2016-07

Reporte de formación complementaria en área de concentración en sistemas embebidos y telecomunicaciones

Tlapa-Juárez, Miguel

Tlapa-Juárez, M. (2016). Reporte de formación complementaria en área de concentración en sistemas embebidos y telecomunicaciones. Trabajo de obtención de grado, Maestría en Diseño Electrónico. Tlaquepaque, Jalisco: ITESO.

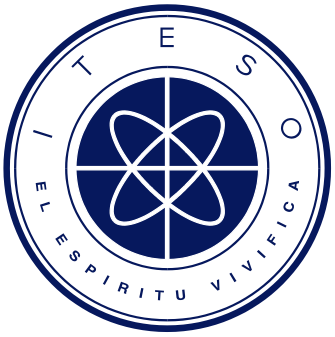
Enlace directo al documento: http://hdl.handle.net/11117/3807

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN DISEÑO ELECTRÓNICO



REPORTE DE FORMACIÓN COMPLEMENTARIA EN ÁREA DE CONCENTRACIÓN EN SISTEMAS EMBEBIDOS Y TELECOMUNICACIONES

Trabajo recepcional que para obtener el grado de

MAESTRO EN DISEÑO ELECTRÓNICO

Presentan: Ing. Miguel Tlapa Juárez

Director: Dr. Raul Campos Rodriguez
Revisor: Mtro. Juan Rafael del Rey Acuña
San Pedro Tlaquepaque, Jalisco. 13 de Junio del 2016.

Contenido

Introducción

El objetivo principal del documento es dar a conocer los proyectos más sobresalientes que se desarrollaron en la Maestría de Diseño Electrónico en el área de Sistemas Embebidos y Telecomunicaciones. Las materias de concentración y proyectos son los siguientes:

- **Taller de Diseño de Tarjetas de Circuito Impreso**: Diseño de un Circuito Impreso para Reproductor de Audio Portátil.
- **Ingeniería de Software en Ambientes Embebidos**: Desarrollo de la Documentación de las Especificaciones de Requisitos de Software para un protocolo de comunicación automotriz.
- **Diseño de Sistemas Operativos en Ambientes Embebidos**: Diseno e implementacion de un modulo de control automotriz.

Se escogieron los siguientes proyectos debido a que desarrollaron diferentes habilidades tanto técnicas en el área de electrónica e informática y de gestión tales como comunicación, liderazgo, trabajo en equipo, administración y desarrollo personal. El primero proyecto me dio las bases para diseñar, conocer a diferentes fabricantes extranjeros que se dedican a manufacturar este tipo de circuitos. El segundo proyecto me permitió conocer a bajo nivel como se implementa un controlador de comunicación automotriz LIN, a partir de estándares automotrices. El último proyecto presento un gran reto porque se integraron conocimientos de varias materias del área de sistemas embebidos partiendo de una metodología que es el modelo V, el uso de un sistema operativo, actuadores y sensores. La educación obtenida en el postgrado del ITESO me ayudo ha ser considerado una opción interesante en una empresa de computo de clase mundial, cubriendo el puesto de Ingeniero de Validación de Sistema para protocolos de comunicación de pcie- express en los nuevos micro servidores.

1. Resumen de los proyectos realizados

1.1. Diseño de un Circuito Impreso para Reproductor de Audio Portátil

1.1.1 Introducción

El presente proyecto consiste en el diseño e implementación de un circuito impreso utilizando la herramienta Allegro, para un reproductor portátil de audio, que consta de 4 capas, fuente de alimentación de voltaje, microcontrolador, decodificador, un puerto usb 2.0, slot memoria micro sd y conectores de audio 3.5mm.

1.1.2 Antecedentes

El amplio sector de los reproductores mp3 ha puesto muy difícil a los consumidores la elección de un modelo determinado, ya que hay un infinidad de marcas, precios, diseños, formas, tamaños, capacidades, etc. de modo que ¿Cómo elegir? ¿Cómo saber cuál es el mejor modelo del mercado?

Para evitar esta problemática se planteó hacer el diseño y desarrollo de un reproductor de audio personalizado a bajo costo, ajustado a nuestras necesidades y preferencias.

Utilizando herramientas de software de diseño de circuitos impreso industrial que actualmente usan los ingenieros de diseño automotriz, conocer las diferentes funciones, compatibilidad y limitantes del mismo. Entender las guías de diseño eléctrico, la teoría electromagnética para no tener problemas con la comunicación entre los diferentes componentes o generar ruido entre los canales. Las tolerancias mecánicas juegan un papel importante cuando se requiere hacer un ensamble final del producto. Por lo que no tener estos conocimientos y experiencia previamente con lleva a invertir mucho tiempo, dinero y demás recursos que son necesarios para la creación de un producto comercial competitivo.

1.1.3 Solución Desarrollada

En la primera parte del proyecto se utilizó un archivo esquemático eléctrico- digital que definía el comportamiento funcional de un reproductor multimedia.

Se empezaron a colocar los componentes eléctricos pesados y grandes en la capa de arriba y los pequeños en la capa de abajo del circuito impreso, tomando en cuenta la funcionalidad de cada circuito, si es analógico o digital.

El microcontrolador principal fue localizado en el centro y cerca de él los componentes de alta velocidad como el dispositivo USB 2.0 micro SD y el decodificador. Además de los capacitores de acoplamiento que son los que filtran los diferentes niveles de voltaje que requiere el microcontrolador.

En la segunda parte del proyecto se procedió a definir los diferentes planos de voltaje y de tierra así como el ruteo de todos los componentes respetando las guías de diseño.

En la tercera parte del proyecto se generó la lista de materiales y se empezó a cotizar con los diferentes vendedores de materiales en Guadalajara y Estados Unidos.

Finalmente se mandó a fabricar el circuito impreso en Estados Unidos.

1.1.4 Análisis de los Resultados

Como parte de los resultados obtenidos se demostró la viabilidad de la metodología utilizada para diseñar el circuito impreso del reproductor multimedia, a un costo muy bajo, de dimensiones 3cm cuadrados aproximadamente. Hubo un retraso de 3 semanas por parte del fabricante del circuito impreso en estados unidos, por lo que no alcanzo el tiempo para soldar los componentes eléctricos para verificar su funcionamiento y compararlo contra productos comerciales.

1.1.5 Conclusiones

Los profesores de la materia, propusieron desde el inicio una metodología y estrategia de trabajo al iniciar el proyecto, lo que ayudo en el desarrollo de las diferentes etapas del proyecto.

El no tener bien definido la posición de los componentes eléctrico-digitales, respetando las guías de diseño, puede ocasionar trabajar horas extras cuando se va rutear.

Los conocimientos adquiridos en esta área se dieron, porque los maestros compartieron su experiencia en el ámbito laboral y por la dedicación del alumno en investigar el uso de las diferentes herramientas y especificaciones de tecnología.

Esta experiencia me sirvió en el desempeño de mis actividades laborales debido a que tenemos que interpretar las diferentes características de los circuitos impresos en los micro servidores de última tecnología.

1.2. Desarrollo de la Documentacion de las Especificaciones de Requerimientos de Software para un protocolo de comunicación automotriz.

1.2.1 Introducción

El proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema es llamado *Ingeniería de Requerimientos*. La meta de la ingeniería de requerimientos es entregar una especificación de requisitos de software correcta y completa.

Los requerimientos son la pieza fundamental en un proyecto de desarrollo de software, en ellos se basan muchos participantes del proyecto para planear el proyecto, los recursos que se usaran en él. Los líderes de proyecto usan los requerimientos como una base para la estimación del esfuerzo necesario para un proyecto. Especificar el tipo de verificaciones que se habrán de realizar al sistema, por ejemplo, cuando se está tratando de satisfacer a cierta norma oficial o estándar. Adicionalmente son la base para planear, la estrategia de prueba a la que habrá de ser sometido el sistema. Los requerimientos son la base sobre la cual se decide si un caso de prueba fue ejecutado exitosamente por el sistema o no. Son el fundamento del ciclo de vida del proyecto. Los requerimientos documentados son la base para crear la documentación del sistema. De ahí su importancia de que deban ser definidos y manejados forma adecuada.

El objetivo de este proyecto fue el desarrollo de la documentación de las especificaciones de requisitos de software para un protocolo de comunicación LIN (Local InterConnect Network) en modo *maestro*, que permite el intercambio de información entre diferentes de mando que conforman un subsistema como por ejemplo el sistema de alarma, la comunicación entre los mandos de volante, el climatizador, etc.

1.2.2 Antecedentes

El protocolo LIN fue creado en 1990 por Volcán Automotive y Freescale, posteriormente fue adoptador por un consorcio formado por los ensambladores de automóviles (Audi, BMW, Daimler Chrysler y Volkswagen), este protocolo ha ido evolucionando en diferentes versiones hasta llegar a la versión 2.2A.

Este protocolo de comunicación es del tipo serial y cuenta con un solo maestro y hasta 16 esclavos. La velocidad de transmisión de datos es de 1 – 20 Kbit/s. Estas características permiten reducir el costo con respecto a un sistema CAN (Network Área Controller) en aquellas situaciones en las que no sea imprescindible una velocidad de transmisión elevada.

El sistema LIN está formado por una unidad de mando que actúa como maestro y otras unidades de mando que actúan como esclavas. También puede funcionar como unidades esclavas elementos sensores y actuadores a los que se les ha dotado de la electrónica necesaria para traducir las señales eléctricas en mensajes. La información entre las unidades de mando o sensores se transmite mediante un único cable. La unidad de mando maestra, define tanto la velocidad de transmisión como el tipo de mensaje y la frecuencia del mismo. Esta unidad es la que esta abonada a otras líneas de transmisión más rápidas como por ejemplo la línea de CAN, por lo que debe llevar acabo la traducción de mensajes entre ambas líneas, incluyendo los referentes a la diagnosis.

Por lo tanto el objetivo fue aplicar la metodología de ingeniería de software, para definir los diferentes tipos de requerimientos funcionales. Se usó como referencia las especificaciones LIN 2.2A liberada por el consorcio automotriz.

1.2.3 Solución Desarrollada

La solución desarrollada se basó en el documento LIN_Specification_Package_2.2A, donde se detalla información acerca de las características eléctricas y funcionales de este protocolo.

Posteriormente se procedió a clasificar los diferentes tipos de requerimientos, el diseño y la arquitectura de los módulos de la interface y del driver de LIN.

En el sistema LIN los mensajes se dividen en 2 partes: encabezamiento y contenido. El encabezamiento solo puede ser escrito por la unidad de mando maestro y el contenido puede estar escrito tanto por una unidad de mando maestro como unidad esclava.

Cuando el contenido es escrito por la unidad de mando maestro el mensaje ordena a las unidades esclavas que utilicen los datos contenidos en las respuestas. Las unidades esclavas escriben la parte correspondiente al contenido en los mensajes donde el encabezamiento del mismo exige que alguna de ellas transmita un dato concreto.

Por lo tanto el modulo maestro de la interface de LIN cuenta con 3 capas de software que son el agendador, que su tarea es ejecutar los diferentes paquetes de mensajes que están almacenados en un buffer y con diferente prioridad, el encabezado del frame que está conformado de la pausa de sincronización, limitación de la sincronización, campo de sincronización y campo de identificación. Por último el contenido del mensaje que contiene los datos del mensaje que es de 8 bytes.

El modulo maestro tiene un driver que está conformado por el transmisor y el receptor de la información. Se incluyeron los diagramas de flujo que describen el flujo de la información y sus diferentes ramificaciones, la explicación detallada de la paridad de bits el algoritmo de integridad de datos.

1.2.4 Análisis de resultados

En resumen, se generó un documento que describe los requerimientos funcionales y no funcionales que el cliente espera que el sistema cumpla y que sirven de base para la etapa de implementación del protocolo.

1.2.5 Conclusiones

Al finalizar este proyecto aprendí de la importancia de definir los requerimientos antes de iniciar un proyecto ya que este documento da claridad y una visión general de que es lo que quiere el cliente.

El no contar con esta documentación causa las siguientes complicaciones: falta de compromiso del usuario, falta de recursos, expectativas no realistas, falta de planeamiento y el final puede ser catastrófico como el cierre completo de un proyecto y de una empresa.

En el ámbito laboral implemente esta metodología al diseñar pruebas de validación funcional para el protocolo de comunicación pci express de micro servidores en la empresa de cómputo donde actualmente trabajo. Además de aprender el funcionamiento de los mecanismos que conlleva un protocolo de comunicación.

1.3. Diseno e implementacion de un módulo de control electronico automotriz

1.3.1 Introducción

En la electrónica automotriz el BCM (Body Control Module) es un término genérico para una unidad de control electrónica (ECU) que es responsable de monitorear y controlar varios accesorios electrónicos en el vehículo.

Típicamente en el auto el BCM controla los elevadores de las ventanillas de las puertas delanteras y traseras, el aire acondicionado, sistema de seguridad, etc. El BCM se comunica con otros ECU's, atraves de protocolos de comunicación CAN, con el objetivo de informar el status de los sensores y en base a esta información tomar la decisión de accionar los actuadores como de las ventanillas, las luces del auto entre otros.

El proyecto, consistió en diseñar e implementar el BCM que se pudiera comunicar con la unidad de control del cluster del automóvil.

1.3.2 Antecedentes

En la Especialidad de Sistemas Embebidos en el ITESO, se tuvo la idea de desarrollar un carro eléctrico prototipo que tuviera incorporado unidades de control que convirtieran la corriente directa a alterna, un banco de capacitores para la carga y descarga de energía, unidades de control donde muestren la información general de auto, unidades de control de motor y un BCM además de toda la electrónica que requiere como microcontroladores, circuitos de comunicación, sensores y actuadores y etapas de potencia.

1.3.3 Solución Desarrollada

Para la implementación del proyecto se siguió el método V, que es un procedimiento uniforme para el desarrollo de productos para el área de tecnologías de la información,
Nos permite hacer una representación gráfica del ciclo de vida del desarrollo de sistemas. En él se resumen las principales medidas que deben adoptarse en relación con las prestaciones en el marco del sistema informático de validación. Este proceso representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. Se describen las actividades y resultados que deben producirse durante el desarrollo del producto. El lado izquierdo de la V representa la descomposición de las necesidades y la creación de las especificaciones del sistema. El lado derecho de la V representa la integración de las piezas y su verificación. V significa verificación y validación. Primero se definieron los requerimientos funcionales y no funcionales del proyecto, esta etapa requirió de varias iteraciones para comunicar claramente las necesidades del cliente.
En la segunda etapa se definió la arquitectura general de software que fue basada en el modelo AUTOSAR que es un estándar de arquitectura de software utilizado en la Industria Automotriz. La arquitectura de hardware se adaptó a las necesidades del proyecto porque de inicio se empezó a trabajar con un sistema embebido automotriz así como la etapa de potencia de la empresa Freescale.
En la tercera etapa de diseño se empezó a refinar cada uno de los bloques que fueron definidos en la etapa anterior. Para el caso de la arquitectura de software se construyó una primera capa de software de bajo nivel que su función es comunicarse directamente con la capa física. La segunda capa de software que se construyo fue la que interpreta la información de la capa anterior. La

última capa de software que se construyo es la de aplicación, que es la que toma la decisión de qué hacer con los datos obtenidos de la capa anterior.
Sin embargo paralelamente se tiene un planificador de tareas cuya función es cargar y enviar a ejecutar las diferentes tareas, con el fin de ordenar las tareas cada una de ellas tiene diferente prioridad y tiempo de ejecución.
Después de esto se procedió a la implementación, generando el código lógico representativo de la arquitectura propuesta de software y la conexión del hardware que dio como resultado un banco de pruebas.
En la última etapa se empieza a verificar y validar cada una de las etapas anteriores haciendo varias iteraciones hasta cumplir con las necesidades del proyecto.

1.3.4 Análisis de los resultados

Los resultados obtenidos fueron los esperados debido a la funcionalidad de la metodología del modelo V. Además se logró identificar áreas de oportunidad para la mejora en la arquitectura de hardware el utilizar otros módulos de etapa de potencia más eficientes y a bajo costo.
En la Arquitectura de software se puede optimizar el código implementado de manera general usando menos recursos en memoria del microcontrolador y hacerlo más eficiente.
Si comparamos el desempeño del módulo construido con uno comercial nos damos cuenta que falta agregar aspectos de seguridad como por ejemplo cuando el BCM, es desconectado de la batería y hay que guardar el contexto de la información.

1.3.5 Conclusiones

El haber utilizado el Método V minimizo los riesgos, la transparencia y el control del proyecto usando enfoques estandarizados como AUTOSAR, y se logró dividir las actividades de manera modular.
Además de reducir los gastos totales durante todo el proyecto y sistema del ciclo de vida.
El aprendizaje y experiencia obtenida en este proyecto fue integral debido a que se desarrollaron diferentes habilidades tanto técnicas como de colaboración en equipo.

2. Conclusiones

Como resultado del tiempo y trabajo invertido por el alumno y la participación activa de mis profesores en la Maestría de Diseño Electrónico, es posible concluir que existe una relación directa entre los conocimientos y experiencia adquirida en esta Institución y el ámbito personal y laboral.

En la parte personal me hicieron una persona más consciente de las diferentes necesidades que hay en mi localidad y como cada uno de nosotros puede contribuir con un granito de arena para que nuestro entorno cambie, y que trabajar en equipo es mejor que ser individualista. Tener una visión más clara de todas las oportunidades que existen en nuestro mundo y como poder integrarse a esos cambios.

En parte laboral me ayudo a entender los procesos y mecanismos de una plataforma operativa comercial que está conformada por hardware y software.

Me dio la oportunidad de ser un candidato interesante para una empresa multinacional en el área de cómputo vendiéndoles mis servicios profesionales en el grupo de validación de sistema para micro servidores.

Finalmente ser miembro del grupo académico de esta institución laborando como profesor, dando un servicio a la comunidad estudiantil.

Mis agradecimientos a Dios por darme la vida y la fortaleza para seguir adelante, a mi familia que siempre me ha apoyado incondicionalmente en todos mis proyectos.

Apéndices

A. DESIGNING OF A PRINTED CIRCUIT BOARD FOR PORTABLE AUDIO REPRODUCER.

Content

Objectives

- Design a Printed Circuit Board for Mp3, using methodologies and software tool.
- Build the Printed Circuit Board accomplish the Industry Standards.

1. What is the Problem?

From a case study defined by Continental Engineers, they require a PCB design for MP3 player with input and output audio, which permit to connect USB Sticks. Compliance with Industry Standards.

2. Requeriments

Number	Type	Tag
1	Functional	F
2	Not Functional	NF
3	Suggestion	S

Number	Type	Description
R1	F	The PCB will be of dimensions n x n centimeters.
R2	F	The PCB will comply with the IPC Standards.
R3	NF	The heavier electrical-mechanical components like: Microcontrollers, audio connectors, usb and memory storage device will have to place in the top layer.
R4	NF	The lighter electrical-mechanical components will have to place in the bottom layer.
R5	F	You will have to define a power layer for different voltages.
R6	F	You will have to use a ground layer.
R7	F	The PCB will have to be building in 4 layers.
R8	F	The material that you will have to use is for manufacturing is FR4.
R9	F	The calculus of trace length shall comply the Oshpark Standards.
R10	NF	The color of card is a consideration of the designer.
R11	S	You will have to add a Bill of Material list included the cost and the vendors.
R12	F	The active and passive electrical components will have to be SMD.
R13	F	In the Ground Layer you will have to separate the analog gnd from digital gnd in order to eliminate interferences.
R14	F	The minimum value of the traces is 6 thousandths.
R15	F	The minimum value of the space between planes is 6 thousandths.
R16	F	You require at least 15 thousandths of free space in the edge of card.
R17	F	You require at least 7 thousandths of annular ring.
R18	NF	The color of card is a consideration of the designer.

Table 1 Requirements

3. Calendar of Events

Calendar of Events																	
Pos	Actividad	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
1	Presentation, Motivation and Description of Project																
2	Theory and PCB Fundamental.																
3	PCB History																
4	Introduction to Allegro Cadence																
5	Introducion to IIPC																
6	Theory of Manufacturing PCB.																
7	Load the Squematic from Cadence																
8	Electromagnetic Theory																
9	Introduction to Constraint Manager																
10	Load Constraints for Placement																
11	Considerations of Power Integrity																
12	Hi-Speed Fundamentals																
13	Load Constraints of spacing and geometry of the project.																
14	Route Differential Pair																
15	Deliver Full Placement																
16	PCB Documentation and create Gerber Files																
17	Deliver Routed PCB without DRC																
18	Manufacturing Documents (Board file + gerbers y mandar a fabricar)																
19	Introduction Libraries Allegro																
20	Creacion del Boom Material																
21	Foundations Transmission Lines																
22	Presenting Project																

Table 2 Calendar Events

4. Theory of PCBs

4.1. PCB

A printed circuit board (PCB) is a set of substrates that provide mechanical support and /or electrical interconnection to electronic systems through routes or tracks of conductive material, engraving on cooper sheets laminated on a nonconductive substrate, commonly bakelite or fiberglass.

4.2. Rigid PCBs

Alternating Layers of Core and Prepreg. Dielectric Core- cured(hardened) coated copper layers on both sides. Dielectric Prepreg which will subsequently be cured by heat and pressure.
The outer layers are formed of prepreg with the copper foil side exterior.
It stack-up is symmetrical about the center of the card on the vertical axis for avoid mechanical stresses in subsequent temperature cycles.

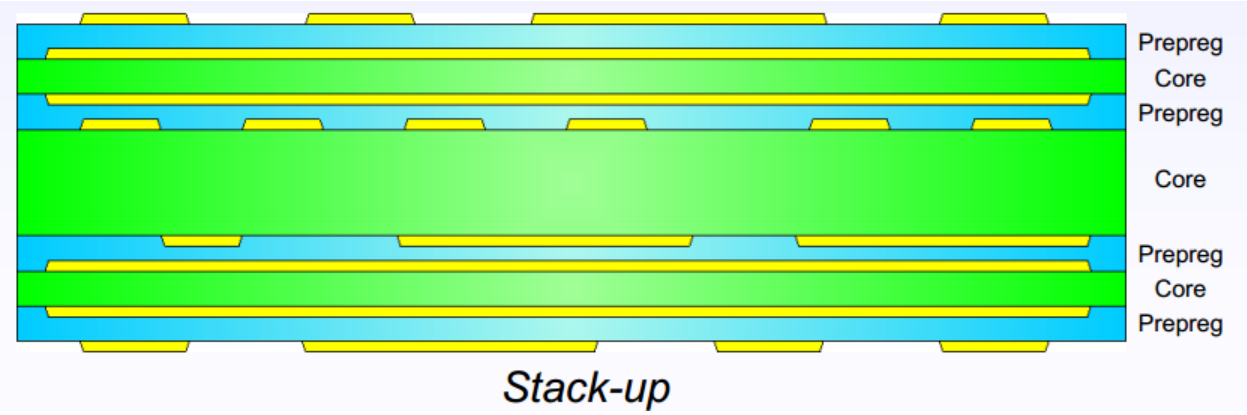


Figure 1 Stackup

4.3. Classification by Application

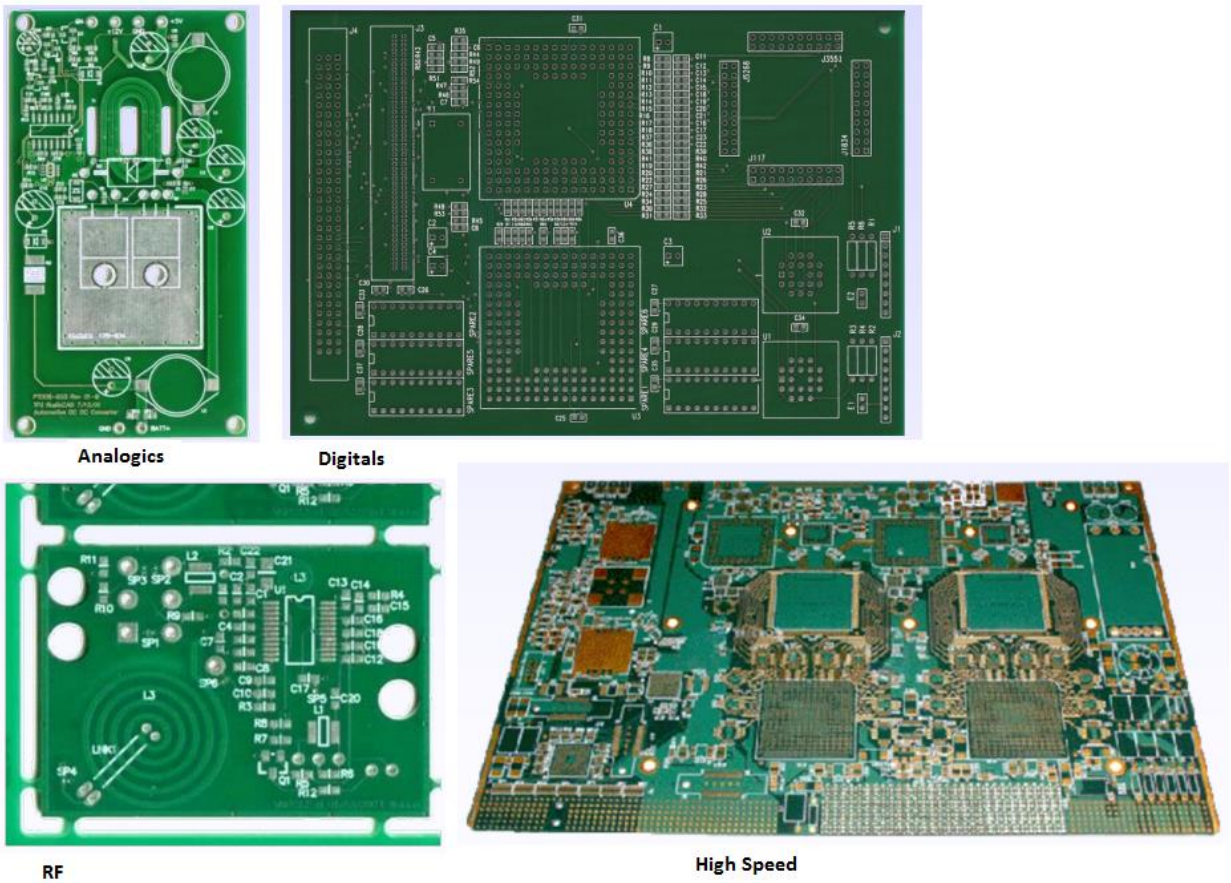


Figure 2 Different Kind of Pcbs

Thickness:

0.004 inches to 0.250 inches aproximately.

Size:

31 inches x 52 inches maximums

If they are smalls, It is convenient to panelize.

Controller Impedance:

Tolerance minimum 10%(typically to 50 Ohms).

4.4. Definitions

Layer of Signal	Usually copper , conduct electricity charge PCB.
Technology	Line Width/ Spacing Strokes
Power Plane	Copper power plane layer dedicated to driving power and ground.
Thermal Relief	Typical way to connect a signal to fill a plane or area.
Stack-up	Accommodating stack-up of layers in a PCB
Area Fill	Copper area in layer aggregate signal.
Copper Balance	Copper that signal layers is added to balance the density copper in ti.
Dielectric	Insulator Separating the different layers of copper generally FR4.
Pad	Footprint copper pad where a component is soldered of SMT
Pin	Drilling and pad of copper used to solder a True Hole Component
Via	Is used to pass a signal to another layer
Solder Mask	Layer generally green that cover a PCB, which is repellent welding.
Silkscreen	Layer drawing, usually white indicating polarity identity and shape of the components.
PasteMask	Template used for placement of the solder paste.
Fiducial	Copper Circle place to guide the automatic assembly machines.
Silkscreen Target	Present in the outer layers of cooper and the silkscreen to align power
Moat	Court power planes, used for noise control electrical
Tooling Hole	Drilling hole tooling used for aligning the layers of PCB Copper coating over the outer layers
Resistors Embedded	Resistors embedded within the PCB formed using a material special resistive.
Beveling	Applied to the edges of the PCB to reduce its thickness

Table 3 Definitions

4.5. Conductive

Material:

Generally Copper

Layers:

From 1 to 60 layers

From 2 layers you will have to build in pairs.

Dimensions:

Thickness in outer layers from 0.125oz to 6oz.

Thickness in inner layers from 0.25 oz to 4 oz (signals) or 5 oz (planes)

Resistivity: $\rho = 1.7 * 10^{-8}\ \Omega \cdot m$

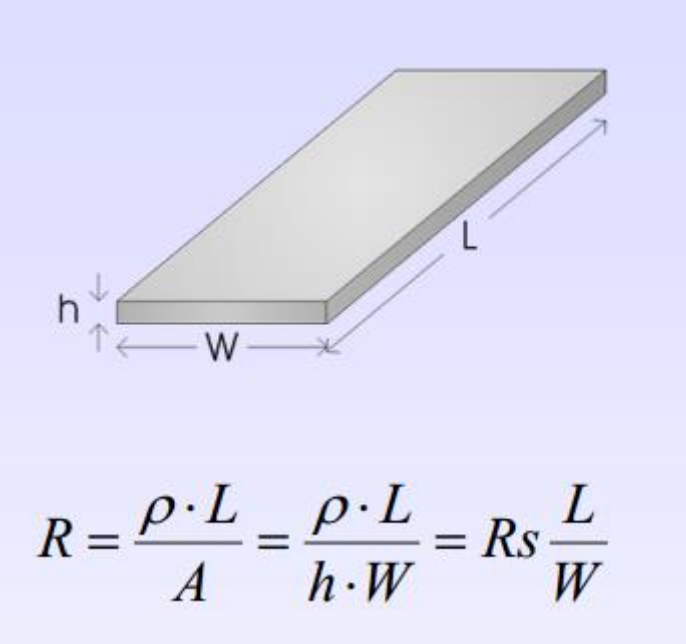


Figure 3 Resistivity

4.6. Copper Laminate Thickness

Depends on:

a) The amount of current it must carry. Any current in a conductor causes heating, among slimmer it gets warm.
b) Quantity of het that must be dissipated. Among thicker copper, more heat will dissipate.
c) Size and Spacing of the lines. Generally they use laminated thin strokes to avoid corrosion on copper base.

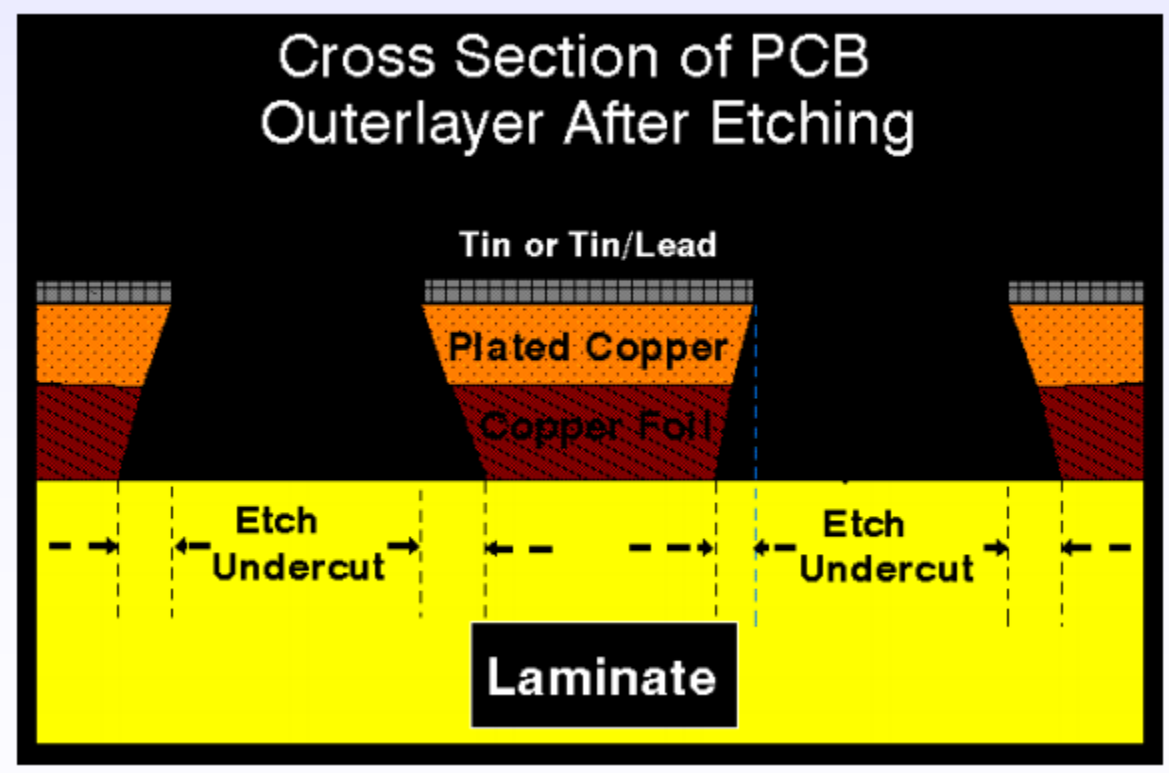


Figure 4 Cross Section of PCB

4.7. Power Planes

The power planes are usually built on a core thin, in order to increase the capacitance between the planes. The connection pins or tracks to power planes, generally through a ‘thermal relief’’.

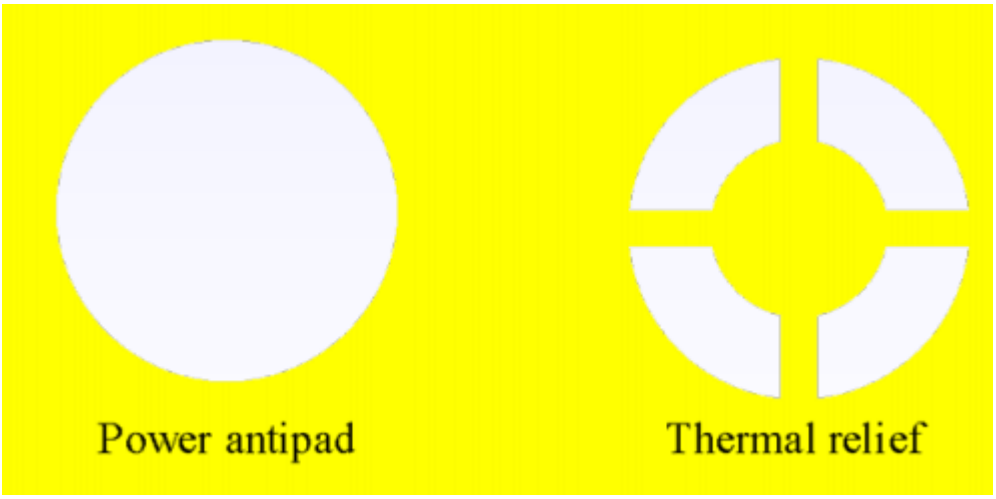


Figure 5 Power Planes

It facilitates connection what for?

a) It can form a capacitor between the planes


Figure 6 Two Planes

b) Low inductance ground signals and power.
c) Isolate electromagnetic noise signals.

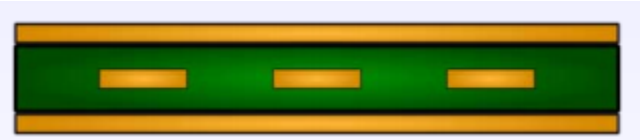


Figure 7 Isolate Electromagnetic

4.8. Stackup

Use the following lamination techniques:

a) Inner layers composed by cores.
b) Outer Layers composed of copper foil and prepreg.

Benefits

a) Improved dimensional stability
b) Reduce Cost

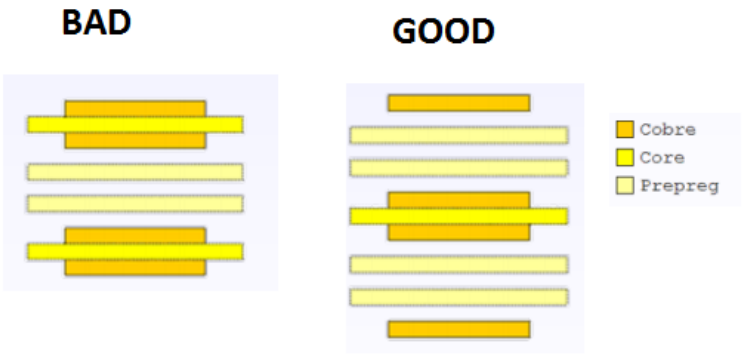


Figure 8 Stackup

4.9. Symmetrical Design

Ensure uniform distribution of three components: prepreg, core and sheet of copper.

Benefits

a) Less bending, twisting or warping
b) Cost Reduction
c) Improved Reliability of the manufacturing process.

5. Develop

5.1. Placement

1) Pleace heavy and large components on the top layer and the small components in the bottom layer.
2) I organized the next form:
   a) Separate the digital part of the analog
   b) The microcontroller place it at the center because they must be close of high speed components such as is the USB, MicroSD and Decode.

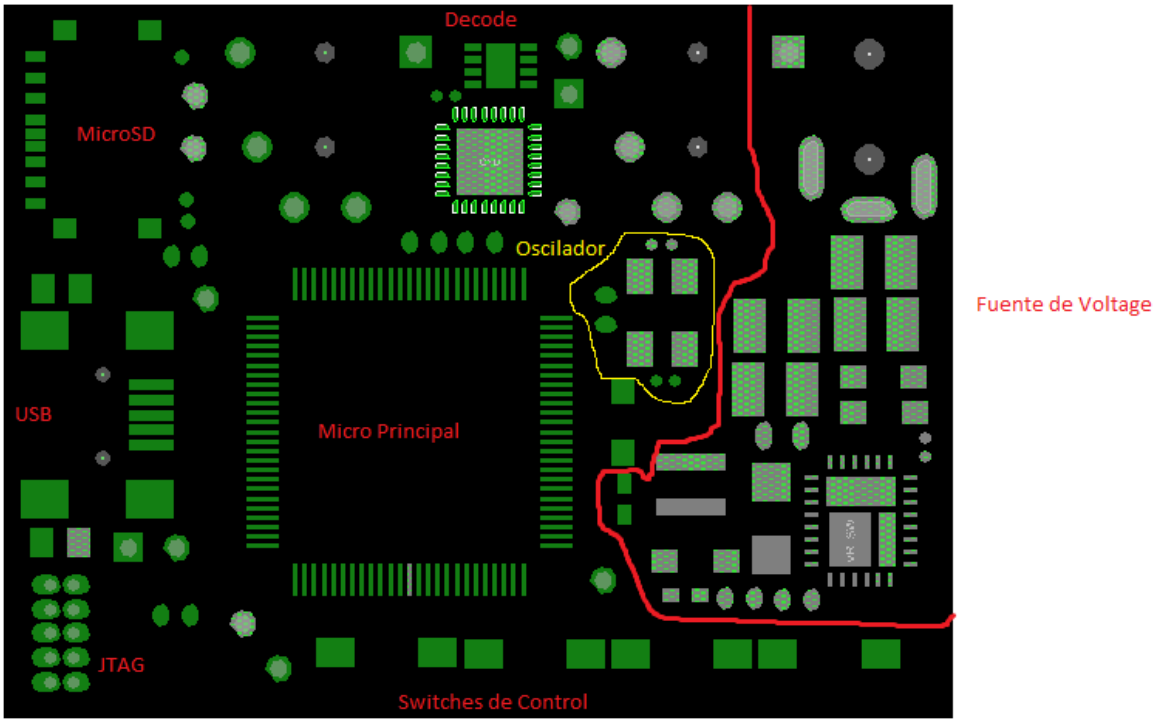


Figure 9 First Placement

c) Place of the Supply Voltage

I used the recommendation of pcb design manufacturer for supply voltage.

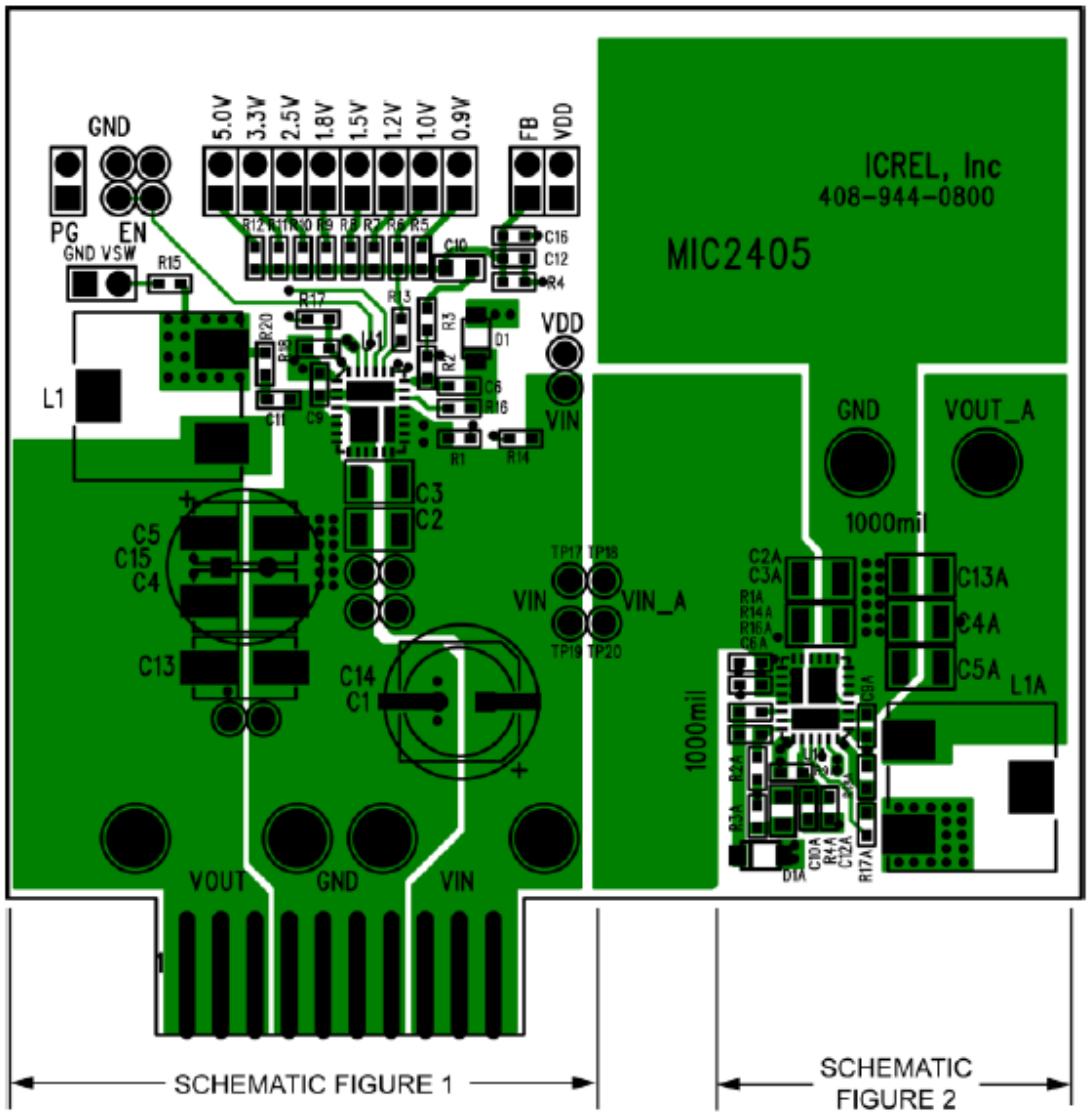


Figure 10 Supply Voltage

It tell us that we must take the placement so that the land is common to the capacitors and input and output. In addition the microcontroller.

The coil should be away from the capacitors because as it is a source type switch, make noise can reach the output.

A coil of Schotky type is used to separate the analog digital signal.

d) The Decode component should be close coupling capacitors near the VCC micro.

e) The USB component is a component of high speed, that uses GSOT05C-GS08 Diode to prevent electric shock when someone touches the device.

   Therefore they must be connected as close to USB connector.

   L1 and L2 of 330 Ohms coils serve to clean up the signal when connecting a wire. Also they must be connected as close to the USB.

f) JTAG connector is a component that will allow me to connect to the host microcontroller, that permit to write information to him.

g) Oscillator is the component that gives me the clock, this should go on top as close to the pin main microcontroller clock.

   You should not connect components on the bottom. Because when it is going to route insulation to prevent noise at the main microcontroller.

h) The microSD connector is where you will be recorded information file manager. It also must be connected near the main microcontroller.

i) The connector who hast the label edge indicate that they are placed on the edge of the card edge.

j) Beware of pads in order not to exceed the line of route keeping.

k) Coupling Capacitors was placed in pairs for each Vcc and GND .

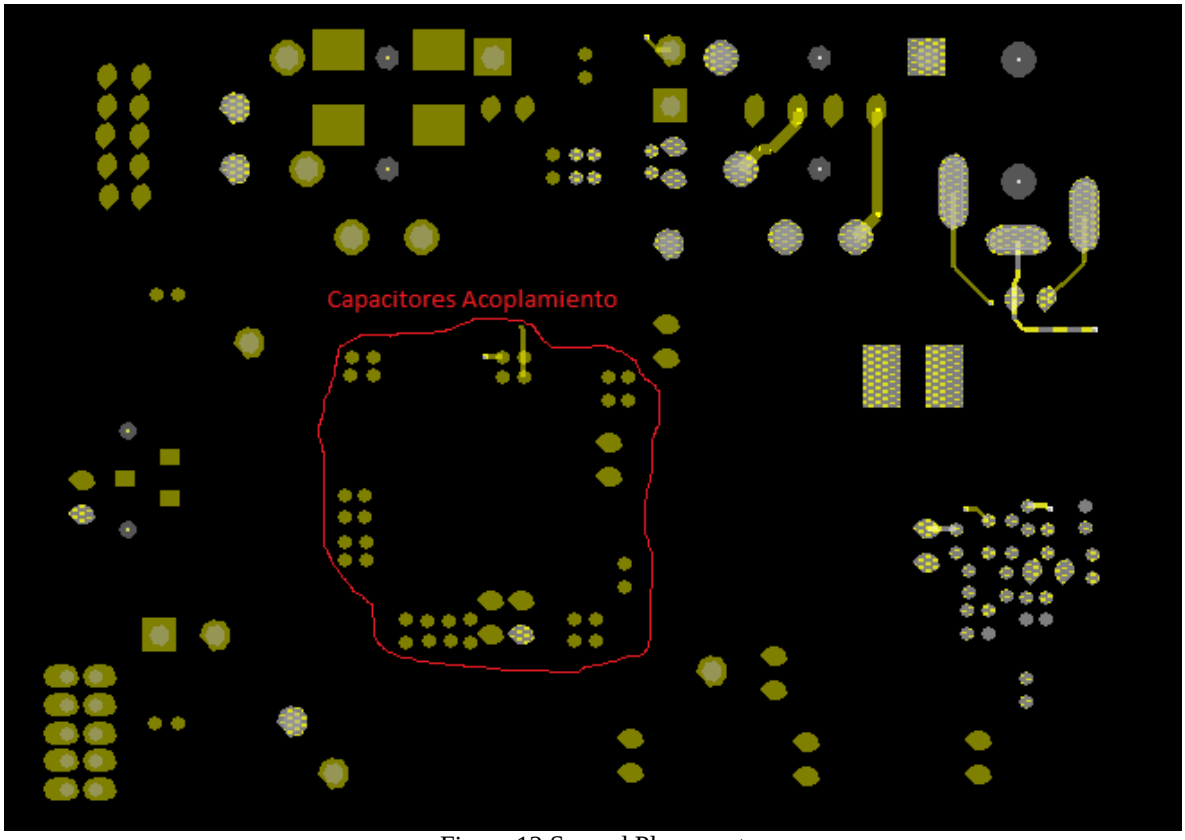


Figure 12 Second Placement

Since the bulks capacitors having voltage source, which are feed all the system. However each time a status change the bulks capacitors are switching 5V and 0V when they don't rise to give the necessary energy to micro they take energy of coupling capacitors.

Finally I clicked the buttom "Rats All" and it permit to visualize the relation between each components.

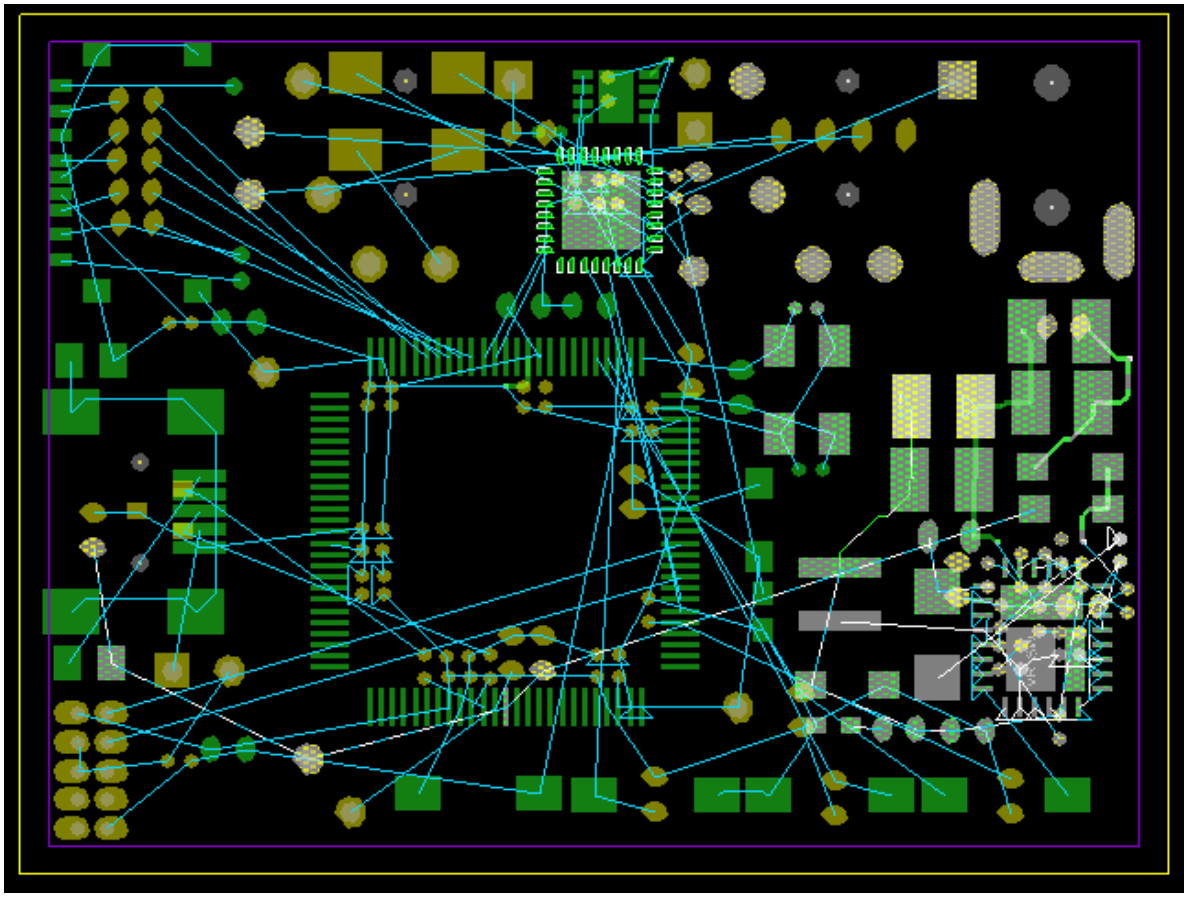


Figure 13 Rats

5.2. Routing

After the Placement I did the routing of MP3 Player PCB.

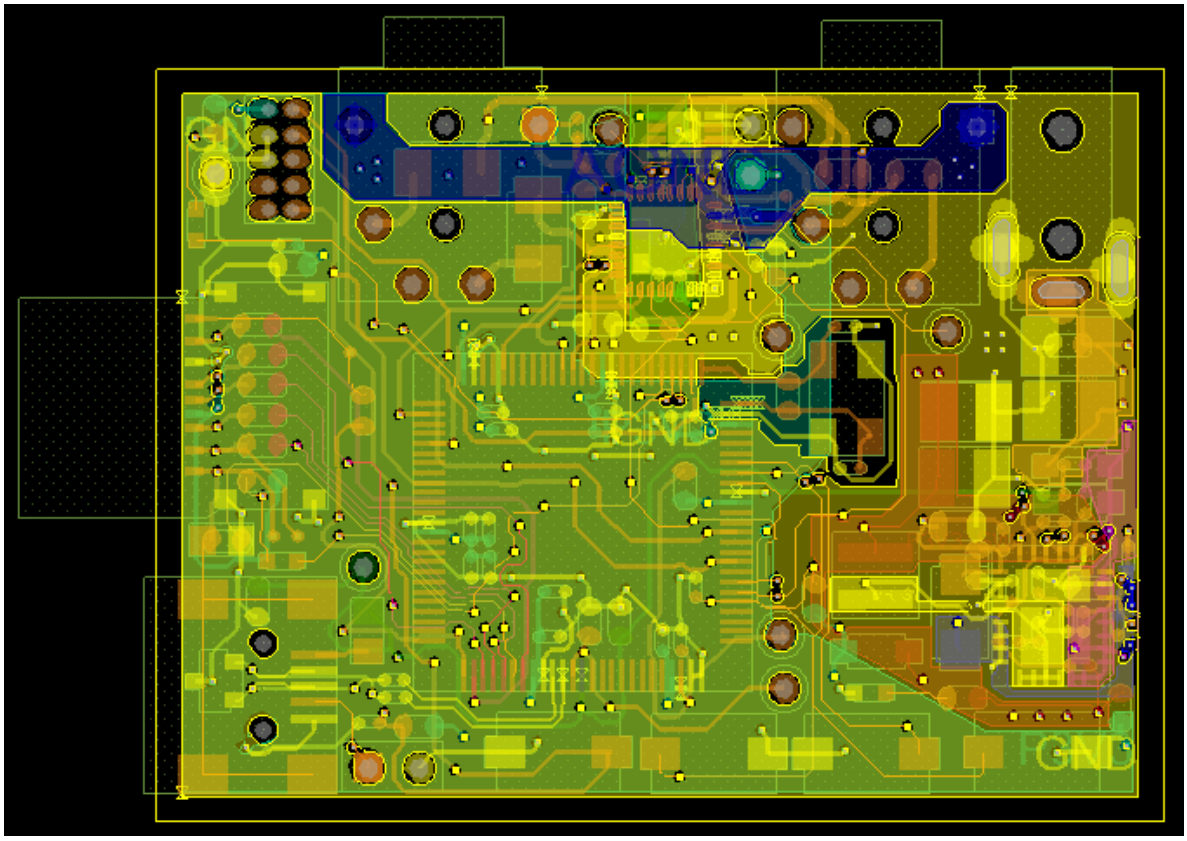


Figure 15 First Routing

After the Placement turned accommodate components MicroSD, because I was far away from the main connection microcontroller.

Then I connected the differential pair of USB 2.0 accommodating resistance so that I will not cross the lines.

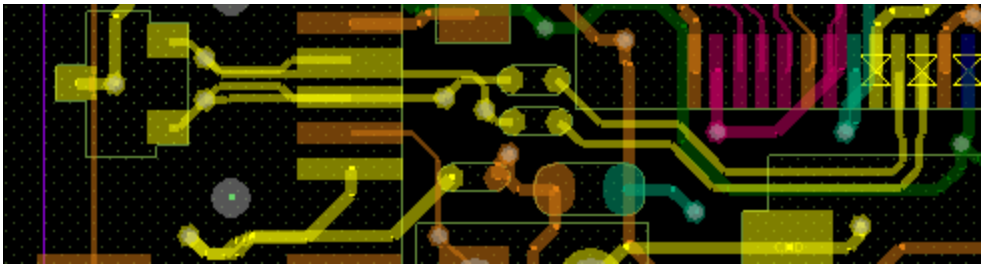


Figure 16 Usb Routing

After I connected the clock signals and the data on the MicroSD to the Microcontroller

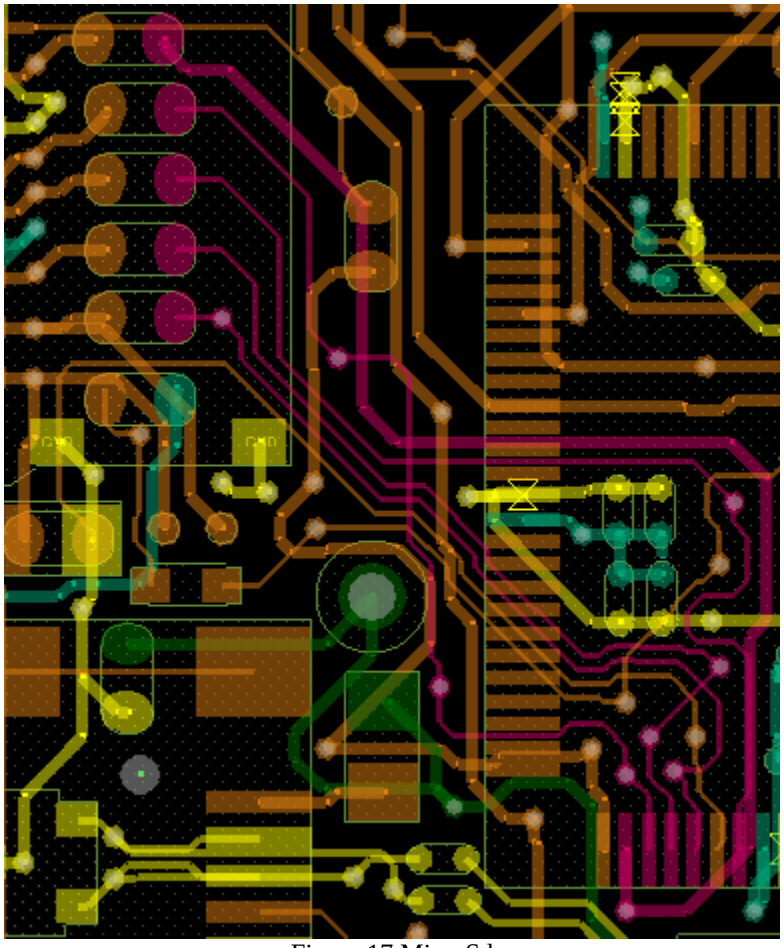


Figure 17 MicroSd

Later I defined the GND Powers

GND- Common Ground

AGND – Analog Earth

I placed the Resistencia Shunt in mode Start Connection together the GND.

I did a hole in the position of clock in order to eliminate noise.

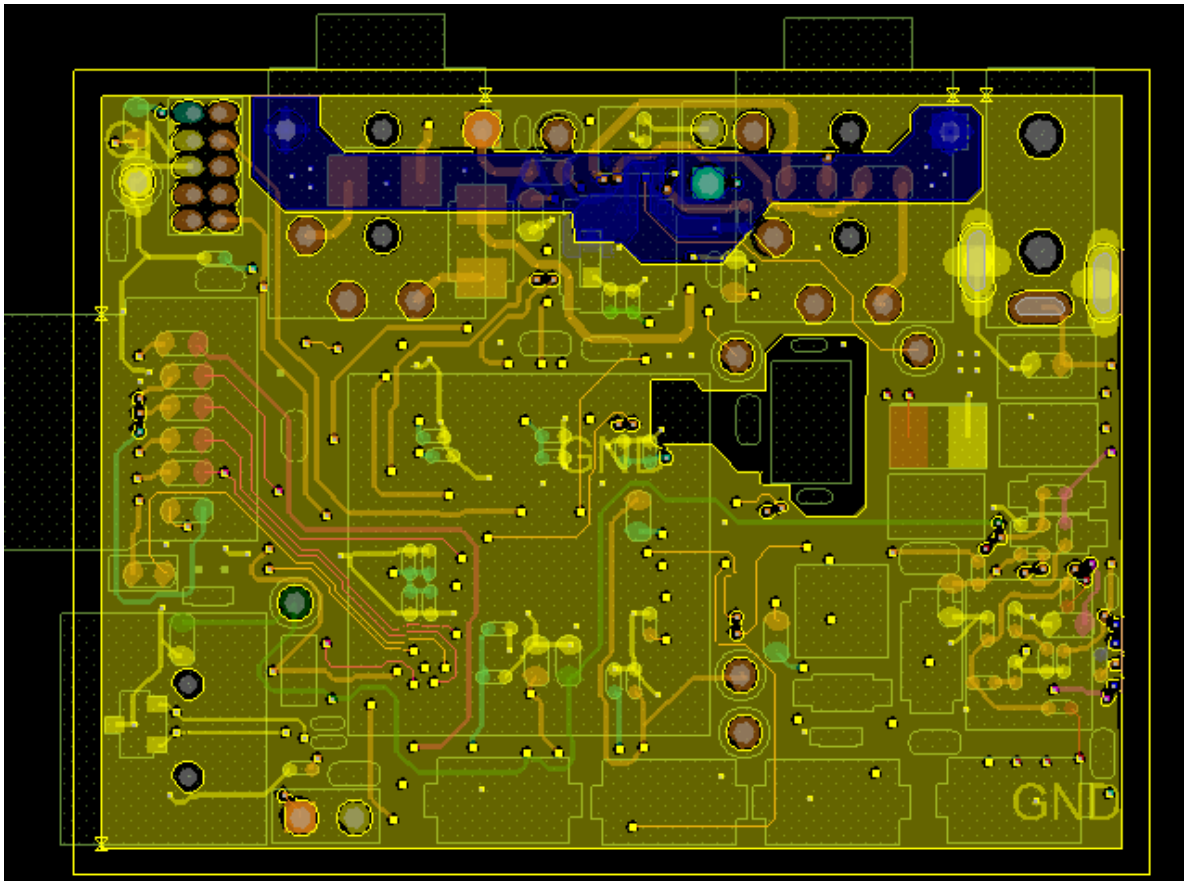


Figure 18 GND Layer

Connection to the Main Clock

It should have to connect near to the microcontroller.

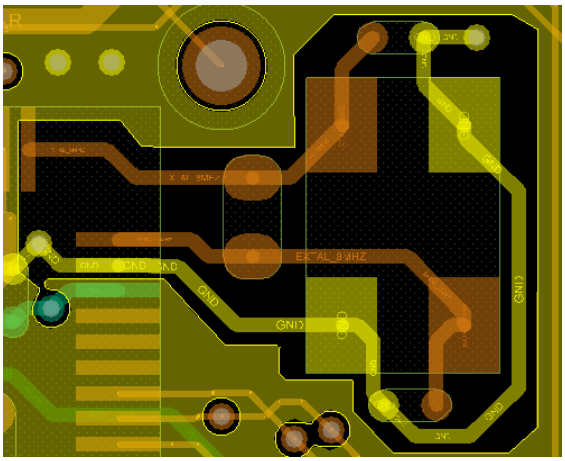


Figure 19 Clock

After I defined the voltage planes for both analog and digital.

Using the specifications of vendor.

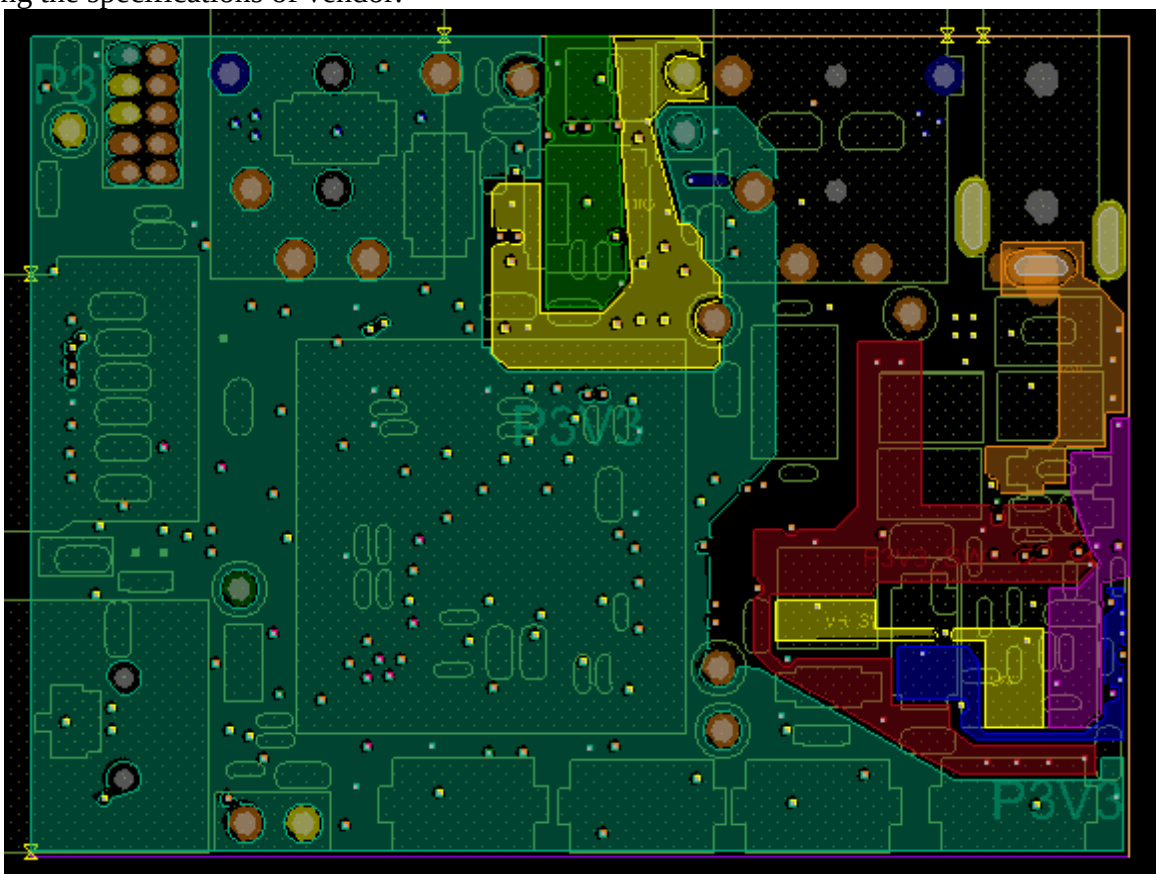


Figure 20 Voltage Layer

The layer was routed and connected the next manner:

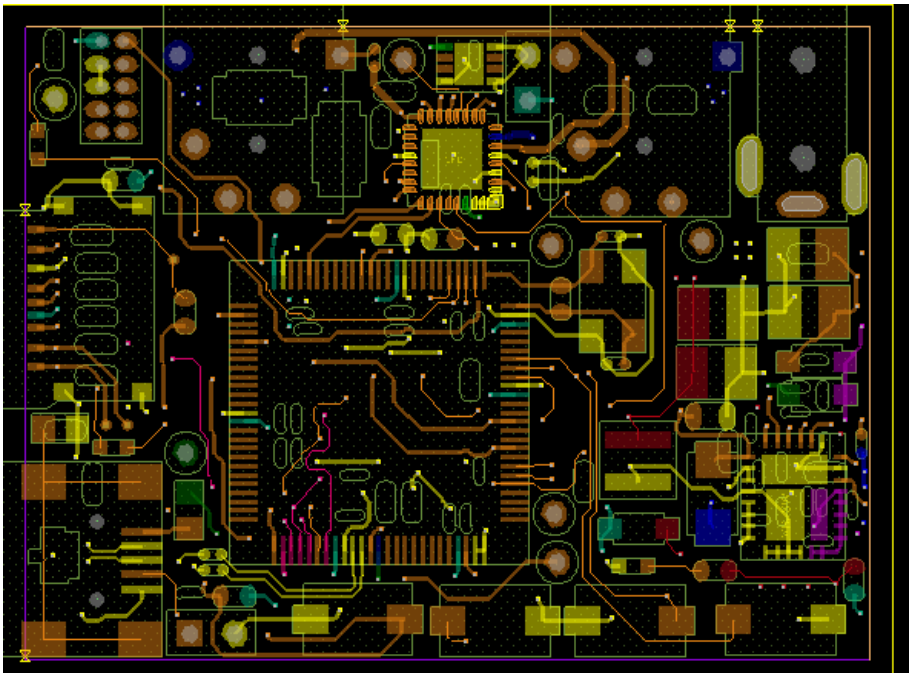


Figure 21 Top Layer

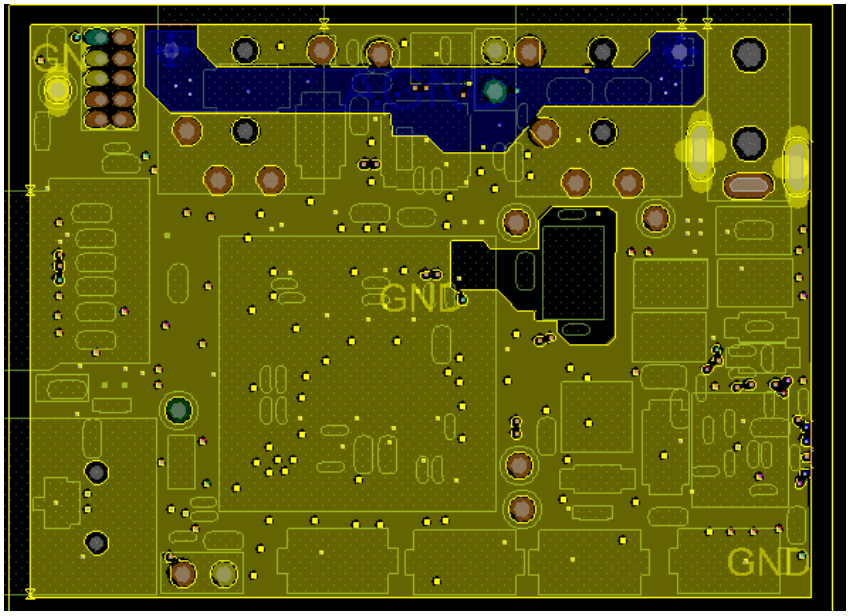


Figure 22 GND Layer

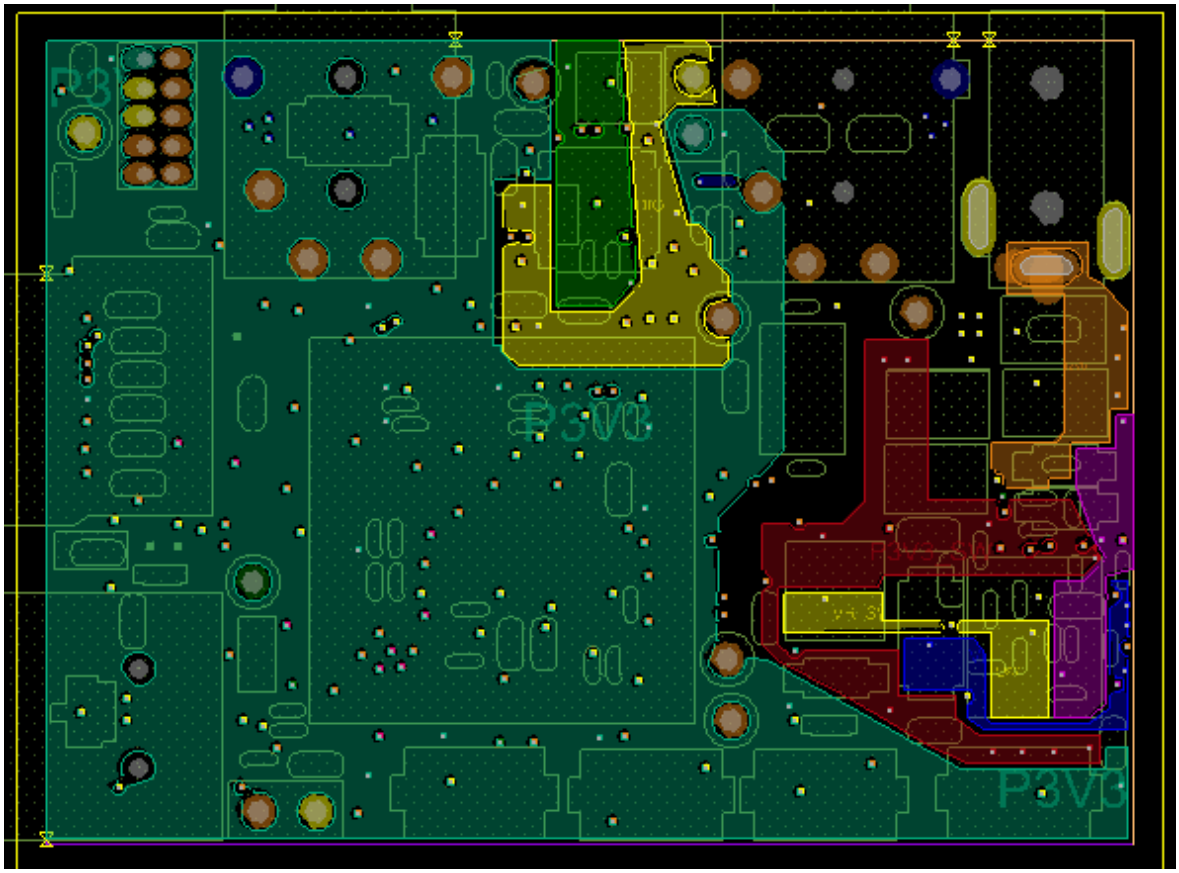


Figure 23 Power Layer

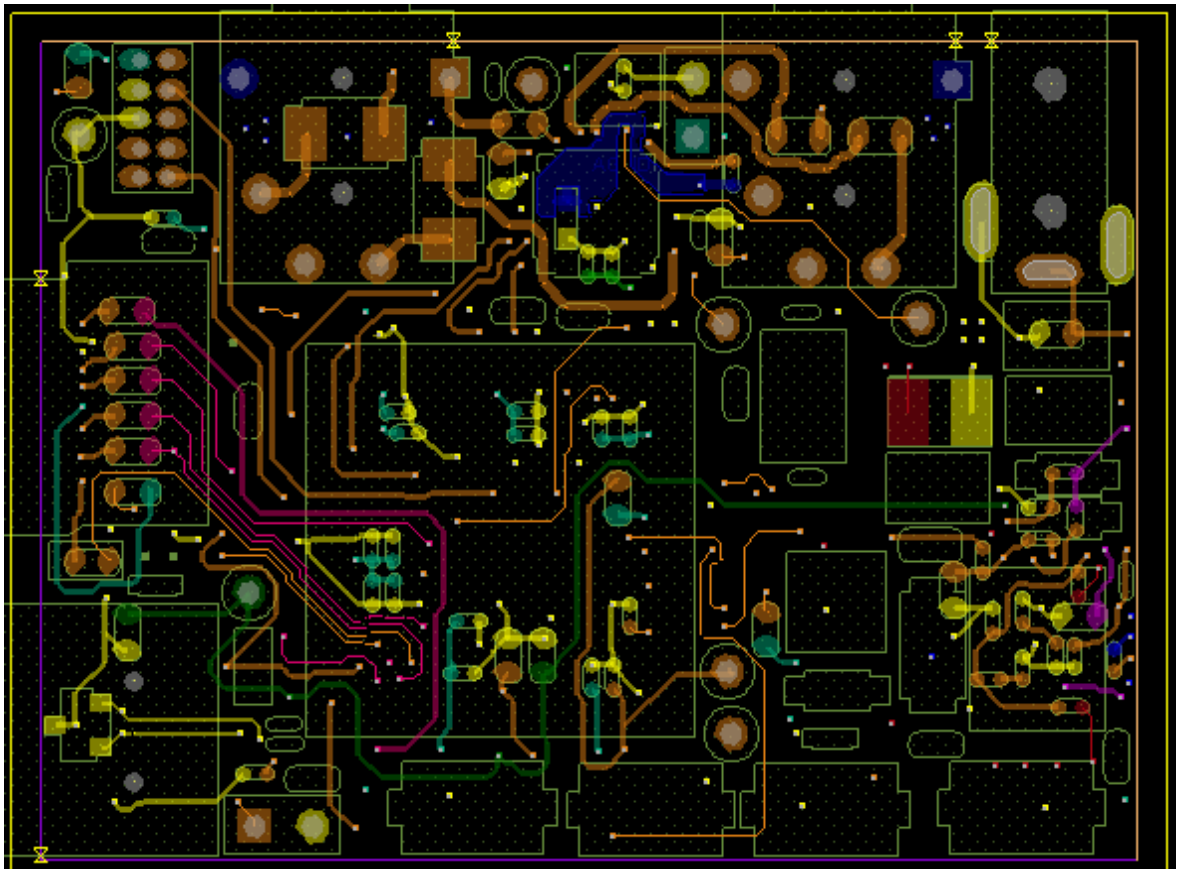


Figure 24 Bottom Layer

5.3. Bill of Material

Allegro permit to generate the bill of material.

My task was to find a vendor who could provide each component.

I established communication with Digikey and Newark Vendors.

ITESO
Periferico Sur Manuel Gomez Morin #8585
Tlaquepaque, Jalisco, Mexico 45604
+52(33) 3669-3434
portal@iteso.mx

Bill Of M Page1

KEY:
Newark part
Digikey part
Alternative part choosen
Part no needed to buy

Item	Quantity	Reference	Part	MFG_NAME01	MFG_PN01	ALTERNATIVE - MFG_I	DESCRIPTION	NEWARK PN	DIGIKEY PN	PRICE +1	TOTAL	PRICE 10+	TOTAL	TOTAL_Formul
1	6	C7,C9,C19,C514,C516,C51	1.0UF	YAGEO AMERICA	CC0402KRX5R6BB105		CAP CER 1.0UF 10V 10% X5R 0402	68R4631		0.125	0.75	0.095	0.57	0.75
2	1	C11	2.2UF	TAIYO YUDEN	LMK107B7225KA-T		CAP CER 2.2UF 10V 10% X7R 0603		587-2983-1-ND	0.17	0.17	0.115	0.12	0.17
3	11	C12,C15,C18,C22,C23,C40,C510,C511,C513,C515,C5	0.1UF	KEMET	C0402C104K8PACTU		CAP CER 0.1UF 10V 10% X5R 0402	78H4791		0.166	1.826	NA	NA	1.826
4	2	C26,C28	22PF	AVX	04025A220JAT2A		CAP CER 22PF 50V 5% C0G 0402	96M1143		0.209	0.418	0.15	0.3	0.418
5	2	C32,C41	1UF	TDK	CGA3E1X7R1E105K080AC		CAP CER 1uF 25V 10% X7R AEC-Q200 0603	04X3151		0.314	0.628	0.261	0.52	0.628
6	3	C39,C44,C495	2.2UF	AVX	08053C225KAT2A		CAP CER 2.2uF 25V 10% X7R 0805	10R5975		1.23	3.69	1.04	3.12	3.69
7	4	C46,C49,C117,C122	0.1UF	KEMET	C0402C104K4RACTU		CAP CER 0.1UF 16V 10% X7R 0402	70K3077		0.154	0.616	NA	NA	0.616
8	2	C113,C114	220uF	AVX	TLNT227M010R1300		CAP TANT 220UF ESR 1.300 OHM 10V 20% 1210		478-6612-1-ND	2.95	5.9	2.44	4.88	5.9
9	5	C115,C509,C512,C519,C5	10uF	TAIYO YUDEN	LMK107BJ106MALTD		CAP CER 10UF 10V 20% X5R 0603	84R9134		0.513	2.565	0.484	2.42	2.565
10	1	C116	3.3UF	KEMET	C0402C335M7PACTU		CAP CER 3.3uF 4V 20% X5R 0402	14N2170		0.474	0.474	0.414	0.41	0.474
11	3	C493,C494,C501	0.1uF	MURATA	GRM155R71H104KE14D	CGA2B3X7R1H104K050	CAP CER 0.1uF 50V 10% X7R 0402	04X3148		0.165	0.495	0.111	0.33	0.495
12	2	C496,C499	1uF	CAPAX TECHNOLOGIES INC.	0603X105K250SNT	CGA3E1X7R1E105K0	CAP CER 1UF 25V 10% X7R 0603	04X3151		0.314	0.628	0.261	0.52	0.628
13	5	C497,C498,C504,C505,C	22uF	MURATA	GRM32ER71E226KE15L		CAP CER 22uF 25V 10% X7R 1210	72R4217		2.26	11.3	NA	NA	11.3
14	1	C500	1000pF	CAPAX TECHNOLOGIES INC.	0402X102J500SNT	C1005C0G1H102J050BA	CAP CER 1000PF 50V X7R 5% 0402	04X3201		0.15	0.15	0.102	0.1	0.15
15	1	C502	4700PF	CAL-CHIP	GMC04X7R472J50NT-L	04025C472JAT2A	CAP CER 4700PF 50V X7R 5% 0402	04X3150	478-[illegible]	0.11	0.11	0.078	0.08	0.11
16	3	D4,D17,D18	YEL/GRN	Bright LED Electronics Corp	BL-HGE36D-TRB	VLMG1300-GS08	LED YEL GRN SGL 30MA 0603	76T8664		0.186	0.558	0.16	0.48	0.558
17	1	D8	B330A	DIODES INC	B330A-13-F		DIODE SCH RECT 3.0A 30V SMA	05R5878	B330A-[illegible]-ND	0.395	0.395	0.368	0.37	0.395
18	3	D14,D15,D16	MBR120VLSFT1G	ON SEMICONDUCTOR	MBR120VLSFT1G		DIODE SCH RECT 1A 20V SMT	10N9424		0.765	2.295	0.442	1.33	2.295
19	1	J8	CON microSD	MOLEX	1050270001		CON 8 SD CARD PUSH SMT 1.1MM SP 75H AU		WM6825CT-ND	1.43	1.43	1.26	1.26	1.43
20	1	J10	CONN USB MINI-	TE Connectivity Ltd	1734035-2		CON 5 USB_MINI_B_RECEPTACLE RA SMT 0.8MM 168H AU		A31727CT-ND	1.6	1.6	1.51	1.51	1.6
21	1	J11	HDR 2X5	ANYTRONIC CORPORATION LI	0922S0205011N6T-2LF	20021111-00010T4LF	HDR 2X5 TH 50MIL CTR 167H AU 91L	82R7568		0.812	0.812	0.732	0.73	0.812
22	2	J18,J48	HDR 1X2 TH	SAMTEC	TSW-102-07-G-S		HDR 1X2 TH 100MIL SP 339H AU 98L		SAM1029-02-ND	0.767	1.534	0.76	1.52	1.534
23	1	J24	CON PWR 3	CUI Inc	PJ1-022		CON 3 PWR JACK RA TH --- 201H AG 118L		CP1-022-ND	1.25	1.25	0.872	0.87	1.25
24	2	J47,J49	AUD_5	CUI STACK	SJ-43515TS		CON 5 AUD JACK 3.2MM SKT RA TH -- 197H SN 079L		CP-43515TS-ND	1.36	2.72	1.53	3.06	2.72
25	2	L1,L2	330 OHM	TDK	MPZ2012S331A	MPZ2012S331AT000	IND FER BEAD 330OHM@100MHZ 2.5A -- SMT		445-1569-1-ND	0.12	0.24	0.1	0.2	0.24
26	1	L57	2.2uH	coilcraft	XAL4020-222MEB	XAL4020-222MEC	IND PWR 2.2UH@100KHZ 5.6A 20% AEC-Q200 SMT	63W4102		2.76	2.76	2.61	2.61	2.76
27	2	R8,R10	33	THYE MING TECH CO LTD	CR-02FL6---33R	CRCW040233R0FKEA	RES MF 33.0 OHM 1/16W 1% 0402	94W4675	311-33.0[illegible]-ND	0.088	0.176	0.049	0.1	0.176
28	10	R17,R35,R37,R40,R42,R49,R649,R655,R659,R66	0	VISHAY INTERTECHNOLOGY	CRCW06030000Z0EA		RES MF ZERO OHM 1/10W -- 0603	52K8014		0.095	0.95	0.041	0.41	0.41
29	1	R31	1.0M	BOURNS	CR0603-JW-105ELF		RES MF 1.0M 1/10W 5% 0603		CR0603-JW-105ELFCT	0.1	0.1	0.029	0.03	0.1
30	3	R53,R650,R651	10K	KOA SPEER	RK73B1JTTD103J	CRCW060310K0JNEA	RES MF 10K 1/10W 5% 0603	52K8066		0.073	0.219	0.031	0.09	0.219
31	3	R72,R657,R658	560	KOA SPEER	RK73H1JTTD5600F	ERJ-3EKF5600V	RES MF 560 OHM 1/10W 1% 0603	64R5353		0.141	0.423	NA	NA	0.423
32	1	R266	2.2K	VENKEL COMPANY	CR0402-16W-222JT	CRCW04022K20FKEA	RES MF 2.2K 1/16W 5% 0402	41B4528	RMCF0402JT2K20CT-	0.1	0.1	NA	NA	0.1
33	1	R635	~~2.21~~ 2.20	KOA SPEER	RK73H1ETTP2R21F	MC00625W040212R2	RES MF 2.21 OHM 1/16W 1% 0402	94W3271	541-2.21[illegible]-ND	0.009	0.009	NA	NA	0.009
34	3	R636,R644,R645	~~1.21~~ 1.20	KOA SPEER	RK73H1ETTP1R21F	MC00625W040211R20	RES MF 1.21 OHM 1/16W 1% 0402	94W3197		0.009	0.027	NA	NA	0.027
35	1	R637	2	KOA SPEER	RK73H1ETTP2R00F	CRCW04022R00FKED	RES MF 2.0 OHM 1/16W 1% 0402	47P3315		0.08	0.08	0.044	0.04	0.08
36	2	R638,R656	10.0K	VISHAY INTERTECHNOLOGY	CRCW040210K0FKED		RES MF 10.0K 1/16W 1% AEC-Q200 0402	52K6595		0.086	0.172	0.019	0.04	0.172
37	1	R641	19.6K	KOA SPEER	RK73H1ETTP1962F	CRCW040219K6FKED	RES MF 19.6K 1/16W 1% 0402	52K6774		0.118	0.118	0.065	0.07	0.118
38	1	R647	2.49K	BOURNS	CR0402-FX-2491GLF		RES MF 2.49K 1/16W 1% 0402		CR0402-FX-2491GLFC	0.1	0.1	0.089	0.09	0.1
39	1	R648	806	KOA SPEER	RK73H1ETTP8060F	ERJ-2RKF8060X	RES MF 806 OHM 1/16W 1% 0402	53W4471		0.148	0.148	NA	NA	0.148
40	1	SH507	0	LAYOUT ELEMENT ONLY	LAYOUT ELEMENT ONLY		ZERO OHM CUT TRACE 0603 PADS; NO PART TO ORDER							0
41	4	SW3,SW4,SW5,SW6	EVQ-PE105K	PANASONIC	EVQPE105K		SW SPST NO 0.05A 12V SMD		P13594SCT-ND	0.72	2.88	0.67	2.68	2.88
42	7	TP5,TP31,TP35,TP36,TP37,TP38,TP39	TEST POINT WHI	COMPONENTS CORPORATION	TP-105-01-09		TEST POINT WHITE 40 MIL DRILL 180 MIL TH 109L							0
43	3	TP20,TP22,TP24	TPAD_030	NOTACOMPONENT	NOTACOMPONENT		TEST POINT PAD 30MIL DIA SMT, NO PART TO ORDER							0
44	1	U3	GSOT05C-GS08	VISHAY INTERTECHNOLOGY	GSOT05C-GS08		DIODE ESD 2LINE 5V SOT23		GSOT05C-E3-08CT-ND	0.49	0.49	3.45	3.45	0.49
45	1	U14	SGTL5000 32QFN	FREESCALE SEMICONDUCTOR	SGTL5000XNAA3R2	SGTL5000XNAA3	IC AUDIO CODEC STEREO 8-27MHZ 1.8-3.3V QFN3:	14R9109		3.19	3.19	3.02	3.02	3.19
46	1	U17	AP7361	Diodes Inc	AP7361-18FGE-7		IC VREG LDO 1.8V 1A 2.2-6V DFN3030-8		AP7361-18FGE-7DICT-N	0.59	0.59	0.494	0.49	0.59
47	1	U61	MIC24051	MICREL INC	MIC24051YJL	MIC24051YJL TR	IC REG BUCK SYNC ADJ 6A 28QFN	06X6397	576-[illegible]-ND	2.81	2.81	NA	NA	2.81
48	1	U62	MK64FN1M0VLL	FREESCALE SEMICONDUCTOR	MK64FN1M0VLL12		IC MCU 1MB FLASH 120MHZ 1.71-3.6V LQFP100							0
49	1	Y1	8MHZ	Abracon Corporation	ABM3B-8.000MHZ-B2-T		XTAL 8MHZ 20PPM PAR 18PF ESR 200OHM SMT	44P2236		0.942	0.942	0.833	0.83	0.942
														58.298
														894.29

Item	$1	q1	$2	q2	$3	q3	$4	q4	$5	q5	$6	q6	$7	q7	$8	q8	$9	q9	$10	q10	$11	q11
2	0.2	1	0.3	2	0.1	3	0.7	4	3.4	5	1	6	1.2	7	1.4	8	1.5	9	1.7	10	1.87	11
8	5.9	2	12	4	70	6	24	8	118	10	35	12	41	14	47	16	53	18	59	20	64.9	22
19	1.4	1	2.9	2	4.1	3	5.7	4	29	5	8.6	6	10	7	11	8	13	9	14	10	15.73	11
20	1.6	1	3.2	2	5.1	3	6.4	4	32	5	9.6	6	11	7	13	8	14	9	16	10	17.6	11
23	1.3	1	2.5	2	3.1	3	5	4	25	5	7.5	6	8.8	7	10	8	11	9	13	10	13.75	11
24	2.7	2	5.4	4	15	6	11	8	54	10	16	12	19	14	22	16	24	18	27	20	29.92	22
25	0.2	2	0.5	4	0.1	6	1	8	4.8	10	1.4	12	1.7	14	1.9	16	2.2	18	2.4	20	2.64	22
29	0.1	1	0.2	2	0	3	0.4	4	2	5	0.6	6	0.7	7	0.8	8	0.9	9	1	10	1.1	11
32	0.1	1	0.2	2	0	3	0.4	4	2	5	0.6	6	0.7	7	0.8	8	0.9	9	1	10	1.1	11
38	0.1	1	0.2	2	0	3	0.4	4	2	5	0.6	6	0.7	7	0.8	8	0.9	9	1	10	1.1	11
41	2.9	4	5.8	8	17	12	12	16	58	20	17	24	20	28	23	32	26	36	29	40	31.68	44
44	0.5	1	1	2	0.5	3	2	4	9.8	5	2.9	6	3.4	7	3.9	8	4.4	9	4.9	10	5.39	11
46	0.6	1	1.2	2	0.7	3	2.4	4	12	5	3.5	6	4.1	7	4.7	8	5.3	9	5.9	10	6.49	11
	18	19	35	38	115	57	70	76	351	95	105	114	123	133	141	152	158	171	176	190	193.27	209

Figure 25 Bill of Material

6. Conclusions

The Requirements Specification establishes the basis for agreement between the customers and the suppliers on what the product is to do. In this case the customer didn't have technical knowledge for describe the product specifications. This fact adds complexity to the stages of requirements and design. The system requirements specification can be achieved through focusing in understand the situation and the needs of the customer. As a result of this approach it can considered that the main goals of this assignment have been fulfilled. Other important thing about requirements in projects where the delivery time is short is necessary achieve all the obligatory requirements. If the time is sufficient all the other requirements can be fulfilled.

I learned that is very important to do very well the placement if we don't want to have problems with the routing stage.

Is necessary to read the documentation about IPC Standards if we have to save time when we want to build a PCB.

I think it was a good experience to design a pcb with the mentors of continental engineers they have a lot of experience using the software tool and knowledges about electrical, high speed and transmission lines.

7. Bibliography

OSHPARK is a PCB Manufacturer

https://oshpark.com/

IPC Association Connecting Electronic Industries

http://www.ipc.org/

Vendors of Electronic Components

http://mexico.newark.com/

http://www.digikey.com.mx/

Freescale Low Power Stereo Codec with Headphone Amp

http://www.nxp.com/files/analog/doc/data_sheet/SGTL5000.pdf

Kinetis K64F Sub-Family Data Sheet

http://cache.freescale.com/files/microcontrollers/doc/data_sheet/K64P144M120SF5.pdf

B. DEVELOPMENT OF DOCUMENTATION OF SOFTWARE REQUIREMENTS SPECIFICATIONS FOR AUTOMOTIVE COMMUNICATION PROTOCOL.

Content

1. Purpose and Scope

This document specifies the software requirements for the Data Link Layer and the Physical Layer of the LIN Protocol according with the OSI reference model (***Figure 1.1)***

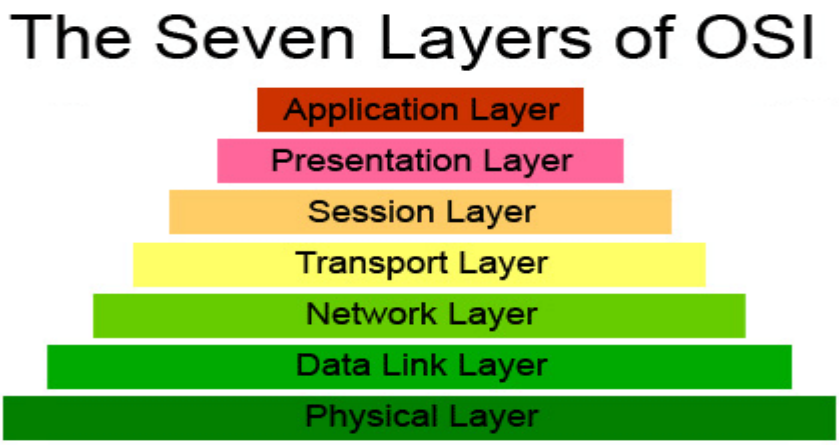


Figure 1.1 (OSI Layers)

The software layers mentioned before will be addressed in the following Basic Software Modules:

A) LIN Physical Layer (Lin)

B) LIN Data Link Layer

The contents of each layer is described in the next (***Figure 1.2***)

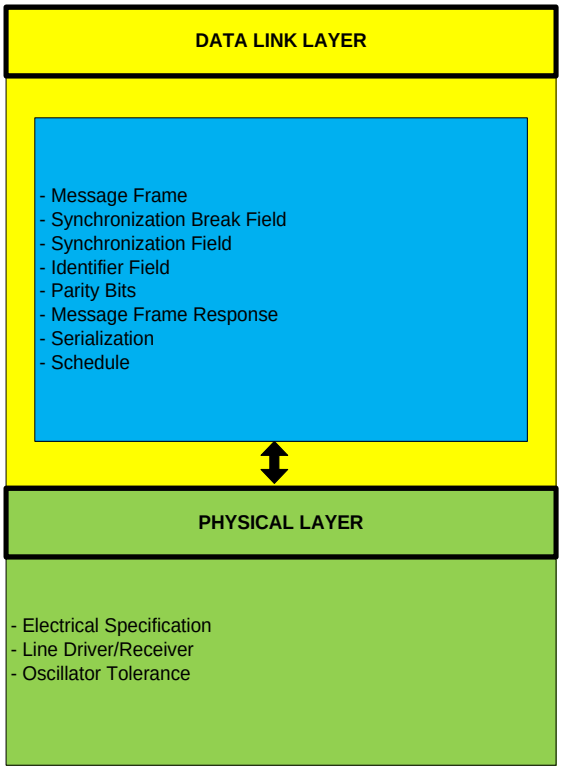


Figure 1.2 (LIN Protocol)

2. ¿What is the problem?

Over the past decades, increasing numbers of functions have developed for motor vehicles, which are intended to make car driving more comfortable, safer and environmentally friendly. In the process, more and more functions are implemented with electronic components, which have a growing need for the exchange of information. These electronic components include both ECUS and sensors and actuators.

For a long time, it was usual to connect sensors and actuators to an ECU via individual wires (see ***Figure 1.3*** Conventional Networking). However the rising number of connections led to thicker and heavier wire harnesses, which had greater weight and space requirements. Furthermore, it was more complicated to produce them for different vehicle versions, because too many customs modifications had to be made. Moreover, the increased number of lines made the systems more

susceptible to errors. Finally, these problems led to continually growing costs for vehicles networking.

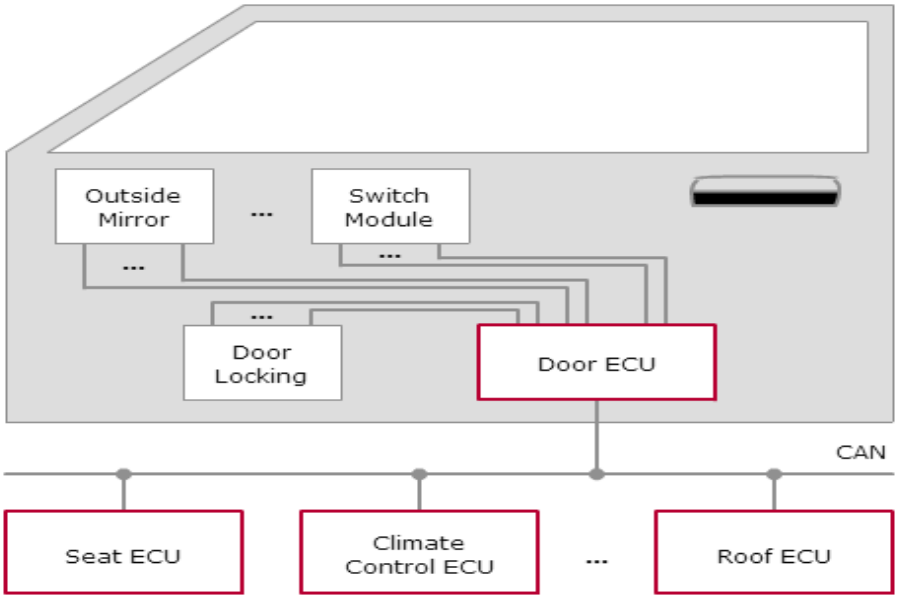


Figure 1.3 Conventional Networking

In 1983 CAN was developed by Robert GmbH, and in 1986 it was presented as a new serial bus system at the SAE Congress. The introduction of CAN enabled bit-serial transmission of multiple signals on one wire pair, which made it possible to reduce the number of lines needed. Nonetheless, CAN was too expensive for connecting cost-effective sensors and actuators in the convenience area. In the mid-1990s some automotive OEMS and suppliers began to develop more economical solutions. However, since these proprietary bus systems involved low part volumes, they only offered limited-cost relief. That is why some OEMS joined together to form the LIN Consortium whose goal was to develop a uniform, standardized and cost-effective communication system (***Figure 1.4***). So they require an hierarchical vehicle network in order to gain further quality enhancement and cost reduction of vehicles. Also the standardization will reduce the manifold of existing low-end multiplex solutions and will cut the cost development, production, service and logistics in vehicle electronics.

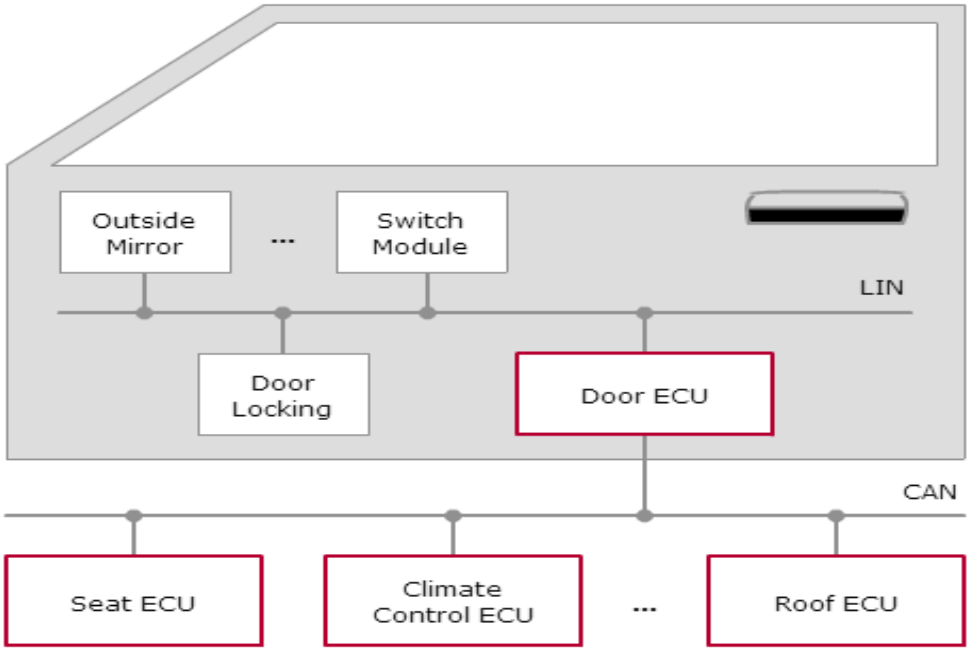


Figure 1.3 LIN Networking

2.1. LIN Protocol Overview

LIN (Local Interconnect Network) is a concept of low cost automotive networks, which complements the existing portfolio of automotive multiplex networks.

The LIN bus is a sub-bus system based on a serial communications protocol. The bus is a single master / multiple slave bus that uses a single wire to transmit data.

To reduce costs, components can be driven without crystal or ceramic resonators. Time synchronization permits the correct transmission and reception of data. The system is based on a UART / SCI hardware interface that is common to most microcontrollers.

The bus detects defective nodes in the network. Data checksum and parity check guarantee safety and error detection.

A LIN Cluster consists of a number of nodes that are interconnected via physical transmission medium (see ***Figure 1.4*** Network). A general distinction is made between two types of nodes. There is always a master that controls bus access, and there are multiple slaves that can send and receive information.

For cost reasons, no dedicated communication controller was implemented.

Instead, the protocol must be integrated as a software component in the microcontroller. It should be clearly noted whether a node is to be implemented as the master or slave. The microcontroller is connected to the transceiver via the serial interface. This interface is referred to as the SCI (Interface Communication Serial).

The transceiver serves to physically connect the bus to the network. This chip converts the logical bit sequence into transmittable bus levels and in the reverse direction. The transceiver has a TX section and an RX section for this purpose.

While the TX section is used to generate the voltages on the bus, the RX section enables evaluation of the received levels. In addition, the transceiver has a mechanism by which a node can be awakened via the bus. This is referred to as the Wakeup.

In LIN physical signal transmission only requires one conductor (single wire) to maintain radiate electrical emissions within limits, the transmission rate is limited to *20kBit/s* for LIN. Another restriction is the maximum recommended number of 16 nodes.

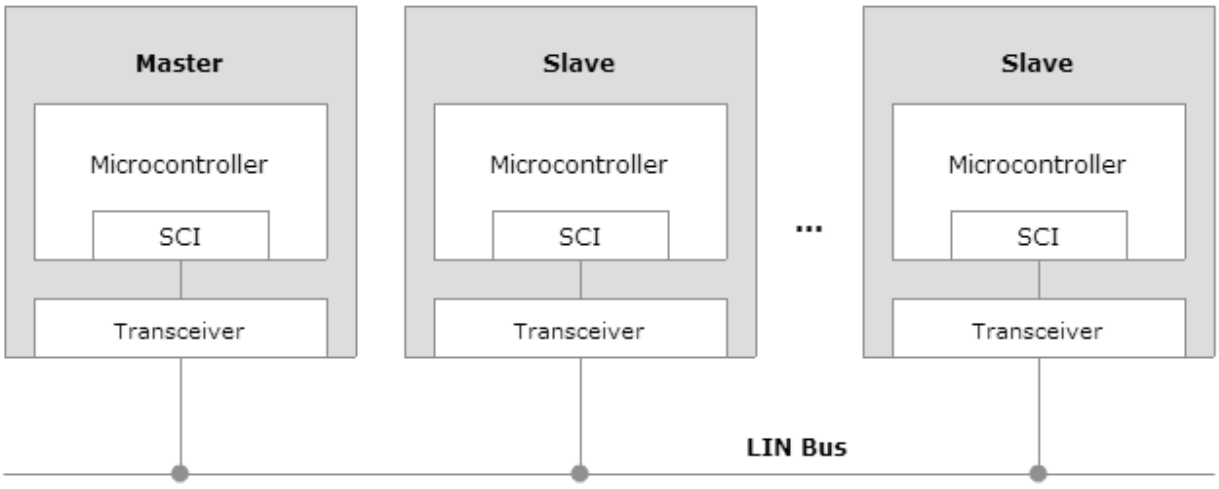


Figure 1.4 LIN Network

The LIN protocol has selectable length of *MESSAGE FRAME* : 0 to 8 bytes and a Data-Checksum security ,and error detection.

2.2. Requirement Structure

There are many kind of requirements such as see ***Figure 1.5***:

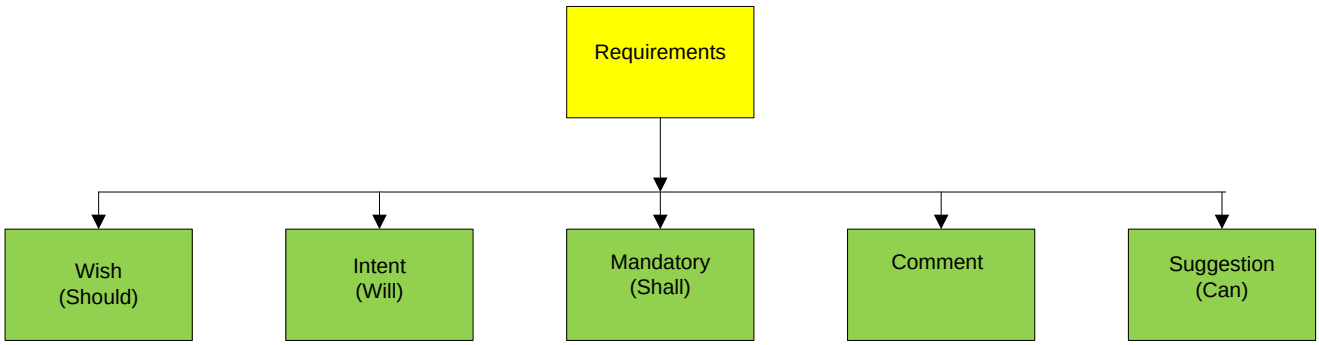


Figure 1.5 Different kinds of requirements.

Each chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines. Definitions for LIN are divided in four chapters.

A) LIN General requirements

B) LIN Interface (Master Node Handler)

- Master Task
- Slave Task Master
- Framing
- Synchronization Break Field
- Synchronization Field
- Identifier Field
- Parity Bits
- Message Frame Response
- Checksum
- Scheduler

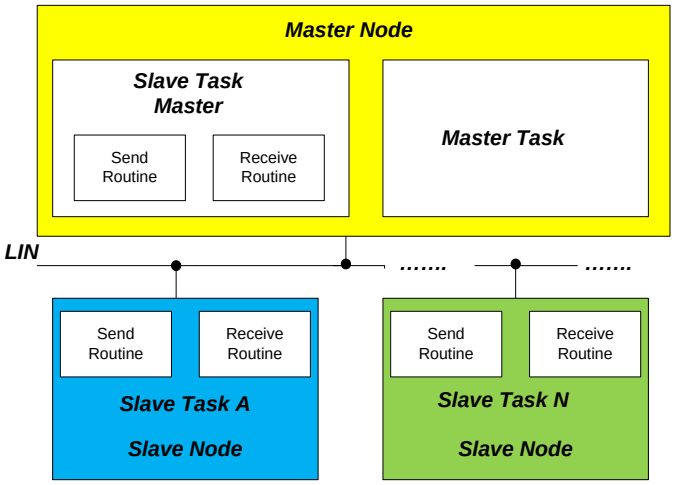
C) LIN Physical Layer

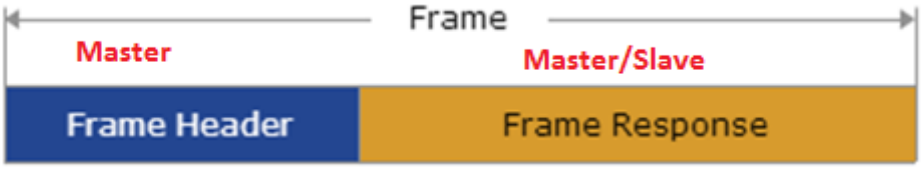
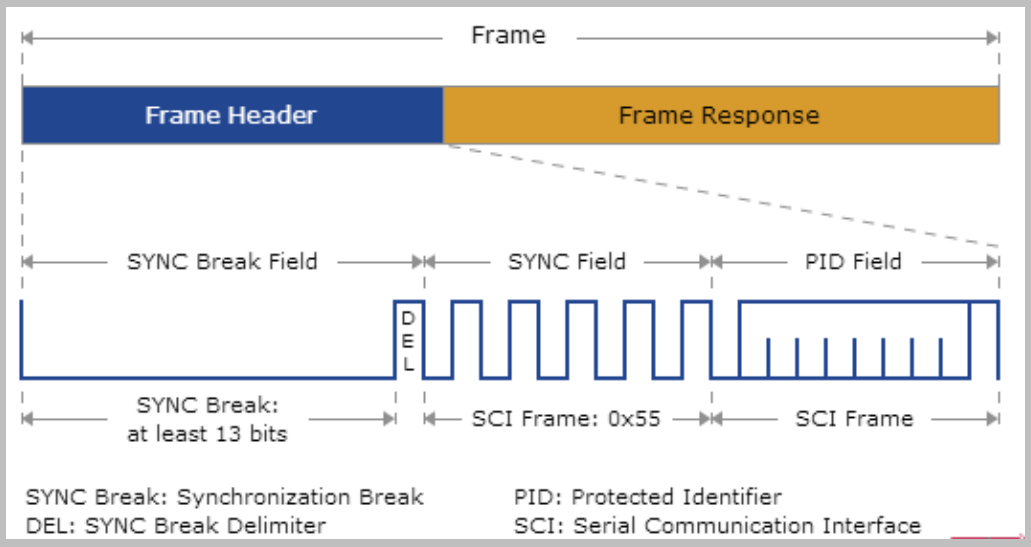
- Serialization

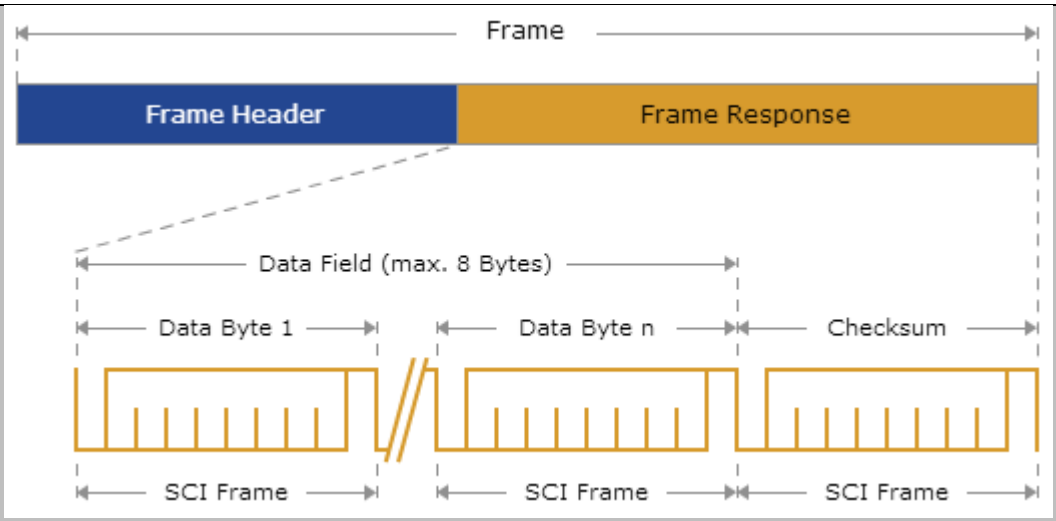
3. LIN Master Node

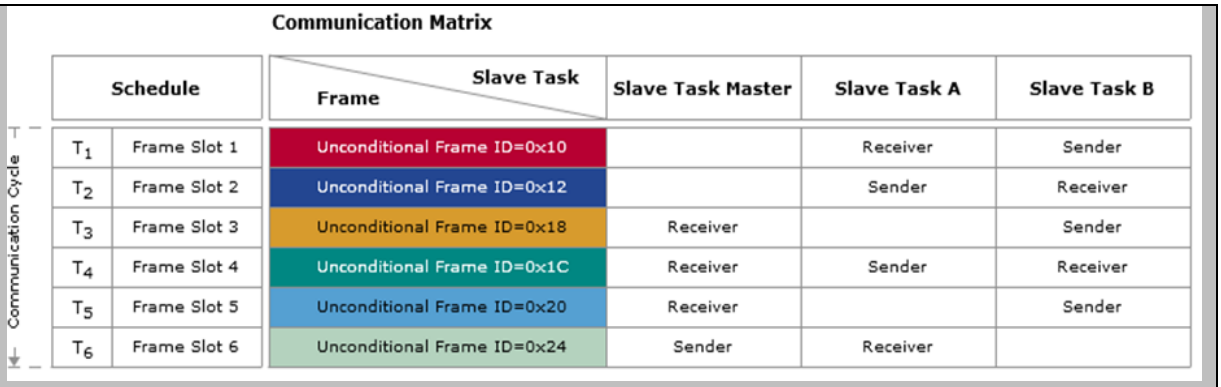
3.1. Functional Requirements

3.1.1 LIN Frame

ID	Functional Requirements
<R1>	The LIN bus ***shall*** be a system based on a serial communications protocol.
<R2>	The Master Node ***shall*** be implemented in a *master task* and a *slave task* ***(Figure 1.6).***  *Figure 1.6 Lin Network*
<R3>	The LIN system configuration ***shall*** be implemented in a single master node / multiple slave nodes ***(Figure 1.6).***
<R4>	The master node ***shall*** have two separated tasks the *master task* and *slave task* master in order to control the traffic on the LIN bus ***(Figure 1.6).***
<R5>	A node in LIN systems ***shall not*** use any information about the system configuration, except for the denomination of the single master node ***(Figure 1.6).***
<R6>	The system ***shall*** have the capability for add new nodes to the network without requiring hardware or software changes in other slave nodes.
<R7>	The system ***shall*** have the capability of simultaneously receive and act upon messages in any number of nodes (it can be achieved thanks to the message filtering).

<R8>	The LIN *Master Task* ***shall*** transmit the message header and one *Slave Task* respond this header (***Figure 1.7).***  *Figure 1.7 Message Frame*
<R9>	The message frame ***shall*** have two components the *header* from the master and the response from *slave* see (***Figure 1.7).***
<R10>	The master task ***shall*** send the header on the LIN Bus.
<R11>	Each Frame Header ***shall*** start with a break signal and is followed by a synchronization field and an identifier field (***Figure 1.8***). In some cases the data field and the check field are sent by the master too.  *Figure 1.8 Message Frame (Header)*
<R12>	The synchronization break ***shall*** enable the slave nodes which have lost synchronization to identify the synchronization field.
<R13>	Slaves ***shall*** use a frame response to answer a request by the master (***Figure 1.9)***

	 *Figure 1.9 Message Frame (Response)*
<R14>	The frame response ***shall*** have a maximum of 8 data bytes (***Figure 1.9***).
<R15>	The useful data ***shall*** be validated by a checksum in order to guarantee the integrity of the transmitted/received data through the LIN bus (***Figure 1.9***).
<R16>	The slave task ***shall*** send back the data field and the check field.
<R17>	The master node ***shall*** be able to trigger a slave to slave communication using a corresponding message ID.
<R18>	The LIN system ***shall*** be capable to synchronize all the nodes in the network in order to ensure the correct transmission and reception of data, it can be achieved measuring the distance between each falling edge in the synchronization field of the message frame.
<R19>	The LIN system ***shall*** monitor the transmitter values on the bus in order to perform an Error Detection process.
<R20>	The System LIN ***shall*** guarantee of latency times for signal transmission.
<R21>	The Master Task ***shall*** be responsible for generations the scheduling of the Frames ***(Figure 2.0)***

	Communication Matrix  Communication Cycle Schedule / Frame / Slave Task Slave Task Master / Slave Task A / Slave Task B T_1 Frame Slot 1 — Unconditional Frame ID=0x10 — Receiver — Sender T_2 Frame Slot 2 — Unconditional Frame ID=0x12 — Sender — Receiver T_3 Frame Slot 3 — Unconditional Frame ID=0x18 — Receiver — Sender T_4 Frame Slot 4 — Unconditional Frame ID=0x1C — Receiver — Sender — Receiver T_5 Frame Slot 5 — Unconditional Frame ID=0x20 — Receiver — Sender T_6 Frame Slot 6 — Unconditional Frame ID=0x24 — Sender — Receiver *Figure 2.0 Scheduler of Master Task*
<R22>	In case a master has detected an inconsistency the master ***shall*** change the message schedule.
<R23>	In case a slave has detected an inconsistency the slave controller ***shall*** save this information.
<R24>	The slave controller ***shall*** be able to provide it on request to the master control unit in form of diagnostic information if **<R23>** occurred.
<R25>	The LIN system ***shall*** have a low power mode (sleep mode) in order to reduce the system's power consumption.
<R26>	The LIN system ***shall*** have a dedicated command for going to sleep mode.
<R27>	The LIN system ***shall*** be recessive during sleep mode.
<R28>	The sleep mode ***shall*** be finished if any dominant period of a minimum length on the LIN Bus occurs.
<R29>	The sleep mode ***shall*** be finished by internal conditions in any bus node.
<R30>	In case of node-internal wake up, a procedure based on use of the WAKE UP SIGNAL ***shall*** be used for alerting the master.
<R31>	On wake up the internal activity of the system ***shall*** be restarted.

3.1.2 LIN Interface (Master Node Handler)

ID	Functional Requirements
<R32>	The Master Task ***shall*** be responsible to generate Synchronization Field ***(Figure 2.0).***
<R33>	The Master Task ***shall*** be responsible to generate Parity Identifier Field ***(Figure 2.0).***
<R34>	When the header is complete the master task ***shall*** send the frame to the LIN BUS.
<R35>	The Master Task ***shall*** be responsible to maintain the correct timing between the frames **.**

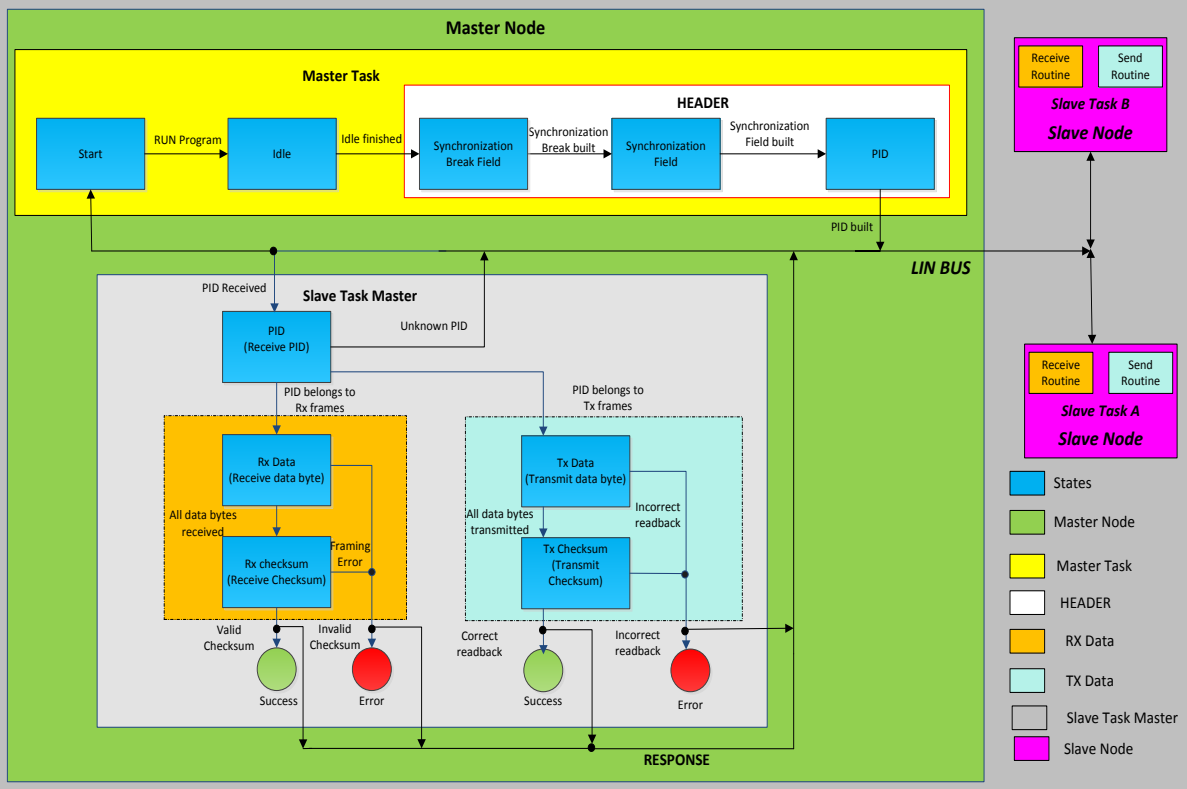


Figure 2.0 State Machine of Master Node

3.1.3 Slave Task Master

ID	Functional Requirements
<R36>	The Slave Task Master ***shall*** transmit the response when it is a sender ***(Figure 2.1).***
<R37>	The Slave Node ***shall*** receive the response when it is a receiver ***(Figure 2.1).***
<R38>	Once received the data from the LIN Bus the PID ***shall*** be capable to identify if the PID known ***(Figure 2.1).***
<R39>	The Slave Task Master ***shall*** be capable to send a response when a corresponding PID sent by the master task is requesting ***(Figure 2.1).***
<R40>	Once received data from the PID ***shall*** be capable to identify rx or tx frame ***(Figure 2.1).***
<R41>	The Rx data ***shall*** be capable to send a response when the data are received ***(Figure 2.1).***
<R42>	After all data are received a checksum byte ***shall*** be validated ***(Figure 1.8).***

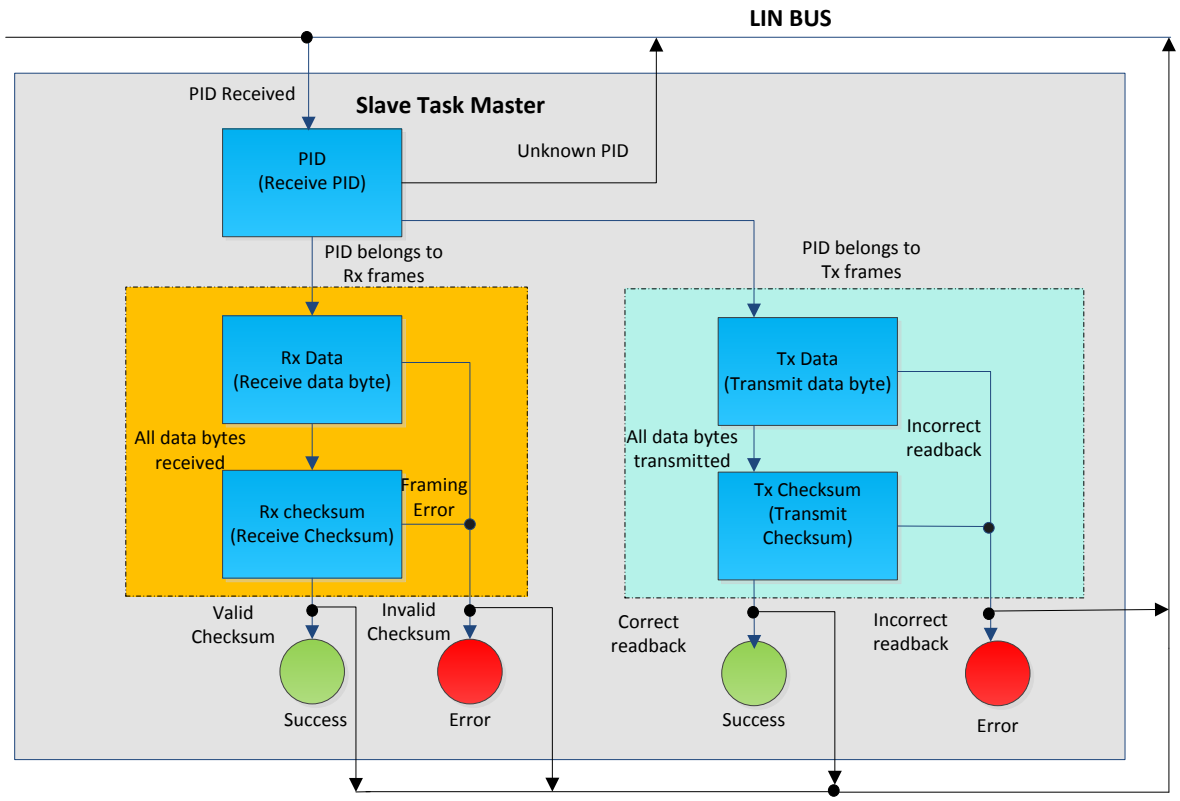


Figure 2.1 Slave Task Master state machine

<R43>	A message ***shall*** be validated only if there is no error detected until the end of the frame ***(Figure 2.1).***
<R44>	The Rx Checksum ***shall*** send an invalid or valid checksum to the LIN Bus ***(Figure 2.1).***
<R45>	Slave task master ***shall*** do nothing when the PID sent by the master task are not appropriate ***(Figure 2.1).***
<R46>	The PID ***shall*** be ensured by the network configuration in order to avoid that more than one slave task is unintentionally responding on a transmitted identifier ***(Figure 2.1).***
<R47>	After all data byes are transmitted a checksum byte ***shall*** be validated ***(Figure 2.1).***
<R48>	Tx data ***shall*** send an incorrect readback message to the LIN BUS, if all data bytes are not transmitted ***(Figure 2.1).***
<R49>	The TX checksum ***shall*** send a correct readback or incorrect readback to the LIN BUS ***(Figure 2.1).***
<R50>	The slave task master ***shall*** generate the checksum byte ***(Figure 2.1).***
<R51>	The slave task master ***shall*** transmit the checksum byte ***(Figure 2.1).***

3.1.4 Synchronization Break Field Framing

ID	**Functional Requirements**
<R52>	The Synchronization Break Field ***shall*** provide a regular opportunity for slave task to synchronize on the bus clock.
<R53>	The Synchronization Break Field ***shall*** consist in two different parts the *first* part consist of a dominant bus value with the duration of *TSYNBRK* or longer ***Figure 2.2***

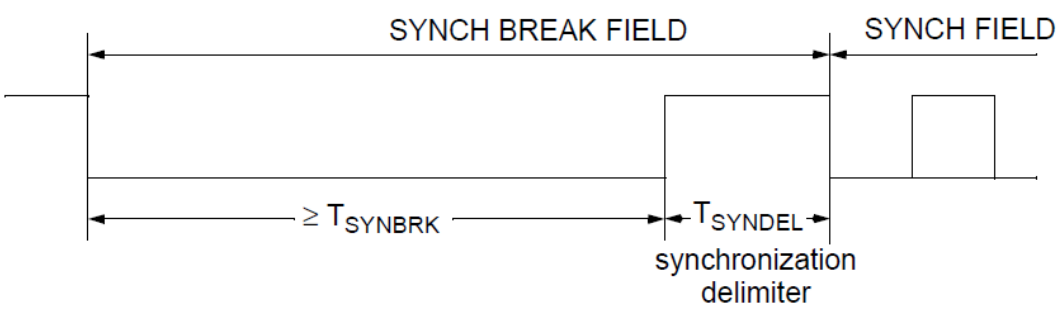


Figure 2.2 Synchronization Break Field.

<R54> The Synchronization Break Field ***shall*** consist in two different parts the *second* part consist of a recessive synchronization delimiter with a minimum duration of *TSYNDEL* see ***Figure 2.2.*** The recessive and dominant logical values are show in the ***table 1.1.***

Logical Value	Bit Value	Bus Voltage
Dominant	0	Ground
Recessive	1	Battery

Table 1.1 Logical Value.

<R55> The Synchronization Break ***shall*** have a minimum bit length of 13 bits, it assured that a slow slave can also detect the start of a data transmission.

<R56> The synchronization Break Delimiter ***shall*** have at least one and maximum of four recessive bits see ***Figure 2.2*** this separate the Sync Break from the Sync Field that follows it, which begins with a dominant start bit due to the use of the SCI for serial communication.

<R57> To ensure that all slaves send and receive at the same clock time, the master ***shall*** transmit the *Sync Break Field* after the *Sync Field* the slaves must synchronize to this.

<R58> A slave node ***shall*** detect such a break if its duration surpasses the period of TSBRKTS, measured in slave bit times see ***Figure 2.3***

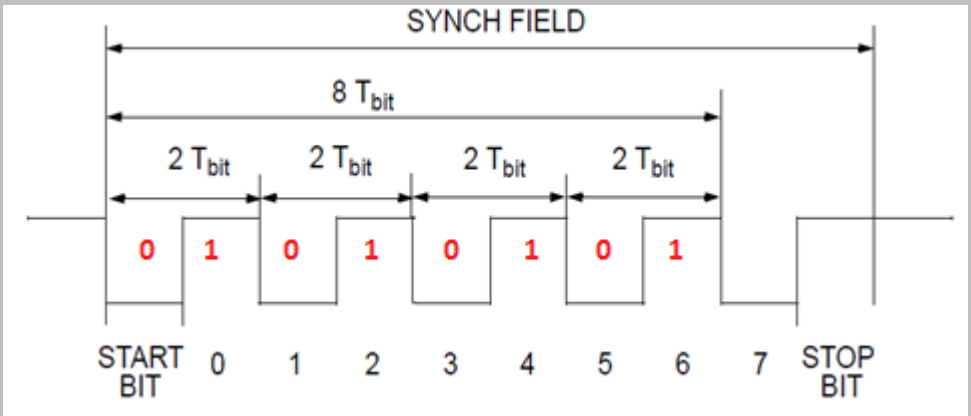
SYNCH BREAK FIELD	LOGICAL	NAME	MIN [T_{bit}]	Nom [T_{bit}]	MAX [T_{bit}]
SYNCH BREAK LOW PHASE	dominant	T_{SYNBRK}	13[a]		-
SYNCH BREAK DELIMITER	recessive	T_{SYNDEL}	1[a]		-
SYNCH BREAK THRESHOLD SLAVE	dominant	T_{SBRKTS}		11[b]	

Timing of the SYNCH BREAK FIELD

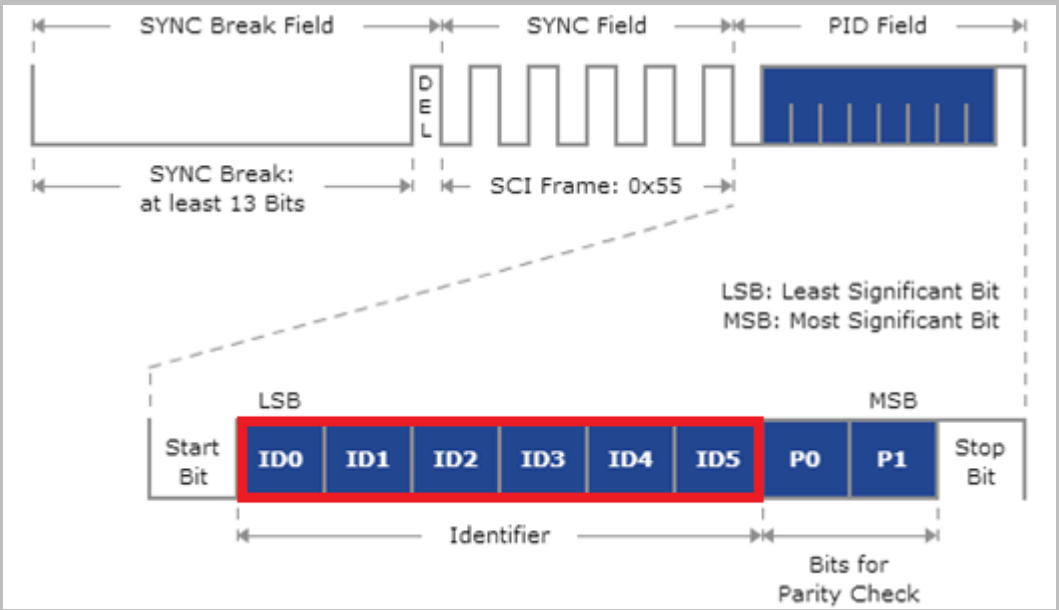
Note a: This bit time is based on the master time base.
Note b: This bit time is based on the local slave time base. It is valid for nodes with a clock tolerance lower than $F_{TOL\_UNSYNCH}$ ( ), e.g. for slave nodes with RC oscillator.
For nodes with a clock tolerance lower than F_{TOL_SYNCH}, e.g. slave nodes with quartz or ceramic resonator ( ). The detection of more than 9 consecutive dominant bits may be regarded as a SYNCH BREAK condition in this case.

Figure 2.3 Timing of the SYNCHRONIZATION BREAK FIELD.

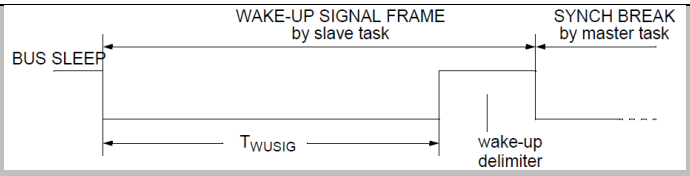
3.1.5 Synchronization Field

ID	Functional Requirements
<R59>	The *Synchronization Field* ***shall*** contain the information for the clocks synchronization.
<R60>	The bit timing ***shall*** be the Master Node.
<R61>	The Synchronization Field ***shall*** consist of the data '0x55' inside a byte field.(***Figure 2.4).***  *Figure 2.4 Synchronization Field.*
<R62>	The Synchronization procedure ***shall*** have to be based on time measurement between falling edges of the pattern see ***Figure 2.4***
<R63>	The falling edges ***shall*** be available in distances of 2,4,6 and 8 bit times see ***Figure 2.4.***

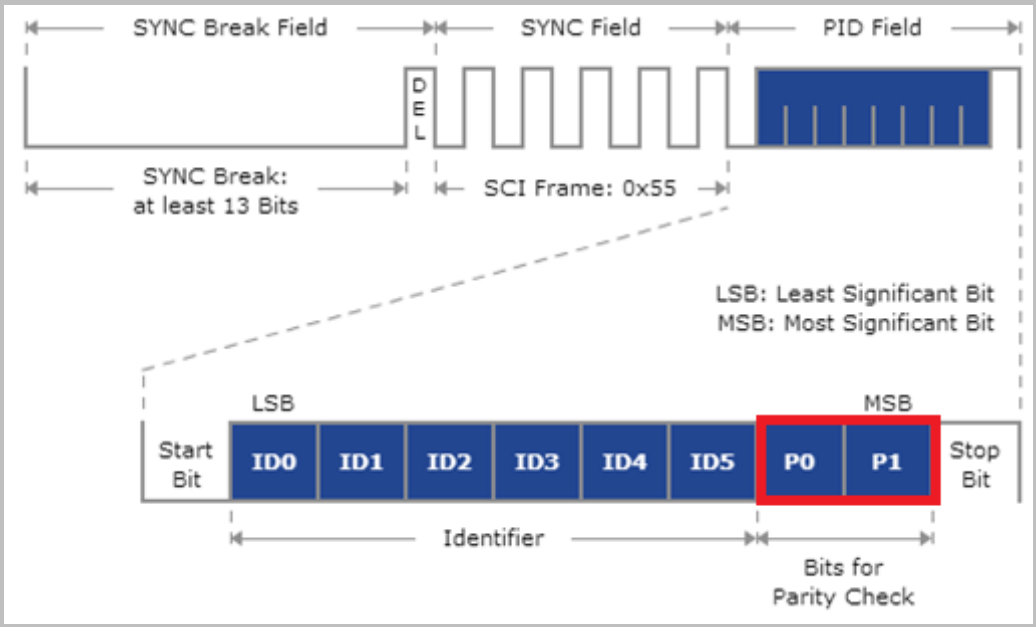
3.1.6 Identifier Field

ID	Functional Requirements
<R64>	The Identifier Field ***shall*** denote the content of a message ***Figure 2.5***  *Figure 2.5 Identifier Field*
<R65>	The Identifier Field ***shall*** describe the meaning of the data in the LIN message.
<R66>	The identifier ***shall*** consist of 6 bits see ***Figure 2.5.***
<R67>	Multiple nodes ***shall*** have the capability for receive simultaneously one message (Multicast) using the message filtering process.
<R68>	The slaves ***shall*** determine, based on the identifier, whether they need to send or receive a response or behave passively.
<R69>	Each node ***shall*** apply a message filtering process when receive and act upon messages.
<R70>	The maximum number of identifiers ***shall*** be 64 values.
<R71>	Four of these identifiers ***shall*** be considered as reserved for special communication purposes such as software upgrades or diagnostics (***Figure 2.6***)

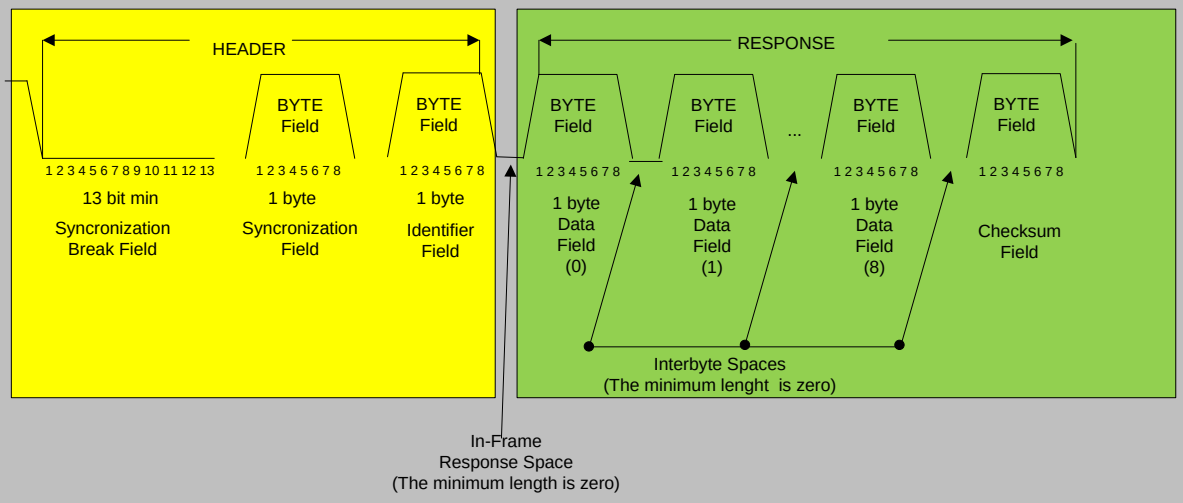
	<table><tr><th>Decimal Value</th><th>Hexadecimal Value</th><th>Description</th></tr><tr><td>60</td><td>0x3C</td><td>Diagnostic Request Master Request Frame</td></tr><tr><td>61</td><td>0x3D</td><td>Diagnostic Response Slave Response Frame</td></tr><tr><td>62</td><td>0x3E</td><td>reserved for future enhancements</td></tr><tr><td>63</td><td>0x3F</td><td>reserved for future enhancements</td></tr></table> *Figure 2.6 Reserved Identifiers*
<R72>	The identifier bits ID4 and ID5 ***shall*** define the number of data fields NDATA in a message (***Figure 2.7***). <table><tr><th>ID5</th><th>ID4</th><th>N_{DATA} (number of data fields) [byte]</th></tr><tr><td>0</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>4</td></tr><tr><td>1</td><td>1</td><td>8</td></tr></table> *Figure 2.7 Control of the number of Data Fields in a MESSAGE FRAME.*
<R73>	The identifier bits ID4 and ID5 ***shall*** divide the set of 64 identifiers on 4 subset of 16 identifiers with 2,4 and 8 data fields, respectively (***Figure 2.7***).
<R74>	The Sleep Mode Command ***shall*** be used to broadcast the sleep mode to all bus nodes. There is no more activity after completion of this message until WAKE-UP SIGNAL on the bus ends the sleep mode.
<R75>	Two extended frame identifiers ***shall*** be reserved to allow the embedding of user defined message formats and future LIN formats into the LIN protocol without violating the current LIN specification.
<R76>	The wake up signal ***shall*** be sent by slave task but only if the bus was previously in sleep mode and a node internal request for wake-up is pending.
<R77>	The wake up signal ***shall*** be generated with the character 0x80.
<R78>	The first field of wake up signal ***shall*** be given by a sequence of *TWUSIG* dominant bits (***Figure 2.8***).

	 *Figure 2.8 Wake up signal frame*
<R79>	After a wake-up signal ***shall*** have been send to the bus, all nodes run through the Start-Up procedure and wait for the master task to send a *SYNCH BREAK FIELD* followed by the *SYNCH FIELD.*
<R80>	If no *SYNCH FIELD* is detected before *TIME-OUT* after *WAKE UP* signal, a new *WAKE UP* signal ***shall*** be issued by the node requesting the first *WAKE UP*.
<R81>	This sequence ***shall*** be issued not more than three times, then the transmission of WAKE UP signals is suspended for a TIME-OUT after three breaks.
<R82>	The re-transmission of a WAKE-UP SIGNAL ***shall*** be allowed only to the node which has an internal request for wake-up pending.

3.1.7 Parity Bits

ID	Functional Requirements
<R83>	The identifier field ***shall*** have 2 reserved bits for double parity protection this combination is referred as the PID (***Figure 2.9***).  *Figure 2.9 Parity Check*
<R84>	The computation of parity bits P0 and P1 ***shall*** be based on exclusive OR logic, where the parity bit P0 represents even parity bit P1 represents odd parity see.
<R85>	Bit P0 ***shall*** be equal to ***Figure 3.0 a***  *Figure 3.0 a) P0* *b) P1*
<R86>	Bit P1 ***shall*** be equal to ***Figure 3.0 b***

3.2. Message Frame Response

ID	Requirement Functional
<R87>	The message frame response ***shall*** zero to eight Data Fields see ***Figure 3.1***  *Figure 3.1 Frame Response*
<R88>	Between the master request (header) and the slave's response there ***shall*** be a response space (***Figure 3.1***).
<R89>	The minimum length of interbyte spaces and the in-frame-response space ***shall*** be zero (***Figure 3.1***).
<R90>	The total maximum length of these spaces ***shall*** be limited by the maximum length of the Message Frame. (***Figure 3.1***).

3.2.1 Checksum

ID	Functional Requirements
<R91>	*The Checksum Field* ***shall*** *contain the inverted module 256 sum over all data bytes see* ***Figure 3.2.***

Message frame with 4 data bytes
Data 0 = 0x4A
Data 1 = 0x55
Data 2 = 0x93
Data 3 = 0xE5

	HEX	Bit Carry	D7	D6	D5	D4	D3	D2	D1	D0	
0x4A	0x4A		0	1	0	0	1	0	1	0	
+ 0x55	0x55		0	1	0	1	0	1	0	1	
	0x9F	0	1	0	0	1	1	1	1	1	
+ Bit Carry	0									0	
	0x9F		1	0	0	1	1	1	1	1	
+0x93			1	0	0	1	0	0	1	1	
	0x132	1	0	0	1	1	0	0	1	0	
+ Bit Carry										1	
	0x33		0	0	1	1	0	0	1	1	
+0xE5	0xE5		1	1	1	0	0	1	0	1	
	0x118	1	0	0	0	1	1	0	0	0	
										1	
	0x19		0	0	0	1	1	0	0	1	Checksum
Invert	0xE6		1	1	1	0	0	1	1	0	Checkbyte
0x19 + Invert	0xFF		1	1	1	1	1	1	1	1	

The resulting checksum is 0x19. The checkbyte is inverted checksum 0xE6
The receiving node can easily check the consistency of the data and the checkbyte by using the same addition mechanism
Checksum + checkbyte must result in 0xFF.

Figure 3.2 Checksum Complete

<R92>	The slave sender task ***shall*** compare each bit placed on the bus with the Checksum Field see ***Figure 3.2.***
<R93>	The slave receiver task ***shall*** check whether frame response transmitted directly after a frame header.
<R94>	During the response space a node ***shall*** switch from the receiving state to the sending state.
<R95>	*First Step* in the Checksum Field the Data 0 of the Response ***shall*** be added with the Data 1. ***(Figure 3.2).***
<R96>	*Second Step* in the Checksum Field if the Result between Data0 and Data1 gives the Bit Carry, then it values ***shall*** be added at the first Result, else nothing ***(Figure 3.2).***
<R97>	*Third Step* in the Checksum Field the Result ***shall*** be added to the next Data and repeat the Second Step until the last data ***(Figure 3.2).***
<R98>	*Four Step* in the Checksum Field the last Result ***shall*** be the checksum ***(Figure 3.2).***
<R99>	*Fifth Step* in the Checksum Field the checksum ***shall*** be inverted.
<R100>	*Six Step* in the Checksum Field the checksum ***shall*** be added with the checksum inverted. ***(Figure 3.2).***

<R101>	*Seven Step* in the Checksum Field the Result between checksum and checksum inverted ***shall*** be **0xFF** (***Figure 3.2***).

3.3. Scheduler

ID	Functional Requirements
<R102>	The Master ***shall*** control all communication in a cluster.
<R103>	The Master ***shall*** have an established sending scheme that is planned by the system designer, this makes communication in the network predictable, because there is a fixed time sequence.(***Figure 3.3***)
<R104>	The Schedule ***shall*** *be organized into SLOTS, which are provided for transmission of one frame each.* (***Figure 3.3***).

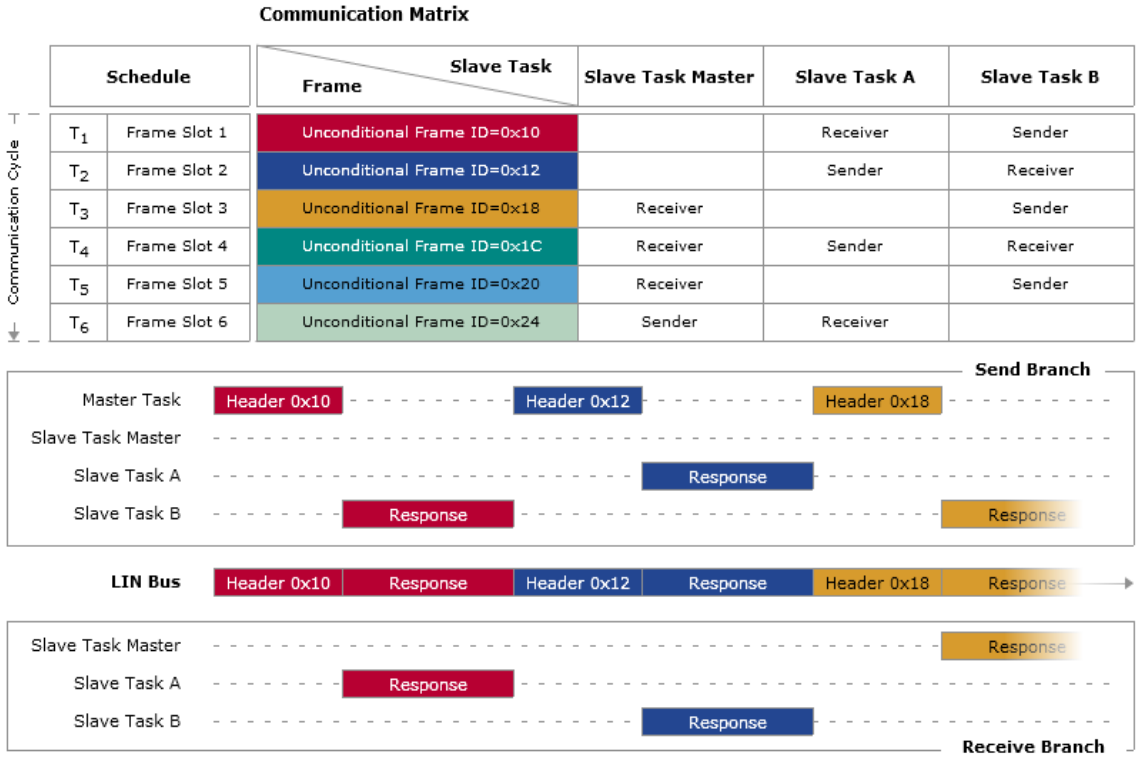
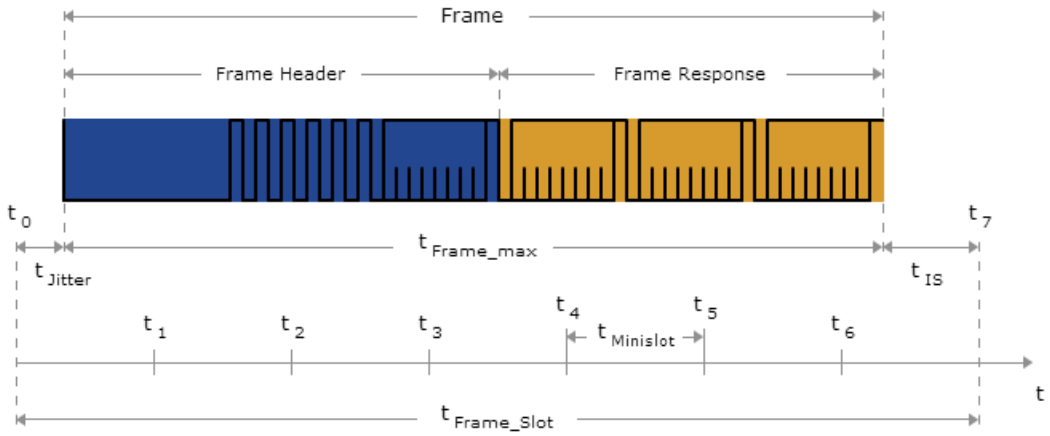
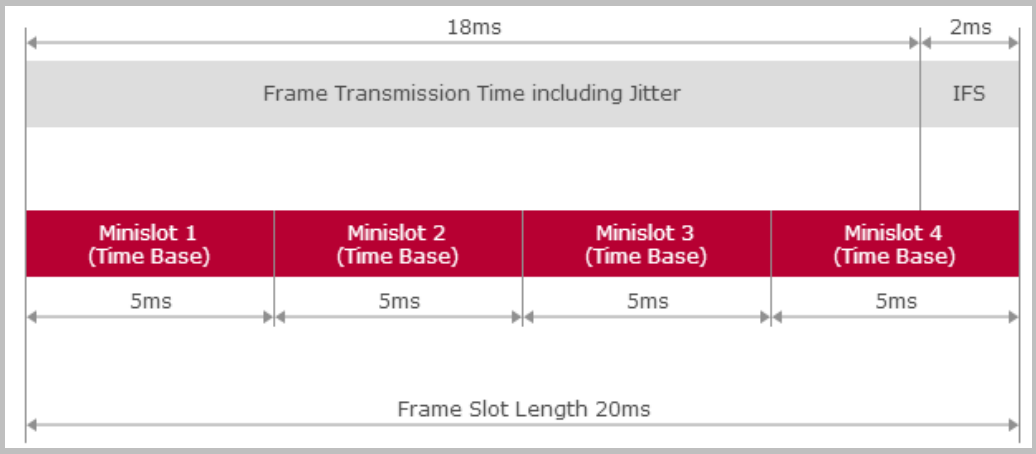


Figure 3.3 Schedule Table

<R105>	The size of frame slot ***shall*** be added a *jitter* value this is the potential time difference between the nominal beginning of a slot and the actually starting point.***(Figure 3.4)***  *Figure 3.4 Time Frame Slot*
<R106>	If a frame not completely fill out the slot, the rest of time period ***shall*** be awaited until the next slot is available its names is IFS(Inter Frame Space).***(Figure 3.5)***  *Figure 3.5 Computing Length Frame Slot*
<R107>	The master task ***shall*** delegate the sending right and controls bus access (***Figure 3.6***)

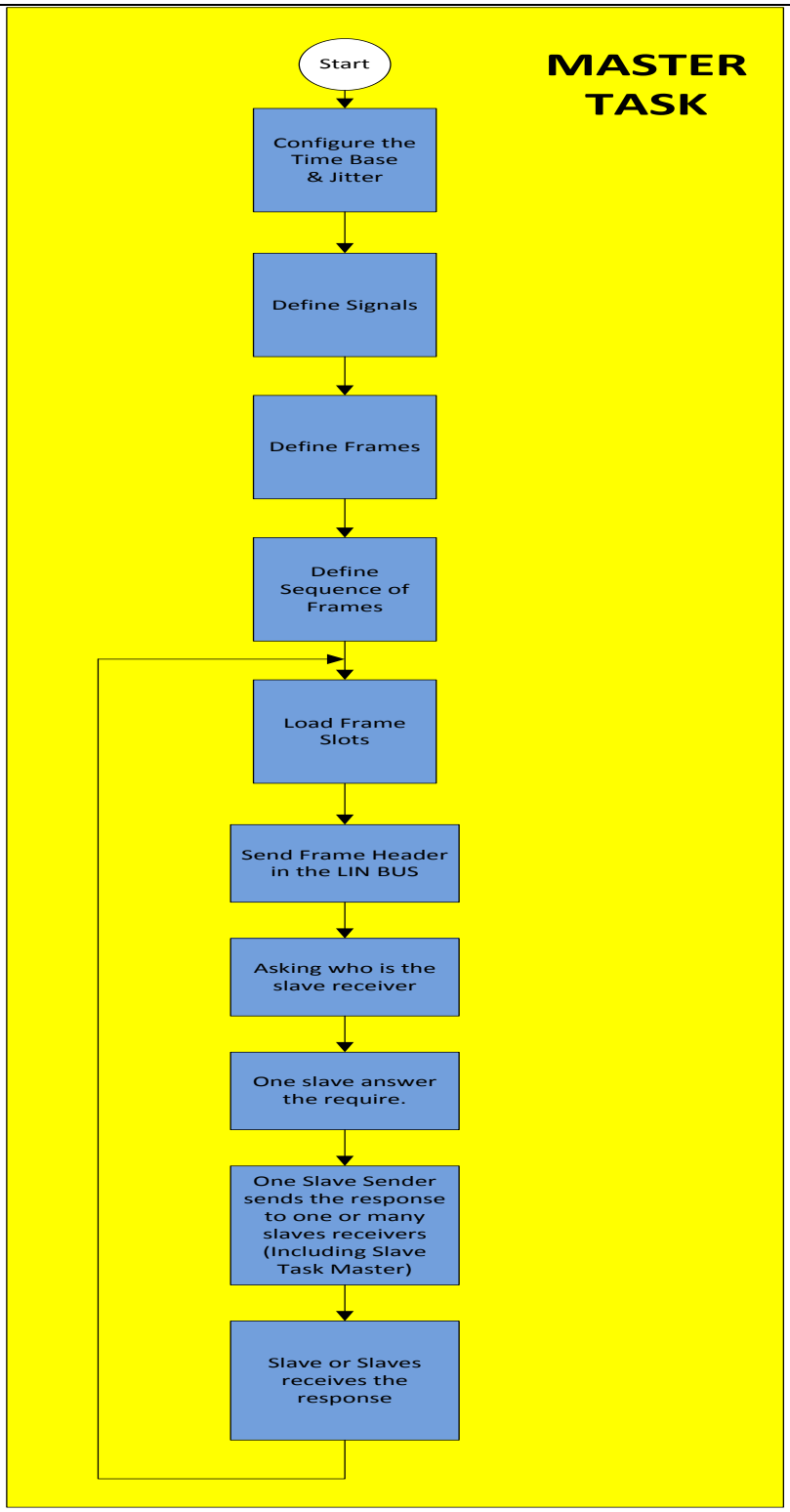
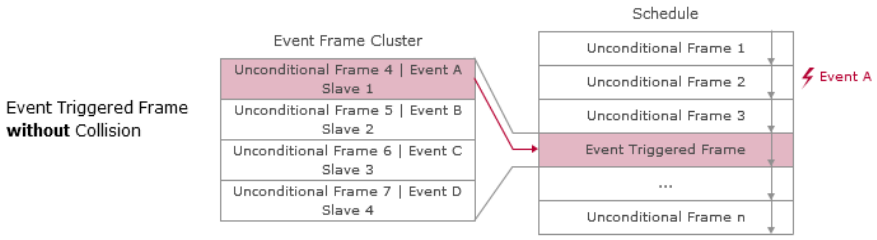
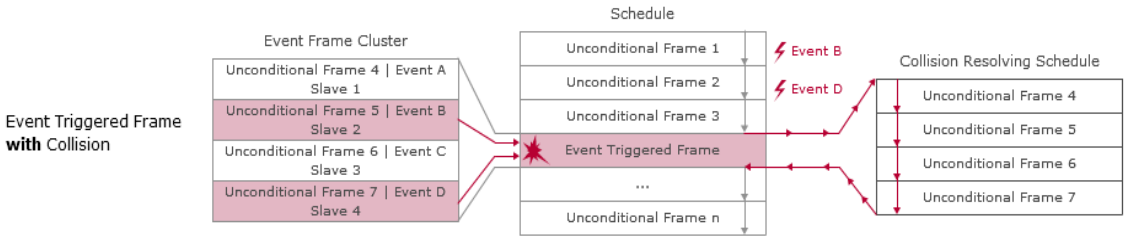
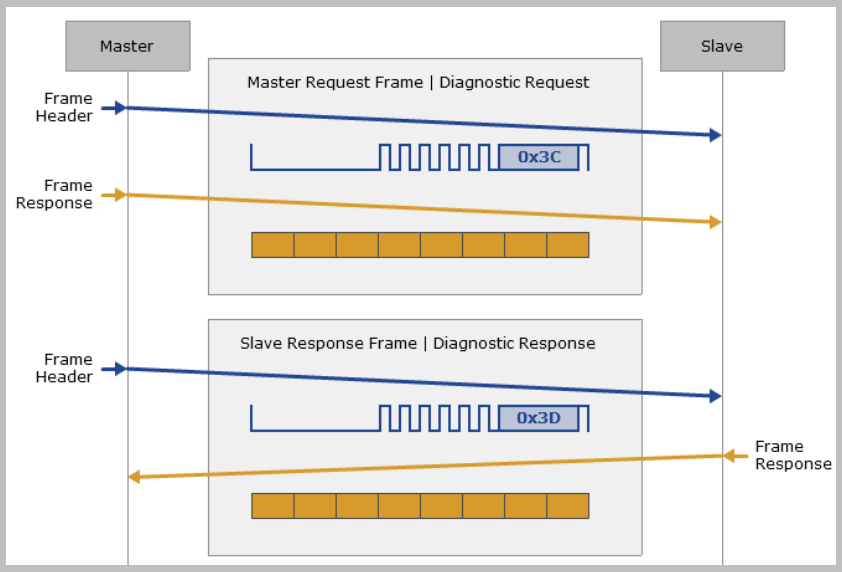


Figure 3.6 State Machine of Scheduler

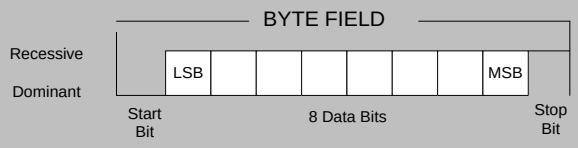
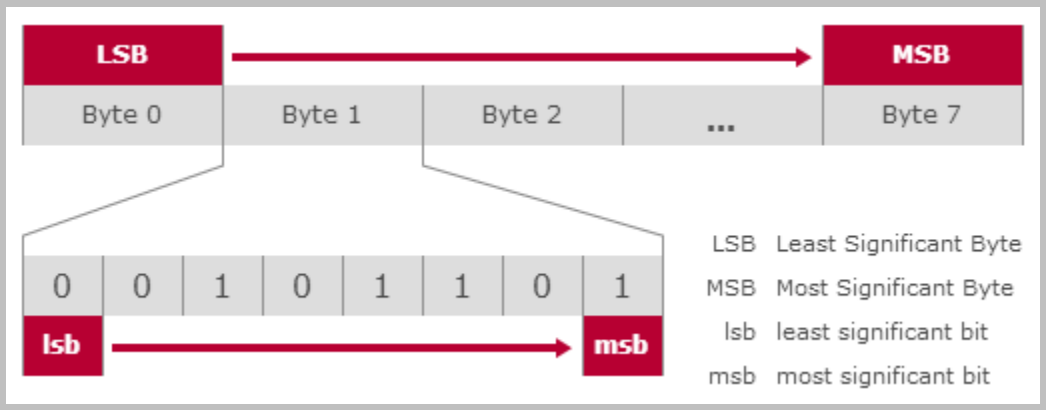
<R108>	*First Step* in the Scheduler ***shall*** be to configure time base.

<R109>	*Second Step* in the Scheduler ***shall*** be to configure the jitter.
<R110>	*Third Step* in the Scheduler ***shall*** be capture signals.
<R111>	*Fourth Step* in the Scheduler ***shall*** be define frames.
<R112>	*Fifth Step* in the Scheduler ***shall*** be define sequence of frames.
<R113>	*Sixth Step* in the Scheduler ***shall*** be load the Unconditional frame slots.
<R114>	*Eight Step* in the Scheduler ***shall*** be to send the FRAME HEADER to the LIN BUS including the Slave Task Master .
<R115>	*Ninth Step* in the Scheduler ***shall*** be to asking trough identifier who is the slave receiver.
<R116>	Tenth Step in the Scheduler, one slave task ***shall*** answer the require.
<R117>	*Eleventh Step* in the Scheduler, One slave sender ***shall*** send the response to one or many Slave Task receivers including slave task Master.
<R118>	*Twelfth Step* in the Scheduler, Slave or Slaves ***shall*** receive the response.
<R119>	*Finally* the start of the sequence ***shall*** be initiate in the Define Frame Slot state.
<R120>	The *Unconditional Frame* ***shall*** consist of header and frame, the master transmits the header onto the bus as a request, then a defined slave answers with the response (***Figure 3.7***).
<R121>	When the master of the *Event Triggered Frame* ***shall*** send a header with the identifier for an event triggered frame, the assigned slaves may append their specific responses to it.(***Figure 3.7***)  *Figure 3.7 Event Triggered Frame without Collision*
<R122>	One response of the *Event Triggered Frame* ***shall*** be sent after the header, the first data byte contains and additional Parity Check.It makes possible to determine which node has sent its response on the bus, to ensure that the length of an event triggered frame is clearly defined, all potential responses have the same number of data bytes.

<R123>	The master of the *Event Triggered Frame* ***shall*** resolve the collisions with a Collision Resolving Schedule (***Figure 3.8***). This is a special sending scheme in which the response of the slaves are polled again and are handled as regular unconditional frames.  *Figure 3.8 Event Triggered Frame with Collision*
<R124>	The master of the *Event Triggered Frame* ***shall*** begin to process the Collision Resolving Schedule after detecting a collision, and it then jumps back to the originally exited Schedule, this guarantees that all open responses that need to be sent ***(Figure 3.8)***.
<R125>	A master request of *Diagnostic Frame* ***shall*** be used as a diagnostic request or to configure slaves ***(Figure 3.9).***  *Figure 3.9 Diagnostic Frame*
<R126>	A slave response of *Diagnostic Frame* ***shall*** be used as a diagnostic response.
<R127>	The *Diagnostic Frame* of Master Request ***shall*** transmit the header and the response (***Figure 3.9***).
<R128>	The *Diagnostic Frame* of Master Request ***shall*** initiate the diagnostic process (***Figure 3.9***).

<R129>	The *Diagnostic Frame* of Master Request ***shall*** reserve the unconditional frame with ID = 0x3C
<R130>	The *Diagnostic Frame* of Slave Response ***shall*** reserve the unconditional frame with ID = 0x3D
<R131>	The *Diagnostic Frame* of Slave Response ***shall*** transmit the response to the diagnostic request.
<R132>	The transmission of the header and the response ***shall*** be used as a diagnostic response.

3.4. LIN COMMUNICATION

ID	**Requirement Functional**
<R133>	Every BYTE Field ***shall*** have a length of 10 Bit Times (***Figure 4.0***)  *Figure 4.0 BYTE FIELD Format*
<R134>	The START BIT ***shall*** mark the begin of the BYTE FIELD and is dominant. (***Figure 4.0***)
<R135>	The transmission ***shall*** be made in ascending order from the *least significant* to the *most significant* byte (***Figure 4.1***).  *Figure 4.1 Byte Order*
<R136>	After the START BIT ***shall*** be followed by 8 DATA BIT's with the LSB First. ***(Figure 4.1)***

<R137>	*The STOP BIT **shall*** mark the end of the *BYTE FIELD and is recessive.* ***(Figure 4.1)***

4. Conclusions

This Project helped me to understand the importance to translate the needs of a client in to technical requirements. Because this information is the base for people who is going to develop the software and hardware architecture. In my next projects I'm going to include this methodology for saving time.

5. Bibliography

Autosar Partnership of vehicle manufacturers, suppliers and other companies from the electronics,

Semiconductor and software industry.

https://www.autosar.org/

LIN Protocol Specification by Freescale

www.cs-group.de/fileadmin/media/Documents/**LIN_Specification**_Package_2.2A.pdf

LIN Protocol Specification by ST

http://www.st.com/content/ccc/resource/technical/document/application_note/10/30/18/b5/90/bc/4c/73/CD00004273.pdf/files/CD00004273.pdf/jcr:content/translations/en.CD00004273.pdf

Vector provides OEMs and suppliers of automotive and related industries a professional an open development platform of tools, software components and services for creating embedded systems.

https://vector.com/

C. DESIGNING AND IMPLEMENTATION OF AUTOMOTIVE ELECTRONIC CONTROL MODULE.

Content

1. Overview Requeriments

The present document describes the requirements given by customer either functional or not functional. The requirements are divided by functional area, this mean, are categorized, there are requirements to Control, Communication, Hardware, etc. This information was extracted from the document “Final Project” that it was provider by costumers. The requirements also are at traceability matrix in order to tracking the progress of them. Once that the requirements was generated, must be review by developers and costumers to verify that none of them is unambiguous, unnecessary or not feasible.

From this document begins to build the architecture and design of the project, hence the importance of remaining well described. From this same document will be generated the test bench to test the proper operation of the system.

The requirements shall have these key words in bold to specify that this requirements must be implemented, or how it should be implemented, or if it is nice to have. Each requirement must be have at least one of the key words.

Each requirements has its own ID, this one conformed by letter and numbers, FR means Functional Requirement and the numbers are incremental for every requirements that has been added into the list.

LINK	KEY WORD
Duty	Shall/Must
Wish	Should
Intent	Will
Suggestion	Can
Comment	…

Table 1. Key Words

1.1. Requirements

This chapter displays the requirements associated to ID. The words in bold shows the severity of the requirement. The ID match with traceability matrix file.

ID	Requirement
FR001	The BCM system **shall** be able to report the temperature value.
FR002	Temperature value **shall** be defined in C degrees.
FR003	Temperature value **shall** be in range of 0 - 99.5 C degrees.
FR004	Temperature resolution **shall** be of decimal value (0.5 C degrees).
FR005	Temperature format **shall** be 7 bits integer and 1 bit decimal. 0b0000000.0
FR006	The BCM system **shall** report the Car Speed.
FR007	Speed value **shall** be defined in Km/h.
FR008	Speed value **shall** be in range of 0 - 120 Km/h.
FR009	Speed value resolution **shall** be of decimal value (0.5 Km/h).
FR010	Speed format **shall** be 7 bits integer and 1 bit decimal. 0b0000000.0
FR011	The BCM system **shall** measure of the fuel tank.
FR012	Fuel value **shall** be defined in Liters.
FR013	Fuel value **shall** be in range of 0 to 45 Liters (normal tank).
FR014	Speed format **shall** be 8 bits. 0b00000000
FR015	The BCM system **shall** measure of the environment humidity.
FR016	Humidity value **shall** be defined in percentage (%).
FR017	Humidity value **shall** be in range of 5% - 95%.
FR018	Humidity value resolution **shall** be of 10%.
FR019	Humidity format **shall** be 7 bits integer and 1 bit decimal. 0b0000000.0
FR020	The BCM system **shall** measure of proximity to an object.
FR021	Object sensing value **shall** be in meters.

FR022	Object sensing value **shall** be in range of 0 to 10.0 meters.
FR023	Object sensing resolution **shall** be of decimal of meters (0.5 meter).
FR024	The battery voltage **should** be measure from 0 to 12 volts
FR025	The battery voltage **shall** have operations mode, low battery and normal voltage.
FR026	In low battery the system **should** be in standby, just can detect key status, and CAN message are lock.
FR027	In normal mode the system **shall** work property.
FR028	**shall** be detect the key status, **can** be two options On or Off
FR029	If the system is turn off, just **can** be turn on the internal, external, stop, and high/low beam lights.
FR030	The system **shall** detect if a brake pedal was pressed or not.
FR031	If the brake pedal was pressed the system **shall** turn on the stop light.
FR032	The system **should** receive CAN message from cluster, with the current status of the stalk position.
FR033	The message received **shall** be a payload of 8 bytes.
FR034	each byte has information about stalk position
FR035	The can message received **should** have the format as is shown in the *table 4.*
FR036	The system **should** send CAN message to cluster, with the current status of the sensors.
FR037	The message sent **shall** be a payload of 8 bytes.
FR038	Each byte **has** information about sensors, brake pedal and key status.
FR039	The can message sent **should** have the format as is shown in the *table 5.*
FR040	The reception and transition of each message **must** be cyclic according with the network schedule table.

FR041	The system **shall** be able process data according with the values in the *table 3* these values are from cluster.
FR042	The system **shall** be capable to open circuit of any the actuators.
FR043	All the actuators **must** to be isolated from the microcontroller.
FR044	All the actuators **shall** be activated across a Relay
FR045	All the digit signals **shall** be 3.3v.
FR046	All the analog signals **shall** be from 0 to 3.3v.
FR047	The system **shall** have 4 analog pines to use like sensors inputs. • Driver assistance • Environment humidity • Fuel Level • Brake pedal
FR048	The system **shall** have 1 analog pin to use like sensor input. • Key position strategy
FR049	The system **shall** have 2 pines to interaction with the actuators. • Interior lighting • Exterior lighting

Table 2. Requirements

O. Horn	L. Beam	H. Beam	T. Right	T. Left	
B4	**B3**	**B2**	**B1**	**B0**	**Function**
0	0	0	0	0	All switches OFF
0	0	0	0	1	Turn Left
0	0	0	1	0	Turn Right
0	0	0	1	1	Not possible
0	0	1	0	0	High beam
0	0	1	0	1	High beam/Turn left
0	0	1	1	0	High beam/Turn right
0	0	1	1	1	Not possible
0	1	0	0	0	Low Beam
0	1	0	0	1	Low Beam/ Turn Left
0	1	0	1	0	Low Beam/ Turn Right
0	1	0	1	1	Not possible

0	1	1	0	0	Not possible
0	1	1	0	1	Not possible
0	1	1	1	0	Not possible
0	1	1	1	1	Not possible
1	0	0	0	0	Optical Horn
1	0	0	0	1	Optical Horn/Turn Left
1	0	0	1	0	Optical Horn/Turn Right
1	0	0	1	1	Not possible
1	0	1	0	0	Not possible
1	0	1	0	1	Not possible
1	0	1	1	0	Not possible
1	0	1	1	1	Not possible
1	1	0	0	0	Not possible
1	1	0	0	1	Not possible
1	1	0	1	0	Not possible
1	1	0	1	1	Not possible
1	1	1	0	0	Not possible
1	1	1	0	1	Not possible
1	1	1	1	0	Not possible
1	1	1	1	1	Not possible

Table 3. Light Position

ECU	ID	Packet Name	Pck Length	Signal Byte	Signal Bit	Signal Name	Signal Functi
Cluster	100h	CLSTR_SGN_LIGTHS	2	1	0	Interior Light	Reports current status of the Interior Light signal (0 = OFF, 1 = ON)
				1	1	Exterior Ligth	Reports current status of the Exterior Light signal (0 = OFF, 1 = ON)
		CLSTR_SGN_STALK	5	1	2	Optical Hotn	Reports current status of the Optical Horn (0 = OFF, 1 = ON)
				1	3	Low Beam	Reports current status of the Low Beam (0 = OFF, 1 = ON)
				1	4	High Beam	Reports current status of the High Beam (0 = OFF, 1 = ON)
				1	5	Turn Rigth	Reports current status of the Turn Right (0 = OFF, 1 = ON)
				1	6	Turn Left	Reports current status of the Turn Left (0 = OFF, 1 = ON)
			1	1	7	Reserved	Unused

Table 4. Message Received

ECU	ID	Packet Name	Pck Length	Signal Byte	Signal Bit	Signal Name	Signal Functi
BCM	200h	BCM_SGN	64	1	0	Temperature	Environment temperature (Celcius)
				2	0	Speed	Car Speed (Km\h)
				3	0	Fuel Level	Measure of the fuel tank
				4	0	Humidity	Measurement of the environment humidity
				5	0	Key Status	Key OFF = 0 Key ON = 1
				6	0	Object Sensing	Measurement of proximity to an object
				7	0	Low Voltage Status	Normal Mode = 0 Low Voltage mode = 1
						Baterry Voltage	Measurement of Batter voltage
				8	0	Reserved	Unused

Table 4. Message Transmitter

Program	Body Control Module											
Reference Documents	BCM_Requirements											
Reviewed by	Miguel Tlapa Juarez											
			Necesary	Unambiguous	Feasible	Implementatio	Complete	Concise	Consistent	Maintainable	Traceable to	Verifiable
DOORS Id	Section	Requirements										
FR001	Temperature	The BCM system shall be able to report the temperature value.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR002	Temperature	Temperature value shall be defined in C degrees.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR003	Temperature	Temperature value shall be in range of 0 - 99.5 C degrees.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR004	Temperature	Temperature resolution shall be of decimal value (0.5 C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR005	Temperature	Temperature format shall be 7 bits interger and 1 bit decimal. 0b0000000.0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR006	Speed	The BCM system shall report the Car Speed.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR007	Speed	Speed value shall be defined in Km/h.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR008	Speed	Speed value shall be in range of 0 - 120 Km/h.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR009	Speed	Speed value resolution shall be of decimal value (0.5 Km/h).	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR010	Speed	Speed format shall be 7 bits interger and 1 bit decimal. 0b0000000.0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR011	Fuel	The BCM system shall measure of the fuel tank.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR012	Fuel	Fuel value shall be defined in Liters.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR013	Fuel	Fuel value shall be in range of 0 to 45 Liters (normal tank).	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR014	Fuel	Speed format shall be 8 bits. 0b00000000	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR015	Humidity	The BCM system shall measure of the environment humidity.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR016	Humidity	Humidity value shall be definded in percentage (%).	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR017	Humidity	Humidity value shall be in range of 5% - 95%.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR018	Humidity	Humidity value resolution shall be of 10%.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR019	Humidity	Humidity format shall be 7 bits interger and 1 bit decimal. 0b0000000.0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR020	Object Sensing	The BCM system shall measure of proximity to an object.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR021	Object Sensing	Object sensing value shall be in meters.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR022	Object Sensing	Object sensing value shall be in range of 0 to 10.0 meters.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR023	Object Sensing	Object sensing resolution shall be of decimal of meters (0.5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR024	Battery Voltage	The battery voltage should be measure from 0 to 12 volts	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR025	Battery Voltage	The battery voltage shall have operations mode, low battery and normal voltage.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR026	Battery Voltage	In low battery the system should be in stand by, just can detect key status, and and CAN message are lock.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR027	Battery Voltage	In normal mode the system shall work propertly	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR028	Key Status	shall be detect the key status, can be two options On or Off	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR029	Key Status	if the system is turn off, just can be turn on the internal, external, stop, and high/ow beam ligths.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR030	Break Pedal	the system shall detect if a break pedal was press or not.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR031	Break Pedal	if the break pedal was press the system shall turn on the stop	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR032	CAN Recibe	the system should recibe CAN message from cluster, with the current status of the stalk position.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR033	CAN Recibe	the message recibed shall be a payload of 8 bytes.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR034	CAN Recibe	each byte has a information about stalk position	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR035	CAN Recibe	the can message recibed should have the format as is show in the *"table 4 Doc requirements"*	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR036	CAN Send	the system should send CAN message to cluster, with the current status of the sensors.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR037	CAN Send	the message sent shall be a payload of 8 bytes.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR038	CAN Send	each byte has a information about sensors, break pedal and key status.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR039	CAN Send	the can message sent should have the format as is show in the *"table 5 Doc requirements"*	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FR040	CAN	The reception and transition of each message must be cyclic according with the network schedule table.	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

FR041	**CAN**	The system shall be able procces data according with the values in the table *"table 3 Doc requirements"* these values	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR042	**Safety**	The system shall be capable to open circuit of any or the actuators.	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR043	**Safety**	All the actuators must to be isolated from the microcontroller.	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR044	**Hardware**	All the actuators shall be activated across a Relay	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR045	**Hardware**	All the digit signals shall be 3.3v.	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR046	**Hardware**	All the analog signals shall be from 0 to 3.3v.	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR047	**Hardware**	The system shall have 4 analog pines to use like sensors inputs. • Driver assistance • Environment humidity • Fuel Level • Brake pedal	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR048	**Hardware**	The system shall have 1 analog pin to use like sensor input. • Key position strategy	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
FR049	**Hardware**	The system shall have 2 pines to interaction with the actuators. • Interior lighting • Exterior lighting	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

Table 5: Traceability matrix

2. Overview Body Control Module

In automotive electronics, body control module (BCM) is a generic term for an electronic control unit responsible for monitoring and controlling various electronic accessories in a vehicle's body. Typically in a car the BCM controls the power windows, power mirrors, air conditioning, immobilizer system, central locking, etc. *"The Load Control can either be directly from BCM or via CAN/LIN communication with remote ECUS. The central body controller often incorporates RFID functions like remote keyless entry and immobilizer* (Texas Instruments, 2013)*".*

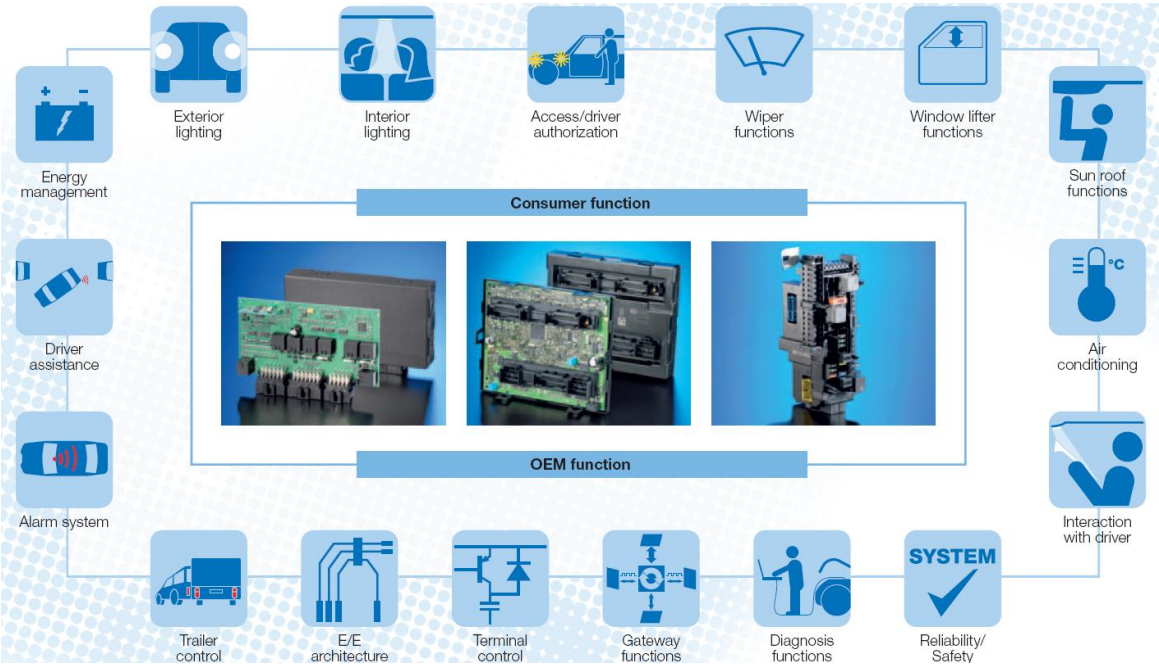


Figure 1. BCM system

Optimization and advanced developments in terms of comfort, safety and variety are already presenting a great challenge to the vehicle electric system. The Body Control Module (BCM) is the core component for the realization of a broad range of functions. This central control unit can combine classic power distribution and the safeguarding of relay and fuse boxes with the advantages of intelligent, micro-controller controlled systems. In addition, BCMs play a deciding role in cost efficiency as they allow for the amount wiring within the vehicle to be significantly reduced by providing interfaces for bus systems. Depending on the architecture approach different

variants of central control units can be created from numerous combination possibilities. From more universal entry-level BCMs to highly integrated variants see *Figure 1*.

The *Figure 2* is a block diagram that describes the interaction with the external world, for instance: measure the temperature inside the car and regular it with the air conditioning, this is the reason whereby the BCM take over the sensors and actuators.

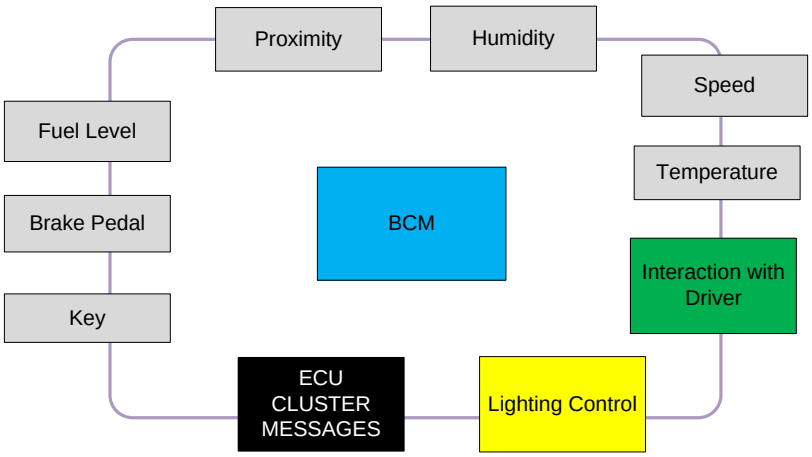


Figure 2: BCM project

The BCM project was developed in based to the requirements given for the costumers, and checked every one by the developers, the requirements are in Requirements section, noteworthy that the scheduler (part of a RTOS) was proposed by the developers team.

3. Architecture

This chapter describes the software and hardware architecture. To software architecture is taken as reference the model AUTOSAR (Automotive Open System Architecture), but does not comply with AUTOSAR, it just used some elements like the sorting of software layers.

3.1. Software Architecture

In order to cover the requirements specification of this implementation, software architecture was proposed, based only in the structure of software layers suggested by AUTOSAR, and it is not compliant with the AUTOSAR spec.
This implementation considers using the next software layers:

- **Application Layer**
- **ECU Abstraction Layer**
- **MCU Abstraction Layer**
- **System services layer**

3.1.1 Operating System

This layer will contain the Scheduler, Hardware configuration and Interrupt handler.
Scheduler: This module will be responsible for managing the CPU usage, some of its features are:

- configured and scaled statically
- amenable to reasoning of real-time performance
- will provides a priority-based scheduling

Interrupt Handler: It module will be in charge of manage the next system interrupts:

- Periodic Timer Interrupts
- Communication Interrupts
- I/O Interrupts

3.1.2 Communication

This layer will be responsible of handling the communication Framework, of CAN and SPI interphases, the I/O management for these interphases and, Network management.

3.1.3 Microcontroller Abstraction

Access to the hardware is routed through the Microcontroller Abstraction layer (MCAL) to avoid direct access to microcontroller registers from higher-level software. MCAL will be used as hardware specific layer that ensures a standard interface to the components of the ECU Software Layer. It will manage the microcontroller peripherals and will provide the components of the ECU with microcontroller independent values. MCAL also will implement notification mechanisms to support the distribution of commands, responses and information to different processes. This layer will include handling of:

- Digital I/O (DIO)
- Analog/Digital Converter (ADC)
- Pulse Width (De)Modulator (PWM, PWD)
- Capture Compare Unit (CCU)
- Watchdog Timer (WDT)
- Serial Peripheral Interface (SPI)
- Controller Area Network Bus (CAN)

3.1.4 Hardware Abstraction

The ECU Abstraction will provide a software interface to the electrical values of any specific ECU to decouple higher-level software from all underlying hardware dependencies.

3.1.5 Aplication Layer

This layer will be in charge of the low power management, enabling the possibly for change the operation mode of the BCM to low consumption mode.

Software architecture is implemented as shown in block diagram of the *Figure 3.*

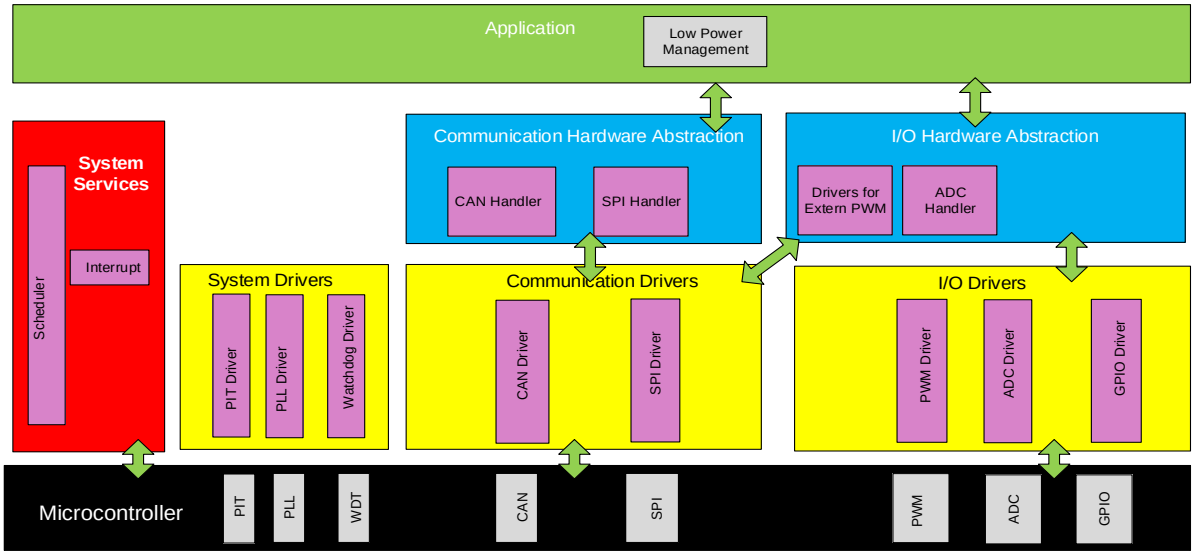


Figure 3: Software architecture

3.2. Hardware Architecture

This section describes how interact all stages in the hardware such as control stages, power stage, etc. The *Figure 4* shown those stages, it also shown that the system work with a closed control system, a clear example is the temperature control (only in automatic mode) the control must keep the same temperature. The sensors are connected into the analog/digital input, known as GPIO. To manage actuators the BCM used as power stage an Extreme Switch board, this one has four PWM input ports, it also has a serial communication (SPI is not used) full description will be seen in hardware design section.

The BCM communicates with other systems through the CAN network. It is able to transmit and receive messages from the cluster.

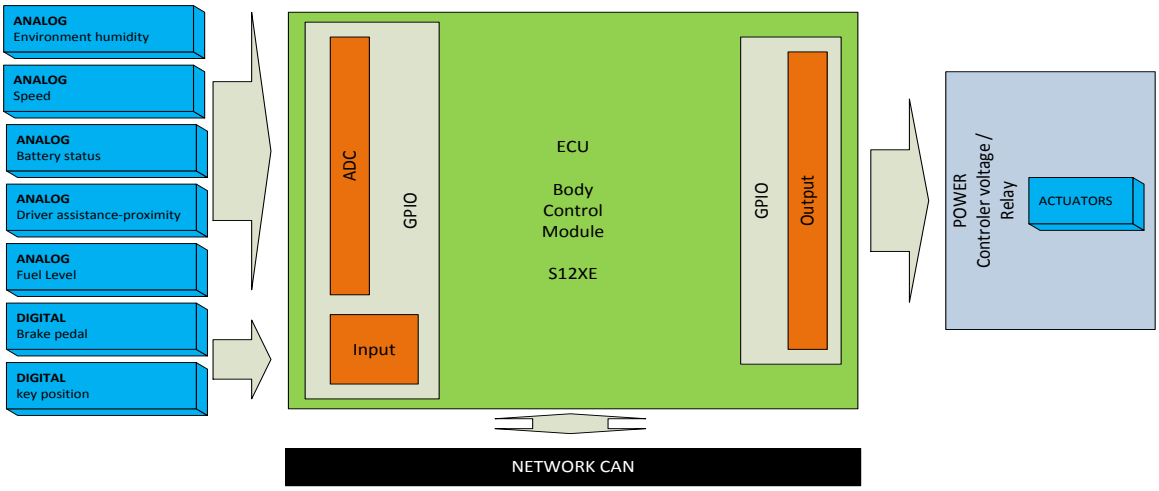


Figure 4: Hardware architecture

The ECU is a kit develop board DEMO9S12XEP100 by Freescale, these are the main characteristics:

- 112-pin LQFP MC9S12XEP100 MCU
- Selectable 4 MHz Crystal and a provision for an oscillator module (socketed)
- Built-in USB to BDM interface for in-circuit debugging
- Provision for a BDM connector for external in-circuit debugging
- Header connectors with all MCU signals
- One CAN connector with transceiver
- Two LIN connector with one transceiver
- One RS-232 connector with transceiver
- Four user LEDs
- Four DIP-switches
- Potentiometer for analog input
- Light sensor
- 2 push buttons
- Reset Push Button

4. Design

This chapter delves into the design of each block of the software and hardware architecture, it also delves into the scheduler design since it is not part of the requirements is the core part of the project.

4.1. Software Design

Taking the previous concept the last Chapter, building the architecture that was adapted to the requirements.

The design exhibit an architectural structure, contain interfaces that reduces the complexity of connections between modules and external environment.

Below is described the cycle life to develop the BCM project. Some of the next steps had to rethink because functionalities were added to the project.

1. **Requirements:** First identify the problem and classify the requirements in functional and not-functional.
2. **Analyze and build the model of problem:** In this part building the first version of model, it was based in AUTOSAR but at the end it just take AUTOSAR as reference.
3. **Postulate a design solution:** In this point, taking the elements and modules necessary for building the architecture of software.
4. **Validate Solution:** V-Model permitted to validate the design.
5. **Refine Design Solution:** This step is iterative in all design.
6. **Implement Solution:** Programming the software design in C language using the IDE of Freescale (Code Warrior 10.3).

4.2. Scheduler

The BCM project does not use a real time operative system as such, rather uses only a part of it, due to is not necessary to use a RTOS to this implementation. The real time scheduler uses a

real time periodic interrupt (implemented by hardware) in order to generate the heartbeat of the operative system, this heartbeat is known as the OS tick.

The flow of interrupt service routine (ISR) of the timer interrupt is described in the *Figure 5*, keep in mind to setting the timer interrupt, must be configured before the PLL.

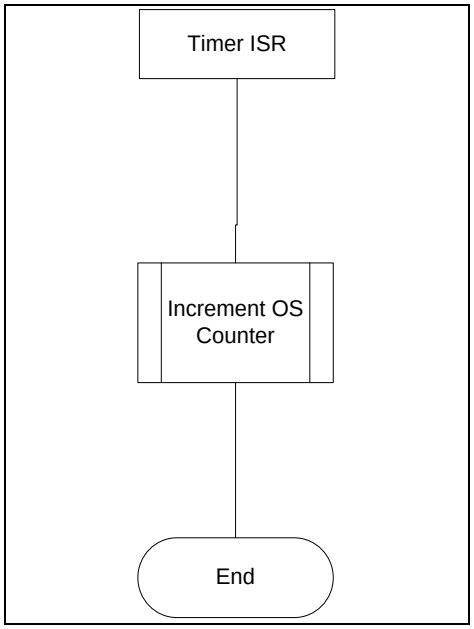


Figure 5: ISR of the periodic timer interrupt

The configuration of the tasks to be executed is performed by the task tables, this tables are a set of periodic task which will be executed for the scheduler to the operation of the BCM.

This implementation can have different task tables to offer flexibility for configure different modes of operation for the BCM. The modes of operation can be selected as *normal mode* and *low voltage mode*. *Table 6* show how these tables are structured.

A Fully cooperative scheduling algorithm is used to schedule the execution of the all task in the system.

Task Name	Functions	Rate [mS]
Task2MS	Debounce Key Status Debounce Brake Pedal	2
Task4MS	Get Battery Level Get Brake Pedal Status	4
Task8MS	Set PWM Duty Cycle Send Data to Cluster Get Lights Signal Status Get Stalk Status	8
Task16MS	Process Received CAN Frames	16
Task32MS	Set PWM Duty Cycle Transmit Stalk Status Transmit Lights Status Transmit Brake Pedal Status	32

Table 6: Task table definitions

At the begin of the execution of the tasks, the selection of the active table must be performed. This means that after the hardware configuration, the real time scheduler needs to find the active task table, validate that the table is the correct one and store the address of the valid table for the Kernel.

This procedure is represented in the flow diagram of the *Figure 6.*

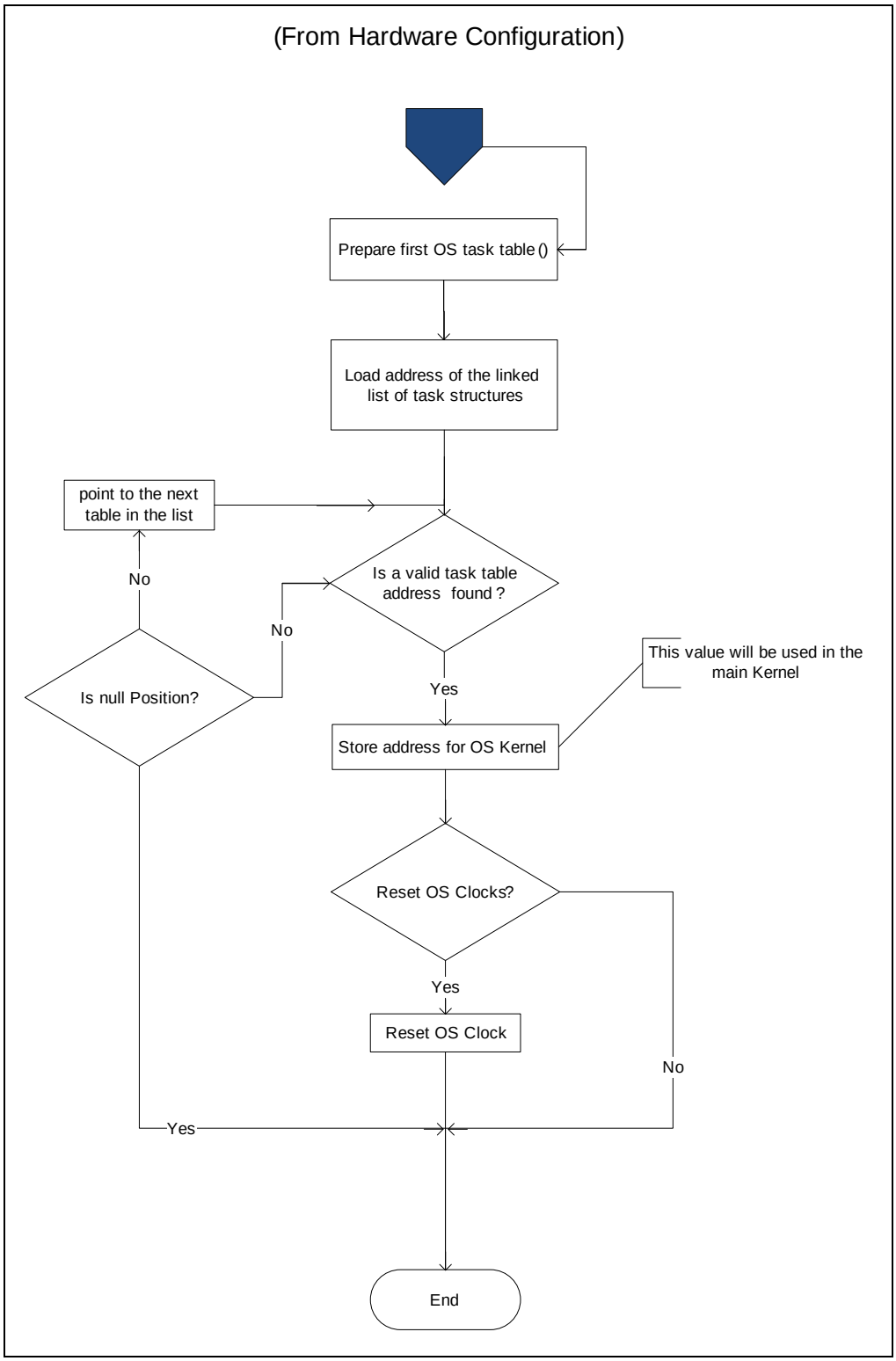


Figure 6: Flow Diagram of the preparation of the first OS task table.

The decision of which task will be executed will be performed by the kernel of the scheduler. The logic implementation of this module is illustrated in *Figure 7*.

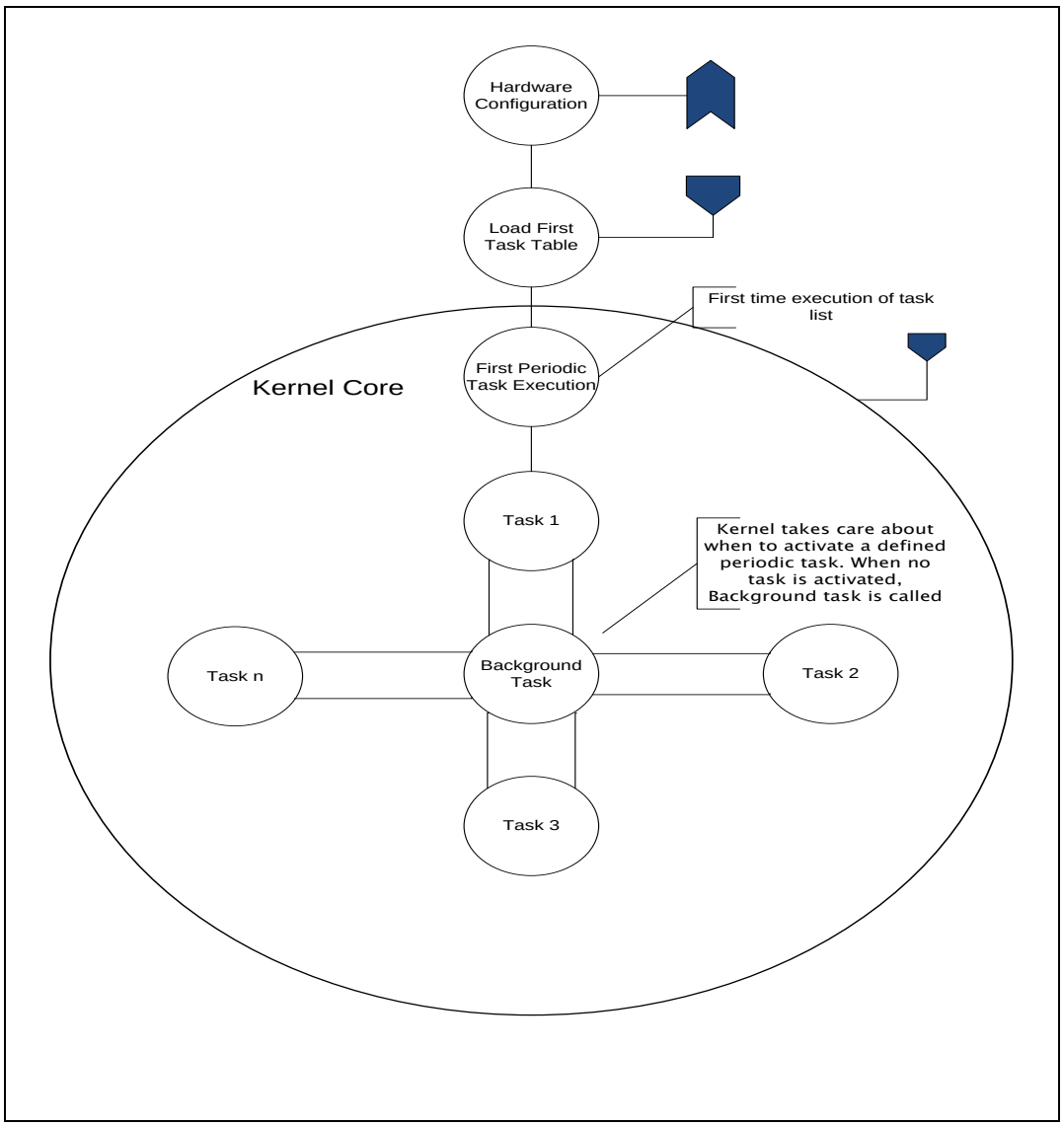


Figure 7: OS kernel block diagram.

4.3. MCU Abstraction Layer

This layer is the lowest software layer of the basic software. It contains internal drivers, which are software modules with direct access to the microcontroller internal peripherals and memory mapped microcontroller external devices. Its task is to make higher software layers independent of microcontroller.

4.3.1 CAN Driver

CAN Bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other a vehicle without a host computer. Also it is a message-based-protocol, designed specifically for automotive. *"The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991"* (Freescale Semiconductor, MC9S12XEP100 Reference Manual, 2012). Some characteristic are:

- Implementation of the CAN protocol — Version 2.0A/B
  - — Standard and extended data frames
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mbps1
  - — Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wake-up functionality with integrated low-pass filter
- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Programmable bus-off recovery functionality
- Separate signaling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)

• Programmable MSCAN clock source either bus clock or oscillator clock

• Internal timer for time-stamping of received and transmitted messages

• Three low-power modes: sleep, power down, and MSCAN enable

• Global initialization of configuration registers

The CAN bus driver allows communication between BCM and Cluster, it is able to send/receive messages, *Figure* **8.**

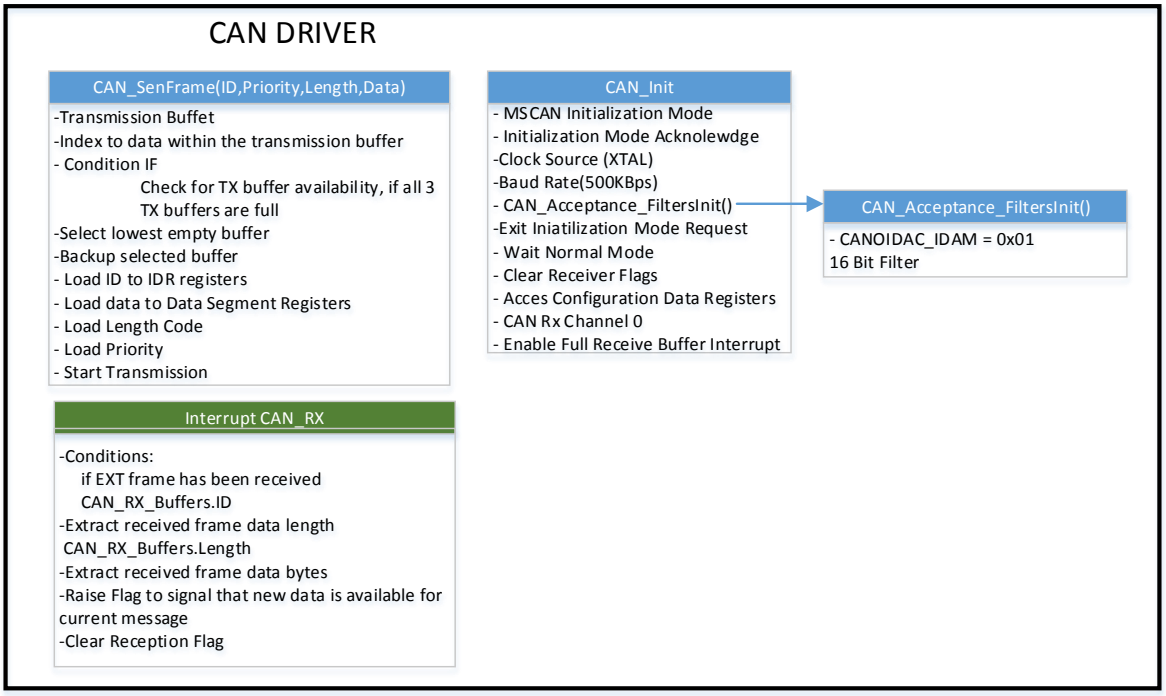


Figure 8: CAN block diagram.

void vfnCAN_init (void)

Description	*This function initialize and configure CAN, wait for initialization mode acknowledge*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called when the all drivers are initialization.*
Post condition	*Enabling MSCAN, configure baud rate, acceptance filters.*
Error Conditions	*None*

void vfnCAN_BaudRateConfig (void)

Description	*Configure the baud rate of CAN.*
Parameter 1	*Void*
Return Value	*void*
Precondition	*Function can only be called once the CAN was initialized.*
Post condition	*CAN CLK, Baud Rate & Pre-scaler will be enabled.*
Error Conditions	*None*

UINT8 u8CAN_SendFrame(UINT32 u32ID, UINT8 u8Prio, UINT8 u8Length, UINT8 *u8TxData)

Description	*Transmit Data Frame*
Parameter 1	*UINT32 u32ID (Identifier which represent the priority of message)*
Parameter 2	*UINT8 u8Prio (Priority based on bus arbitration)*
Parameter 3	*UINT8 u8Length (Length section of data)*
Parameter 4	*UINT8 *u8TxData (Data to be transmitted)*
Return Value	*NO_ERR (if the frame was sent successfully, it return NO_ERR)*
Precondition	*Function can only be called once the CAN was initialized, and baud rate as well.*

Post condition	*CAN CLK, Baud Rate & Pre-scaler will be enabled.*
Error Conditions	*None*

4.3.2 PWM Driver

The pwm driver is used to control the intensity of the light such as external or internal. The BCM system has pwm signals that are connected into the input of the Extreme switch, this driver has four functions that are shown in *Figure 9.*

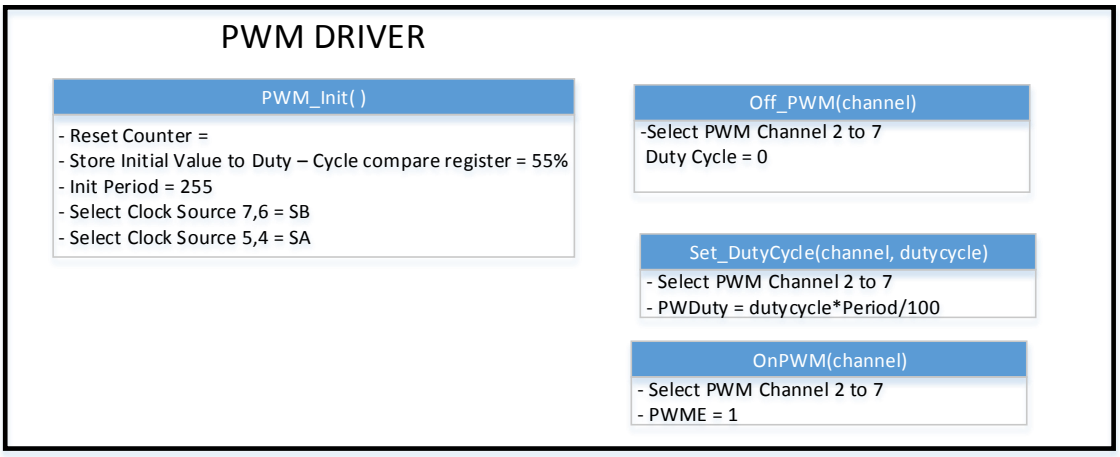


Figure 9: PWM block diagram

void PWM_init (void)

Description	*This function initialize and configure the pwm.*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called when the all drivers are initialization.*
Post condition	*The pwm is configured but not enable yet, to use a channel must first invoke the OnPWM() and Set_DutyCycle() functions.*
Error Conditions	*None*

void On_PWM (int channel[0:4])

Description	*This function set the duty cycle to channel selected.*
Parameter 1	*This function receiver only the parameter channel and it will be an integer, there are constants to pass as argument in this function.*

Return Value	*void*
Precondition	*Function can only be called once the pwm was initialized.*
Post condition	*The PWM is working now, but it has a default duty cycle, is necessary to modify it using a Set_ DutyCycle() function.*
Error Conditions	*None*

void Off_PWM (int channel[0:4])

Description	*This function turn off the channel selected.*
Parameter 1	*This function receiver only the parameter channel and it will be an integer, there are constants to pass as argument in this function.*
Return Value	*Void*
Precondition	*Function can only be called once the channel was turned on.*
Post condition	*The channel selected was turned off.*
Error Conditions	*None*

void Set_DutyCycle (int dutycycle, int channel[0:4])

Description	*With this function can be set the duty cycle from 10% to 90%.*
Parameter 1	*This function receiver two arguments: duty cycle and channel they will be integers.* *Duty cycle: Is a value from 10 to 90, it represented as percent.*
Return Value	
Precondition	*Function can only be called once the channel was turned on.*
Post condition	*The channel selected will be change its duty cycle work.*
Error Conditions	*None*

4.3.3 ADC Driver

The major part of the sensors that used the BCM are analogs, these sensors are connected into ADC port to get a digital value. The ADC can be configured to 10 or 12 bits of resolution. With

the purpose to get the best measure is configured as 12 bits and it is working in interrupt mode. The sensors that are connected in ADC ports are: temperature, humidity, speed, voltage battery, proximity, they are describe in hardware design section.

The functions contained in the ADC driver are shown in

Figure 10.

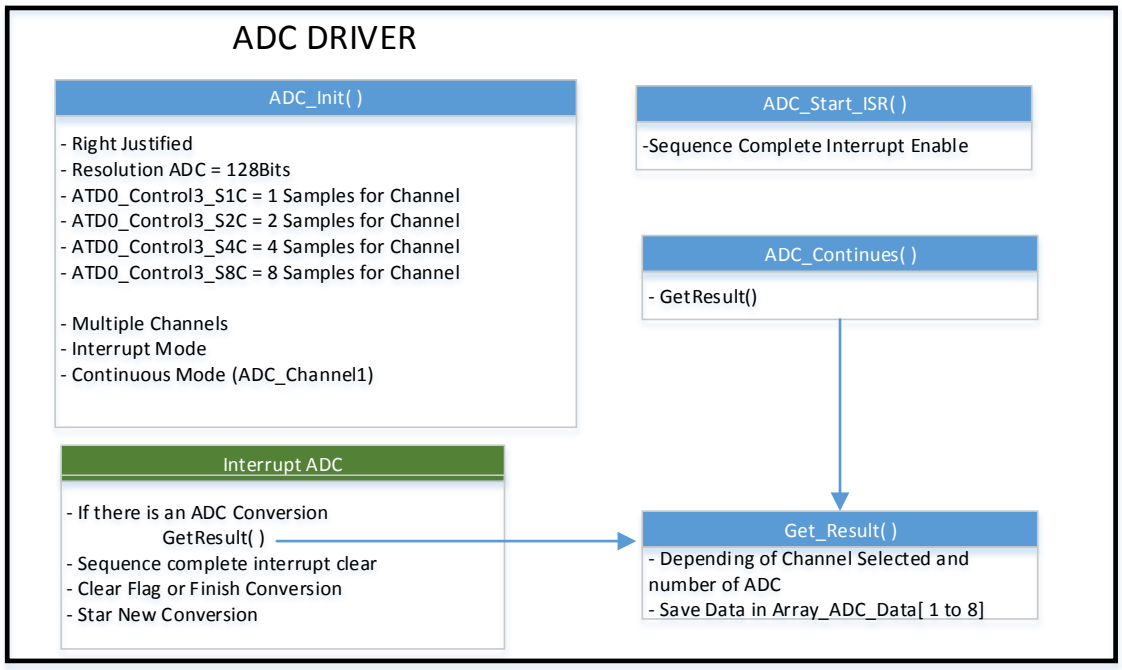


Figure 10: ADC block diagram

void ADC_init (void)

Description	*This function initialize and configure the ADC.* *Resolution, Numbers of Samples for channels* *Multiple Channel Conversions*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called once the channel was turned on.*
Post condition	*The adc is configured but not enable yet.*

Error Conditions	*None*

void vfnADC_START_ISR (void)

Description	*This function initializes the ADC ISR.*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called when the ADC is initialized*
Post condition	*When complete the sequence the interrupt is enable.*
Error Conditions	*None*

void vfnGetResult (void)

Description	*This function choose the channels of ADC*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Depend of value of MASK_SAMPLE.*
Post condition	*Return channels selected.*
Error Conditions	*None*

void vfnADC_Continues(void)

Description	*ADC continues converting the next value.*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called once the channel was reading*
Post condition	*Continue reading the next conversion of ADC.*
Error Conditions	*None*

4.3.4 GPIO Driver

The GPIO port are used to indicate the status in the system performance. To do that, only are used the LEDs from kit develop board, BCM reports if CAN bus lost the communication or if it is working in low power. The key and the brake pedal are connected also in a GPIO ports. The function to use a port are shown in

Figure **11**.

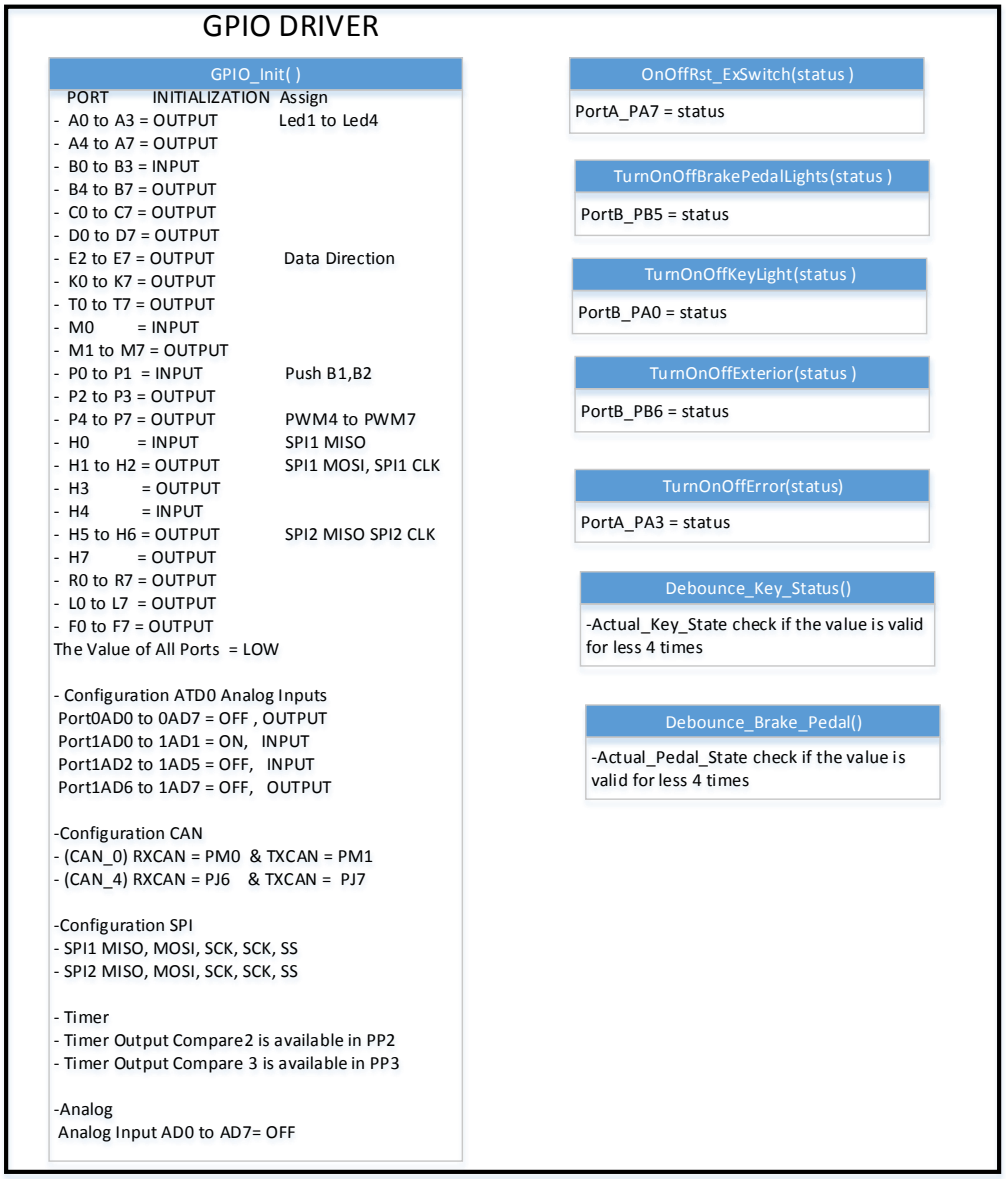


Figure 11: GPIO block diagram

void GPIO_init (void)

Description	*This function initialize and configure the Input/Ouput Ports of Microcontroller.* *INPUT & OUTPUT (Turn OFF) the logic level is LOW* *Receiver and Transmitter of CAN* *Timers* *ADC*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Function can only be called when the all drivers are initialization.*
Post condition	*This GPIO will be configured*
Error Conditions	*None*

void TurnOnOffBrakePedalLights(UINT8 status)

Description	*Show status of BrakePedalLights.*
Parameter 1	*UINT8 u8status* *Status Value* *1= Active, 0 = Not Active*
Return Value	*Void*
Precondition	*Read Status of Port assigned to Pedal.*
Post condition	*Turn ON/OFF BrakePedalLights.*
Error Conditions	

void TurnOnOffKeyLight(UINT8 u8Status)

Description	*This function describe the status of Key.*
Parameter 1	*UINT8 u8Status* *Status Value* *1= Active, 0 = Not Active*

Return Value	*Void*
Precondition	*Read Status of Port assigned to Pedal.*
Post condition	*Continue reading the next conversion of ADC.*
Error Conditions	*None*

void TurnOnOffExterior(UINT8 u8Status)

Description	*This function describe the status of Exterior light.*
Parameter 1	*UINT8 u8Status* *Status Value* *1= Active, 0 = Not Active*
Return Value	*Void*
Precondition	*Read Status of Exterior.*
Post condition	*Turn ON/OFF Exterior.*
Error Conditions	*None*

void TurnOnOffInterior(UINT8 u8Status)

Description	*This function describes the status of interior lights.*
Parameter 1	*UINT8 u8Status* *Status Value* *1= Active, 0 = Not Active*
Return Value	*Void*
Precondition	*Read Status of Interior lights.*
Post condition	*TurnON/OFF Interior lights*
Error Conditions	*None*

void vfnDebounceKeyStatus(void)

Description	*This function check the Status of Key*
Parameter 1	*Void*
Return Value	*Void*

Precondition	*Function can only be called when the user wants to initialize the car.*
Post condition	*Initialize the mean system*
Error Conditions	*None*

4.3.5 PIT Driver

The PIT driver are describe in *Figure 12*.

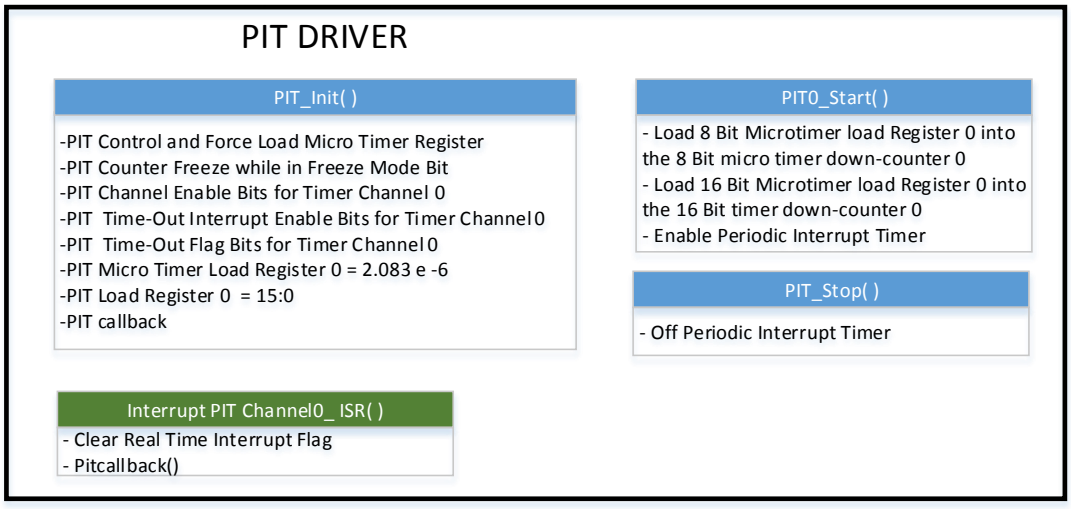


Figure 12: PIT block diagram.

void PIT_init (Ptr_to_fctn callback)

Description	*This function initializes registers to configure the Periodic Interrupt Timer.*
Parameter 1	*Ptr_to_fctn callback* *When the Periodic Interrupt Timer is finished*
Return Value	*callback*
Precondition	*Interrupt Timer is generated.*
Post condition	*The Segment Time will be generated.*
Error Conditions	*None*

void vfnPIT0_Start (void)

Description	*Initialize the count of the Periodic Interrupt of Time*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*Ostick start incrementing is count.*
Post condition	*Enable Periodic Interrupt Timer.*
Error Conditions	*None*

void vfnPIT_Stop(void)

Description	*Stop the count of the Periodic Interrupt of Time.*
Parameter 1	*Void*
Return Value	*Void*
Precondition	*None*
Post condition	*Periodic Interrupt of Time will be stopped.*
Error Conditions	*None*

4.3.6 Watchdog Driver

The watchdog (WD) timer is a common feature on many MCUs. The purpose of the watchdog is to allow the system or application a means to recover in the case of errant code execution or other events that may cause uncontrolled operation of the MCU. Typically, a watchdog is a continuously running timer that may be configured by the application so that it expirers or rolls over at a predetermined time interval. This interval is usually determined by the system clock frequency and a watchdog timeout value that is set by the application.

The application must perform some specific action before the time occurs, which causes a reset of the watchdog timer, and a restart of the timeout count. The required action may be writing a specific location in memory, setting or clearing a bit, or some other method. The application must service the watchdog periodically at intervals short enough to prevent the timeout. The WD must have the same time value that the task with the less execution time, in this case is 2ms.

4.3.7 Memory Checksum

The Memory Checksum is a small size code, its purpose is detecting errors that may have been introduced during its transition or storage. The integrity of the data can be checked at any later time by re-computing the checksum and comparing it with the stored one. If the checksums match, the data was likely not accidentally altered. The checksum algorithm will yield high probability when the data is accidentally corrupted, if the checksums match, the data has the same high probability of being free of accidental errors.

Once the code is finished, the developers use a part of code to calculate the 1s complement checksum, this value is put in the memory. To verify checksum when the system is running, the background function calculate the checksum and add the 1s complement checksum value which had been put in the memory, and if the result is different to zero, therefore the memory was corrupted.

In the linker file must assigned a memory space to save the value of the 1s complement.

```
CHECKSUM_F       INTO  ROM_FEFE;
ROM_FEFE     =   READ_ONLY   DATA_NEAR IBCC_NEAR  0xFEFF TO   0xFEFF;
```

With the pragma directive it assigned the value to a variable, this value will be stored into the memory space that it had been reserved.

```
#pragma DATA_SEG CHECKSUM_F
UINT8 u8CheckSumValue = 0x7C;
#pragma DATA_SEG Default
```

Now, it add up all the memory flash together with the checksum value put on previously in the memory flash, and the result must be “0”.

The *Figure 13* is a screenshot when the memory flash has been added up. The letter A is the result of add up all the memory flash and letter B is the value that was put on into the memory flash.

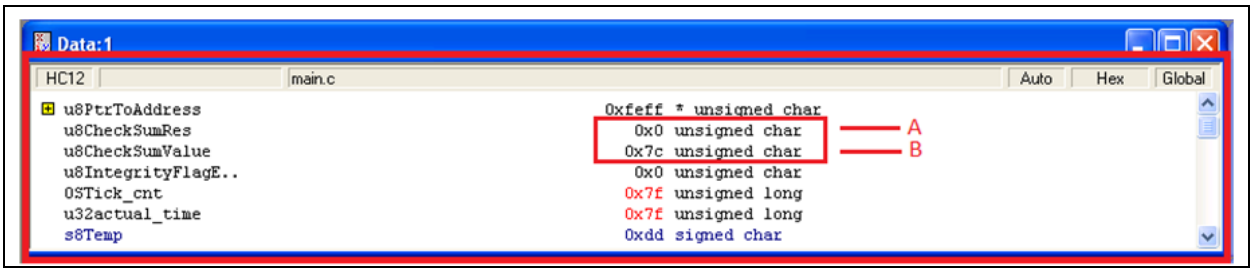


Figure 13: Checksum

4.4. ECU Abstraction Layer

This layer mainly makes use of the MCU functions has the low level drivers, particularly this layer not make use of the registers *"This layer interfaces the drivers of the Microcontroller Abstraction Layer. It also contains drivers for external devices. It offers an API for access to peripherals and devices regardless of their location (microcontroller internal/external) and their connection to the microcontroller"* (Mitidieri, 2008).

It task is to make higher software layers independent of ECU hardware layout. An interface contains the functionality to abstract the hardware realization of a specific device for upper layers. It provides a generic API to access a specific type of device independent on the number of existing devices of that type an independent on the hardware realization of the different devices. The interface does not change the content of the data.

In general the interface are located in this Layer. A handler is a specific interface which controls the concurrent, multiple and asynchronous access of one or multiple clients to one or more drivers, it does not change the content of the data.

How to apply this layer in BCM project will be described in the next sections.

4.4.1 CAN Handler

It contains some functions that permit to obtain the signals of different sensors sending by cluster and send messages with the system status, *Figure 14* has these functions.

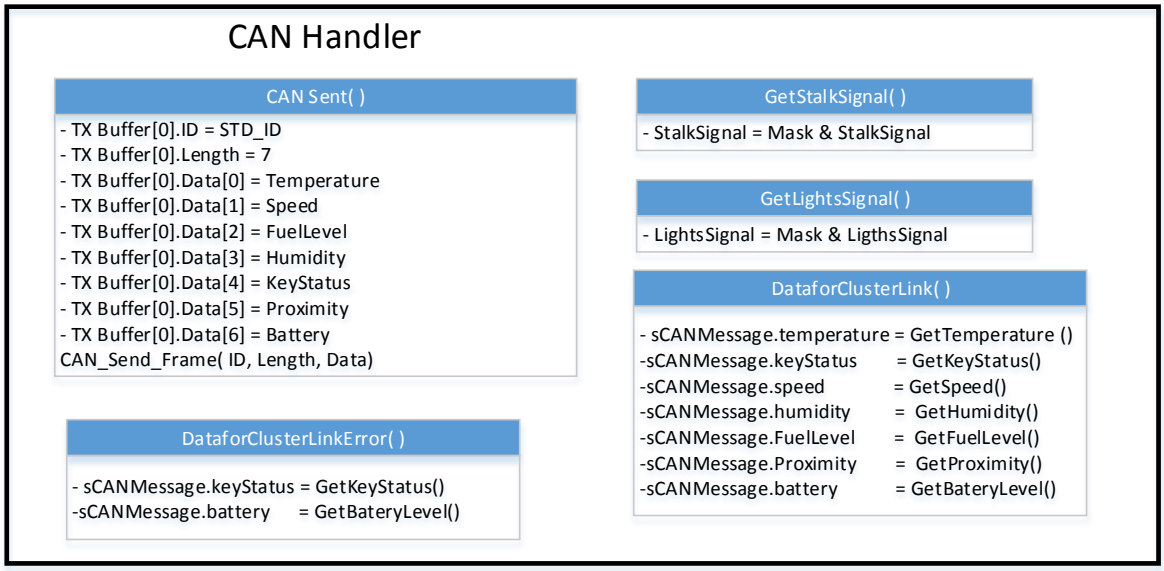


Figure 14: CAN Handler Block Diagram

4.4.2 ADC Handler

It permits to get the values for Proximity, Humidity, Temperature and Battery Level sensors, the *Figure 15* shows the functions that are using in this driver.

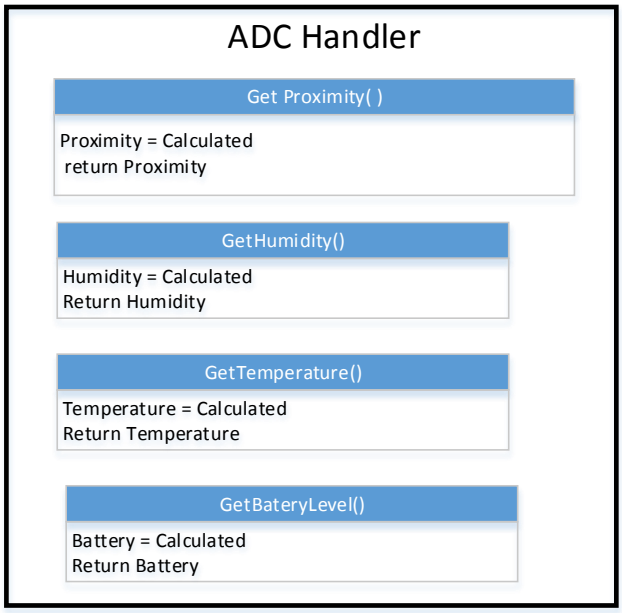


Figure 15: ADC handler block diagram

Below is described the flow diagrams to perform a conversion to the value that the user can interpret, for instance to degree, RH%, km/h. take in mine, these values are send to cluster and this one is shown in the display. These functions are in a low priority task.

4.4.2.1 Temperature

Temperature: a centigrade degree is equal to 10mv, this value is given by LM35 sensor and it is linear. The ADC resolution is 12 bits which is equal to 4095 units, and the CONSTCENT constant defined is 500 due to 5v in the input of the microcontroller.
As well the sensor is linear if it has 300mv in the output, which means, the temperature is 30 °C, now 300mv in digital value are 250u, if the operation is performed it will get the value in Celsius.

$$Cent = \frac{ADCValue * CONSTCENT}{RESOLUTION} = \frac{250 * 500}{4095} = \frac{1250000}{4095} = 30.52°C$$

As well is not using float point the result is 30 °C.

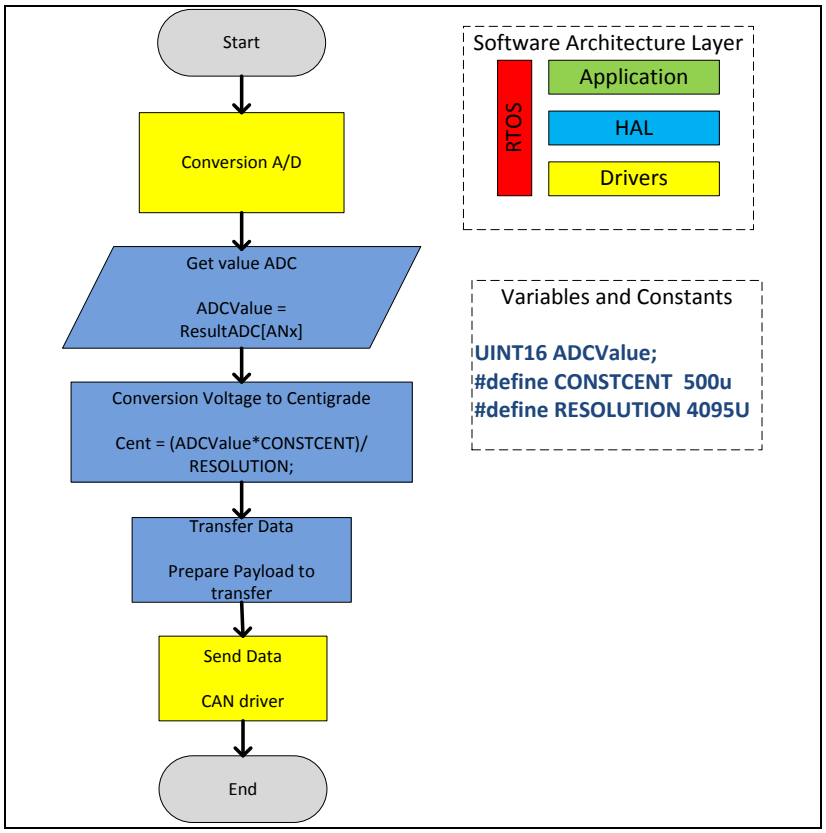


Figure 16: Converter ADC Digital Value to Centigrade.

4.4.2.2 Humidity

Humidity: The humidity range from 10 %RH to 90 %RH, which means, in voltage range are 1 to 2.9v, it has an offset of 1v or 833 units, and the max range is 2497 units. If the operation is performed it will get the value in %RH. Assuming that it has a 2v in the output, which means, the humidity is 60 %RH.

$$HUM = \frac{ADCValue * MAX\_PORC}{MAX\_RANGE} = \frac{1666 * 90}{2497} = 60.04\ \%RH$$

As well is not using float point the result is 60 %RH.

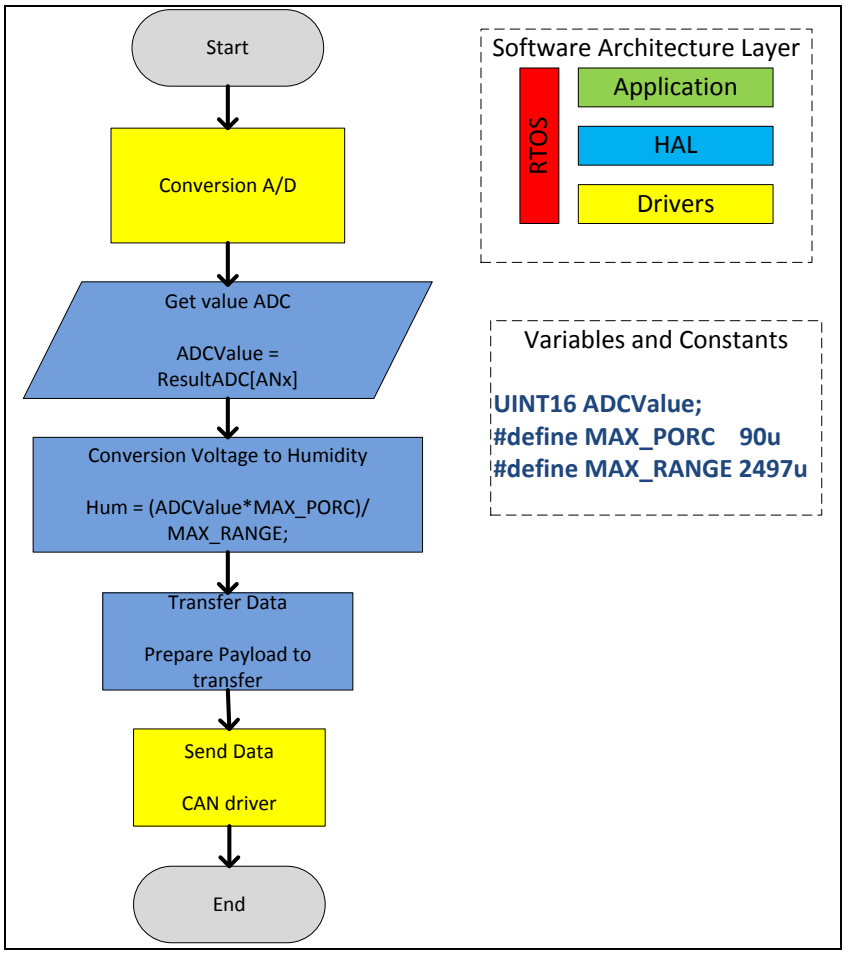


Figure 17: Converter ADC Digital Value to Humidity

4.4.2.3 Voltage Battery

Battery Voltage: this is the only part of code that it is on the application layer. The CONSTVOLT constant defined is obtained divided 4095 (ADC resolution) by 12 (max battery voltage). If the operation is performed it will get the value in Voltage. Assuming that it has a 4.5v in the output, which means, the voltage is 11v.

$$Voltage = \frac{ADCValue}{CONTSVOLT} = \frac{3750}{341} = 10.99v$$

As well is not using float point the result is 11v.

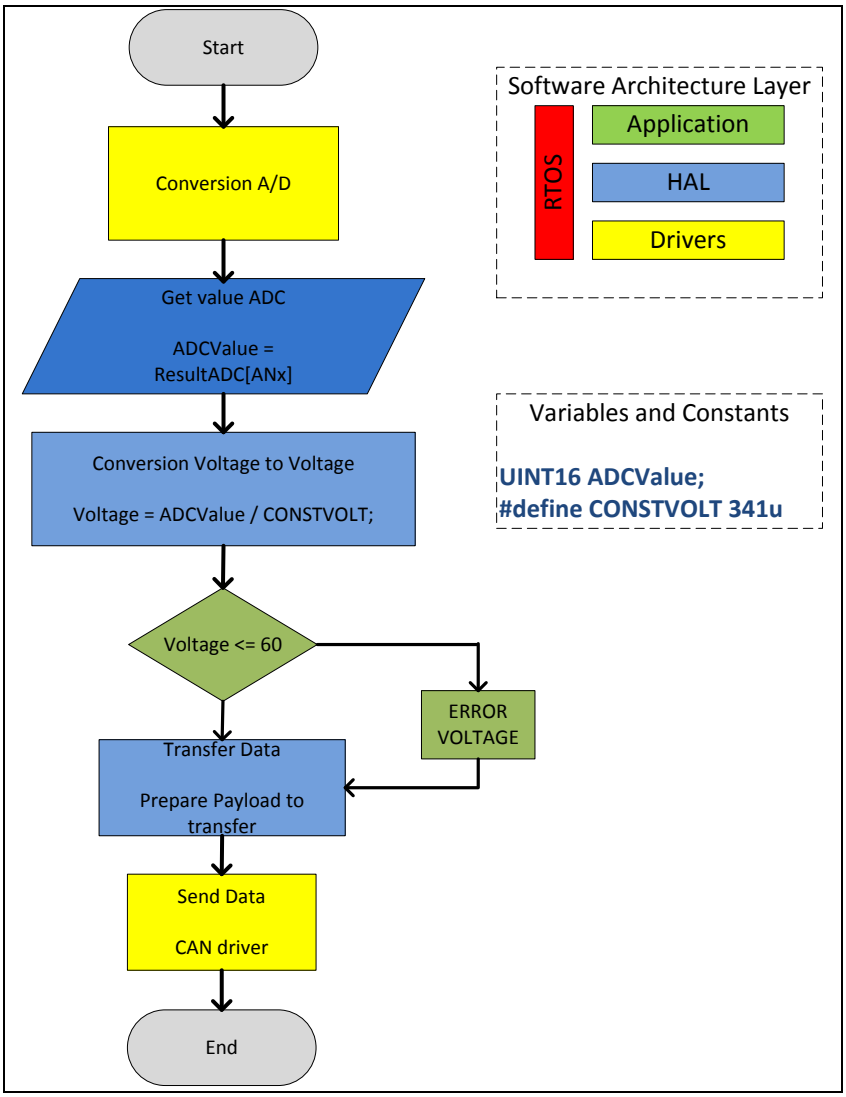


Figure 18: Converter ADC Value to Voltage

4.4.2.4 Proximity

Proximity: to explain this flow chart it will be used an example, assuming that the proximity sensor has an object to a 4 meters (400cm), its output voltage is 1.53, this voltage once converted in digital value is 1280 units. The proximity resolution is:

$$Res_{Sensor} = \frac{5}{512} = 9.8mv = 1\ in\ or\ 2.54cm$$

And the ADC resolution is:

$$Res\_ADC = \frac{5}{4095} = 1.22mv$$

If Res_Sensor is divided by Res_ADC the result is 8, which means, each 8 units is equal to 2.54cm. The RESOLUTION_IN_CM constant is Res_Sensor but multiplied by 100, in order to does not used float point, at the end of the operation the result is divided by 100 to get the value in centimeters.

$$Dist = \frac{\left(\frac{ADCValue}{PRS\_TO\_ADC\_UNIT}\right) * RESOLUTION\_IN\_CM}{GO\_TO\_CM} = \frac{\left(\frac{1280}{8}\right) * 254}{100} = 398.14cm$$

The result was about 4 meters.

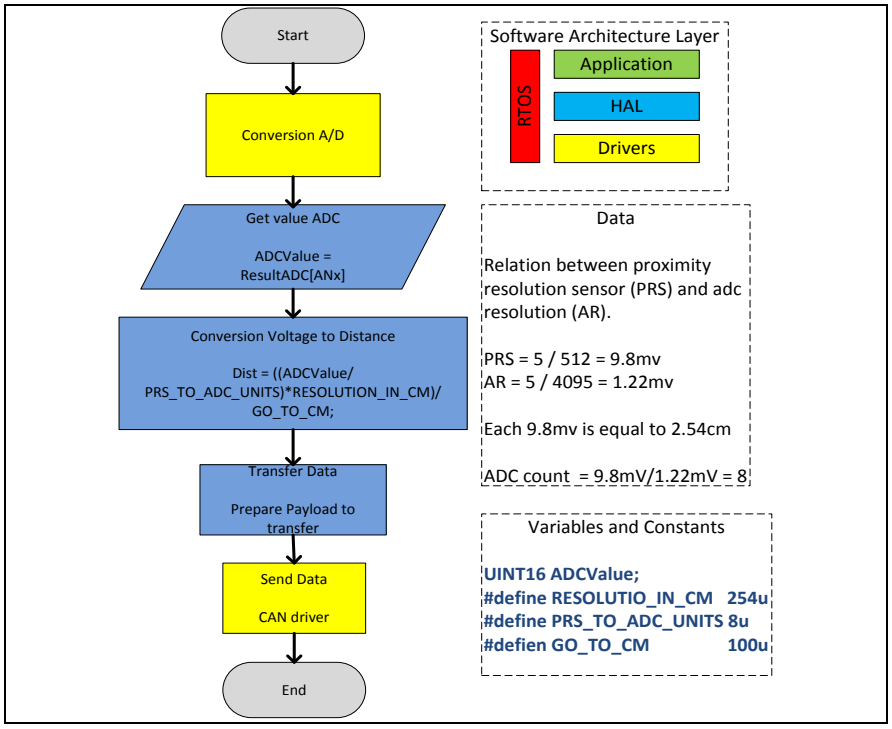


Figure 19: Converter ADC to Distances

4.5. Hardware Design

This section has a description of the sensors, its schematic circuit, characteristic curve and diagram flow to make a conversion from analog to digital, such as equations and formulas to perform a conversion. It also has a brief description about each IC that are using in the project.

4.5.1 Bill of Materials

The *table 2* shows all components that are used in the BCM project. Should be noted that the components of *Table 7* were not used due to it just was a prototype and were used only potentiometers, but in the code is implemented as if it were using the sensors that are shown in the *Table 7*.

#	Description	Part Number	Vendor	Qty
1	Microcontroller	DEMO9S12XEP100	Freescale	1
2	Extreme switch	MC35XS3400	Freescale	2
3	Sensor Temperature	LM35	Texas Instruments	1
4	Sensor Humidity	HIH-4030	Honeywell	1
5	Sensor Speed	HS130-400	Sensoronix	1
6	Sensor Proximity	MB-1000	Maxbotix	1
7	Optocoupler	4N32	Vishay	2
8	Cap. Ceramic SMT 0.1uF/50v	C0805C104M5RACTU	Kemet	3
9	Resistor SMT 100K-5%	CRCW2512100KJNEGHP	Vishay	1
10	Resistor SMT 80K-5%	CRCW251280KJNEGHP	Vishay	1
11	Resistor SMT 50K-5%	CRCW251250KJNEGHP	Vishay	1
12	Resistor SMT 690R-5%	CRCW251260RJNEGHP	Vishay	1

Table 7: Bill of materials

4.6. Sensors

The BCM uses a variety of sensors both analog and digital, in this way can understand the environment and it take action through of the actuators, and send the status to the cluster and show it to the users.

Those sensors that give its value in a voltage range must be connected into the ADC port, to change the analog value to a digital value. The others sensors that give a digital value, are connected into a digital port but before that, it must be isolated through an optocoupler.

The *table 8* shows which sensors are analog and which ones are digitals, all of them work with voltage level not more than 5v.

Sensor	Output
Temperature	Analog
Humidity	Analog
Speed	Analog
Voltage Battery	Analog
Proximity	Analog
Key status	Digital
Brake Pedal	Digital

Table 8: output's sensors

4.6.1 Temperature Sensor

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. This sensor was implemented in the BCM project as it shows in *Figure 20.*

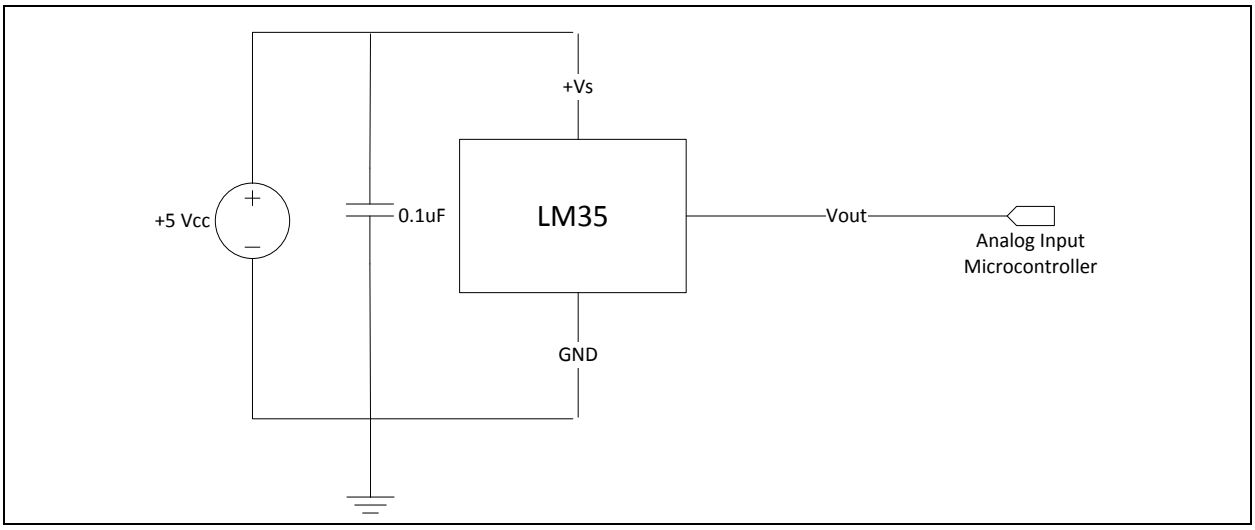


Figure 20: Sensor Temperature Schematic

This is a typical performance characteristic to the LM35 sensor. The

Figure 21 describe the behavior of the device with different range of voltage and temperature, it can see that the behavior is linear. The formulas to change the value from analog to digital were taken from the

Figure 21.

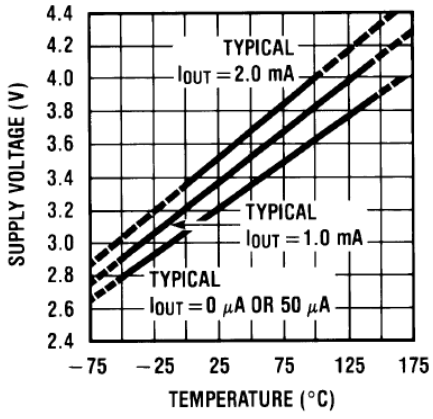


Figure 21: LM35 Performance

4.6.2 Humidity Sensor

HIH-4030 humidity sensor measures relative humidity (%RH) and outputs the result as an analog voltage on the pin labeled OUT. The sensor output can be read by an ADC on a microcontroller and varies nearly linearly with humidity, which makes the readings easy to

process. The humidity sensor is powered with 5 V and typically draws 200 μA. The pins on the board have a 0.1" spacing, making them compatible with 0.1" male header pins and standard breadboards and per boards. The *Figure 22* shows the circuit implemented in the BCM.

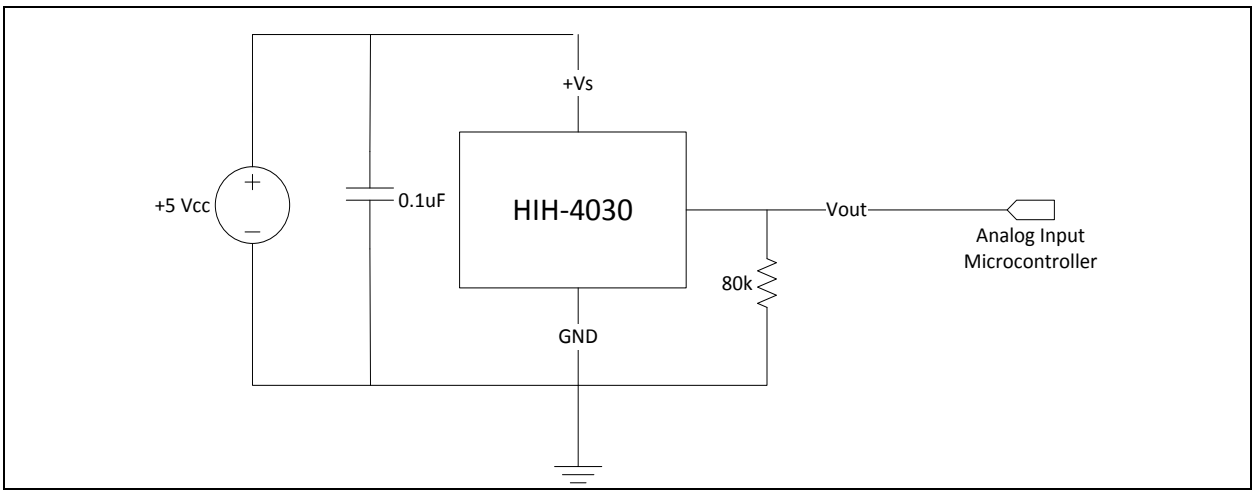


Figure 22: Sensor Humidity Schematic

Ranges of humidity measurements are from 10 to 90 degrees. This way, we can see that the output voltages that must be measure with the analog port (ADC) are from 1 to 3.5 volts. Note that the best case is the bold line.

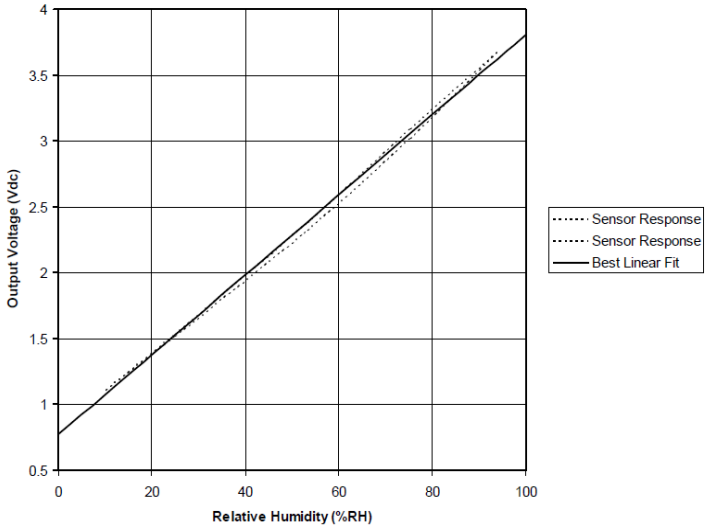


Figure 23: HIH-3040 Performance

4.6.3 Speed Sensor

The Hall element is constructed from a thin sheet of conductive material with output connections perpendicular to the direction of current flow. When subjected to a magnetic field, it responds with an output voltage proportional to the magnetic field strength. The voltage output is very small (µV) and requires additional electronics to achieve useful voltage levels. When the Hall element is combined with the associated electronics, it forms a Hall Effect sensor" (Honeywell .Inc, 2005).

Hall-Effect Zero speed sensors provide very precise measurements of movement event at zero speed which makes the Hall-Effect zero speed sensors ideal for speed measurements. Hall-Effect sensors provide digital output with constant amplitude signal regardless of variation of the speed.

This sensor was implemented in the BCM project as it shows in *Figure 24*.

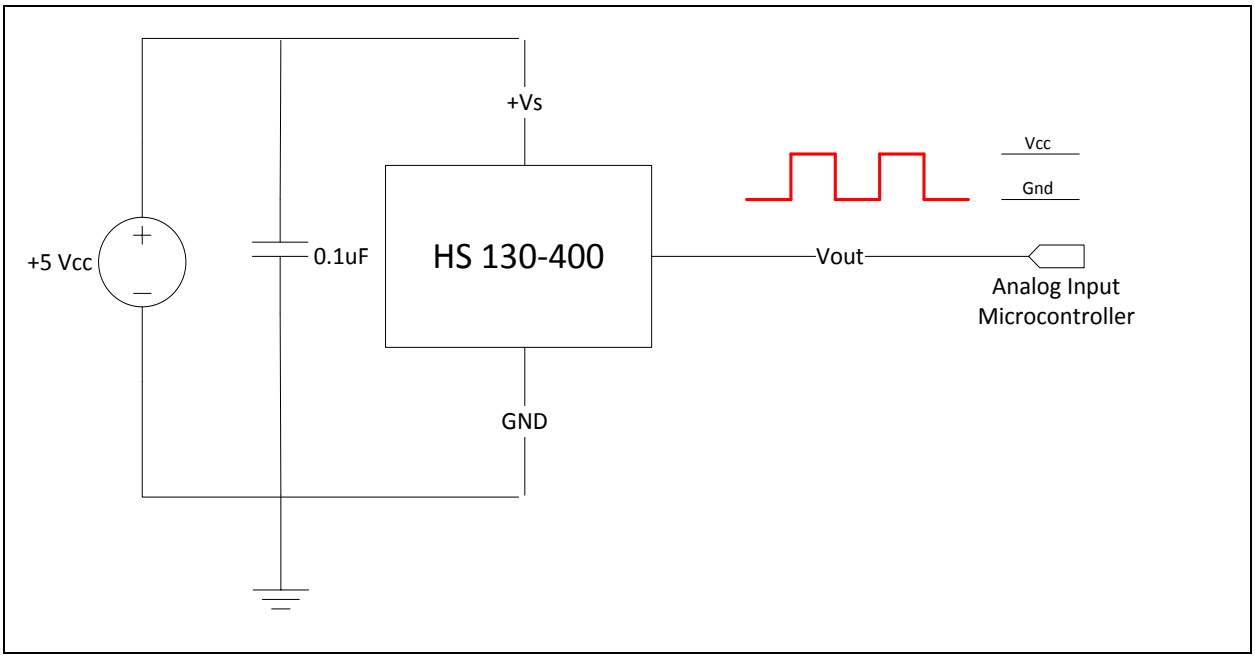


Figure 24: Sensor Speed Schematic

To explain the *Figure 25* must make clear the operation of this sensor. Air gap is a distance between target gear and the sensor, the gear pitch is teeth number plus 2 divided by outside diameter. The output from this sensor is a digital signal (square wave), the frequency output depend of two factors, the teeth number and the speed of the shaft.

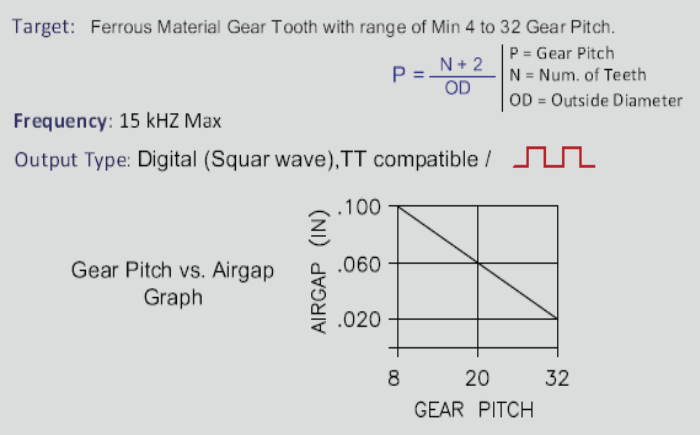


Figure 25: Performance Gear Pitch/Airgap Graphic

4.6.4 Voltage Battery Sensor

The voltage value of the battery is measurement using a voltage divider. This one return a value from 0 to 4.5v. The resistor value using in the circuit of the *Figure 26* must be above the kilo ohms to get a small amount current. To obtain the R1 resistor value is explained in the *equation 1*.

[1]	Equation to calculate resistor value.	Vin = 13.5 Vout_max = 4.5 R1 = 100k R2 = X	$R2 = \frac{R1 * Vout_{max}}{Vin - Vout_{max}} = \mathbf{50}k$

Equation 1: Calculated value of R2

In this way can be calculated the low voltage condition, this is the only part where the BCM makes a decision, that's mean, is the only code that has the application layer.

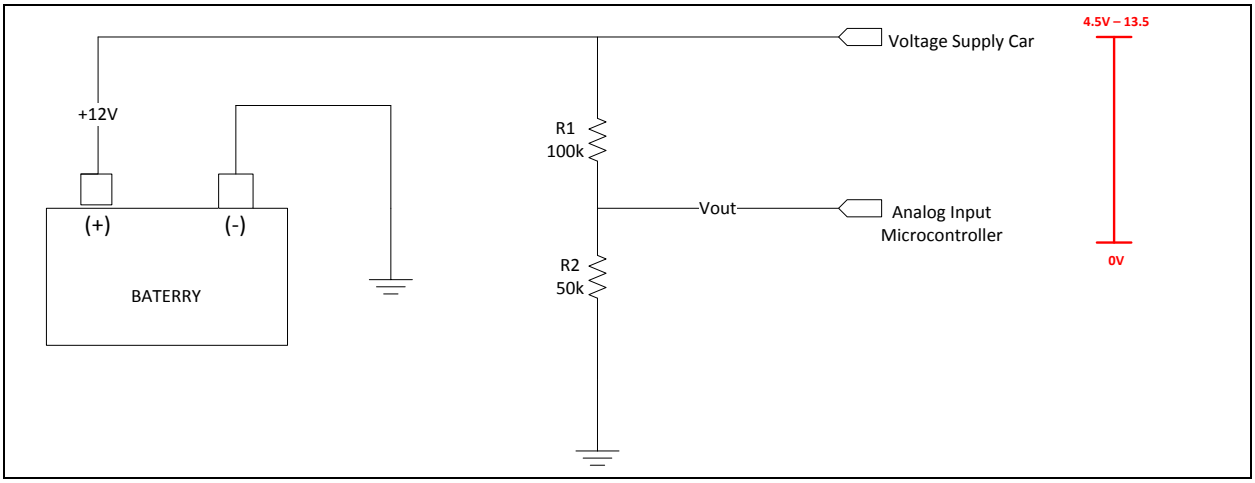


Figure 26: Sensor Status Battery

4.6.5 Proximity Sensor

This compact sonar range detects objects from 0 to 6.45 m with a resolution of 2.5 cm for distances beyond. Unlike other sonar range finders, the LV-MaxSonar has virtually no dead zone: it can detect even small objects up to and touching the front sensor face. The LV-MaxSonar provide three outputs types: analog output with a scaling factor of (Vcc/512) per inch, serial output Rx/Tx the RS232 standard, and a PWM output, the distance can be calculated using the scale factor of 147uS per inch, but in this case is only used the analog output. The *Figure 27* shows how was implemented this device.

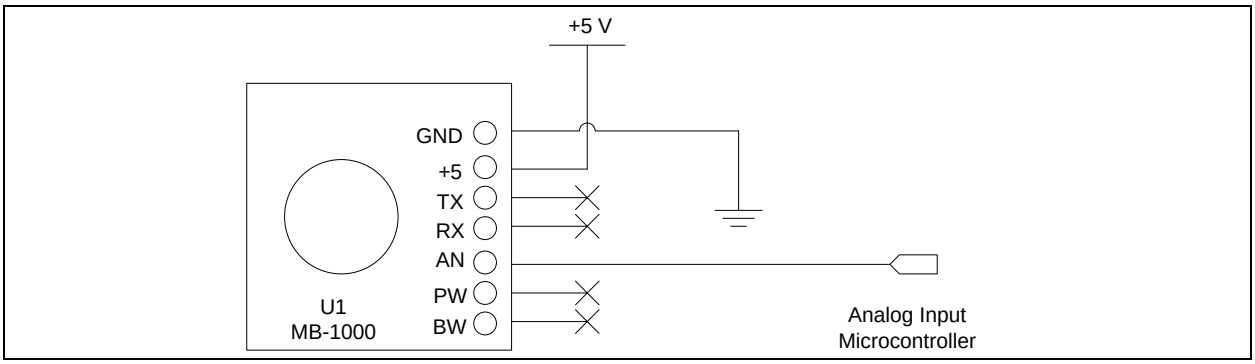


Figure 27: Sensor Proximity

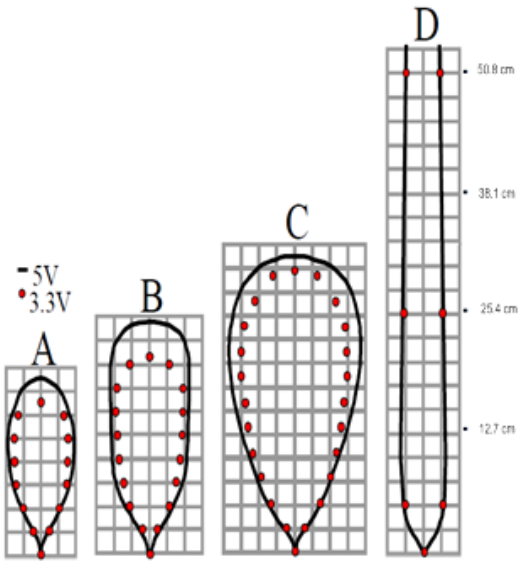


Figure 28: beam characteristics

A) 0.635-cm diameter dowel, note the narrow beam for close small objects.

(B) 2.54-cm diameter dowel, note the long narrow detection pattern.

(C) 8.25-cm diameter rod, note the long controlled detection pattern.

(D) 27.94-cm wide board moved left to right with the board parallel to the front sensor face and the sensor stationary.

Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.

4.6.6 Key Sensor

The key status signal is captured through an input digital of the microcontroller, when the car is turn on, this signal goes to 5v momently and it is debouncing by microcontroller, noteworthy that key status signal is monitoring in the high priority task due to importance for the operation of the rest of the system, namely, if the car is turn off the BCM does not send any information to the cluster, BCM just perform some functionalities like turn on and off the lights.

[3]	Equation to calculate resistor (R1) value.	Vin = 12 Icc_max = 17mA R1 = X	$R1 = \frac{Vin}{Icc_{max}} = \mathbf{690}R$

Equation 2: Get R1 resistor value

The circuit of the *Figure 29* for check the key status signal uses only an optocouple to isolate the battery voltage and uses a digital voltage of 5 volts, the R1 resistor is calculated according to the *equation 2*.

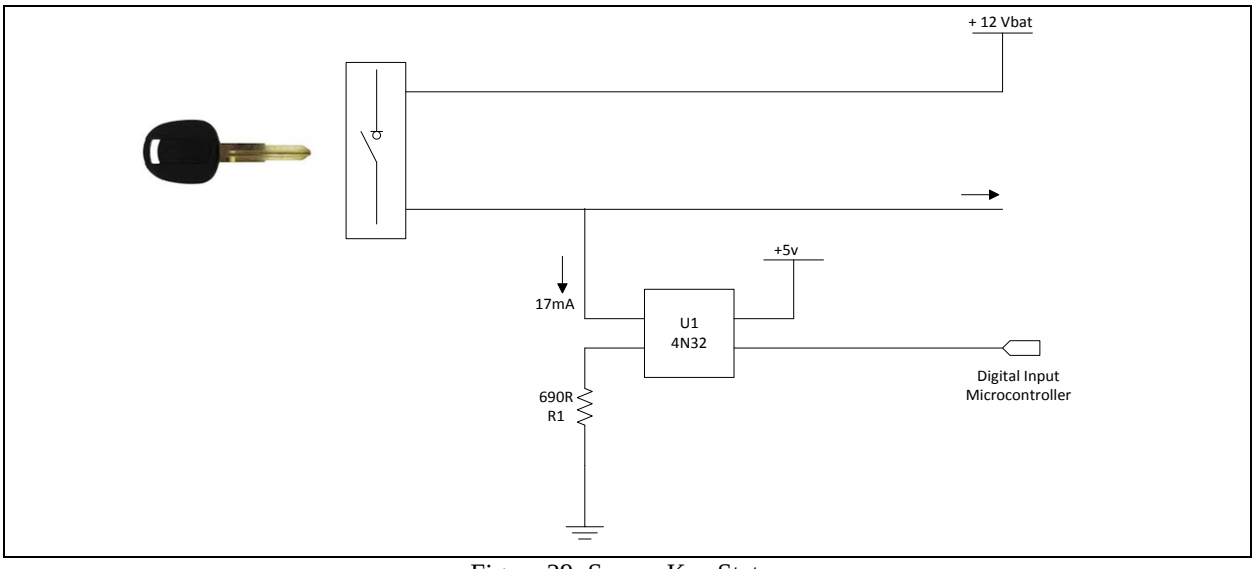


Figure 29: Sensor Key Status

4.6.7 Brake Pedal Sensor

This circuit of the *Figure 30* is similar to the key status circuit and it is inside of a high priority task as well. The brake pedal signal is debounced by a task of the scheduler, this signal also handles the brake pedal lights.

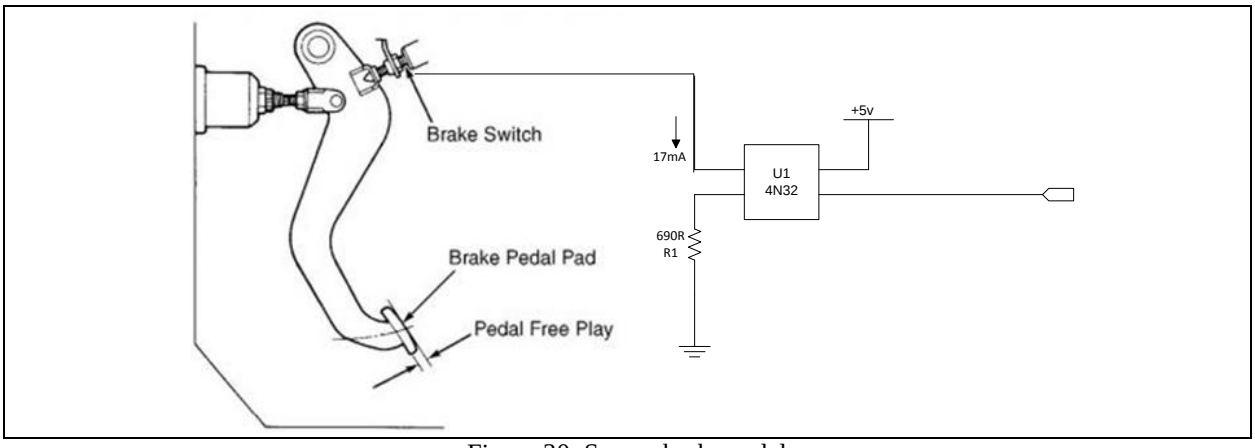


Figure 30: Sensor brake pedal

The R1 resistor is calculated according to *the equation 3*.

[3]	Equation to calculate resistor (R1) value.	Vin = 12 Icc_max = 17mA R1 = X	$R1 = \frac{Vin}{Icc_{max}} = \mathbf{690}R$

Equation 3: Get R1 resistor value

4.7. Actuators

The BCM handles all the light in the system, these lights are considerate as actuators, but some of them need a load greater than that afforded the microcontroller, so that is necessary to use an extreme switch board to control all the loads. The cluster send us information through CAN to indicate the lights status. This way the BCM knows that light should be off and which one should be on. Worth mentioning that the voltage and current measurements are beyond the scope of this project.

The *Figure 31* describes how the microcontroller, extreme switch boards and actuators interact.

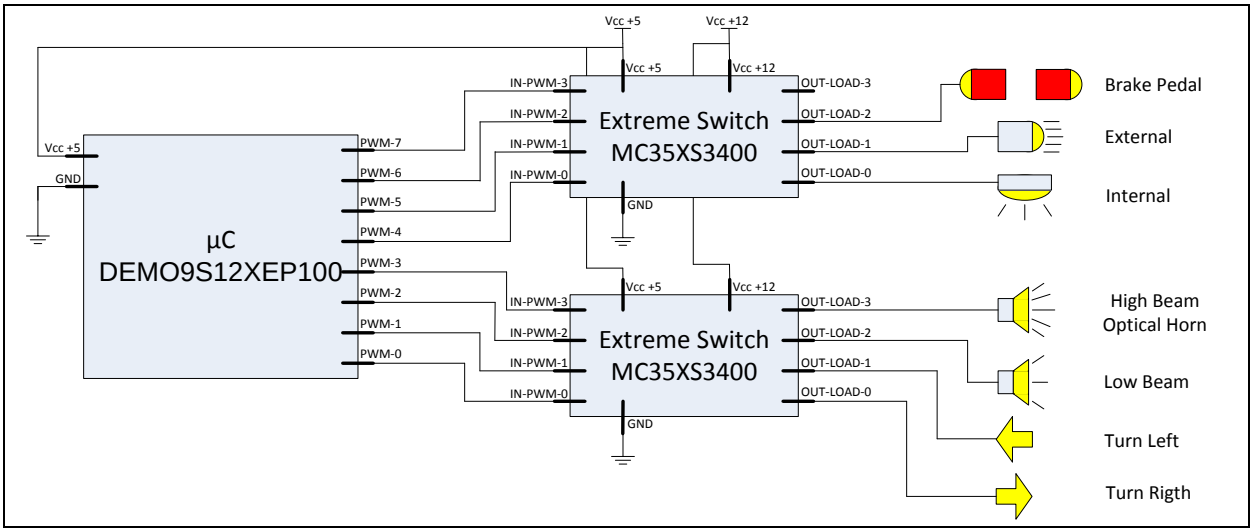


Figure 31: Actuators and extreme switch

4.7.1 Lights

The BCM take over lights based of a message that is receiver from the cluster, how are active the lights are described below:

Brake Pedal: these lights are active when the brake pedal is pressed, and only when the car is on.

External: also called fog lights, it can active only when the car is on.

Internal: are the internal lights of the car, and it can active even if the car is off.

High Beam: The beam of a vehicle's headlight that provides long-range illumination, it can active only when the car is on.

Low Beam: The beam of a vehicle's headlight that provides short-range illumination, it can active only when the car is on.

Turn Left: indicates when you take the left direction, it can active only when the car is on.

Turn Right: indicates when you take the right direction, it can active only when the car is on.

4.7.1 Extreme Switch

Different types of lamps (e.g. halogen, xenon, LED) are used in a variety of lighting functions, such as low and high beam headlights, daytime running lights, brake lights, indicators and others. Being a safety element for the driver, but also a prime source of energy consumption, modern lighting systems are based on relays replacement for enhanced reliability and wiring harness reduction for weight and fuel saving. The extreme switch product family of intelligent high-side power switches provides extensive diagnostics and fail-safe functionality. Paired with lamp load profile tailoring, these switches address the requirements of modern lighting systems.

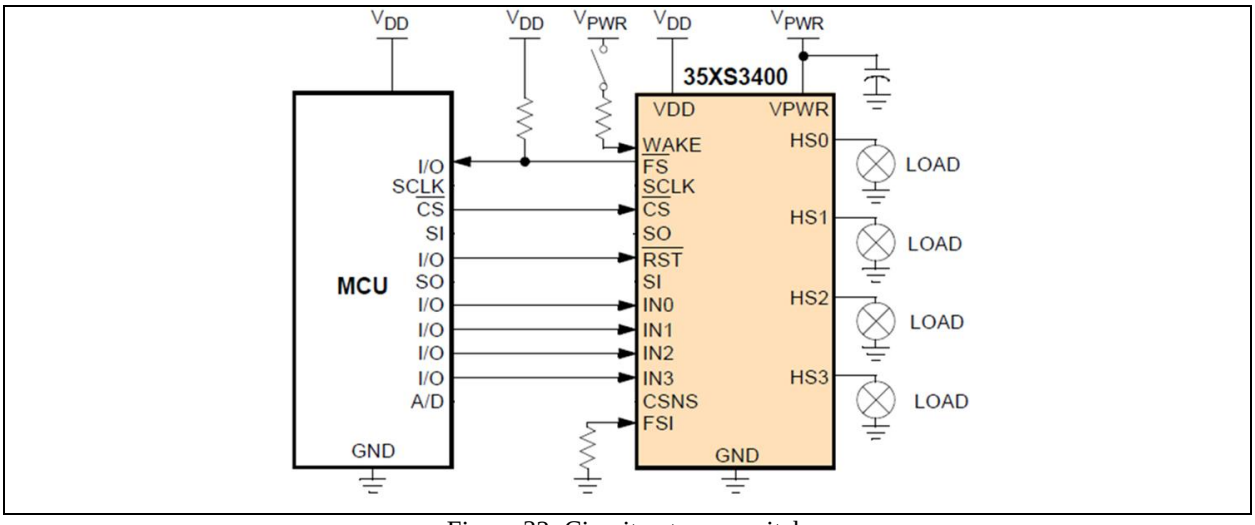


Figure 32: Circuit extreme switch

To this application are used the PWM ports to control the loads of the extreme switch, the fail-safe functionality does not used. The function that control the PWM are in a low priority task due to low importance of the systems, that means, one of the light does not turn on, no matter, because it does not jeopardize the lives of users. The *Figure 32* shows how is implemented the extreme switch circuit in the BCM project.

5. Test Cases

5.1. Software

The purpose of this document is to describe a test procedure of functionality for the Body Controller Module (SYSTEM/INTEGRATION Testing). This test procedure must be performed by the test manager and development team leader with assistance from the individual developers as required.

Test cases described in this document are grouped to be executed following the preconditions each test case indicates respectively. Each test case contains a number of steps that must be executed under the series which have been defined.

Each test case that is included in this document represents a set of conditions under which a tester will determine the properly function of each feature of the BMC, the main goal of test cases specified in this document is to provide an environment in which hardware/software interfaces can be tested.

The interfaces of the BMC are considered as specific pieces which provide access to hardware/software resources of this module, this interfaces gives to the tester an easier way to address the validation of a complex product.

5.1.1 Scheduler

TC-BMC01				
Type	Black Box			
Requirement Reference				
Pre-condition	1.- The debugger tool is running in the PC. 2.- The user has connected the develop board to the PC via USB. 3.- The microcontroller has been flashed with the software module release. BMC_v1.0. 4.- The user has connected 5 digital probes of a logic analyzer in 5 pins of the connector J102 of the development board according the next relation: • 1st probe - J102.5 • 2nd probe J102.7 • 3rd probe J102.9 • 4th probe J102.11 • 5th probe J102.13 The logic analyzer must be powered and configured to measure the period of the waveforms that have described above.			
Post-condition			**TC- BMC02**	
Purpose	Validate the periodicity and correct execution of the tasks handled by the scheduler.			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	In the debugger window select the "Run" option	The microcontroller starts with execution of the pre-loaded code. Once the initialization procedure has been performed and the scheduler is in execution state, the logic analyzer must display 5 squared waveforms. Period of 1st waveform = 2ms Period of 2nd waveform = 4ms Period of 3rd waveform = 8ms Period of 4th waveform =16ms Period of 5th waveform =32ms		

5.2. Sensors

5.2.1 Analogic to Digital Converter (ADC)

TC-BMC02				
Type	Black Box			
Requirement Reference				
Pre-condition	1.- Change SW5 in the chassis of the BMC from Sensors to external power supply option. 2.- Connect a variable voltage power supply (0- 5V) in the next pins of the connector J101 in the development card: • J101.24 – ADC channel 0 • J101.22 – ADC channel 1 • J101.20 – ADC channel 2 • J101.18 – ADC channel 3 • J101.16 – ADC channel 4		**TC-BMC01**	
Post-condition			**TC- BMC03**	
Purpose	Validate the functionality of ADC module (Hardware and software).			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name u8arrayADC_data	In the watch window: u8arrayADC_data[0] = 0 u8arrayADC_data[1] = 0 u8arrayADC_data[2] = 0 u8arrayADC_data[3] = 0 u8arrayADC_data[4] = 0		
2	Power on the voltage power supply and set the output voltage as 0V	In the watch window: u8arrayADC_data[0] = 0 u8arrayADC_data[1] = 0 u8arrayADC_data[2] = 0 u8arrayADC_data[3] = 0 u8arrayADC_data[4] = 0		
3	Set the output voltage to 1 volt.	In the watch window: u8arrayADC_data[0] = 51 u8arrayADC_data[1] = 51 u8arrayADC_data[2] = 51 u8arrayADC_data[3] = 51		

		u8arrayADC_data[4] = 51		
4	Set the output voltage to 3 volts.	In the watch window: u8arrayADC_data[0] = 153 u8arrayADC_data[1] = 153 u8arrayADC_data[2] = 153 u8arrayADC_data[3] = 153 u8arrayADC_data[4] = 153		

5.2.2 Digital Input Output (GPIO)

TC-BMC05				
Type	Black Box			
Requirement Reference				
Pre-condition			**TC-BMC03** **TC-BMC04**	
Post-condition			**TC- BMC06**	
Purpose	Validate the functionality of GPIO module (Hardware and software).			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Press button SW501 in the development card	Development board Leds: LD502 = On LD503 = Off LD504 = Off LD505 = Off		
2	Press button SW501 in the development card	Development board Leds: LD502 = Off LD503 = On LD504 = Off LD505 = Off		

5.2.3 Low Voltage Detection

TC-BMC06				
Type	Black Box			
Requirement Reference				
Pre-condition	1.- SW5 in the chassis of the BMC is in external power supply option. 2.- The BMC Module is turned off 3.- A CAN protocol analyzer tool, is connected to the CAN bus of the BMC module (channel 0).		**TC-BMC02** **TC-BMC04**	
Post-condition			**TC- BMC07**	
Purpose	Validate the functionality of Application software layer.			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Connect J101.24 (ADC channel 0) to a variable voltage power supply (0-12v)			
2	Set Vout of the power supply to 12 v			
3	Turn On the BMC Module			
4	Set Vout of the power supply to 7 v	In the analyzer display: CAN Frame: ID =0x0333 Message = 0x22 None of the other features of the BMC must be available.		

5.2.4 Environment Humidity

TC-BMC07				
Type	Black Box			
Requirement Reference				
Pre-condition	1. SW5 in the chassis of the BMC is in "Sensors" option.		**TC-BMC02**	
Post-condition			**TC- BMC04**	
Purpose	Validate the functionality of humidity sensor			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers	In the watch window: aCAN_Tx_Buffers[0].data[3] = CurenTHumVal In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 00 00		
2	Use a humidity external sensor to get the ambient humidity.	In the watch window: aCAN_Tx_Buffers[0].data[0] = CurenTHumVal In the analyzer display: CAN Frame: • ID =0x0222 Message = 00 00 CurenTHumVal 00 00 00		

5.2.5 Speed Sensor

TC-BMC08				
Type	Black Box			
Requirement Reference				
Pre-condition	1. SW5 in the chassis of the BMC is in "Sensors" option.		**TC-BMC02**	
Post-condition			**TC- BMC04**	
Purpose	Validate the functionality of speed sensor			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers	In the watch window: aCAN_Tx_Buffers[1].data[3] = 0 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 00 00		
2	Set the speed of the car to 20 km/H	In the watch window: aCAN_Tx_Buffers[0].data[0] = 20 In the analyzer display: CAN Frame: • ID =0x0222 Message = 00 14 00 00 00 00		
3	Set the speed of the car to 80 km/H	In the watch window: aCAN_Tx_Buffers[0].data[0] = 80 In the analyzer display: CAN Frame: • ID =0x0222 Message = 00 50 00 00 00 00		
4	Set the speed of the car to 120 km/H	In the watch window: aCAN_Tx_Buffers[0].data[0] = 120 In the analyzer display: CAN Frame: • ID =0x0222 Message = 00 78 00 00 00 00		

5.2.6 Fuel Level Sensor

TC-BMC09				
Type	Black Box			
Requirement Reference				
Pre-condition	1. Fuel tank level it's empty 2. SW5 in the chassis of the BMC is in "Sensors" option.		**TC-BMC04**	
Post-condition			**TC- BMC04**	
Purpose	Validate the functionality of fuel level sensor			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers	In the watch window: aCAN_Tx_Buffers[0].data[2] = 0 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 00 00		
2	Set the level of the fuel tank to 25%	In the watch window: aCAN_Tx_Buffers[0].data[2] = 25 In the analyzer display: CAN Frame: • ID =0x0222 • Message = = 0x00 00 19 00 00 00		
3	Set the level of the fuel tank to 50%	In the watch window: aCAN_Tx_Buffers[0].data[2] = 50 In the analyzer display: CAN Frame: • ID =0x0222 • Message = = 0x00 00 32 00 00 00		
4	Set the level of the fuel tank to 75%	In the watch window: aCAN_Tx_Buffers[0].data[2] = 75 In the analyzer display: CAN Frame:		

		• ID =0x0222 • Message = = 0x00 00 4B 00 00 00		
5	Set the level of the fuel tank to 100%	In the watch window: aCAN_Tx_Buffers[0].data[2] = 100 In the analyzer display: CAN Frame: • ID =0x0222 • Message = = 0x00 00 64 00 00 00		

5.2.7 Proximity Sensor

TC-BMC10				
Type	Black Box			
Requirement Reference				
Pre-condition	1. SW5 in the chassis of the BMC is in "Sensors" option.		**TC-BMC02**	
Post-condition			**TC- BMC04**	
Purpose	Validate the functionality of proximity sensor			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers	In the watch window: aCAN_Tx_Buffers[0].data[5] = 0 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 00 00		
2	Set an object to 2.5 meters of the proximity sensor	In the watch window: aCAN_Tx_Buffers[0].data[5] = 250 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 FA 00		
3	Set an object to 1.5 meters of the proximity sensor	In the watch window: aCAN_Tx_Buffers[0].data[5] = 150 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 96 00		
4	Set an object to 0.5 meters of the proximity sensor	In the watch window: aCAN_Tx_Buffers[0].data[5] = 50 In the analyzer display: CAN Frame: • ID =0x0222		

		• Message = 0x00 00 00 00 32 00		

5.2.8 Temperature Sensor

TC-BMC11			
Type	Black Box		
Requirement Reference			
Pre-condition	1. SW5 in the chassis of the BMC is in "Sensors" option.	**TC-BMC02**	
Post-condition		**TC- BMC04**	
Purpose	Validate the functionality of temperature sensors.		
Test Description			
Action	**Expected response**	**Current response**	**Comments**
Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers	In the watch window: aCAN_Tx_Buffers[0].data[2] = 0 In the analyzer display: CAN Frame: • ID =0x0222 • Message = 0x00 00 00 00 00 00		
Use a thermometer to get the ambient temperature.	In the watch window: aCAN_Tx_Buffers[0].data[0] = CurenTemVal In the analyzer display: CAN Frame: • ID =0x0222 Message = 0xCurenTemVal 00 00 00 00 00		

5.3. Actuators

5.3.1 Pulse Width Modulator (PWM)

TC-BMC03				
Type	Black Box			
Requirement Reference				
Pre-condition	1.- SW4 in the chassis of the BMC is in "No load" option. 2.- Connect oscilloscope probes in the next pins in the development card: • J101.32 – PWM channel 3 • J101.34 – PWM channel 4 • J101.36 – PWM channel 5 • J101.40 – PWM channel 6 • J101.08 – PWM channel 7 3.- Configure all used channels of the oscilloscope to measure duty cycle of the PWM signals.		**TC-BMC01**	
Post-condition			**TC- BMC04**	
Purpose	Validate the functionality of PWM module (Hardware and software).			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name u8arrayPWM_data	In the watch window: u8arrayPWM_data[0] = 0, u8arrayPWM_data[1] = 0, u8arrayPWM_data[2] = 0, u8arrayPWM_data[3] = 0, u8arrayPWM_data[4] = 0, In the oscilloscope display: CH1 dtyCycle = 0% CH2 dtyCycle = 0% CH3 dtyCycle = 0% CH4 dtyCycle = 0% CH5 dtyCycle = 0%		
2	Press button SW501 in the development card during 10 seconds in order to			

	switch execution mode of the BMC from normal operation to self-test operation mode.			
3	Wait 5 seconds	In the watch window: u8arrayPWM_data[0] = 100, u8arrayPWM_data[1] = 100, u8arrayPWM_data[2] = 100, u8arrayPWM_data[3] = 100, u8arrayPWM_data[4] = 100, In the oscilloscope display: CH1 dtyCycle = 39.2% CH2 dtyCycle = 39.2% CH3 dtyCycle = 39.2% CH4 dtyCycle = 39.2% CH5 dtyCycle = 39.2%		
4	Wait 5 seconds	In the watch window: u8arrayPWM_data[0] = 150, u8arrayPWM_data[1] = 150, u8arrayPWM_data[2] = 150, u8arrayPWM_data[3] = 150, u8arrayPWM_data[4] = 150, In the oscilloscope display: CH1 dtyCycle = 58.82% CH2 dtyCycle = 58.82% CH3 dtyCycle = 58.82% CH4 dtyCycle = 58.82% CH5 dtyCycle = 58.82%		
5	Wait 5 seconds	In the watch window: u8arrayPWM_data[0] = 200, u8arrayPWM_data[1] = 200, u8arrayPWM_data[2] = 200, u8arrayPWM_data[3] = 200, u8arrayPWM_data[4] = 200, In the oscilloscope display: CH1 dtyCycle = 78.43% CH2 dtyCycle = 78.43% CH3 dtyCycle = 78.43% CH4 dtyCycle = 78.43% CH5 dtyCycle = 78.43%		
6	Wait 5 seconds	In the watch window: u8arrayPWM_data[0] = 255, u8arrayPWM_data[1] = 255, u8arrayPWM_data[2] = 255,		

		u8arrayPWM_data[3] = 255, u8arrayPWM_data[4] = 255, In the oscilloscope display: CH1 dtyCycle = 100% CH2 dtyCycle = 100% CH3 dtyCycle = 100% CH4 dtyCycle = 100% CH5 dtyCycle = 100%		

5.3.2 Power Light Controller

TC-BMC12				
Type	Black Box			
Requirement Reference				
Pre-condition	1. SW4 in the chassis of the BMC is in "Full load" option.		**TC-BMC03** **TC-BMC04**	
Post-condition				
Purpose	Validate the functionality of Power Relay controller (Hardware and software).			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x01	BMC Lights: Optical Horn = Off Low Beam = Off High Beam = Off Turn Right = Off Turn Left = On		
2	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x02	BMC Lights: Optical Horn = Off Low Beam = Off High Beam = Off Turn Right = On Turn Left = Off		
3	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x03	BMC Lights: Optical Horn = Off Low Beam = Off High Beam = On Turn Right = On Turn Left = Off		

4	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x04	BMC Lights: Optical Horn = Off Low Beam = Off High Beam = On Turn Right = Off Turn Left = Off		
5	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x05	BMC Lights: Optical Horn = Off Low Beam = On High Beam = Off Turn Right = Off Turn Left = On		
6	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x06	BMC Lights: Optical Horn = Off Low Beam = On High Beam = Off Turn Right = On Turn Left = Off		
7	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x09	BMC Lights: Optical Horn = Off Low Beam = On High Beam = Off Turn Right = Off Turn Left = On		
8	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x10	BMC Lights: Optical Horn = On Low Beam = Off High Beam = Off Turn Right = Off Turn Left = Off		
9	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x11	BMC Lights: Optical Horn = On Low Beam = Off High Beam = Off Turn Right = Off Turn Left = On		
10	Using the protocol analyzer tool, the frame ID =0x0100 Message = 0x12	BMC Lights: Optical Horn = On Low Beam = Off High Beam = Off Turn Right = On Turn Left = Off		

5.4. Communications

5.4.1 Control Area Network (CAN)

TC-BMC04				
Type	Black Box			
Requirement Reference				
Pre-condition	1.- Connect a CAN protocol analyzer tool, to the CAN bus of the BMC module (channel 0).		**TC-BMC02**	
Post-condition			**TC- BMC05**	
Purpose	Validate the functionality of CAN module (Hardware and software).			
Test Description				
Step	**Action**	**Expected response**	**Current response**	**Comments**
1	Add to the watch window of the debugger tool the variable name aCAN_Rx_Buffers	In the watch window: aCAN_Rx_Buffers [0] .ID = 0x0000 0000 aCAN_Rx_Buffers [0] .data[0]= 0x00 aCAN_Rx_Buffers [0] .data[1]= 0x00 aCAN_Rx_Buffers [0] .data[2]= 0x00 aCAN_Rx_Buffers [0] .data[3]= 0x00 aCAN_Rx_Buffers [0] .data[4]= 0x00 aCAN_Rx_Buffers [0] .data[5]= 0x00 aCAN_Rx_Buffers [0] .data[6]= 0x00 aCAN_Rx_Buffers [0] .data[7]= 0x00 In the analyzer display: CAN Frame: ID = 0x0000 Message = 0x00 00 00 00 00 00 00 00		

2	Using the protocol analyzer tool, the frame ID = 0x0100 Message = 0x00 11 22 33 44 55 66 77	In the watch window: aCAN_Rx_Buffers [0] .ID = 0x0000 0100 aCAN_Rx_Buffers [0] .data[0]= 0x00 aCAN_Rx_Buffers [0] .data[1]= 0x11 aCAN_Rx_Buffers [0] .data[2]= 0x22 aCAN_Rx_Buffers [0] .data[3]= 0x33 aCAN_Rx_Buffers [0] .data[4]= 0x44 aCAN_Rx_Buffers [0] .data[5]= 0x55 aCAN_Rx_Buffers [0] .data[6]= 0x66 aCAN_Rx_Buffers [0] .data[7]= 0x77 In the analyzer display: CAN Frame: ID =0x0100 Message = = 0x00 11 22 33 44 55 66 77		
3	Add to the watch window of the debugger tool the variable name aCAN_Tx_Buffers			
4	Using the protocol analyzer tool, send the frame ID = 0x0244 (CAN TX test command of the BMC)	In the watch window: aCAN_Tx_Buffers [2] .ID = 0x0000 0100 aCAN_Tx_Buffers [2] .data[0]= 0x00 aCAN_Tx_Buffers [2] .data[1]= 0x11 aCAN_Tx_Buffers [2] .data[2]= 0x22 aCAN_Tx_Buffers [2] .data[3]= 0x33		

		aCAN_Tx_Buffers [2] .data[4]= 0x44 aCAN_Tx_Buffers [2] .data[5]= 0x55 aCAN_Tx_Buffers [2] .data[6]= 0x66 aCAN_Rx_Buffers [2] .data[7]= 0x77 In the analyzer display: CAN Frame: ID =0x0644 Message = = 0x00 11 22 33 44 55 66 77		

6. Conclusions

This project has provided an example of effective use of design methodology throughout the development of a body controller module. In the architecture selection phase, a simple approach was implemented in order to cover all the project requirements with a low effort of design and codification. From the software development point of view, this approach allows adopt some of the advantages that standardized automotive software architecture brings, but is not restricted to be full complaint with it. The selection of the software architecture was based in AUTOSAR according to characteristics and working principle of BCM, analyzed the external interface and system architecture, and designed a common software structure and basic module that has been applied successfully in the software design of a BCM for one car model.

With the software architecture selected, the focus then shifted to development and validation of software and hardware interfaces. The control system architecture was used to systematically outline the specific requirements of the control strategy for the body controller module. Control software was then developed using C programming language, using modular software architecture facilitate team work and code reusability. To validate and improve the functionality of the BCM a simple prototype box was then developed. This simple prototype allowed implement self-test routines by software thus decreasing the time of validation for product functionality.

One of the main goals of this project was develop a BMC that achieve a steady performance in a real vehicle test, with practical value and significance, although this has not been possible to carry out due to lack of resources. Future work that could build off this thesis could include the implementation of this BMC and collecting real world data to compare results against what was expected to be.

7. Bibliography

AUTOSAR. (2013). Automotive Open System Architecture. Retrieved from http://www.autosar.org/

Continental AG. (2013). Body Control Modules – Hidden But Essential For Every Car. Retrieved from

http://www.conti-online.com/www/automotive_de_en/themes/passenger_cars/interior/body_security/body_control_units_en.html

Freescale Semiconductor. (2012). MC9S12XEP100 Reference Manual. *Covers MC9S12XE Family*. Freescale Semiconductor. Retrieved from http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12XEP100RMV1.pdf

Freescale Semiconductor. (2013). DEMO9S12XEP100: Demo Board for the 16-bit MC9S12XE and XS-Families. Retrieved from http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DEMO9S12XEP100

Freescale Semiconductor. (2013, February). KIT eXtreme Switch Evaluation Board. *KTXSWITCHG3UG*. Retrieved from http://www.freescale.com/files/analog/doc/user_guide/KTXSWITCHG3UG.pdf

Honeywell .Inc. (2005, March). HALL EFFECT SENSING AND APPLICATION. *MICRO SWITCH Sensing and Control*, p. 8. Retrieved from http://sensing.honeywell.com/index.php?ci_id=47847

MaxBotix. (2012). Ultrasonic Sensors by MaxBotix® Inc. *Ultrasonic Sensor Component Modules for OEMs, Engineers, & More*. Retrieved from http://www.maxbotix.com/

Mitidieri, A. (2008, December). “Cooperate on standards, compete on implementation.”. *AUTOSAR – Automotive Open System Architecture*, p. 16. Retrieved from http://www.automotive-spin.it/uploads/4/mitidieri-4W.pdf

Sensoronix. (2012). Hall Effect Zero Speed Sensors. Retrieved from http://www.sensoronix.com/

SparkFun Electronics. (2013). Humidity Sensor. *HIH-4030 Breakout*. Retrieved from https://www.sparkfun.com/products/9569