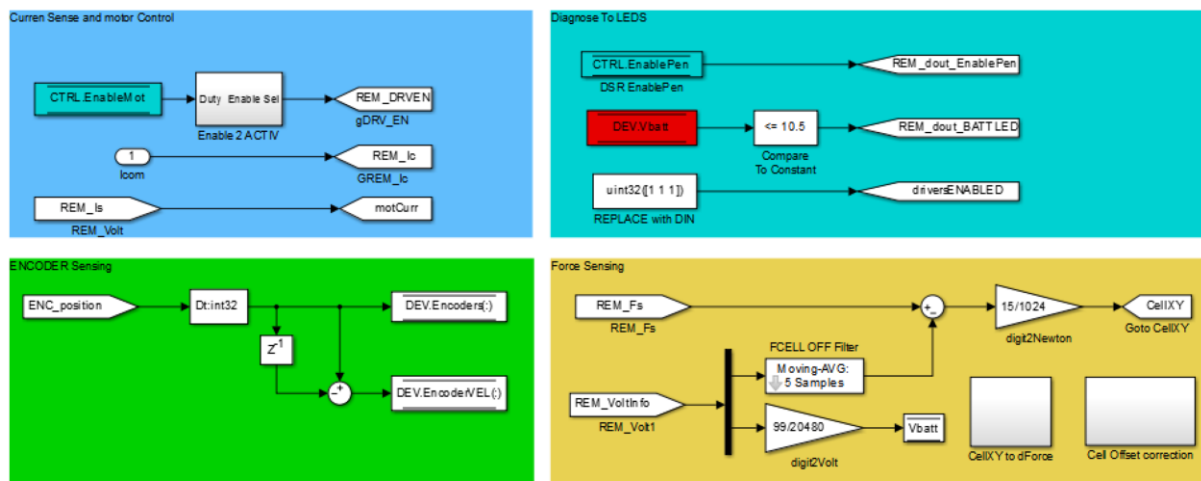


Figure 6: Low Level Hw Io In Simulink (Embeds User Code)

The preliminary tests have been carried out on the following internal board being developed for another laboratory project (REMEDI).

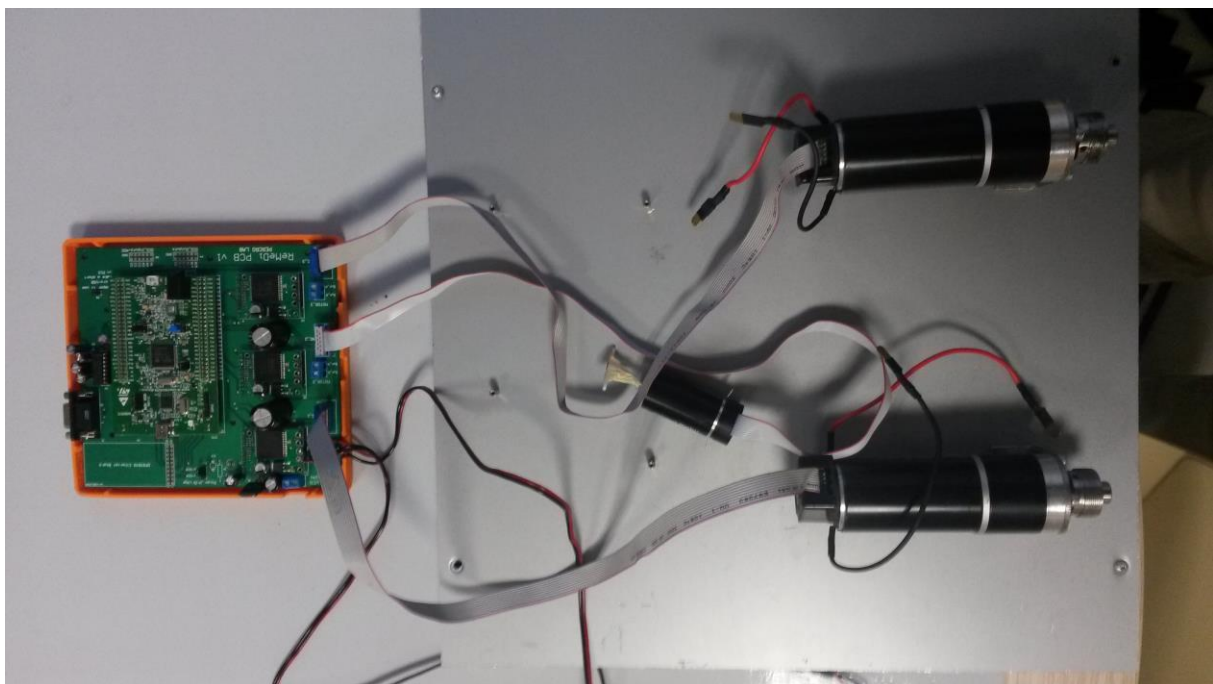


Figure 7: PERCRO Board And 1st TEST Configuration (Encoders Only)

The REMEDI project has several common features with the proposed target (three motor control, serial and wireless communication) and the board has allowed to accelerate the design process by connecting the old motore setup to the existing board. With this setup several activities have been accelerated. Among these:

- Implementing a new development environment based on the previous laboratory development experience. Using Matlab Toolchain a completely new development environment has been developed. This new toolchain allows to design develop and test embedded controllers based on standard matlab tools without any other paid software. In particular the proposed architecture (inherited from the developments within the REMEDI projects) allows the following benefits:
- Using the GCC-ARM toolchain;
- Using the novel STM_HAL libraries;
- Merging with the STM32Cube design product;
- Support any CortexM processor from STM (including the forthcoming F7)
- Joining automatically the CUBE project with matlab schematics;
- Allow realtime processor in the loop tuning as well simultaneous classic debugging (using ColDE development toolchain);

Using the PERCRO@internal board a preliminary control scheme has been developed.

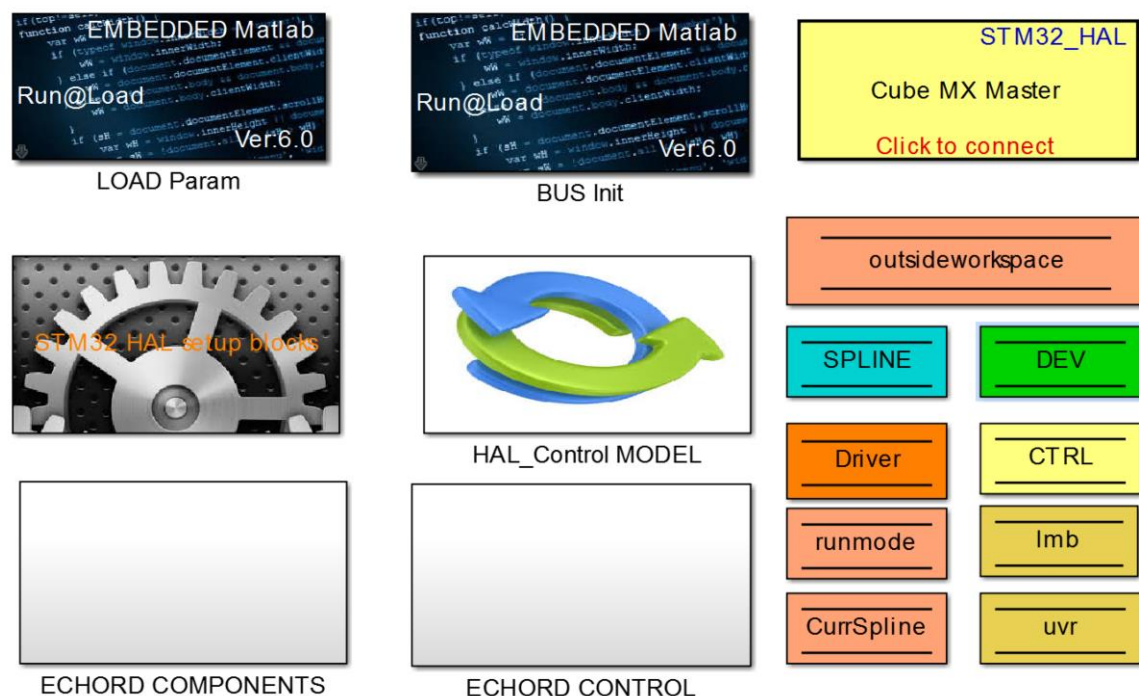
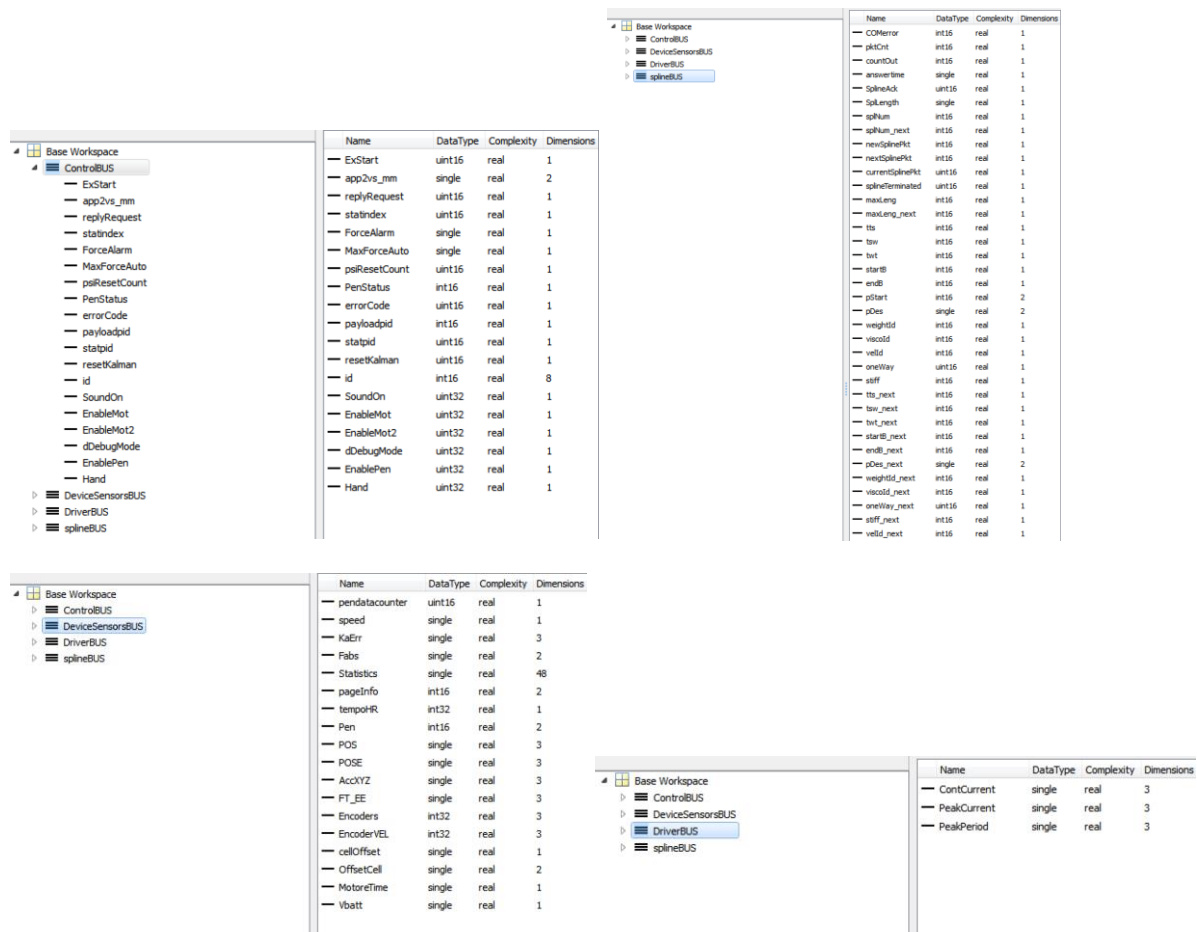


Figure 8: Top Level Control Structure

The control scheme exposes several new features with respect the pre-existing software. The huge amount of datastore memory employed for IO control have been completely replaced with structured data store that makes use of Simulink busses. This option allows not only having lesser datastore around the schematic but also

allows Simulink to reduce the shared variable from 110+ to only four major variables. The benefit is that shared information is now stored into structures with homogeneous meanings (Drivers, CTRL, DEV and SPLINE). An overview of the structures is documented in the following three:



The figure consists of three screenshots of the MATLAB Base Workspace, each showing a different structure: ControlBUS, DeviceSensorsBUS, and DriverBUS. Each screenshot includes a tree view on the left and a table of variables on the right.

ControlBUS Structure:

Name	DataType	Complexity	Dimensions
ExStart	uint16	real	1
app2vs_rm	single	real	2
replyRequest	uint16	real	1
statIndex	uint16	real	1
ForceAlarm	single	real	1
MaxForceAuto	single	real	1
psResetCount	uint16	real	1
PenStatus	int16	real	1
errorCode	uint16	real	1
payloadpid	uint16	real	1
statpid	uint16	real	1
resetKalman	uint16	real	1
id	int16	real	8
SoundOn	uint32	real	1
EnableMot	uint32	real	1
EnableMot2	uint32	real	1
dDebugMode	uint32	real	1
EnablePen	uint32	real	1
Hand	uint32	real	1

DeviceSensorsBUS Structure:

Name	DataType	Complexity	Dimensions
pendatacounter	uint16	real	1
speed	single	real	1
KaErr	single	real	3
Fabs	single	real	2
Statistics	single	real	48
pageInfo	int16	real	2
tempoHR	int32	real	1
Pen	int16	real	2
POS	single	real	3
POSE	single	real	3
AccXYZ	single	real	3
FT_EE	single	real	3
Encoders	int32	real	3
EncoderVEL	int32	real	3
cellOffset	single	real	1
OffsetCell	single	real	2
MotoreTime	single	real	1
Vbatt	single	real	1

DriverBUS Structure:

Name	DataType	Complexity	Dimensions
COMError	int16	real	1
pktCnt	int16	real	1
countOut	int16	real	1
answerTime	single	real	1
splineAck	uint16	real	1
splLength	single	real	1
splNum	int16	real	1
splNum_next	int16	real	1
newSplinePkt	int16	real	1
nextSplinePkt	int16	real	1
currentSplinePkt	uint16	real	1
splineTerminated	int16	real	1
maxLang	int16	real	1
maxLang_next	int16	real	1
fts	int16	real	1
low	int16	real	1
lvt	int16	real	1
startI	int16	real	1
endI	int16	real	1
gStart	int16	real	2
uDes	single	real	2
weightId	int16	real	1
viscId	int16	real	1
velId	int16	real	1
oneWay	uint16	real	1
stiff	int16	real	1
ts_next	int16	real	1
lvt_next	int16	real	1
lvt_next	int16	real	1
startI_next	int16	real	1
endI_next	int16	real	1
uDes_next	single	real	2
weightId_next	int16	real	1
viscId_next	int16	real	1
velId_next	int16	real	1
stiff_next	int16	real	1
stiff_next	int16	real	1

Integration Panel (Version 3) Screenshot:

Name	DataType	Complexity	Dimensions
ContCurrent	single	real	3
PeakCurrent	single	real	3
PeakPeriod	single	real	3

Figure 9: Busses architecture

Using structures we had the opportunity to name and document each Simulink entity thus minimizing program and debugging errors for most of the similar name signals. Within the scheme, new special interface has designed to handle the whole building process, from STMCUBEMX design to the target deploy.

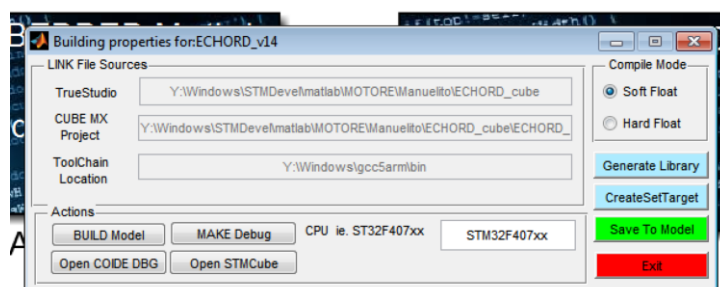


Figure 10: Integration Panel (Version 3)

Within the controller six special blocks have been designed to manage the ECHORD/MOTORE++ specific hardware:

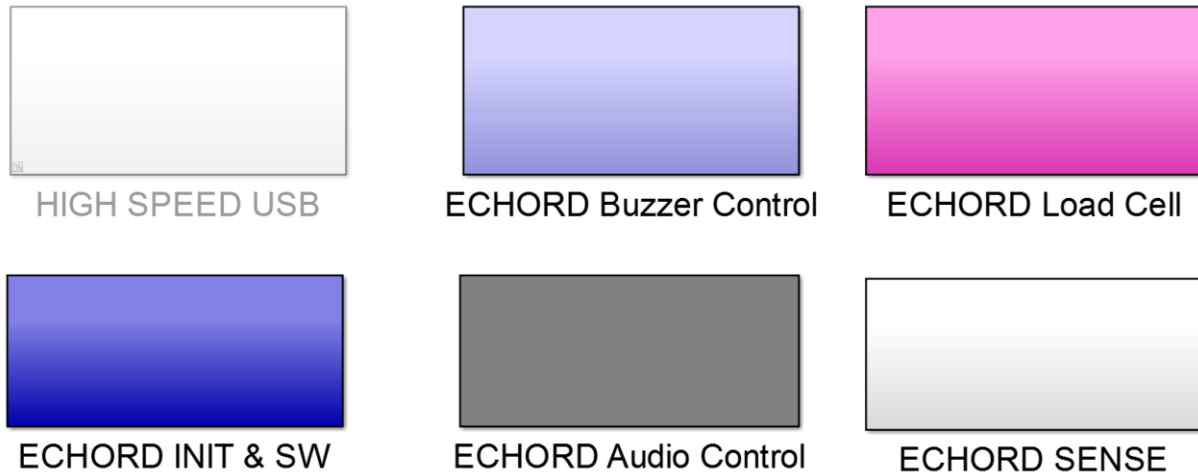


Figure 11: Subsystem of MOTORE++ specific drivers (ad hoc development)

Here it follows a detail on how the PEN sensing has been organized with the newer structure:

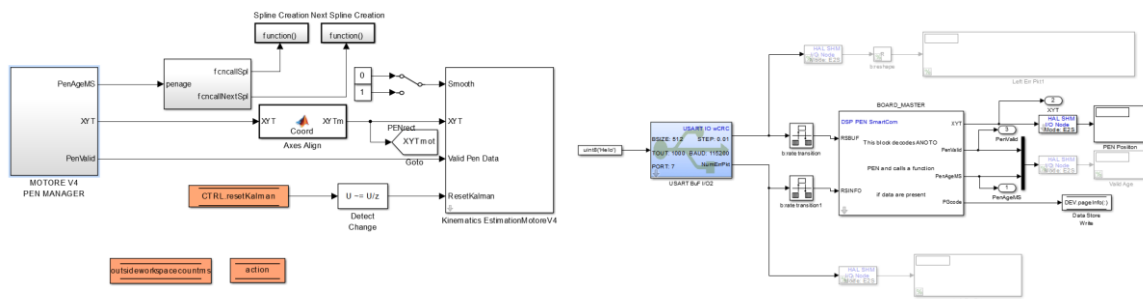


Figure 12: Example of the new pen driver design

Below this level of detail, specifically developed MEX function and Matlab-TLC blocks provide function inlining that support handling the host communication and the hardware management with the STM specific hardware. The design and the development of these blocks have been tested to properly work with the available board based on STM32F407.

Each block manages part of the IO in a way that is fully compliant with the STM_HAL library and such that all levels of code optimization can be implemented on the device.

In particular this allow a smart DMA management which delegates to basic HAL libraries and DMA architecture the burden to update memory coherently with data IO from/to any peripheral.

Some of the blocks inherit precedent result from research projects. For instance the Serial communication block does allow bidirectional data handling with any hosting PC and the same block can both be used as a data generator for the embedded target and a data consume from an hosting PC. This approach ease the debugging in the sense that the user can directly play a simulation to manage IO of data with a target application.

At the top level of interaction Simulink and STMCube integration is manage in a basic device control block.

The base device control is instead based on the following sub-diagram:

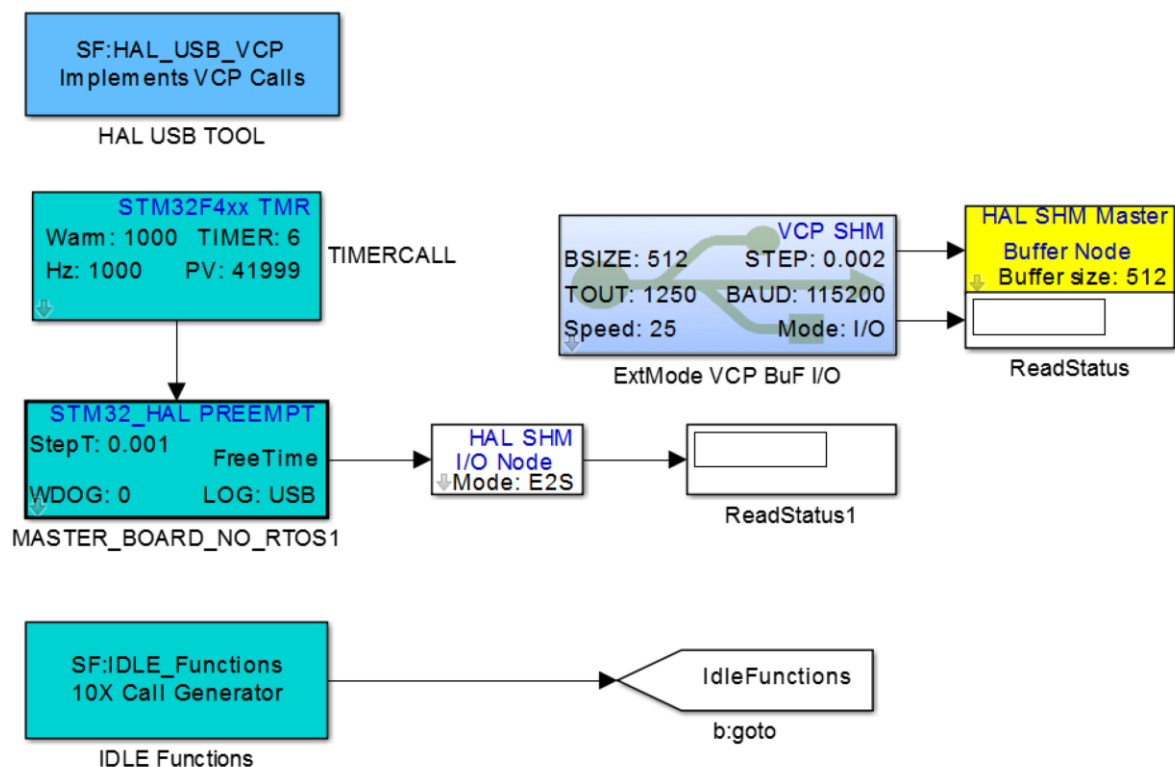


Figure 13: Scheduler subsystem

There are three main components:

- The cyan components relates to the process scheduling. One device timer can be programmed to generate regular interrupt which consequently trigger the simulation of the overall schematic as a interrupt driven procedure call.
- Even being prioritized in the arm architecture each interrupt is non-pre-emptable. Hence we designed a specific stack manipulation scheduler (names HAL PREEMPT) that allows pre-emptable interrupts. This is particularly rele-

vant when user has the need to run multithreaded task at different sampling times.

- In addition during the idle time, the main loop has been modified in order to execute sequentially up to 10 function call procedures.
- The blue components handle the onboard USB device existing on the micro-controller. Using this device it is possible to print debug information, or as in most cases to executes diagnostic with processor in the loop.

The diagnostic with the processor in the loop is handled by the yellow master block, which is one of several blocks connected thorough the schematic. The se block collect in a shared memory all data which are required to exchange with an external Simulink schematic. Here the user has the following option. When the scheme is built these block are inserted in the target platform, alternatively, if the blocks are played, the USB interface queries the remote target to know the value stored in the block and exhibit them as an output of the simulation. In such a way, playing a schematic has the effect of inspecting values running in the remote target.

The test procedures have been executed by connecting directly the existing motore Hardware to the available board. Hence the introduction of the new rocessor in the loop design has allowed us to simplify the diagnostic operation.

Most of the hardware can be intercepted at low level and forwarded to an external host system to diagnose the proper operation.

The operation performed include:

- Checking the new software for PEN acquisition and decoding (Result OK);
- Checking the new communication stack (Result OK);
- Checking the new motor control drivers (Results OK, with remarks);
- Checking the new ADC and force acquisition (Result Fail).

The new motor control drivers worked well but the new checking procedure allowed us to diagnose some noise issues previously unseen with the precedent MOTORE architecture. In these days we are developing to have a better control strategies that takes into account the non-lineary and other driver related issues to have a better current control;

The Force acquisition failed in the sense that the existing PERCRO board do not offer sensor pre-amplification for the low voltage signal produced by the force sensors. We discussed this point with the consortium and agreed that the FORCE sensing will be diagnosed as soon as the MOTORE++ official electronic will be ready.

3 Deviations & Reasons

Nothing to mention