Title: LIN Driver
SW Component LIN_EA V1.10

History

Issue status (Index)	Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)	Author Department	Check/Release Department	Description
1.0	Draft 27-Oct-15	Edgar Escayola	Adrián Zacarías	Creation of the document
1.1	Draft 28-Oct-15	Edgar Escayola	Adrián Zacarías	Changes at: Purpose, References, and realization constraints and targets.
1.2	Draft 29-Oct-15	Adrián Zacarías	Edgar Escayola	Addition of abbreviations and definitions, diagrams and descriptions.
1.3	Draft 30-Oct-15	Adrián Zacarías	Edgar Escayola	Class diagram and functions definitions.
1.4	Draft 30-Oct-15	Edgar Escayola	Adrián Zacarías	Implementation of traceability in the document.
1.5	Release 1-Dec-15	Edgar Escayola	Adrián Zacarías	Changes from the review implemented.
1.6	Draft 2-Dec-15	Edgar Escayola	Adrián Zacarías	Implementation of the comments from the coach: Architecture diagram, use case diagram, sequence diagram, functional decomposition and definitions of new functions.
1.7	Draft 6-Dec-15	Adrián Zacarías	Edgar Escayola	Functional decomposition and sequence diagram updated. Addition of the following functions: Header_handler, Data_reception, Led_OFF_State, Led_ON_State, Led_TOGGLING_State, Slave_FALSE_State and Slave_TRUE_State. LIN transceiver, master's features and error handling added.
1.8	Draft 7-Dec-15	Edgar Escayola	Adrián Zacarías	Traceability updated and frequency of task0 was changed to 10ms.
1.9	Draft 7-Dec-15	Adrián Zacarías	Edgar Escayola	Addition of circuit used for the transceiver.
1.10	Release 7-Dec-15	Edgar Escayola	Adrián Zacarías	Remarks from the review implemented. Additional information to the circuit design of the transceiver and versions of documents.

Table of Contents

1 Purpose

This document has been created to describe the design specifications of the application LIN Driver. It consists on the implementation of the LIN protocol in the TRK-MPC5606B development board and an application, which will be a slave number 1 in a group of one master and four slaves.

Req. Id. 1.0, 1.1, 1.2

2 Definitions and abbreviations

2.1 Definitions

Cmd_NONE	Command to do nothing.
Cmd_LED_on	Command to turn the LED on.
Cmd_LED_off	Command to turn the LED off.
Cmd_LED_toggling	Command to trigger the blinking LED of the sequence.
Cmd_disable_slv	Command to disable the slave mode in the slave node.
Data	The response of a frame carries one to eight data bytes, collectively called data.
Dominant	A cero value in the LIN bus.
Frame	A frame consists of a header and a response. The reply frame for a node configuration or a diagnostic request is a response.
Header	A header is the first part of a frame; it is always sent by the master task.
ISR	Interrupt Service Routine
LED	Light Emitting Diode
LFDIV	LINFlex Divider. Intermediate variable for Baud Rate calculation.
LIN	Local interconnect Network
LIN_EA	Name of the system designed in this project.
LINFBRR	LIN Fractional Baud Rate Register
LINFlex	Local Interconect Network Flexible controller.
LINIBRR	LIN Integer Baud Rate Register
Node	A node is an ECU (electronic control unit). However, a single ECU may be connected to multiple LIN clusters.
PIT	Periodic Interrupt Timer
Recessive	A one value in the LIN bus.
Rx	Reception.
Slave node	A node that contains a slave task only, i.e. it does not contain a master task.
TOGGLING	Status where the LED blinks.
Tx	Transmission.

Req. Id. 1.13, 1.14, 1.15, 1.17, 1.18

2.2 Type name definition

Type Name	Elements					
	0	**1**	**2**	**3**	**4**	**5**
t_cmdType	cmd_NONE	cmd_LED_on	cmd_LED_off	cmd_LED_toggling	cmd_disable_slv	cmd_enable_slv
t_LEDstat	OFF	ON	TOGGLING			
t_boolean	FALSE	TRUE				
array	AZSEEV					
Scalar	1					

Req. Id. 1.12, 1.16, 1.19, 1.20, 1.21

References

N°	Document name	Reference	Revision
1	LIN Specification Package	LIN/Documents/LIN-Spec_2-2A.pdf	2.2A
2	Traceability Matrix – LIN_EA	LIN/Documents/1.0 Requirements/Traceability Matrix – LIN_EA.xls	1.10
3	MPC5607B Microcontroller Reference Manual	LIN/Documents/MPC5607BRM_Reference_Manual.pdf	7.2
4	Quick Start Guide TRK-MPC5606B	LIN/Documents/Quick_Start_Guide.pdf	3
5	LIN Network Definition	LIN/Documents/LIN_Network_Database.xls	1.0
6	AVT-718 User's manual	LIN/Documents/AVT-718.pdf	C1

3 Realization constraints and targets

3.1 TRK-MPC5606B's features

- MPC5606B MCU (144-pin LQFP).
- On-board JTAG connection via open source OSBDM circuit using the MPC9S08JM MCU.
- MCZ3390S5EK system basis chip with advanced power management and integrated CAN transceiver and LIN 2.0 interface.
- CAN interface.
- LIN interface with 1.3, 2.0, 2.1, and J2602 protocol versions supported.
- Analog interface with potentiometer.
- High-efficency green LEDs.
- 4 PushButtons.
- Serial communication interface.
- External power 9V DC to 12V DC regulated down to 5V DC.

Req. Id. 3.3

3.2 Master's features

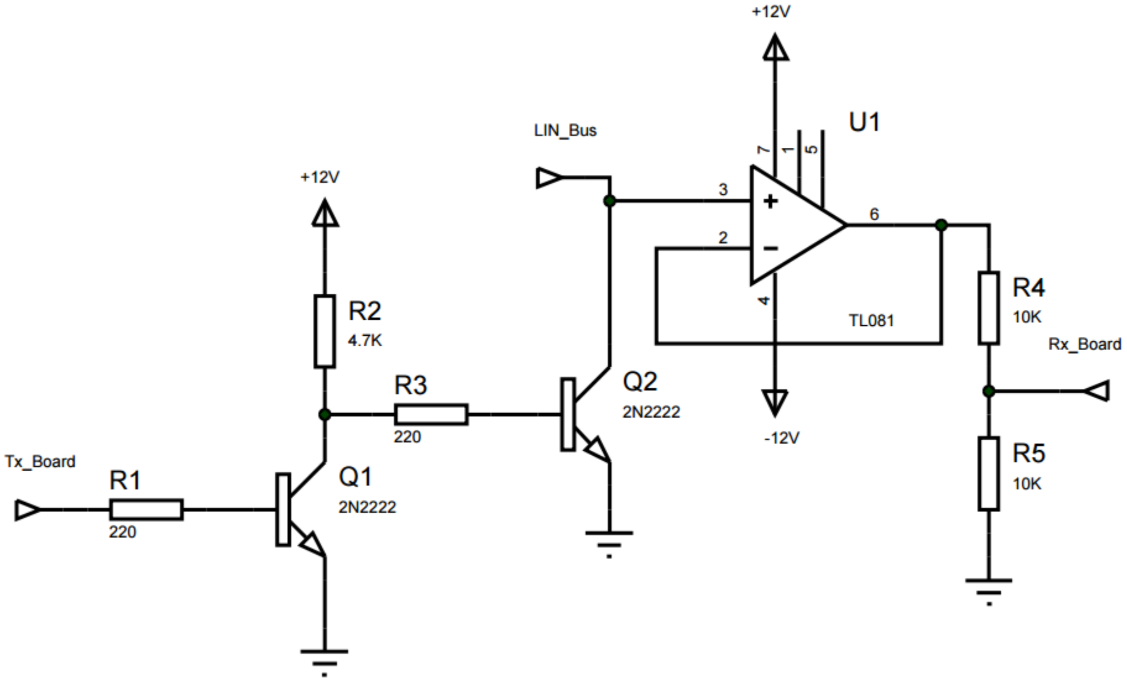
The master node was implemented using the AVT-718 Multiple Protocol Interface unit, which supports, among others, the LIN protocol specification 1.2.

Specifications
Overall size (inches): 7.3 x 6.1 x 2.0.
Weight: 18 oz.
Operating temperature range: 0 to +70 degrees C.
Input voltage range: +9.0 VDC to +30.0 VDC.
Power dissipation: 3.0 watts (nominal).
The factory default setting for the AVT-718 is 19.2 kbaud.

Since the master node only has the specification 1.2 of the LIN protocol, this will be the one used in this project. This implies that a classic checksum will be used.

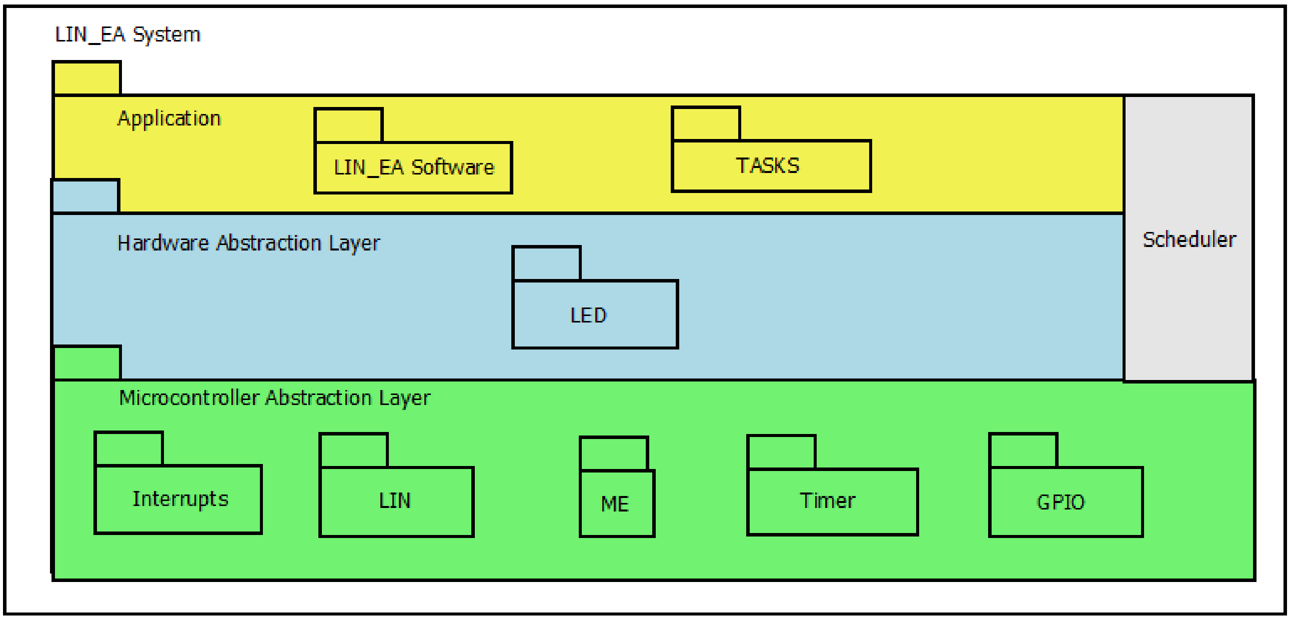
3.3 LIN transceiver

A LIN transceiver was implemented in order to connect the 5 volts logic of the Tx and Rx pins of the microcontroller to the 12 volts LIN-bus. A circuit was designed for this purpose. Two transistors 2N2222 were used to force the LIN Bus to low level. An operational amplifier TL082 was used as a buffer in order to be able to lower the voltage with a voltage divider array of resistors. The following diagram illustrates de circuit used.



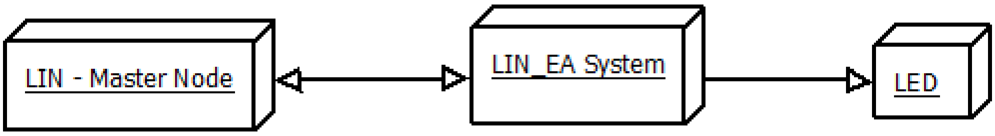
4 SW Conceptual design

4.1 Architecture design



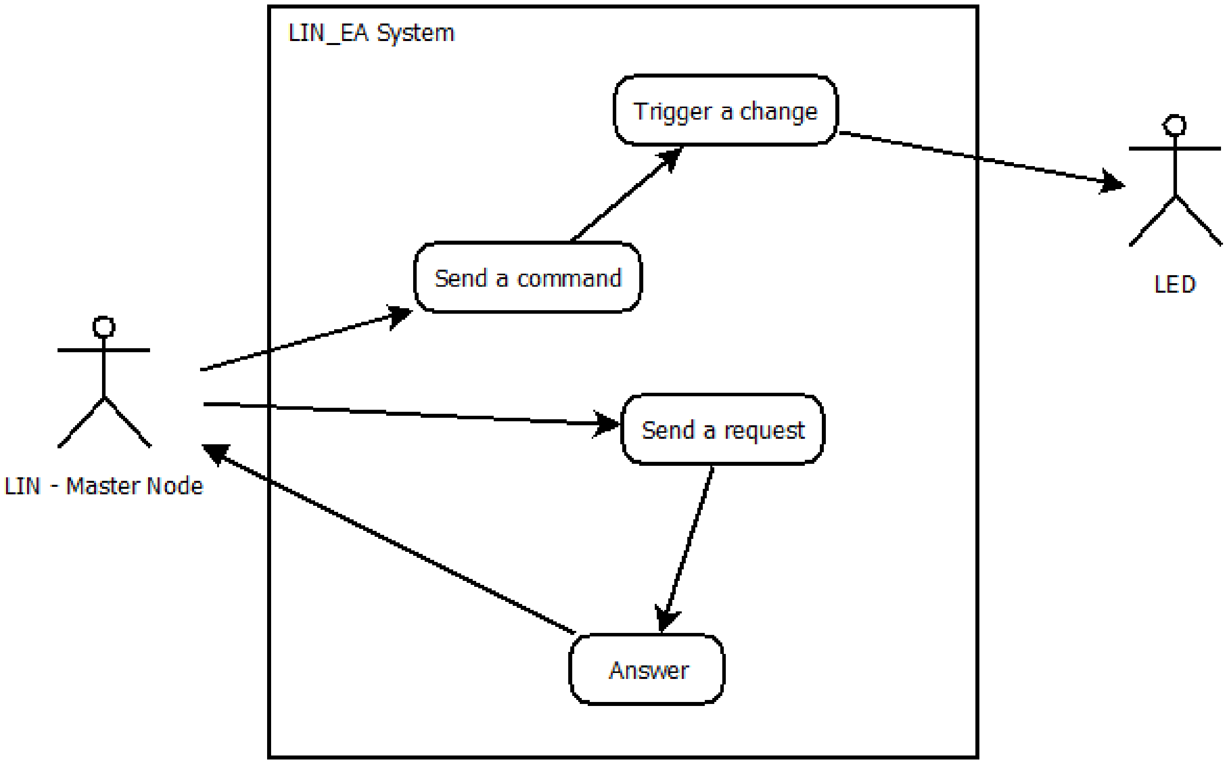
The architecture diagram show the layers in which the system is divided. In the Microcontroller Abstraction Layer, there are the modules where the interrupts, timers, GPIO, LIN interface and general configurations are handled. In the Hardware Abstraction Layer, the system manages the external modules as LEDs. In the Application layer, there is the main software and the tasks from the scheduler. Parallel to the hardware and application layer there is the scheduler, which communicates with each of the layers.

4.2 Deployment Diagram



The deployment diagram shows the interaction of the LIN_EA system with external systems. There is a bidirectional communication with a LIN master node and an output to a LED. The master node sends commands, which can affect the system and LED state.

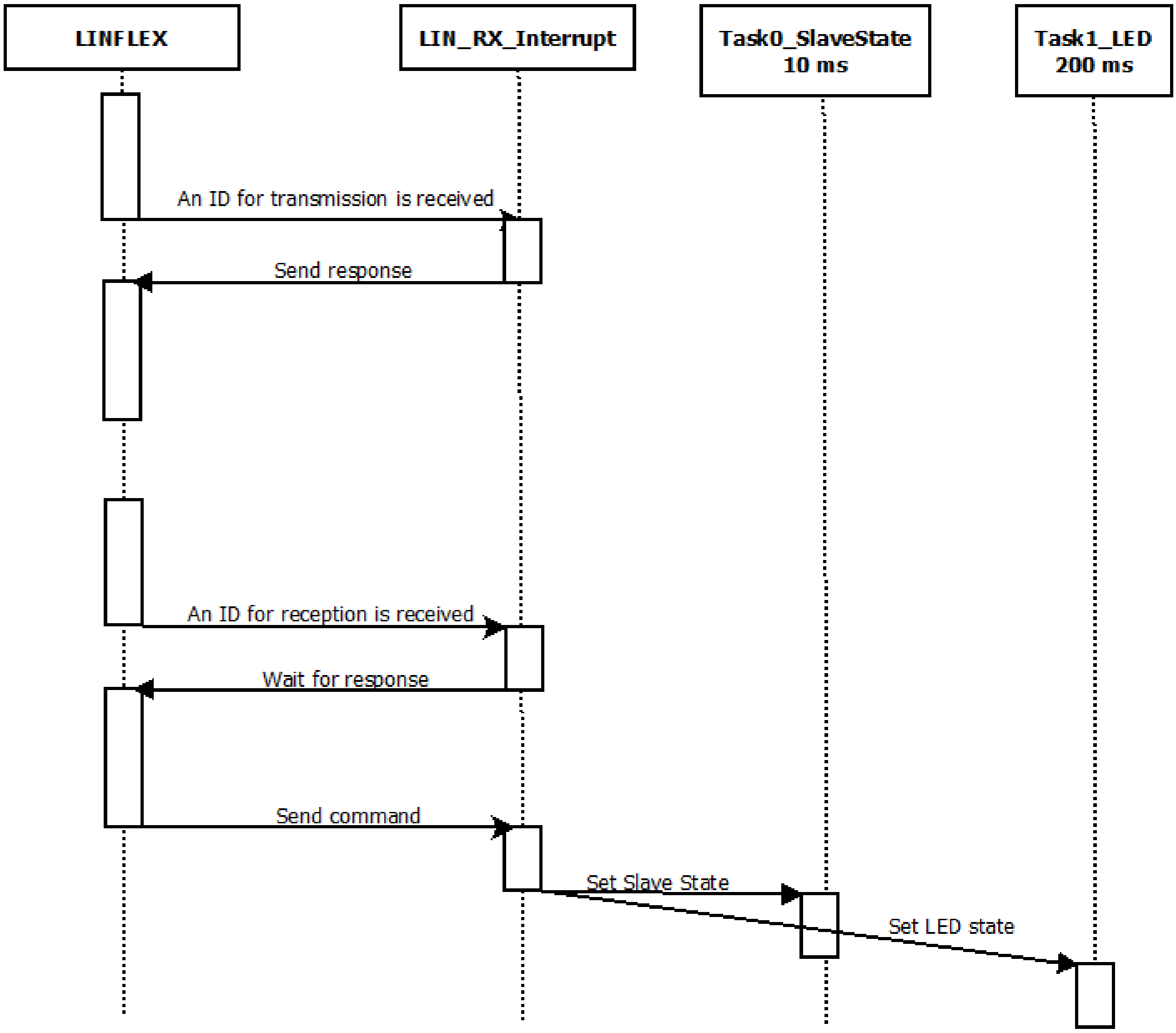
4.3 Use case diagram



Users	Description
LIN – Master Node	The master node sends data to the slave in order to affect the system.
LED	There is a LED which is controlled by the system.

Case	Description
Send a command	This case triggers a transmission by the master in order to communicate a command to be executed.
Trigger a change	This case evaluates the command sent.
Send a request	This case triggers a transmission by the master to communicate a request to receive status information.
Answer	This case triggers the transmission to the LIN – Master Node of the solicited information.

4.4 Sequence diagram



This diagram describes the interaction between different modules of the software and the microcontroller. The LINFLEX controller from the microcontroller is in charge of receiving the messages that arrive to the system through the LIN protocol. Then it sends them to the interrupt.
If it matches for transmission, the LIN_RX_interrupt fills the buffer and triggers the transmission of the data. Lastly, LINFLEX sends the data and checksum. If it matches for reception, the LIN_RX_interrupt sets the length of the data that will be received. After the reception is done, the data is taken by the interrupt and marks them as available for task0_SlaveState and task1_LED.

4.5 Error handling

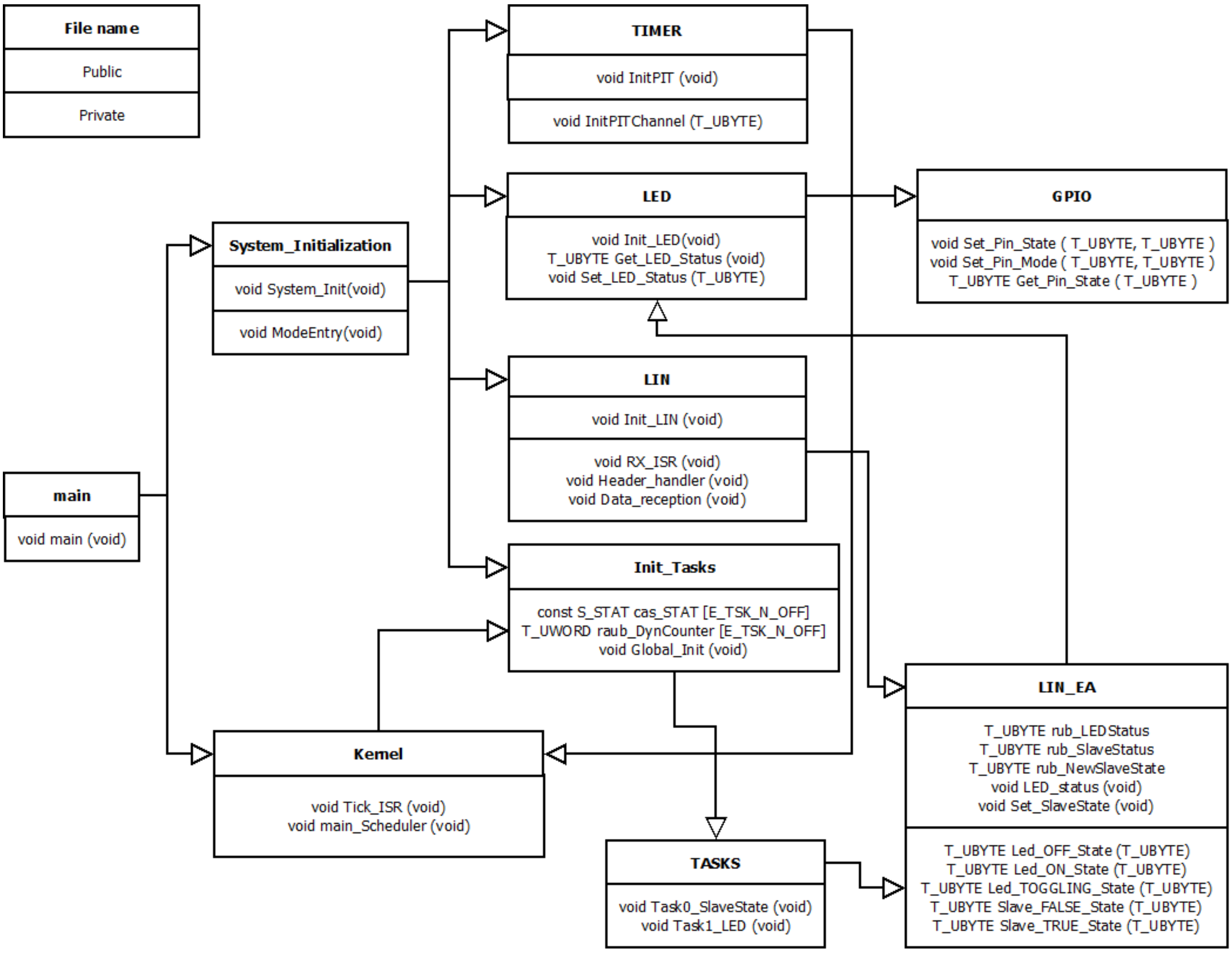
In order to consume less CPU resources, the hardware of the MPC5606B handles the signals of the LIN-bus, managing the errors and only generating interrupts when the software needs to specify the length of the response field, specify the data that is going to be sent or when the data is ready to be read. The following errors alter the internal state machine of the LINFlex controller of the MPC5606B.

Error	Description	LINFlex response
Bit error	During transmission, the value read back from the bus differs from the transmitted value.	• Stops the transmission of the frame after the corrupted bit. • Returns to idle state.
Framing error	A dominant state has been sampled on the bit of the currently received character (Sync field, identifier, or data field).	If encountered during reception: • Discards the current frame. • Returns immediately to idle state.
Checksum error	The computed checksum does not match the received checksum.	If encountered during reception: • Discards the received frame. • Returns to idle state.
Header error	An error occurred during header reception (break delimiter error, inconsistent sync field, header timeout)	If encountered during header reception, a break field error, an inconsistent sync field, or a timeout: • Discards the header. • Returns to idle state.

In case that a received signal is longer than expected, it only takes the bytes that the software was expecting. Since the checksum would not be the real checksum but another byte of data, the checksum would be incorrect, generating a checksum error, discarding the received frame and returning the internal state machine to idle state.

5 SW Component internal breakdown

5.1 Functional Decomposition



File	Description
Main	Main module that runs the main function. It is divided in system's initialization and execution of the scheduler.
System_Initialization	The function of this module is to call the functions that initializes the mode of operation, peripherals, and the scheduler.
LIN_EA	It is in charge of executing the main application of the program. It contains two main functions, which also contain the state machines to control the LED status and the slave state.
Kernel	The scheduler is being executed here. It handles the main configurations and the tick interrupt.
TASKS	This module contains the periodic tasks that are executed by the

	scheduler.
Init_Tasks	This module contains the global initializations which are needed for the correct execution of the scheduler.
LIN	This module contains the LIN driver. It consists of an initialization of the LIN controller and an interrupt for reception.
TIMER	This file contains the configurations that must be done to achieve the periodic interrupt that gives the Ticks to the scheduler.
GPIO	This module handles the registers needed to configure ports and change state of pins.
LED	This module configures and handles the state of the LED.

Function Description and Dynamic Behavior

5.2 LIN

5.2.1 Function void Init_LIN (void)

Description	This function initializes the LIN controller of the board.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

The LIN controller must be set to initialization mode in order to set its configurations. First the RX and TX pin are set as INPUT and OUTPUT, respectively. The break detection threshold is set to 11 dominant local slave bit times. Another configuration is to set the endianness to Little Endian, the initial baud rate is set, auto-baud rate is enabled and the interrupt is configured. Before leaving the function, the software sets the LIN controller to normal mode.

In order to calculate the initial Baud Rate (BR) the following formula is used:

$$\mathrm{LFDIV} = \mathrm{LINIBRR} + \frac{\mathrm{LINFBRR}}{16}$$

$$BR = \frac{f}{16 * LFDIV}$$

If $\mathrm{f} = 64$ MHz, $\mathrm{LINIBRR} = 416$, and $\mathrm{LINFBRR} = 11$, then:

$$\mathrm{LFDIV} = \mathrm{LINIBRR} + \frac{\mathrm{LINFBRR}}{16} = 416 + \frac{11}{16} = 416.6874$$

$$BR = \frac{f}{16 * LFDIV} = \frac{64\ x\ 10^6}{16 * 416.6874} = 9599.51 \approx 9600\ Symbols/second$$

Req. Id. 1.3, 1.30, 2.0, 2.1, 2.7, 2.11.

5.2.2 Function void RX_ISR (void)

Description	This function is called every time a reception interrupt is generated from the LINFlex controller. It calls the Header_handler or Data_reception function if the Header Reception Flag or Data Received Flag are set, respectively.
Parameter 1	Void
Return Value	Void
Precondition	Basic configuration of LINFlex must be done and the interrupt must be initialize in order to run this function.
Post condition	Does not apply
Error Conditions	Does not apply

5.2.3 Function void Header_handler (void)

Description	This function is called every time a header is received. It is in charge of discard an invalid ID or setting the parameters to receive or send the response.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

In order to consider a valid header, this shall start with a falling edge of the break field, have a break and sync field, and ends after the end of the stop bit of the protected identifier field. A frame shall be constructed of a number of fields, one break field followed by four to eleven byte fields.

Req. Id. 2.3, 2.4, 2.5, 2.8.

The following messages ID (considering parity) are valid for reception:

Message name	Msg ID	Msg Data Length (byte)	Message publisher	Message subscriber	Signal length (bits)	Signal type	Signal Description
MASTER_CMD_ALL	0xCF	1	Master	Slave 1, 2, 3, 4.	4	T_cmdType	Command for all nodes
MASTER_CMD_SLV1	0x50	1	Master	Slave 1	4	T_cmdType	Command for node 1.

The following messages ID (considering parity) are valid for transmission. After the reception of one of those IDs, the function fills the buffer with the data described below and triggers the transmission. This transmission begins the response which starts at the end of the stop bit of the protected identifier field and ends after the stop bit of checksum field.

Message name	Msg ID	Msg Data Length (byte)	Message publisher	Message subscriber	Signal length (bits)	Signal type	Signal Description
SLAVE1_RSP	0x20	2	Slave 1	Master	2	T_LEDstat	Return LED status
					1	T_boolean	Return node status
SLAVE1_ID	0xF0	7	Slave 1	Master	8	Scalar	Return team number
					48	Array	Return initials of team members

Req. Id. 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 2.0, 2.1, 2.6.

5.2.4 Function void Data_reception (void)

Description	This function is called every time a Data Reception Flag is set which is after the checksum of the data received is validated. It is in charge of setting flags depending on the command received in the response.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id. 2.2, 2.9, 2.10.

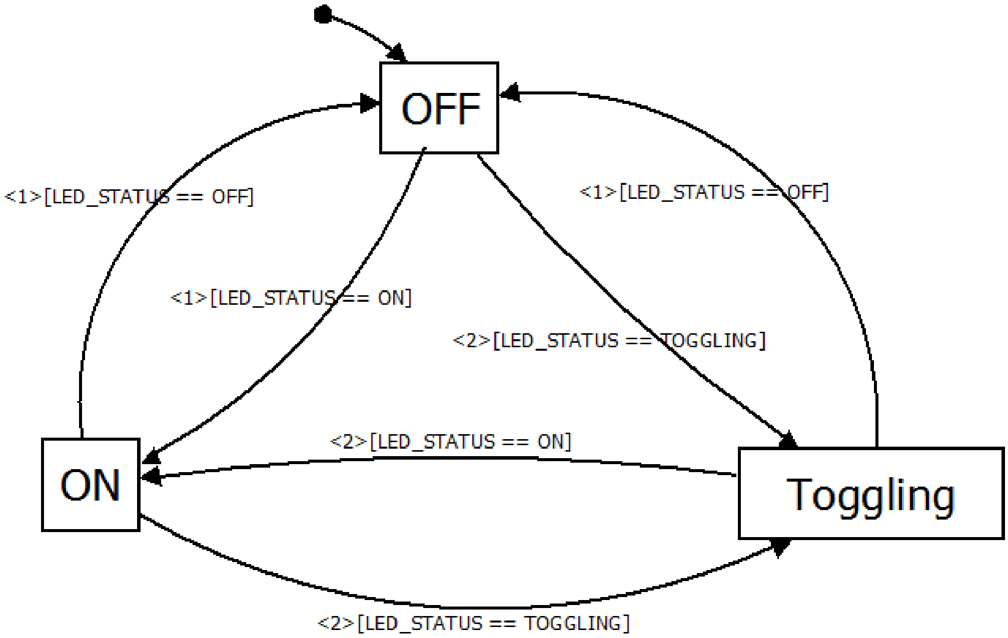
5.3 LIN_EA

5.3.1 Function void LED_status (void)

Description	This function is called every time the task 1 is executed. It has a state machine, which controls the response's reception after a valid command is received.
Parameter 1	Void
Return Value	Void
Precondition	Basic configuration of LINFlex must be done and the interrupts must be initialize in order to run this function.
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id. 3.0, 3.1

Dynamic Behavior
State Chart



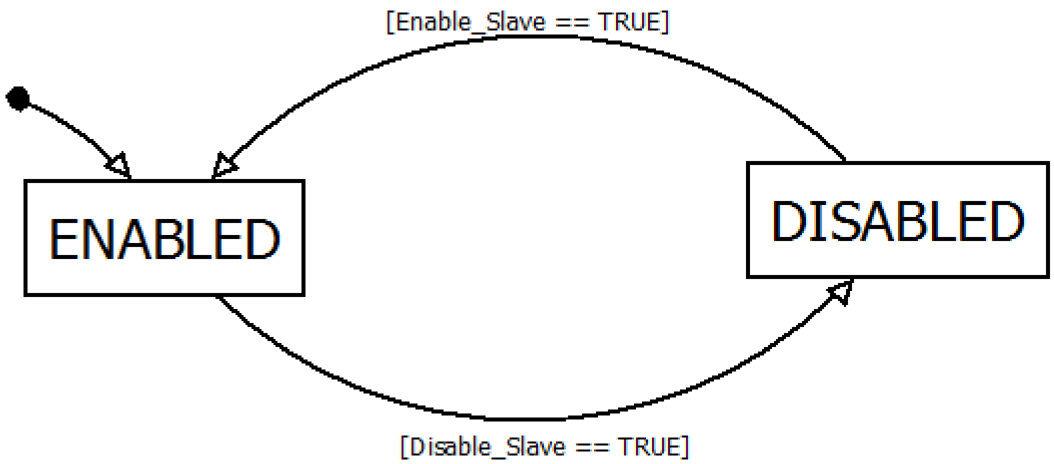
State	Description
OFF	State in which the LED is OFF. Req. Id. 1.24
ON	State in which the LED is ON. Req. Id. 1.23
Toggling	State in which the LED toggles. Req. id. 1.25

5.3.2 Function void Set_SlaveState (T_UBYTE)

Description	This function is executed every 10 milliseconds in the task 0. It has a state machine, which controls the response's reception after a valid command is received.
Parameter 1	T_UBYTE. Flag that might trigger a change in the internal state machine.
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id. 3.2

Dynamic Behavior
State Chart



State	Description
Enabled	State in which the slave node acts fully in slave mode. The state transition is triggered by the command cmd_disable_slv. The command cmd_none shall not trigger any change. Req. Id. 1.26, 1.28
Disabled	State in which the system will not accept any other command than the command cmd_enable_slv. The command cmd_none shall not trigger any change. Req. Id. 1.27, 1.29

Req. Id. 1.22

5.3.3 Function T_UBYTE Led_OFF_State (T_UBYTE)

Description	Function that implements the OFF state in the state machine of the function LED_status.
Parameter 1	T_UBYTE. Receives the state variable so it could change it.
Return Value	T_UBYTE. Returns the state variable.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.3.4 Function T_UBYTE Led_ON_State (T_UBYTE)

Description	Function that implements the ON state in the state machine of the function LED_status.
Parameter 1	T_UBYTE. Receives the state variable so it could change it.
Return Value	T_UBYTE. Returns the state variable.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.3.5 Function T_UBYTE Led_TOGGLING_State (T_UBYTE)

Description	Function that implements the TOGGLING state in the state machine of the function LED_status.
Parameter 1	T_UBYTE. Receives the state variable so it could change it.
Return Value	T_UBYTE. Returns the state variable.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.3.6 Function T_UBYTE Slave_FALSE_State (T_UBYTE)

Description	Function that implements the FALSE state in the state machine of the function Set_SlaveState.
Parameter 1	T_UBYTE. Receives the state variable so it could change it.
Return Value	T_UBYTE. Returns the state variable.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.3.7 Function T_UBYTE Slave_TRUE_State (T_UBYTE)

Description	Function that implements the TRUE state in the state machine of the function Set_SlaveState.
Parameter 1	T_UBYTE. Receives the state variable so it could change it.
Return Value	T_UBYTE. Returns the state variable.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.4 TASKS

5.4.1 Function void Task0_SlaveState (void)

Description	This task is executed every 10 milliseconds. Its main function is to call the state machine that handles the slave status of the system.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.4.2 Function void Task1_LED (void)

Description	This task is executed every 200 milliseconds. Its main function is to call the state machine that handles the status of the LED.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.5 Init_Tasks

5.5.1 Function void Global_Init (void)

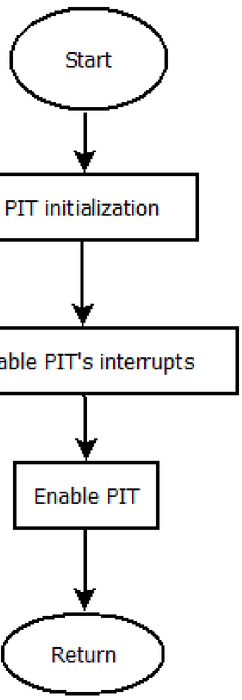
Description	This function initializes the scheduler.
Parameter 1	Void
Return Value	Void
Precondition	This function should be called before executing the scheduler.
Post condition	The scheduler's functionalities can be used.
Error Conditions	Does not apply.

5.6 TIMER

5.6.1 Function void InitPIT (void)

Description	PIT is initialized.
Parameter 1	Void
Return Value	Void
Precondition	This function is called in the beginning of the main program to initialize the PIT.
Post condition	The interrupts every 5ms will be generated.
Error Conditions	Does not apply

Dynamic Behavior
Activity diagram



5.6.2 Function void InitPITChannel (T_UBYTE)

Description	This function configures the given channel of the PIT timer.
Parameter 1	T_UBYTE PIT channel which must be configured.
Parameter 2..n	Does not apply
Return Value	Void
Precondition	This function should be called in the beginning of the main application.
Post condition	The initialized channel of the PIT will be ready to use.
Error Conditions	Does not apply.

5.7 LED

5.7.1 Function void Init_LED (void)

Description	This function initializes the pin used for the LED.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	The LED will be able to be changed.
Error Conditions	Does not apply

5.7.2 Function T_UBYTE Get_LED_Status (void)

Description	This function returns the state of the LED.
Parameter 1	Void
Return Value	T_UBYTE. Corresponds to the logic state of the LED. It can be either 0 or 1, meaning off or on, respectively.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.7.3 Function void Set_LED_Status (T_UBYTE)

Description	This function changes the LED status depending on the parameter received.
Parameter 1	T_UBYTE. It can receive either 0 or 1 to turn it off or on respectively.
Return Value	Void
Precondition	The function Init_LED should be executed before.
Post condition	The LED status will change to the one received in the parameter.
Error Conditions	Does not apply

5.8 GPIO

5.8.1 Function void Set_Pin_State (T_UBYTE, T_UBYTE)

Description	Changes the logic level of the output pin selected.
Parameter 1	T_UBYTE. Corresponds to the pin number that should be affected.
Parameter 2	T_UBYTE. It can receive either 0 or 1 to turn the pin off or on respectively.
Return Value	Void
Precondition	The mode of the pin selected should be OUTPUT.
Post condition	The logic level for the selected pin, will be the one selected in the second parameter.
Error Conditions	Does not apply

5.8.2 Function void Set_Pin_Mode (T_UBYTE, T_UBYTE)

Description	This function changes the pin mode of the selected pin to the selected mode.
Parameter 1	T_UBYTE. Corresponds to the pin number that should be affected.
Parameter 2	T_UBYTE. It can receive a value according to the following definitions: 0 -> OUTPUT, 1 -> INPUT, 2 -> LIN_TX, 3 -> LIN_RX.
Return Value	Void
Precondition	Does not apply
Post condition	The selected pin will act as the selected pin mode.
Error Conditions	Does not apply

5.8.3 Function T_UBYTE Get_Pin_State (T_UBYTE)

Description	This function returns the state of the given pin.
Parameter 1	T_UBYTE. Corresponds to the pin number which state is unknown.
Return Value	T_UBYTE. Corresponds to the logic state of the pin. It can be either 0 or 1, meaning off or on, respectively.
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.9 KERNEL

5.9.1 Function void Tick_ISR (void)

Description	This function is the one that handles the clock Ticks in order to trigger the tasks. This interrupt runs periodically every 5 milliseconds according to the configuration of the PIT.
Parameter 1	Void
Return Value	Void
Precondition	This function is called when a PIT interrupt is generated.
Post condition	It will literally interrupt the flow of the program to implement its code.
Error Conditions	Does not apply

5.9.2 Function void main_Scheduler (void)

Description	This function contains the main function of the scheduler which controls the timing for each of the tasks.
Parameter 1	Void
Return Value	Void
Precondition	The Global_Init function should be executed before.
Post condition	Does not apply
Error Conditions	Does not apply

5.10 Main

5.10.1 Function void main (void)

Description	This function runs first by default. It is divided in system initialization and execution of the scheduler.
Parameter 1	Void
Parameter 2	Does not apply.
Return Value	Void
Precondition	Does not apply.
Post condition	Does not apply.
Error Conditions	Does not apply.

5.11 System_Initialization

5.11.1 Function void System_Init (void)

Description	This function calls the functions that initializes the mode of operation, peripherals, and the scheduler.
Parameter 1	Void
Parameter 2	Does not apply.
Return Value	Void
Precondition	This should be the first function called in the main program.
Post condition	It will be possible to use the microcontroller with the configurations done.
Error Conditions	Does not apply.

5.11.2 Function void ModeEntry (void)

Description	It initializes the mode of operation.
Parameter 1	Void
Parameter 2	Does not apply.
Return Value	Void
Precondition	This should be the first function called in the main program.
Post condition	It will be possible to use the microcontroller with the configurations done.
Error Conditions	Does not apply.