INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE TOULOUSE

_____

Detailed specifications of automotive application

APPROVALS			
Function	**Name(s)**	**Date**	**Signature**
Author(s)	**Antoine Dubois**	**01/02/2015**	
Reviewer(s)	**Patrick Tounsi**		

Glossary:

CAN	Controller Area Network
BCM	Body Control Module
LCM	Light Control Module
SPI	Serial Peripheral Interface
HCS	Headlights Control System
SBC	System Basis Chip

Description:

This document is the specifications report of class 2015 project, for the automotive electronics class. It comes with the following examples:

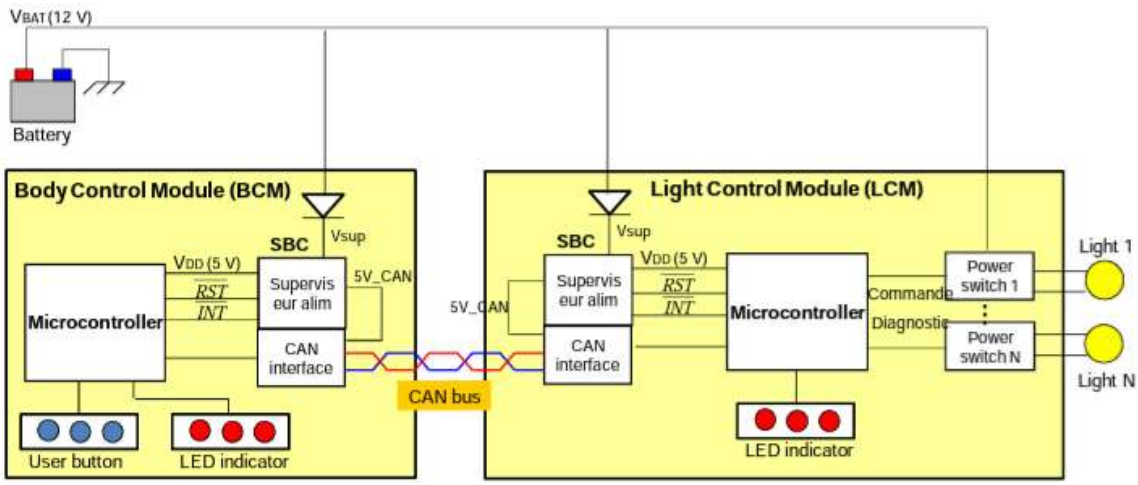
- Visio original documents (algorigram)
- Excel specification documents
- Software source code and project for general application
- Software source code to demonstrate low power and clock management
- Software source code to demonstrate SBC advanced application

Summary

I. Presentation of the application

1. Global Schematic

The application developed in class is composed of two modules (BCM and LCM) which communicate through CAN protocol. They are both powered by a battery or the alternator (Power supply from 8 to 16V typical). Any other module can be added at the CAN bus in the future application.



II. Detailed specifications

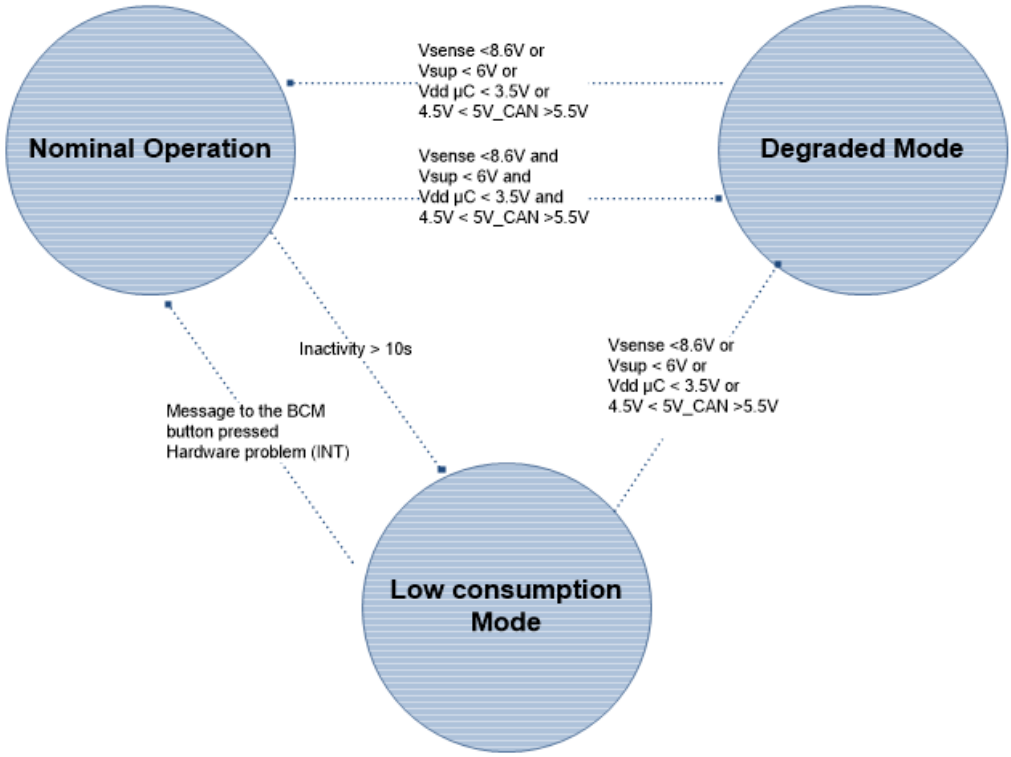
1. List of requirements

Requirements list			
Section	**Requirements**	**Description**	**Comments**
Diagnostic	Pass in degraded mode if the battery voltage, **Vsense < 8,6V**	The microcontroller checks the battery voltage value (Vsense). If **Vsense** voltage becomes lower than 8.6 V, the SBC sends a pulse on the INT line and notifies the risk of low battery (light indicator). The BCM and the LCM then pass in degraded mode.	Realized with SBC, not tested
Diagnostic	Pass in degraded mode if the battery voltage, **Vsup < 6V**	if the **Vsup** becomes lower than 6 V, the SBC sends a pulse on the INT line and notifies the risk of low battery (light indicator). The BCM and the LCM then pass in degraded mode.	Realized with SBC, not tested
Diagnostic	Disable the SBC if the voltage **Vsup < 4V**	If the voltage **Vsup** is lower than 4V, the Vdd voltage supplied by the SBC is disabled.	Realized with SBC, not tested
Diagnostic	Detect low voltage supply Vdd: **3.5V < Vdd < 4.5V**	If the supply voltage **Vdd** of the microcontroller is between 3.5V and 4.5 V, the SBC sends a pulse on the INT line. The microcontroller notify (or not) the risk of low battery voltage by a light indicator. The BCM and the LCM microcontrollers pass in degraded mode.	Realized with SBC, not tested
Diagnostic	Detect low voltage supply **Vdd < 3.5V**	If the supply voltage **Vdd** of the microcontroller becomes lower than 3.5 V, the BCM and the LCM microcontrollers will be reset.	Realized with SBC, not tested
Diagnostic	SBC fault error detection **5V_CAN**	If a problem is detected on the supply voltage **5V_CAN** of the CAN interface (overcurrent, under voltage, over temperature), this last one goes into Sleep mode until the problem disappears.	Realized with SBC, not tested
low consumption	Enter low consumption mode if **inactivity > 10s**	STOP0 mode: the BCM and the LCM enter in this mode if the period of inactivity exceeds 10s. THEY check before if the can is in sleep mode and disable.	Timer PIT1 count inactivity time Go to Run 2 and then low power mode

low consumption	Exit low consumption mode	The exit of the low consumption mode is done by: -a microcontroller request - a CAN massage - a Low light - a hardware problem.	Interrupt from the SBC wake up the microcontroller mode in Low power Mode. For BCM, periodic timer wake the microcontroller to check luminosity and buttons.
CEM	Reduce electromagnetic emissions	The FM modulation of the PLL of each microcontroller is enabled in order to avoid a synchronization problem. The CAN interfaces outputs (CAN_H and CAN_L) and the power outputs are configured in **slow slew rate**.	PLL active, tested
Debit	Regulate the CAN bit rate at **1 Mbit/s** in normal mode	The CAN bus operates with the CAN 2.0B specification. The bit rate must be greater than 1 Mbits/s in nominal operating mode.	Tested
Debit	Regulate the CAN bit rate at **125 Kbit/s** in degraded mode	The CAN bus operates with the CAN 2.0B specification. The bit rate will be reduced to 125 Kbit/s in degraded mode.	Not implemented. Freescale expert advice was to stay at 1Mbit/s. No reason to switch at 125kHz. More consumption and less efficient.
Piloting Command	Turn on/off the headlights	When the user presses the headlights turn on/off button, the headlights must be turned on or turned off.	Tested
Piloting Command	Turn on/off the left indicator	When the left indicator button is pressed, it must be turned on or turned off.	Tested
Piloting Command	Turn on/off the right indicator	When the right indicator button is pressed, it must be turned on or turned off.	Tested
Diagnostic	Request a diagnostic of the driver	The BCM demand the average current consumed and the switches command state.	The average current is periodically send to the BCM, and fault will be send to. Not tested.
low consumption	Enter the CAN transceiver in listen only mode	If the CAN controller transmits nothing for 2s, it must pass in listen only mode.The microcontroller ensure that the CAN interface is in sleep mode and the CAN controller is disable	The SBC go to Listen Only. The microcontroller turns off the CAN peripherals to reduce Energy consumption.

low consumption	Enter the CAN transceiver in disable mode	If the CAN controller transmits nothing for 10s, it must pass in disable. Before entering the STOP0 mode, the microcontroller ensure that the CAN interface is in sleep mode and the CAN controller is disable	The SBC go to Listen Only. The microcontroller turns off the CAN peripherals to reduce Energy consumption.
Safety	Detect CAN transceiver bus OFF mode	The CAN transceiver enters in this mode when a CAN problem is detected.	Interrupt happens when bus Off detected. No treatment. Not tested
Communication	Identify the Bus off state inputs/outputs	In the case of errors in transmission or reception, only the inputs and outputs of the Bus Off state are identified.	Requirement needs more precision
Communication	Display transmission/reception errors after **5 consecutive Bus Off state**	After 5 consecutive entries in the Bus Off state, an error is displed on the dashboard and the current request (demand) is aborted.	Not implemented
Communication	Exit of the Bus Off state for the BCM	The exit of the Bus Off state is done by a request from the controller	Not implemented
Communication	Exit of the Bus Off state for the LCM	For LCM, the exit of the Bus Off state is automatic. According to CAN 2.08 protocol specification: 128 occurrences of 29 recessive bits.	Automatic
Piloting Command	Receive information form BCM	Reception of frames from the BCM: Reception the information delivered by the light sensor.	Tested
Safety	Manage over current problem > **10ms** **20A < courrent < 75A**	The detection of an overcurrent is determined by a lower threshold of **20A** and a high threshold of **75A**. When the current exceeds the lower threshold for longer than **10ms**, the corresponding power output is inhibited.	Implemented but not working. Use the SPI interruption default pin.
Safety	Manage switches problem	If a problem is reported on a switch, an indicator light indicates the nature of the problem and the control of the switch is off until the problem disappears. When the problem disappears, the DCM or LCM sends a message to BCM.	Implemented but not working. Use the SPI interruption default pin.
Diagnostic	Check switches states	The BCM can request a diagnostic of the power switches controlled by the DCM.	Implemented but not working. Use the SPI interruption default pin.

2. BCM specifications



Nominal operation

This is the operation mode in which the BCM switches if there is no anomaly. In this mode the BCM works with a 64 MHz clock frequency provided by an FM-PLL module from an 8 MHz external oscillator. BCM in this mode can transmit and / or receive frames. The frames must be sent with a rate of 1M bit / sec. The following tasks must be carried out when the BCM enter in this mode:

- Measure the current
- Detect power supply fault
- Read the CAN messages
- Communicate with the headlights
- Check the switch states

Fail-soft mode (or degraded mode)

BCM enters this mode when problems are encountered on the supply voltage. The following tasks must be performed when the microcontroller enters in this mode.

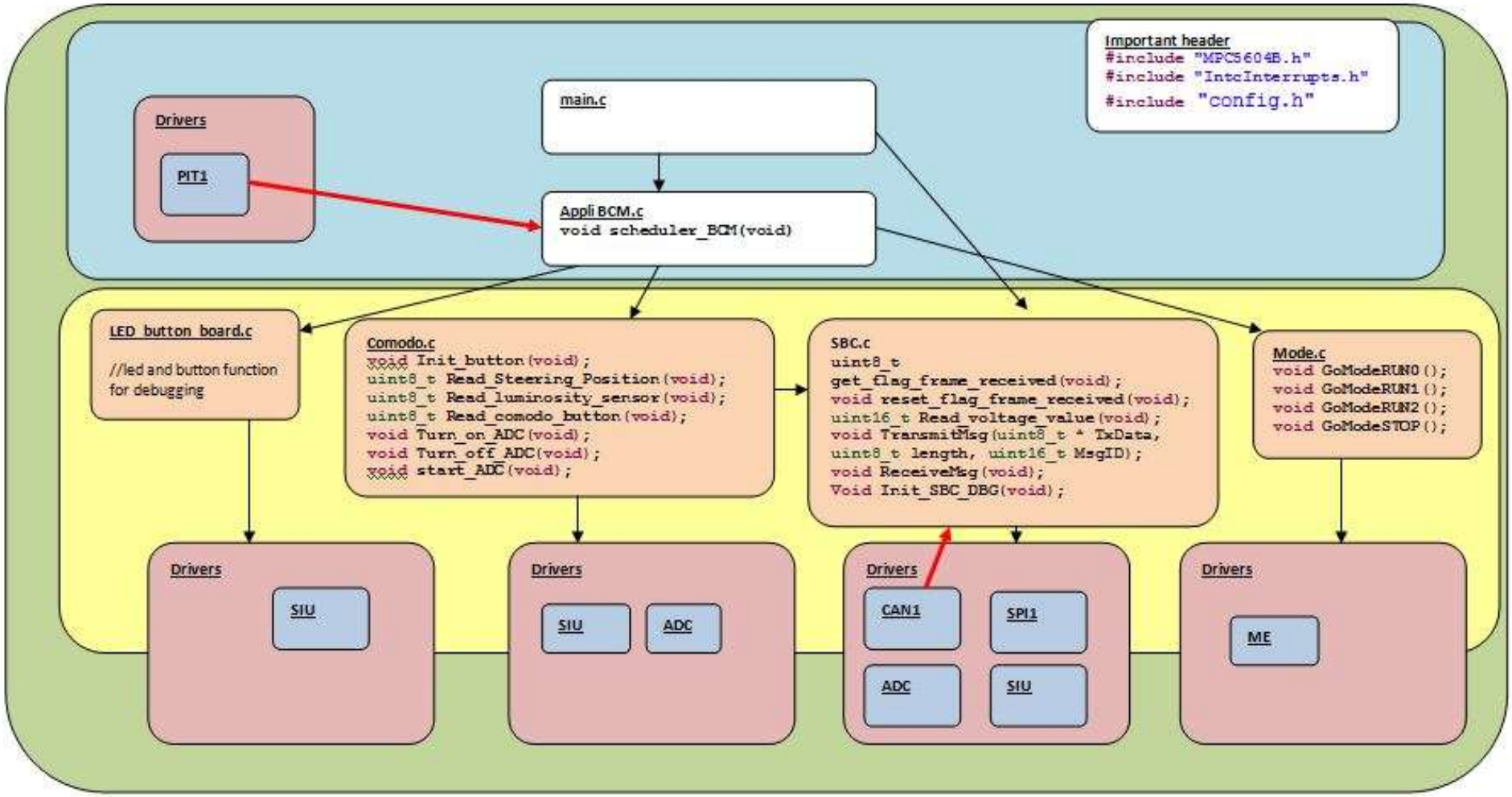
- Measure the current
- Measure voltage to switch to leave failsoft mode
- Read the CAN messages

- Communicate with the headlights
- Check the switch states
- Disable the PLL
- Reduce the speed to 125 Kbit / s
- Put the Clock external
- Light indication

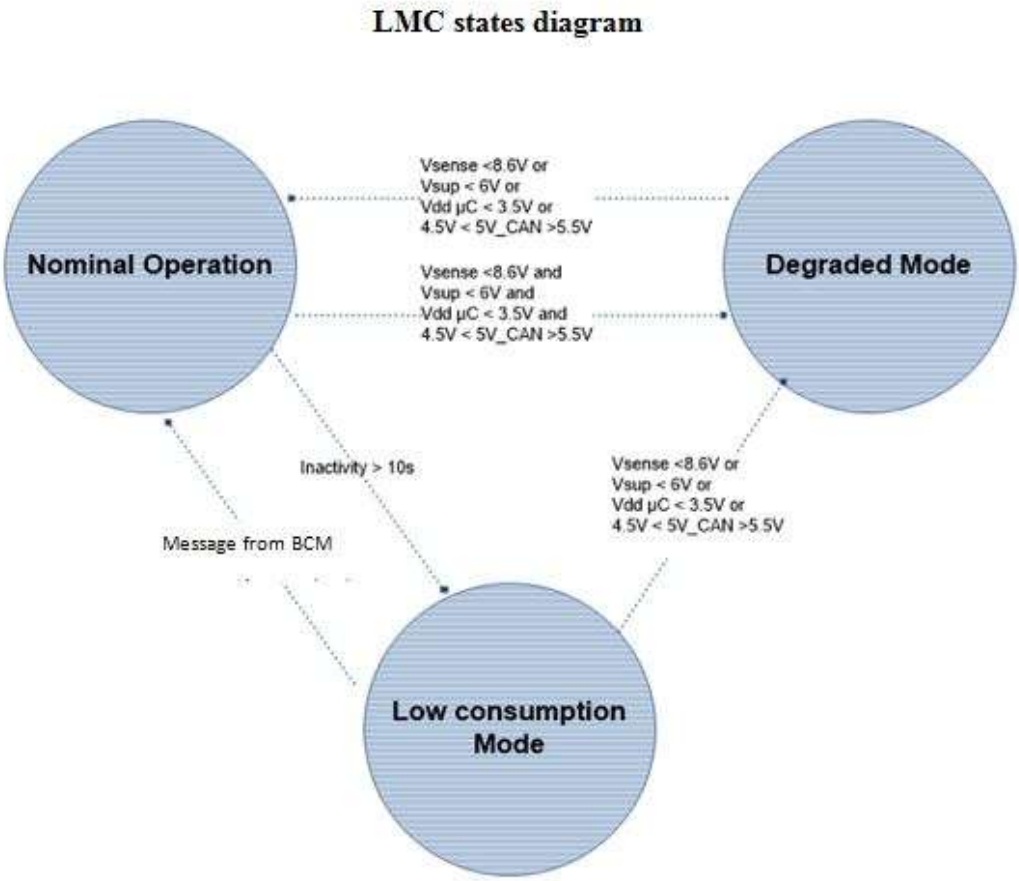
Low consumption mode

The microcontroller enters in this mode after 10s of inactivity. In this mode the BCM change the source of its oscillations, and go to another RUN mode (RUN2). It periodically wakes up to check button and ADC value, if the state stays the same the microcontroller go back to sleep mode and wait the next PIT period to wake up again.

Project structure

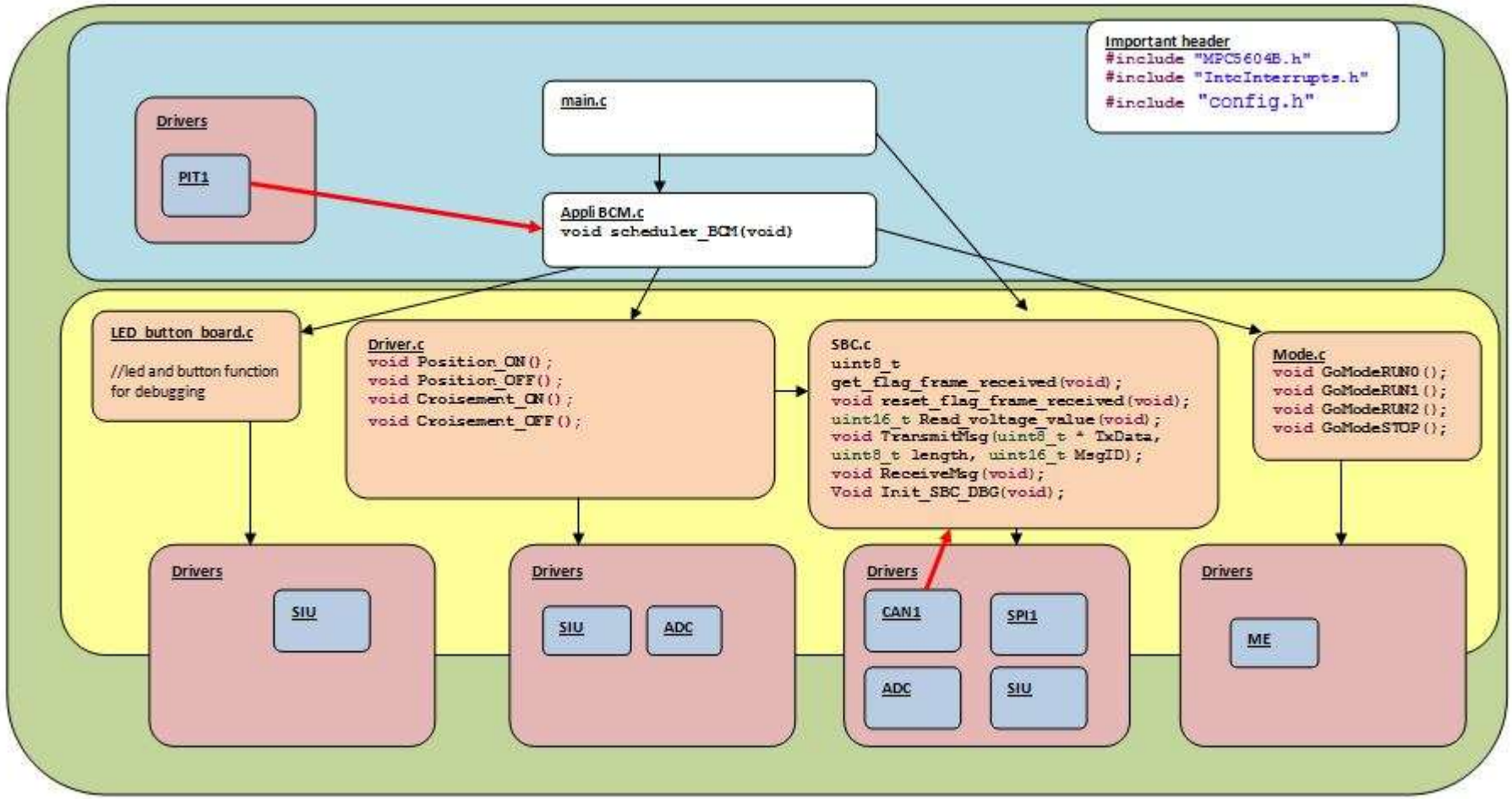


2.1. LCM details specifications



The normal and degraded modes are similar to the BCM, except the LCM received BCM light command and change the lights value.

Low consumption mode
The microcontroller enters in this mode after 10s of inactivity (only if the lights are off). In this mode the CAN flex is turned off and the only way to wake up the LCM, it is to send it a CAN frame. The SBC will send a wake up command on the interrupt pin.


Important header
#include "MPC5604B.h"
#include "IntcInterrupts.h"
#include "config.h"
Drivers
PIT1
main.c
Appli BCM.c
void scheduler_BCM(void)
LED_button_board.c
//led and button function for debugging
Driver.c
void Position_ON();
void Position_OFF();
void Croisement_ON();
void Croisement_OFF();
SBC.c
uint8_t
get_flag_frame_received(void);
void reset_flag_frame_received(void);
uint16_t Read_voltage_value(void);
void TransmitMsg(uint8_t * TxData,
uint8_t length, uint16_t MsgID);
void ReceiveMsg(void);
Void Init_SBC_DBG(void);
Mode.c
void GoModeRUN0();
void GoModeRUN1();
void GoModeRUN2();
void GoModeSTOP();
Drivers
SIU
Drivers
SIU
ADC
Drivers
CAN1
SPI1
ADC
SIU
Drivers
ME

III. Realization details:

1. Mode and clock source

To respect the state diagram below, the source code uses three different RUN mode.

RUN0: External crystal 8MHz with PLL and FM enable 64MHz
It is the normal mode: we do not try to optimize consumption. The FM modulation is enable and can be observed and the spectrum analyzer. Electromagnetic emissions are reduced.

RUN1: Failsoft mode: we disable the PLL, the main clock only come from the external oscillator so the main clock frequency is decreased by 8.

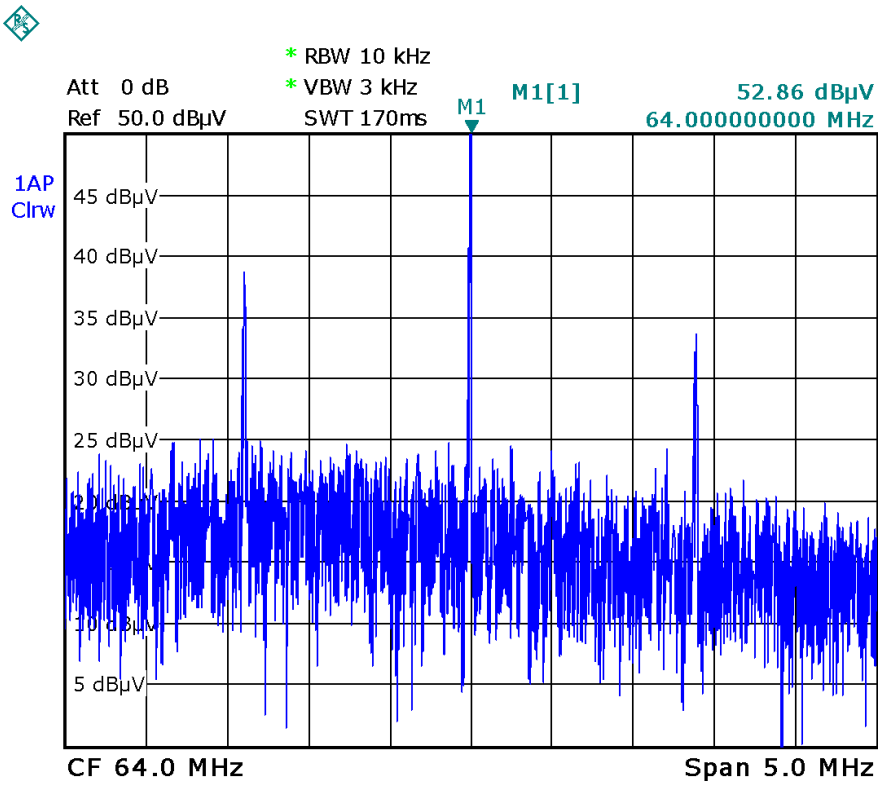
RUN2: Wake up mode: in this mode the microcontroller is in low consumption state, it just woke up from the sleep mode and check the state of the buttons and go back to sleep mode if nothing happens. It is only defined for the BCM.
Clock source come from low RC oscillator.

Sleep Mode: only the PIT1 is enable, and some external interrupt. It waits an interrupt to wake up and go to RUN 2. The clock source comes from the RC oscillator.

2. PLL modulation

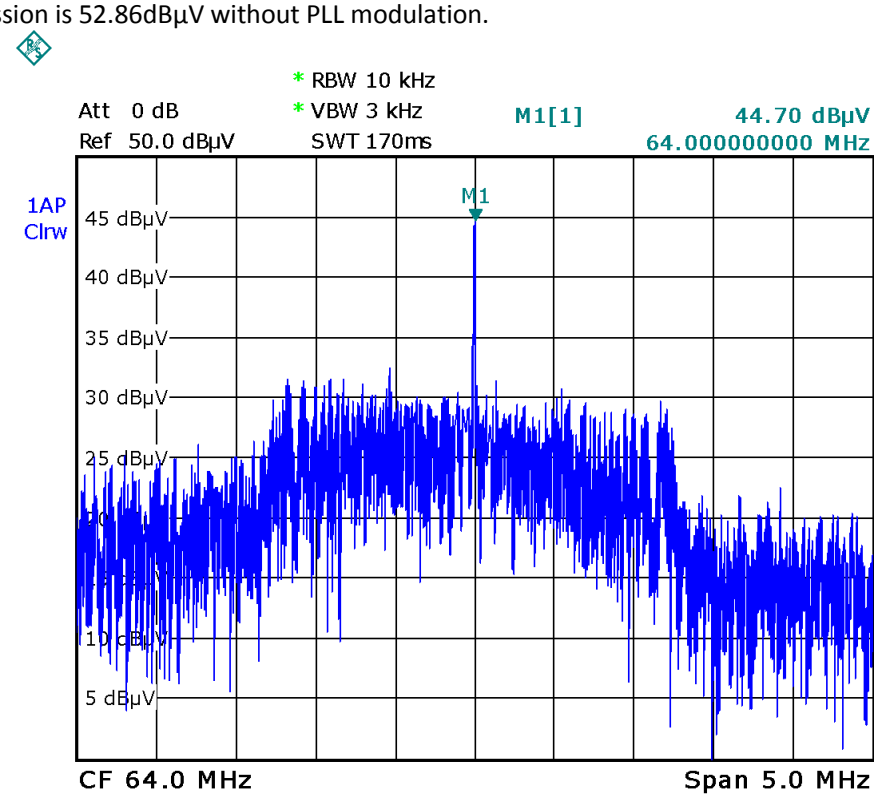
In order to reduce electromagnetic emission, the frequency modulation is active on the MCP5604B. Some tests were realized to measure the electromagnetic spectrum.

- Vertical scale in dBuV, 30dB pre-amplification with a 9kHz to 2Ghz bandwidth..
- Central frequency = 64Mhz (PLL frequency)
- Bandwidth resolution : 10kHz
- SPAN = 5MHz (sampling frequency) (64+/-2.5MHz)

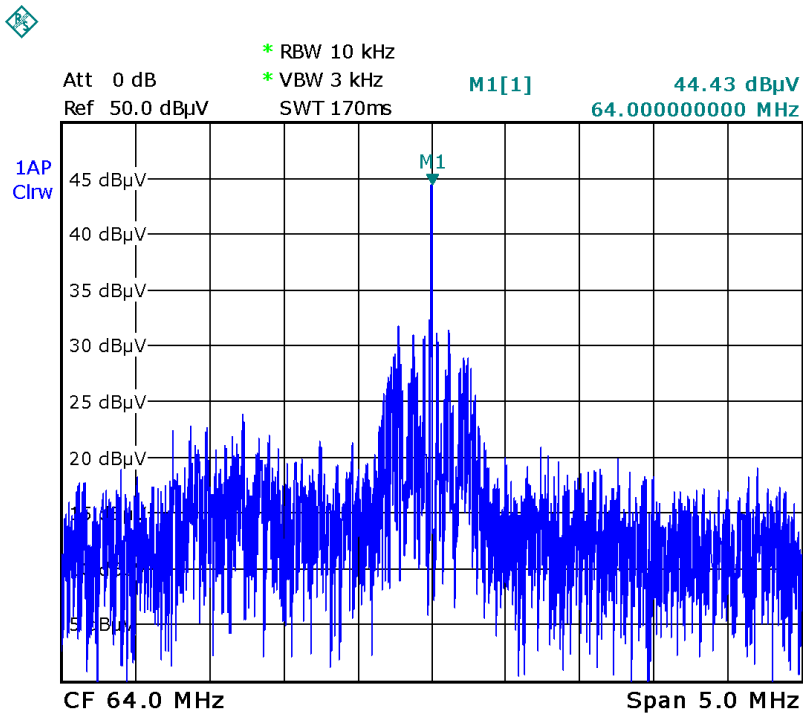


MODULATION OFF

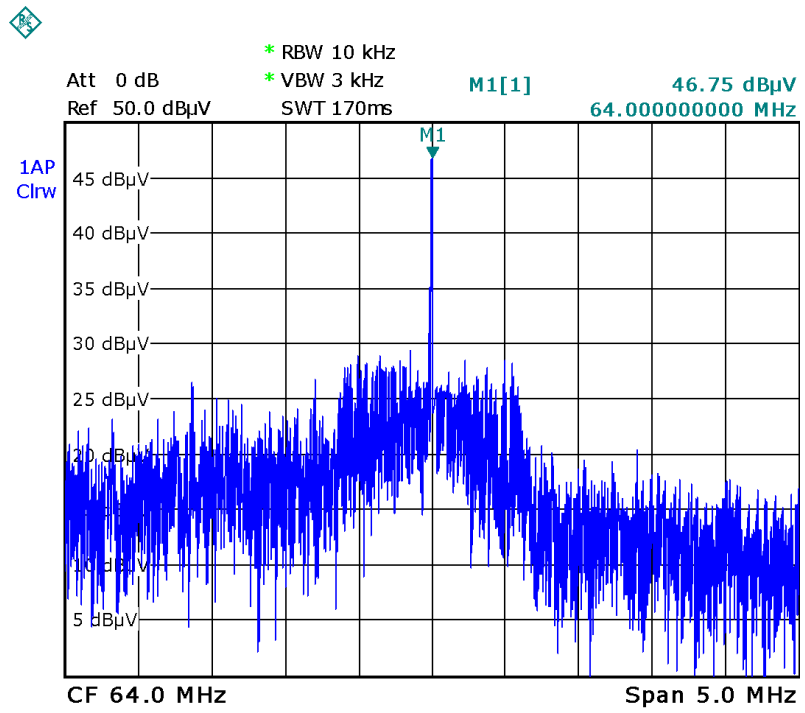
The 64MHz emission is 52.86dBµV without PLL modulation.



MODULATION ON: fmod = 100k, mod depth = 2%



MODULATION ON: fmod = 25kHz, mod depth = 2%



MODULATION ON: fmod = 100kHz, mod depth = 1%

PLL modulation reduces spectral density. The power stays the same but we reduce the emission at 64MHz and increase the range of frequency emission.

3. SBC Management

The SBC is not just a transceiver. We are using it to respect some requirement.

CAN transceiver	The SBC is the CAN transceiver of both applications.	Implemented and tested
Debug mode	The SBC can be configured with or without the Debug diode. Some extra commands SPI commands has to be sent by the microcontroller periodically. See debug mode in datasheet	Implemented in a special demo source code, and tested
Voltage supply fault detection	SBC sends an interrupt when the voltage is low (Vsup, 5V CAN or another voltage). To leave failsoft mode the voltage on the Mux resistor output is periodiccaly converted to verify if the power supply came back to its normal value	Not implemented
Wake up from low power mode	When the module is in Low power, the SBC sends an interrupt to wake the microcontroller, when it received a CAN frame.	Implemented 50%

4. Driver management

The KIT33981BPNAEVB includes the MC33984 power driver and allows us to drive two loads of high power HS0 and HS1 (>30A).

It has to do the following task.

- Turning on and off the loads
- Giving the fault value when the fault is detected (over current, open load).
- Represent the current value

It communicates through SPI protocol and General Input/Output.

The settings with SPI commands do not work well for some commands, so the settings of fault detection and fault detection is not done yet.

5. CAN

- The CAN module send and receive correctly frame at 1Mbit/s.

- However, when we go to RUN 1 and RUN2, the clock prescaler has to be modified to respect frequency variation and requirements.

- Also, the interrupt on CAN reception does not work, so a polling is currently implemented on the main.

IV. Feedback:

6. Team organization

We first split the team in two parts the System engineers and the software engineers. The first team was responsible of studying the requirements and the feedback of other group, and develops the application. The software engineering team was configuring the peripherals, and implemented the software. This is sum up in the following document:

	Séance 1		Séance 2		Séance 3		Séance 4	Bilan mis-séance, rendu par groupe	Séance 5		Séance 6		Séance 7		Séance 8		Séance 9	Rendu final	Séance 10		
Personnes																					Personnes
Romain	Spécifications			Rendu liste spec, algorigramme	Perfectionnement des spec			Rendu du documents de spécifications complet, prêt pour le codage, et renvoi au client pour validation	Confic wdt, LP et interrupt du SBC									rendu code à reprendre SBC	Video et demo		Romain
Chaymaa									Codage timer LCM		document pin def + code test		Debug LCM bouton et trame				rendu spécifications	Rapport spécification V1			Chaymaa
Ousmane									Realisation document structure code		Code ADC capteur lumineux				rendu spécifications						Ousmane
Asmaa									Codage timer BCM		document pin def + code test		Debug LCM bouton et trame				rendu spécifications				Asmaa
Abd.	Spécifications		Installation CW, SPI	Programmation MC33984 (travail avec marine et Jad pour communiquer avec switch SPI et PWM)				Démo Switch	MC33984 (SPI, PWM, GPIO, config registre)						Debug et demo LCM			rendu code demo main appli			Abd.
Camille	Installation CW		Mode Low Power	Programmation demo Reset, Reveil, Low Power				Programme de démo et explication	Fichier Mode.c		PLL reglage et mesure		Code ADC capteur lumineux		ADC pour SBC		Debug passage mode LP	rendu code demo LP			Camille
Clément													Debug passage mode LP								Clément

<table>
<tr><td>Thomas</td><td rowspan="3"></td><td>CAN</td><td>Programmation CAN Demo, deux messages différents envoi et reception</td><td>Programme de demo et explication</td><td>CAN et SBC Config avec Mounier</td><td>LCD etude faisabilité</td><td colspan="2" rowspan="2">Debug BCM bouton et trame</td><td rowspan="3">rendu code demo main appli</td><td rowspan="5"></td><td>Thomas</td></tr>
<tr><td>Hajar</td><td>SPI (avec Abd)</td><td>Cablage BCM et Interraction avec Larbi pour avoir une plateforme fonctionnel</td><td></td><td rowspan="2">Connexion bonton, fichier comodo.c</td><td rowspan="2">ADC volant</td><td>Hajar</td></tr>
<tr><td>Larbi</td><td>GPIO</td><td>Programmation programme de démo, détection de bouton, entré dans le mode interruption, allumage de voyant lumineux</td><td>Programme de démo et explication</td><td>document connectique et config Driver</td><td>Debug et demo BCM-LCM</td><td>Larbi</td></tr>
<tr><td>Marine</td><td colspan="2" rowspan="2">Prise en main switch</td><td rowspan="2">Cablage et programme demo switch (ce qui a était fait l'année précedente + SPI)</td><td rowspan="2">Démo Switch</td><td colspan="2" rowspan="2">MC33984 (SPI, PWM, GPIO, config registre)</td><td>MC33984 SPI</td><td>Debug et demo LCM</td><td rowspan="2">rendu code demo LCM</td><td>Marine</td></tr>
<tr><td>Jad</td><td>Code ADC volant</td><td>Debug et demo LCM</td><td>Jad</td></tr>
</table>

7. For 2016 ESE group

Advice to start the project:

- Split the team in system (4 or 5 organized students) and software engineers (the rest). Every software engineers had to be specialized in one section.
- The main source code demonstrates the Normal mode. Both LCM and BCM are in the same project, you have to change one define in config.h.
- The normal mode works well, but the mode changes are commented. It should work, but because of time constraint the test was not complete 100%
- The CAN clock period and the Timer period has to be changed if we change mode, to cope with frequency variation.
- The SBC part is critical, you should start as soon as possible, because it might be complex, and use M. Mounier documents and experience. Some work can be inspired with our demo program even if it was not complete.
- SPI commands does not work, it is probably a configuration problem.

Ask M. Tounsi about our email address, we might remember a little what we did ! (adubois.insa.toulouse@gmail.com for example).

Annexe 1 : La liste des correspondances matériels /logiciels:

LCM:

SPI (SPI0)

Commande	PINs
PWM	A2,A3,A10,A11
CLK	A14
CS0	A15
S0	A12
SI	A13
RST	A5,A4
Wake	A6,A0
FS	A7,A9
Internal Counter	A8 //à vérifier
CS1	PB12

BCM:

Commande	PINs
Veilleuse	A6
Code	A7
Phare	A8
Clignotant gauche	A3
Clignotant droit	A14

SBC:

Commande	PINs
MOSI	PH0
MISO	PH1
SCLK	PH2
~CS	PH3
MUX-out	PD15
~INT	PA1