|  |  |
| --- | --- |
| **Title:** | **Engine Control System**  **SW Component MPC5606B** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 1.0 | Draft  28-0ct-15 | Kevin Nevarez | Adan Martinez | Creation of the document |
| 1.1 | Draft  29-Oct-15 | Kevin Nevarez | Adan Martinez | Modification |
| 1.2 | Draft  31-Oct-15 | Kevin Nevarez | Adan Martinez | Modification |
| 1.3 | Draft  1-Nov-15 | Kevin Nevarez | Adan Martinez | Modification |
| 2.0 | Draft  22-Nov-15 | Adan Martinez | Kevin Nevarez | Adding new design. |
| 2.1 | Draft  23-Nov-15 | Kevin Nevarez | Adan Martinez | Adding new charts and comments. |
| 2.2 | Draft  23-Nov-15 | Adan Martinez | Kevin Nevarez | Linking the content and last chart added. |

**Table of Contents**

[1 Purpose 4](#_Toc440629132)

[2 Definitions and abbreviations 4](#_Toc440629133)

[3 Realization constraints and targets 5](#_Toc440629134)

[4 SW Conceptual design 5](#_Toc440629135)

[4.1 Use Case Diagram 5](#_Toc440629136)

[4.2 Deployment Diagram 6](#_Toc440629137)

[4.3 Architecture Software Design 6](#_Toc440629138)

[4.4 Sequence Diagram 8](#_Toc440629139)

[5 SW Component internal breakdown 9](#_Toc440629140)

[5.0.1 Class Diagram. 9](#_Toc440629141)

[5.1 Functional Decomposition 12](#_Toc440629142)

[5.2 Function *<void> <Set\_baudrate> (void)* 12](#_Toc440629143)

[5.3 Function *<void> <init\_Slave\_mode> (void)* 12](#_Toc440629144)

[5.4 Function *<void> <LED\_StateMachine> (le\_slvcmd T\_CMDTYPE)* 13](#_Toc440629145)

# Purpose

Understanding the behavior of a scheduler (replacing an operating system) oriented to control systems of an internal combustion engine, in turn to control the speed and position of a DC motor (replacing the internal combustion engine) in an automotive system where the current speed and the same failures are reported through a network of CAN in synchronization with a defined position of the motor.

# Definitions and abbreviations

**Definitions**

**Abbreviations**

CAN Controller Area Network

SW Software

MCU Microcontroller Unit

LED Lighting Emitting Diode

LIN Local Interconnect Network

ECU Electronic Controller Unit

PIT Periodic Interrupt Timer

SIUL System Integration Unit Lite; microcontroller peripheral responsible of some I/O functions

HAL Hardware Abstraction Layer

MAL Microcontroller Abstraction Layer

**References**

|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** | **Reference** |
| 1  2  3 | Traceability Matrix Can Project.xls  ES\_Requirements.doc  AEP CAN database.xlsx |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Realization constraints and targets

The engine control module is connected to a CAN network where a master module will be commanding data through defined messages.

It will also have a DC motor speed which is controlled depending on the received commands.

Encoders are electromagnetic transducers or optical that convert shaft rotation into output pulses. May be counted to measure the revolutions of the drive shaft or said shaft angle. These provide information and positioning from system closed loops.

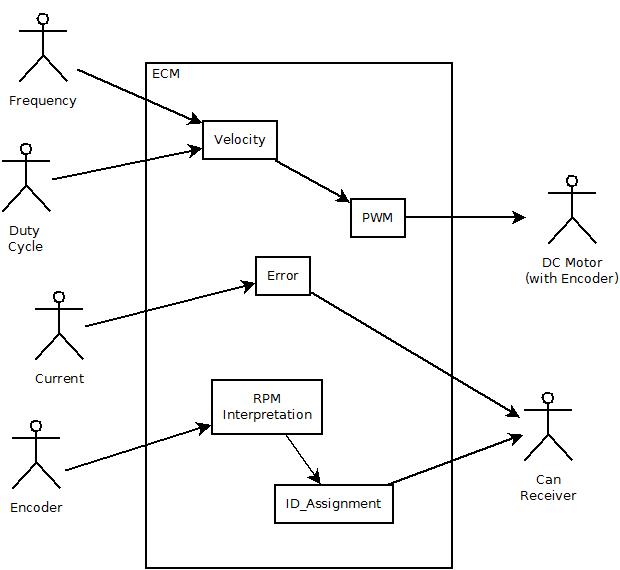
A rotating counter can census number of discrete positions per revolution. The number is called dots per revolution.

The speed of an encoder is in units of account per second. The encoders can measure the angle of the motor shaft and positioning screw indirectly to report the position, but they can also measure the response of rotating machines directly.

# SW Conceptual design

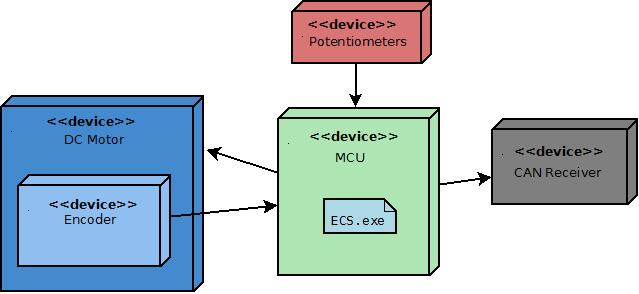
## Use Case Diagram

For this project, use case diagram shows how the potentiometer shall work to control the frequency and the duty cycle of the motor with help from the MCU. In addition, another potentiometer shall simulate the current value so the MCU can simulate the motor’s error. Subsequently the motor change its state, its encoder shall send the rpm’s from the motor to the MCU and after interpretation shall send it to the CAN receiver.



## Deployment Diagram

The Deployment Diagram explain the hardware interconnection between the devices of the project. In this case, show how the potentiometers connected to the MCU are available to control the DC motor and after that the encoder communicate to the MCU the Motor’s rpm’s. The MCU interpret this rpm’s and send messages to the CAN receiver.

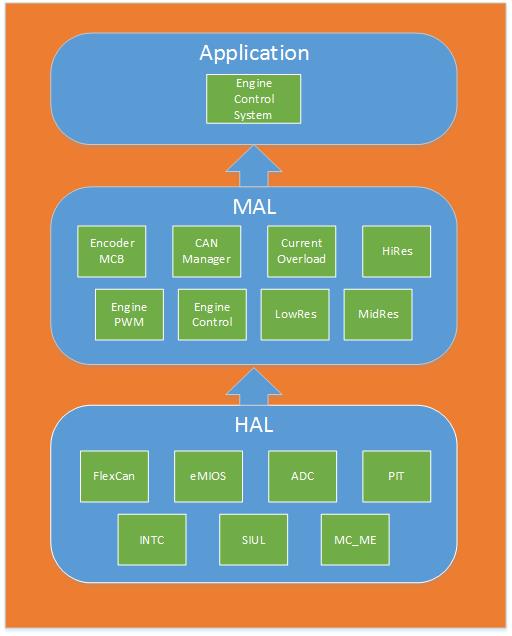


## Architecture Software Design

*The Engine Control System is a SW component which offers an application layer to communicate the motor status by CAN protocol and interpret them. It is made of 3 abstraction layers which are:*

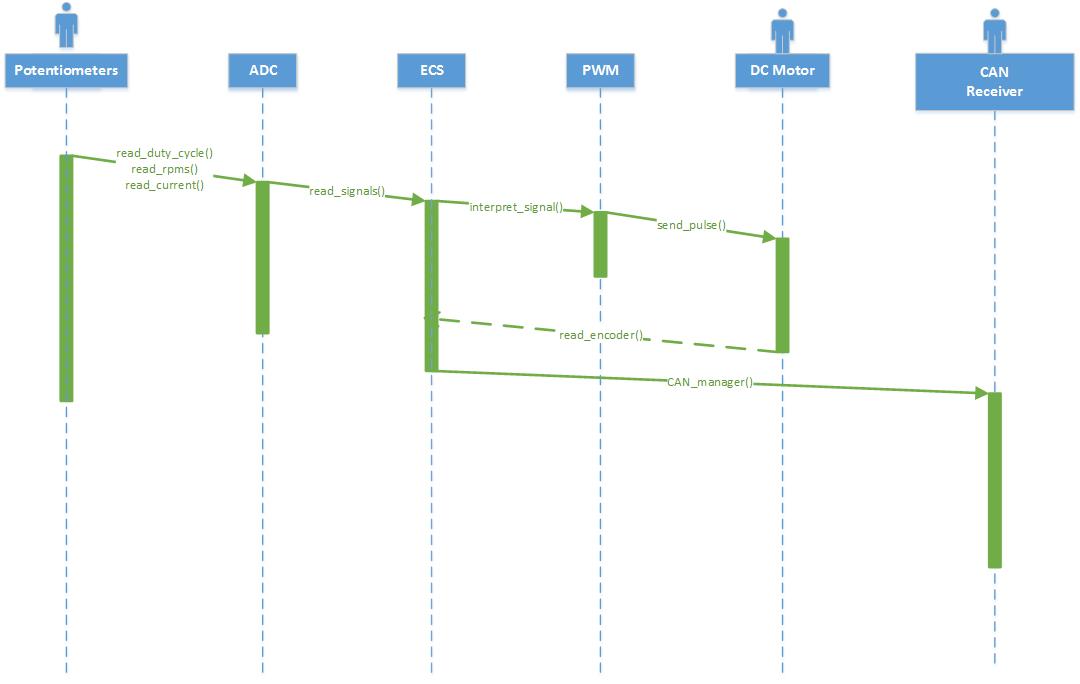
* *Hardware Abstraction Layer (HAL) it consist of the lower layer which is linked to the architecture of the MCU peripherals and modules in charge to perform the application. PIT module is responsible for offering a timer to meet the software requirements and functionalities linked to the scheduler operation built. SIUL module allows full control over I/O ports making possible to develop the Driver to configure the embedded IO peripherals. INTC module’s purpose is to manage interrupts generated over the system. MC\_ME module contains configuration, control and status registers so that the system can operate under a specified. The MCU offers a flexible module for implementing CAN communication supporting great variety of functionalities to prevent CPU overloads configured on flex CAN.*

* *Microcontroller Abstraction Layer (MAL) contains all the drivers that are developed in this project and a support to the implementation of state machines related to. Here can be found all the driver’s in charge to manage the motor status and the states of the potentiometers which control the motor state and the current failure.*
* *On the other hand, application layer consist in the all system software that receives the signals, interpret them and send the messages necessary to the CAN receiver.*



## Sequence Diagram

The way how components in software interacts between layers and upon events over time are defined in this sequence diagram. Here’s the presentation of how the signals are received from the Potentiometers layer to the ADC layer in the MCU. The Engine Control System read this signals, interpret them and order the PWM layer to activate the motors functionality. In response that the motor is working, a signal shall be receive from the encoders motor and interpreted by the ECS. Subsequently, the ECS shall send the correspondent message to the CAN receiver and shows the status from motor and data.



# SW Component internal breakdown

This section shall show the way the software interconnect the process from reading the ADC signal from the potentiometer interpret them by the MCU and send the signal to the PWM. From the encoder signal to the reading and sending of the message by the CAN protocol.

The code design and explanation of the complete system shall be shown at next.

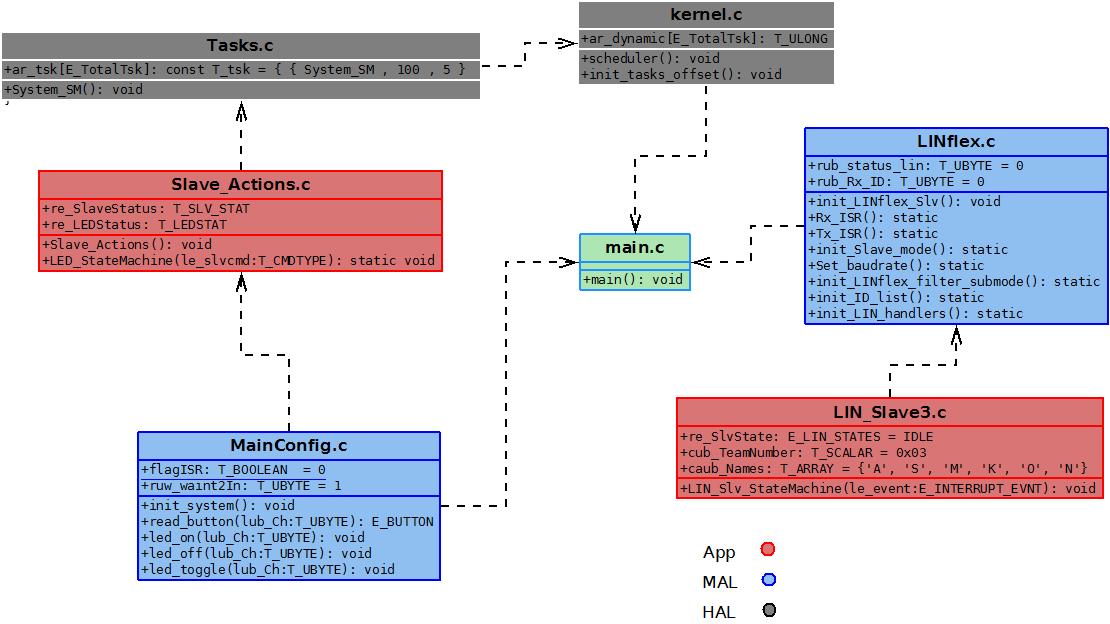
# 5.0.1 Class Diagram.

**Object Model Diagram name:**

Engine Control System

**Description:**

Internal Design of the Engine Control System



**Contained Elements**

|  |  |
| --- | --- |
| **List of Methods** |  |
| **Declarations:** | void Set\_baudrate(void) |
| **Documentation:** | It shall Sets the baudrate. |
| **Declarations:** | void init\_Slave\_mode(void) |
| **Documentation:** | Starts the slave mode configuration. |
| **Declarations:** | void LED\_StateMachine(T\_CMDTYPE le\_slvcmd) |
| **Documentation:** | State machine of the behavior of the LED, executed every 50ms. |
| **Declarations:** | void Slave\_Actions(void) |
| **Documentation:** | Change the Status of the slave and the LED. Called every 50ms. |
| **Declarations:** | void System\_SM(void) |
| **Documentation:** | Init function of Scheduler module. |
| **Declarations:** | void led\_on(T\_UBYTE lub\_Ch) |
| **Documentation:** | Turn on the selected LED. |
| **Declarations:** | led\_off(T\_UBYTE lub\_Ch) |
| **Documentation:** | Turn off the selected LED. |
| **Declarations:** | void led\_toggle(T\_UBYTE lub\_Ch) |
| **Documentation:** | Toggle the LED status selected. |
| **Declarations:** | void set\_pin( vuint8\_t PCR\_ch , vuint8\_t value) |
| **Documentation:** | Assign the value that it’s send to the channel that it’s send. |
| **Declarations:** | LIN\_Slv\_StateMachine  (E\_INTERRUPT\_EVNT le\_event) |
| **Documentation:** | State machine for the actualization of the parameters of the slave 3. |

## Functional Decomposition

Overview of functions and their dependencies shown by a Static Function Tree

**Function Description and Dynamic Behavior**

## Function <void> <Set\_baudrate> (void)

|  |  |
| --- | --- |
| **Description** | It shall Sets the baudrate.. |
| **Parameter 1** | *NA* |
| **Parameter 2..n** | *NA* |
| **Return Value** | *NA* |
| **Precondition** | *The linflex must be on init mode.* |
| **Post condition** | *The baud rate is setted.* |
| **Error Conditions** |  |

## Function <void> <init\_Slave\_mode> (void)

|  |  |
| --- | --- |
| **Description** | Starts the slave mode configuration. |
| **Parameter 1** | *NA* |
| **Parameter 2..n** | *NA* |
| **Return Value** | *NA* |
| **Precondition** | *The correct initialization of the startup code..* |
| **Post condition** | *Set on init mode the LINflex.* |
| **Error Conditions** |  |

## Function <void> <LED\_StateMachine> (le\_slvcmd T\_CMDTYPE)

|  |  |
| --- | --- |
| **Description** | State machine of the behavior of the LED, executed every 50ms. |
| **Parameter 1** | *The status of the LED when you call the function.* |
| **Parameter 2..n** | *NA* |
| **Return Value** | *NA* |
| **Precondition** | *The periodic task is called* |
| **Post condition** | *Switch the status of the LED* |
| **Error Conditions** |  |

LED State Machine is defined for 3 current states depicted below. According to design the LED behavior may turn from one to all other states being capable to execute all functions regardless the state the system is in. The events that moves the State machine are given as commands thus communicating the LIN state machine with the application layer.

