|  |
| --- |
| EMBBEDED SYSTEMS SPECIALIZATION PROGRAMM |
| G2 Task #1: Lin Frame |
| Communications Software Development for Embedded Environments |

|  |
| --- |
| **Feliciano Angulo Angulo**  Email: [se704751@iteso.mx](mailto:se704751@iteso.mx)  **Juan Pablo Garcia**  Email: [se716963@iteso.mx](mailto:se716963@iteso.mx)  **Jesus Íñiguez**  Email: [se717040@iteso.mx](mailto:student3_email@iteso.mx) |

**29/03/2018**

Embedded Systems Specialization Program

[www.sistemasembebidos.iteso.mx/alumnos](http://www.sistemasembebidos.iteso.mx/alumnos)

I T E S O A. C., Universidad Jesuita de Guadalajara

Periférico Sur Miguel Gómez Morín #8585, Tlaquepaque, Jalisco, México

Technical Report Number: ESE-O2014-001

® ITESO A.C.

|  |
| --- |
| ***Abstract:***  *This document shows how LIN protocol is implemented using an UART driver in in SAMV71 XPlained evaluation board. Lin and UART implementation are AUTOSAR compliant, whit this implementation is possible to transmit LIN frame through UART, Scope’s captures are included to show LIN frame from Tx line.* |
| ***Keywords:***  *LIN*  *UART*  *AUTOSAR*  *FRAME* |

# Table of Contents

[Table of Contents ii](#_Toc510451989)

[1. Table of Figures iii](#_Toc510451990)

[2. Introduction 1](#_Toc510451991)

[2.1. Requirements 1](#_Toc510451992)

[3. Functional Description 3](#_Toc510451993)

[3.1. AUTOSAR Implementation 3](#_Toc510451994)

[3.2. LIN functions Implementation 5](#_Toc510451995)

[3.2.1. State description 6](#_Toc510451996)

[3.2.2. IDLE 6](#_Toc510451997)

[3.2.3. SEND\_BREAK 6](#_Toc510451998)

[3.2.4. SEND\_SYNC 7](#_Toc510451999)

[3.2.5. SEND\_PID 7](#_Toc510452000)

[3.2.6. SEND\_EOT 7](#_Toc510452001)

[3.2.7. SEND\_RESPONSE 7](#_Toc510452002)

[4. Results 7](#_Toc510452003)

[5. Conclusions 9](#_Toc510452004)

# **Table of Figures**

[Figure 1 AUTOSAR file structure 3](#_Toc510452005)

[Figure 2 AUTOSAR layers 4](#_Toc510452006)

[Figure 3 Lin\_Init implementation 5](#_Toc510452007)

[Figure 4 Lin\_SendFrame implementation 5](#_Toc510452008)

[Figure 5 Lin\_Isr implementation 6](#_Toc510452009)

[Figure 6 LIN frame 7](#_Toc510452010)

[Figure 7 Header capture from Oscilloscope 8](#_Toc510452011)

# Introduction

LIN (Local Interconnect Network) is a serial network protocol used for communication between components in vehicles. LIN implementation is based in UART/SCI hardware, so, when a microcontroller does not have a LIN peripheral integrated but a UART, it can be implemented over this UART easily.

The purpose of this work is to demonstrate the above description, Lin can be implemented over UART.

## Requirements

Provide a SW solution with defined interfaces that will allow to form the LIN Frame through the UART controller available in the SAM V71 controller.

Software must be based on the attached Base Project.

Requirements are as follows:

Transmit a single LIN frame sending a Break - Synch - and a Data Byte with a recurrence of 10ms.

### Architecture

The SW must be compliant with the AUTOSAR SW layered architecture, so a File Structure must be provided. This means that (if using UART interfaces) the UART driver interfaces will be at MCAL Layer and LIN interfaces will be at the ECU Abstraction Layer.

File Structure:

../source

->/bsw

->/mcal

->/com

->/uart

(uart files here)

->/ecual

->/com

->/linif

(lin files here)

->/services

->/system

->/scheduler

(scheduler files here)

Note: In the case that uart driver is used from the provider, file structure shall be kept AS IS.

### Implementation

The LIN functions to be provided are:

void Lin\_Init (uint16\_t LinBaudrate)

- This function will configure the lower layer UART driver

- Baudrate will as per LinBaudrate parameter

- Interrupts shall be configured for each data byte to be transmitted or received

- This function shall provide to the lower layer a function callback (Lin\_Isr) to be invoked at any of the RX or TX UART interrupts

void Lin\_SendFrame (uint8\_t LinPid)

- This function will send a predefined header as per the LIN protocol with the rate define in the Lin\_Init function.

- This function shall be asynchronous, i.e. it will trigger the "send command" and will continue its operation without waiting for the header to be completely sent over the bus.

- The header shall be composed in order of:

- 1. Break = (from 10 to 13 bit times) ideally >= 13 bit time

- 2. Synch = 0x55

- 3. ID = LinPid

In order to support the underlined LIN SW component, a state machine can be provided:

typedef enum

{

IDLE,

SEND\_BREAK,

SEND\_SYNC,

SEND\_PID,

SEND\_RESPONSE

}LinStateType;

The sequence shall be as follows (since no response shall be provided for this task)

IDLE -> SEND\_BREAK -> SEND\_SYNC -> SEND\_PID -> IDLE

IDLE: No activity

SEND\_BREAK: Send break process is about to start or in progress

SEND\_SYNC: Send sync process is about to start or in progress

SEND\_PID: Send pid process is about to start or in progress

Note: Lin\_SendFrame will start the state machine by changing the state to SEND\_BREAK as long as there is no a frame transmission/reception in progress.

void Lin\_Isr(void)

- UART TX/RX interrupts:

- will invoke this function

- will serve as input transitions to manage the LIN state machine (transitions and actions)

Table 1.1 Example of Table Caption 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# Functional Description

## AUTOSAR Implementation

To make implementation AUTOSAR compliant, the file structure was created according to requirement number 1, it can be seen in figure 1 with required files in specify folder.

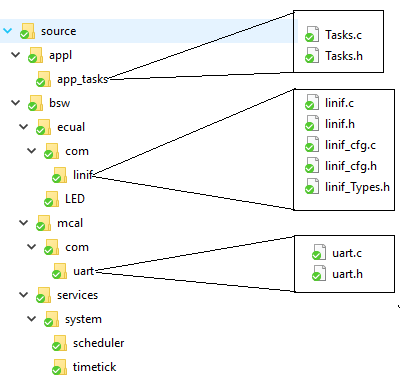


Figure 1 AUTOSAR file structure

Interaction between files showed in Figure 1, are based in AUTOSAR layer format, Figure 2 shows such layers, according to this, UART driver is part of MCAL, LIN is implemented as part of ECU Abstraction Layer and a 10 ms task is part of the application layer, all executions are controlled by the scheduler, which is part of service laver.

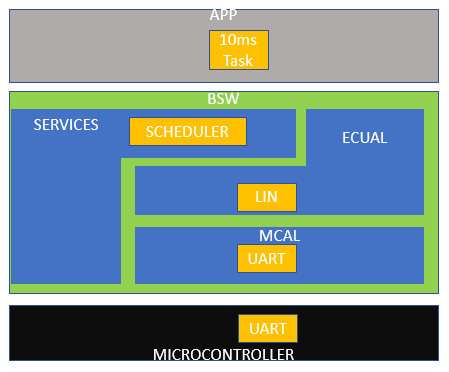


Figure 2 AUTOSAR layers

## LIN functions Implementation

There are 3 main function in Lin to be implemented according to requirement #2, Lin\_Init, Lin\_SendFrame and Lin\_Isr,

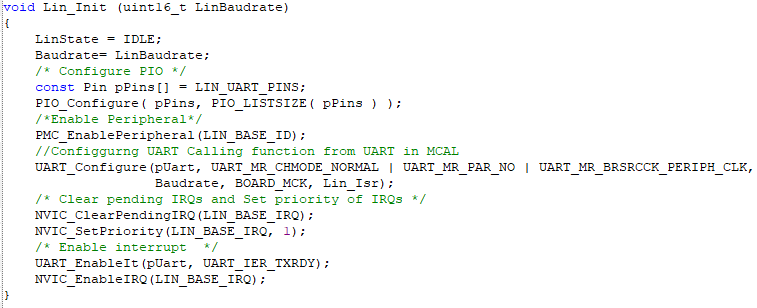


Figure 3 Lin\_Init implementation

As was mentioned before, LIN implementation is in ECU abstraction layer, and according to AUTOSAR, lower interface(MCAL) must be call from here, in this case, needed function from UART driver must be called here, as can be seen in Figure 3, Lin\_Init function initialize UART driver, first enabling used GPIO (details are in Table 2), then UART\_Configure function is called to configure UART, and at the end interruptions are set.

Table 1 UART I/O Lines details

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Signal | IO Line | Peripheral |
| UART2 | URXD2 | PD25 | C |
| UART2 | UTXD2 | PD26 | C |

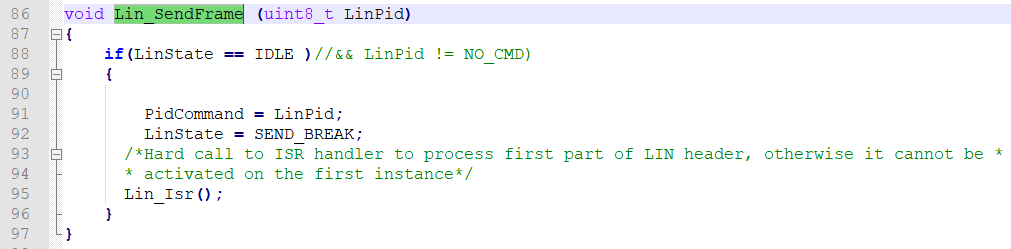


Figure 4 Lin\_SendFrame implementation

Lin\_SendFrame is set to initialize a LIN frame message transmission (see Figure 4), LIN frame transmission is implemented based in a state machine for send each part of the frame.

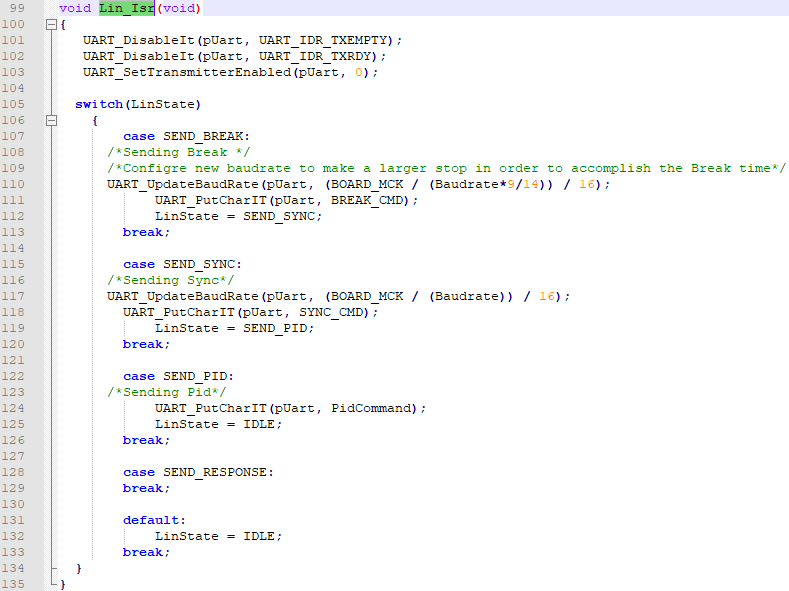


Figure 5 Lin\_Isr implementation

In Figure 5 can be seen Lin\_Isr function, there, is implemented a state machine for control LIN frame states, SEND\_BREAK; SEND\_SYNC, SEND\_PID, SEND\_EOT and SEND\_RESPONSE. Those state are described in next section.

### State description

### IDLE

No transmission is done here.

### SEND\_BREAK

Break is sent here, BREAK consist in a minimum of 13 recessive bits (0), since in a normal UART driver only can be send a max of 8 recessive bits, is necessary to reduce official LIN baud rate to make 8 recessive bits look like at least 13 recessive bits. Baud rate update is done in this state just before send BREAK\_CMD (0x55) value.

### SEND\_SYNC

Here basically is needed send the 0x55 sync byte, but because in previous STATE baud rate was reduced, here baud rate is set to the official LIN baud rate before send 0x55

### SEND\_PID

Protected Identifiers is sent here just after SEND\_SYNC finish.

### SEND\_EOT

Not implemented

### SEND\_RESPONSE

Not implemented

# Results

The complete LIN frame standard is showed in Figure 6, however, target in this work is only send the header (Break, Sync and PID).

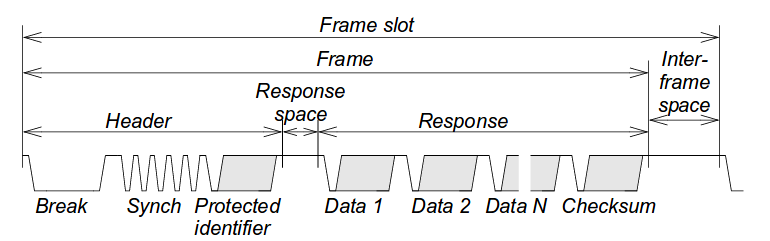


Figure 6 LIN frame

Using the oscilloscope is possible to capture Lin frame for validate that our implementation is sending the header according to the standard. Figure 7 shows the result.

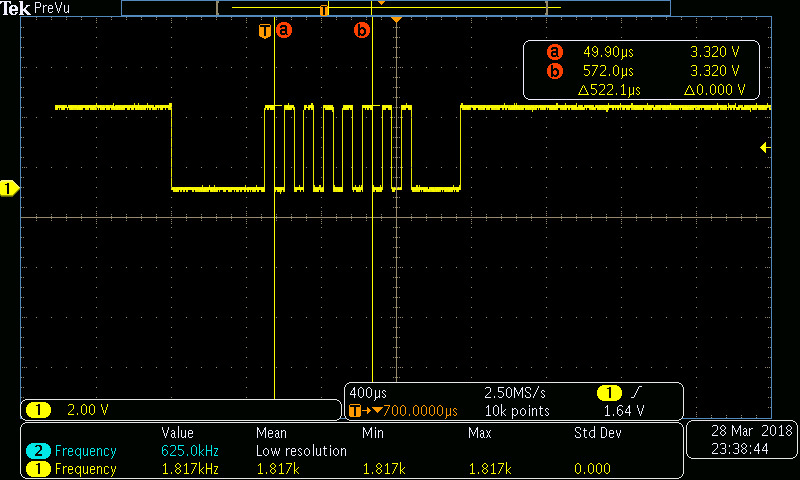


Figure 7 Header capture from Oscilloscope

# Conclusions

It was demonstrated that LIN frame can be sending using an UART driver since LIN is a Serial Based protocol.

So far, only the header was implemented, which is sent by a LIN master node, but this is a first step to create a complete LIN driver to be used by master and slave nodes.