Applications in ARM boards

This document describes the possibilities to develop an application for an ARM-based board using the services offered by emBODY framework.

With emBODY it is possible to write applications at different level of complexity: from the very simple main() function up to HW support and more and more structured services.

For instance, the HL library offers the simple main() function paradigm and easy support for MPU peripherals and attached chips. The abstraction layer allows writing libraries and simple applications by isolating the complexity of lower layers under invariant APIs. Finally, embedded objects help to develop complex applications with many concurrent or serialised services (eg., IP sockets, CAN transmission, protocols, control algorithms, error handling, etc.).

As an example, the EMS board uses the services of emBODY at different levels. The eLoader uses the abstraction layer and a utility library to manage its simple duties (reading shared memory, a permanent storage in EEPROM, jumping to the relevant process). The eUpdater uses a basic set of embedded objects as it also must manage IP communication. The eApplication uses additional embedded objects to also manage: a more structured protocol, CAN gateway mode, motor control, error handling, all inside a tight control loop.

Approval History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Author | | Date | Approved | | Date |
| 0.1 | Accame | iCub Unit |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 0.1 |  | M. Accame | First edit of the document |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 The emBODY for ARM 1](#_Toc377130190)

[1.1 The structure of emBODY 1](#_Toc377130191)

[1.2 The low level 1](#_Toc377130192)

[1.3 The middle level 1](#_Toc377130193)

[1.4 The high level 2](#_Toc377130194)

[2 The applications 3](#_Toc377130195)

[2.1 Low level 3](#_Toc377130196)

[2.1.1 Simple main with HL-CORE CMSIS 3](#_Toc377130197)

[2.1.2 Basic HW peripheral support with HL-CORE STM32LIB 3](#_Toc377130198)

[2.2 Middle level 3](#_Toc377130199)

[2.2.1 Board device support with HL-PLUS 3](#_Toc377130200)

[2.2.2 Multitasking with OOSIIT 3](#_Toc377130201)

[2.3 High level 3](#_Toc377130202)

[2.3.1 Simple portability with abstraction layer 3](#_Toc377130203)

[2.3.2 Simple multitasking with embOBJ 3](#_Toc377130204)

[2.3.3 IP communication with embOBJ 3](#_Toc377130205)

[2.3.4 Tight control loop plus IP protocol services with embOBJ 4](#_Toc377130206)

# The emBODY for ARM

The emBODY framework is a layered SW environment which can be used at any level to obtain the desired trade-off of SW complexity and performance for the target application.

The development of a highly efficient PWM controller should be done at low level close to HW registers. On the other hand, an application which must manage IP and CAN communication, structured protocols, motor control, data acquisition, error handling, etc. should be developed in a more structured and encapsulated mode.

Here is what the emBODY framework offers.

## The structure of emBODY

STRUCTURE OF EMBODY

HL - CORE (CMSIS + STM32LIB)

HL - PLUS

EMBEDDED OBJECTS

SERVICES

ABSTRACTION LAYER (OSAL, HAAL, IPAL)

OOSIIT (RTOS)

EVENTVIEWER

LOW LEVEL

MID LEVEL

HIGH LEVEL

**Figure 1**: The structure of emBODY.

## The low level

It contains the basic structure for developing simple applications. It is grouped under the library HL (Hardware Level) which is the customisation of ARM CMSIS and the STM32 standard peripheral library offered by ST Microelectronics.

The HL so far supports MPUs by ST Microelectronics such as STM32F103, STM32F107 and STM32F407.

## The middle level

It extends the low level in several aspects.

* HL-PLUS simplifies management of some peripherals of the MPU (e.g., I2C communication, CAN initialisation, GPIO basic management) and it gives support to selected external chips (some EEPROMs, gyroscopes, accelerometers, etc.).
* Eventviewer allows viewing timing diagrams in run time in the compiler IDE.
* OOSIIT gives the possibility of organising the application with an open-source fully pre-emptive RTOS optimised for ARM environment.

## The high level

The high level is formed by an abstraction layer which gives separate services using a coherent API. The OS services are provided by OSAL, HW services by HAAL, and basic IP services by IPAL.

These libraries can be used directly, possibly using only one of them, or can be used by some libraries of function (called services) or by the embOBJs, which are a library of objects for the embedded environment.

The applications

## Low level

### Simple main with HL-CORE CMSIS

See eBtest\arch-arm\libs\midware\hl-tests\test-core\test-it\core-cmsis.uvproj.

See also eBtest\arch-arm\libs\midware\eventviewer-tests which is used to test the eventviewer.

### Basic HW peripheral support with HL-CORE STM32LIB

See eBtest\arch-arm\libs\midware\hl-tests\test-core\test-it\ core-stm32f1.uvproj.

## Middle level

### Board device support with HL-PLUS

See eBtest\arch-arm\libs\midware\hl-tests\test-plus\test-it.

### Multitasking with OOSIIT

See development test project in eBtest/arch-arm/libs/midware/oosiit-tests.

## High level

### Simple portability with abstraction layer

See its use in eBcode/arch-arm/board/ems001/env/eLoader.

### Simple multitasking with embOBJ

See example in eBtest/arch-arm/board/many/appl/EOMapplicationBasic.

### IP communication with embOBJ

See example in eBtest/arch-arm/board/many/appl/EOMapplicationIPnet.

See its use in eBcode/arch-arm/board/ems001/env/eUpdater.

### Tight control loop plus IP protocol services with embOBJ

See example in eBtest/arch-arm/board/many/appl/EOMapplicationCTRLoop.

See its use in eBcode/arch-arm/board/ems001/appl/v01

See the relevant document to know more details about how the application works.