Topicos Selectos en Sistemas Digitales / Topicos Selectos de Computación I (CUCEI – Universidad de Guadalajara)

Title: LIN Master Node Design Document

Maturity:	draft
Author(s):	Carlos F. Calvillo Cortes
Version:	0.1
Distribution:	Public
Security Classification:	None

Document History.

Ver. x.y	Date yyyy-mm-dd	Maturity draft/ reviewed/ released	Author Name/ Department	Description
0.1	2013-11-25	draft	Carlos Calvillo / I BS	Document creation. First draft
				-

Table of Contents:

1. Introduction

This document describes the LIN master node implementation in the Freescale® Kinetis platform for the lecture “Topicos Selectos en Sistemas Digitales / Computación I” of the “Universidad de Guadalajara”.

1.1. Abbreviations and Definitions

LIN	Local Interconnect Network
RX	Reception
TX	Transmission
MSTR	Master
SLV	Slave
MSG	Message
HAL	Hardware Abstraction Layer

1.2. References

No.	Document	Revision or Date	Link (if any)
/0/	LIN-Spec_2-2A.pdf	2.2A	http://www.lin-subbus.org/
/1/			

2. Software Layers

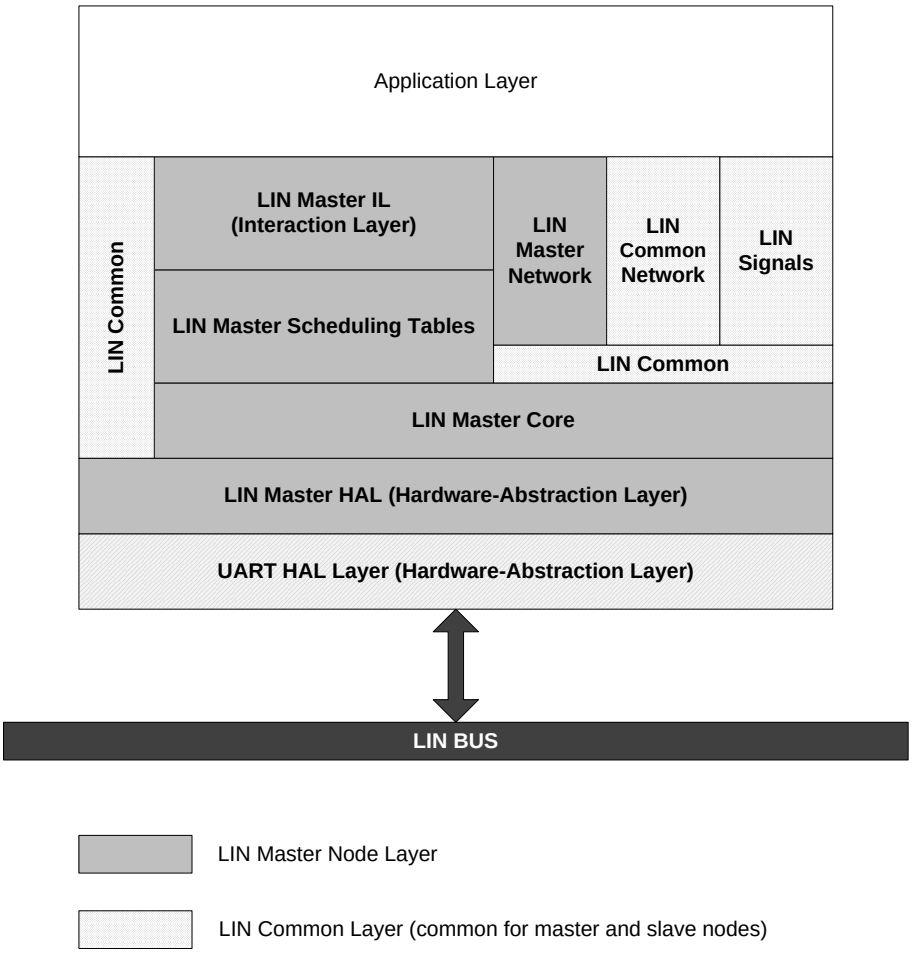

Figure 1: Software Layers

2.1. Layer descriptions

Layer	Description
Application Layer	This layer correspond to the application functionality which uses the LIN data being received by LIN and writes the data being transmitted by LIN.
LIN Master Node Layer	This layer implements the functionality of the LIN Master node
LIN Common Layer	This layer provides implementation of functionality that is common either for a LIN master node or a LIN slave node.
UART Layer	This layer implements all UART related interfaces to be used by the LIN functionality.

2.1. Functional Blocks descriptions

Functional Block Name	Description	Related source code files
LIN Master IL	Provides the “skeleton” of the callbacks being called	LIN_mstr_hal.c

	by the LIN master node. Some of those callbacks are associated to messages defined in the LIN network so this file changes each time any of the the LIN messages changes in the LIN network definition.	LIN_mstr_hal.h
LIN Master Network	Defines and implements all functionality related to the LIN messages of the network where the master node is connected. This file changes each time any of the the LIN messages changes in the LIN network definition.	LIN_mstr_network.c LIN_mstr_network.h
LIN Master Scheduling Tables	Defines and implements all functionality related to the LIN master scheduling tables. This file changes each time a scheduling table or messafe changes in the LIN network.	LIN_mstr_sched.c LIN_mstr_sched.h
LIN Master Core	Implements the core functionality of the LIN master node, i.e., the functionality for sending and receiving LIN frames. This functionality does not depend on the LIN network definition so it does not change upon changes of it.	LIN_mstr_core.c LIN_mstr_core.h
LIN Master HAL	Implements the interfaces of the LIN master core functionality with the UART hardware.	LIN_mstr_hal.c LIN_mstr_hal.h
LIN Common	Provides functionality that is common among any node of the LIN network, i.e., either the master node or any slave node. For example, the checksum calculation is implemented in this functional block.	LIN_common.c LIN_common.h
LIN Common Network	Provides definitions of the LIN network which is common to all LIN nodes (master and slaves). For example, the ID and data length of the messages, etc.	LIN_common_network.h
LIN Signals	Provides data structures to ease the access of the individual signals of the LIN frames.	LIN_signals.h
UART HAL	Implements the API's for sending and receiving bytes with the UART hardware.	UART_hal.c UART_hal.h

3. LIN Master Core State Machine

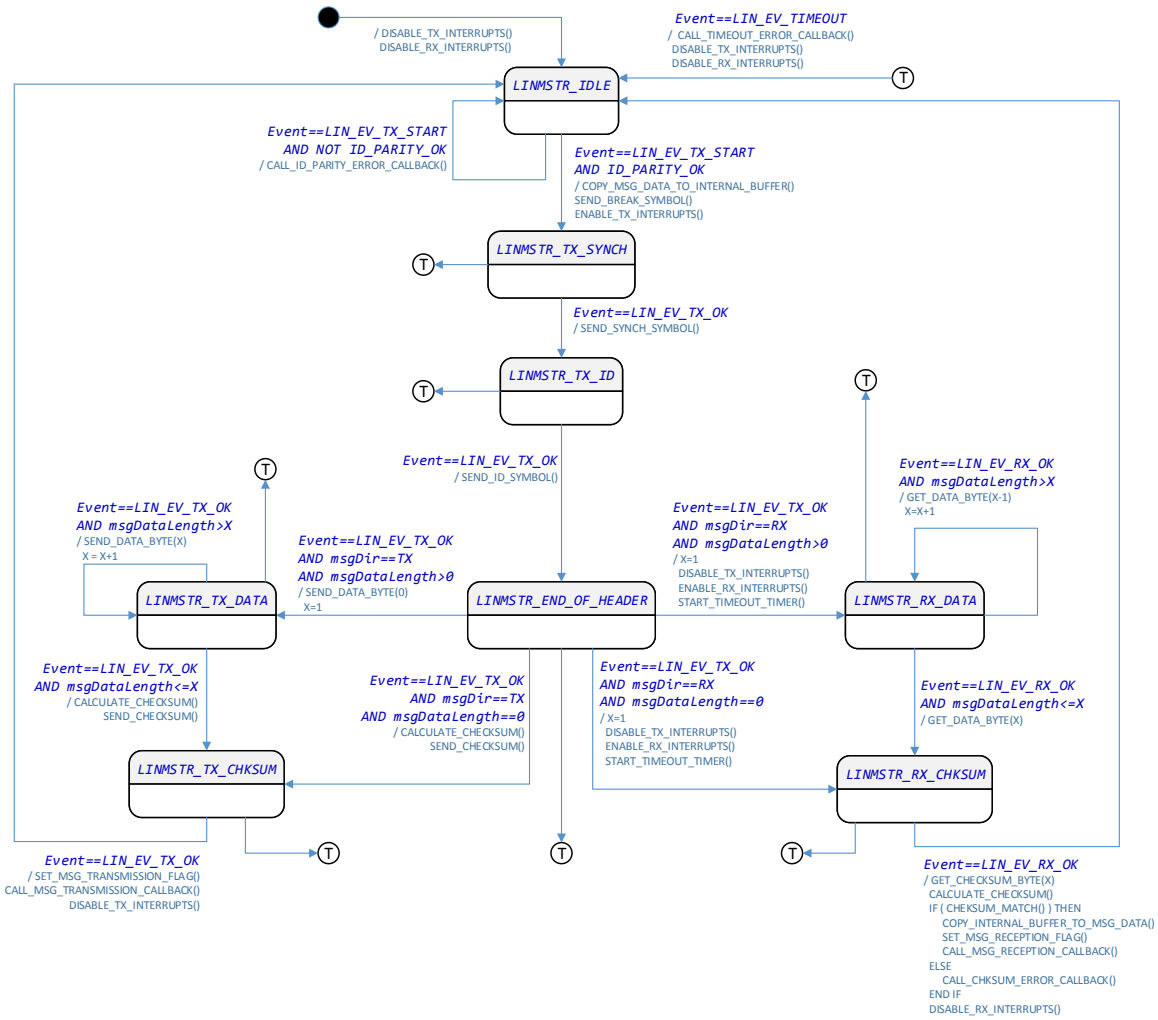

Figure 2: LIN Master Core State Machine

3.1. State Descriptions

State	Description
LINMSTR_IDLE	State machine will stay in this IDLE state as long as no LIN frame transmission has been requested by the scheduling table algorithm. When a frame transmission is requested, this state will trigger the transmission of the BRAKE symbol.
LINMSTR_TX_SYNCH	Wait for successful transmission of BRAKE symbol, when this happens, transmission of SYNCH symbol is triggered in this state
LINMSTR_TX_ID	Wait for successful transmission of SYNCH symbol, when this happens, transmission of message ID is triggered in this state

LINMSTR_END_OF_HEADER	Wait for successful transmission of message ID, once this happens two main things may occur: - If the Master node (this node) is the publisher of the data for this message then this state triggers the transmission of the first DATA byte. If message data size is 0, then triggers transmission of CHECKSUM byte. - If a Slave node is the publisher of the data for this message then this state prepares the state machine for DATA bytes reception. If message data size is 0, then the following expected byte is the CHECKSUM byte.
LINMSTR_TX_DATA	Wait for successful transmission of the first DATA byte, if more DATA bytes are required for transmission then this state transmits all the remaining DATA bytes. After transmission of the last DATA byte then the CHECKSUM byte is triggered for transmission.
LINMSTR_TX_CHKSUM	Wait for successful transmission of the CHECKSUM byte, when this happens, this state flags the correct transmission of the LIN frame (via a flag and a callback call)
LINMSTR_RX_DATA	Wait for successful reception of DATA bytes from the publishing slave. After reception of the last DATA byte then the state is changed in order to receive the CHECKSUM byte.
LINMSTR_RX_CHKSUM	Wait for successful reception of the CHECKSUM byte, when this happens, this state compares the received CHECKSUM byte agains a calculated checksum. If both match then this state flags the correct reception of the LIN frame (via a flag and a callback call) and copies the data to the global LIN messages buffer, if a mismatch occurs, all received data is ignored and an error flagged (via a callback call).

3.2. Conditions and Events

Event Name	Description
LINMSTR_EV_TX_START	Indicates the beginning of a LIN frame transmission. It is called by the scheduling table algorithm.
LINMSTR_EV_TX_OK	Indicates the successful transmission of a byte via the UART hardware.
LINMSTR_EV_RX_OK	Indicates the successful reception of a byte via the UART hardware.
LINMSTR_EV_TIMEOUT	Indicates that the state machine has not finished appropriately the transimission or reception of a LIN frame after certain time.
LINMSTR_EV_ERROR	Indicates an error by for example: UART error in reception or transmission of bytes.

3.3. Functions/Actions

Formal Name (Name in the state machine diagram)	Description
COPY_MSG_DATA_TO_INTERNAL_BUFFER()	If the Master node is the publisher of the LIN message to be transmitted then copy the message data from the global LIN buffer (`raub_LINmsgsDataBuffer`) to the local unified buffer inside the state machine

	(`laub_linUnifiedRxTxBuffer`). The direction of a LIN message (`msgDir`) is set to TX when the Master node is the publisher of such message, otherwise, the direction is set to RX.
COPY_INTERNAL_BUFFER_TO_MSG_DATA()	Copies data from the local unified buffer (`laub_linUnifiedRxTxBuffer`) to the global LIN buffer (`raub_LINmsgsDataBuffer`) for the message being received.
ID_PARITY_OK	Returns TRUE if the parity bits of the LIN message ID symbol is correct, otherwise, returns FALSE.
SEND_BREAK_SYMBOL()	Triggers the transmission of the LIN BRAKE symbol (start bit + 13 bit times with value of 0 + stop bit)
SEND_SYNCH_SYMBOL()	Triggers the transmission of the LIN SYNCH symbol (character 0x55).
SEND_ID_SYMBOL()	Triggers the transmission of the LIN message ID (LIN protected identifier).
SEND_DATA_BYTE(X)	Triggers the transmission of LIN message data byte X where X=1,2,…,n with n=LIN message data length (`msgDataLength`).
GET_DATA_BYTE(X)	Get the data byte X of the LIN message being processed. Where X=1,2,…,n with n=LIN message data length (`msgDataLength`).
CALCULATE_CHECKSUM()	Returns the calculated checksum based on the LIN message ID and the transmitted data when the Master node is the publisher of the data, or the received data otherwise.
SEND_CHECKSUM()	Triggers the transmission of the calculated checksum byte (calculated with function CALCULATE_CHECKSUM)
GET_CHECKSUM_BYTE()	Get the checksum byte of the LIN message being processed.
CHEKSUM_MATCH()	Compares the calculated checksum byte (with function CALCULATE_CHECKSUM) against the received checksum byte (with function GET_CHECKSUM_BYTE), if both are equal then returns “1” (TRUE) otherwise returns “0” (FALSE).
SET_MSG_TRANSMISSION_FLAG()	Each LIN message defined in the network has an associated flag in the LIN master core functionality. This function writes “1” (TRUE) to the flag associated with the LIN message and indicates that the message has been correctly transmitted.
SET_MSG_RECEPTION_FLAG()	Each LIN message defined in the network has an associated flag in the LIN master core functionality. This function writes “1” (TRUE) to the flag associated with the LIN message and indicates that the message has been correctly received.
CALL_MSG_RECEPTION_CALLBACK()	Each message being received by the LIN node has its own associated callback function which is called each time such LIN message has been received correctly.
CALL_MSG_TRANSMISSION_CALLBACK()	Each message being transmitted by the LIN node has its own associated callback function which is called each time such LIN message has been transmitted correctly.

CALL_CHKSUM_ERROR_CALLBACK()	This function is called each time a mismatch between the calculated checksum and the received checksum is detected.
CALL_TIMEOUT_ERROR_CALLBACK()	This function is called each time a timeout error has occurred. This timeout indicates that a expected response has not been received on time.
CALL_ID_PARITY_ERROR_CALLBACK()	This function is called each time a LIN msg ID is scheduled for transmission but the parity bits of this ID are wrong.
DISABLE_TX_INTERRUPTS()	Disables transmission interrupts from the UART HW.
DISABLE_RX_INTERRUPTS()	Disables reception interrupts from the UART HW.
ENABLE_TX_INTERRUPTS()	Enables transmission interrupts from the UART HW.
ENABLE_RX_INTERRUPTS()	Enables reception interrupts from the UART HW.

4. Kinetis board PIN routing

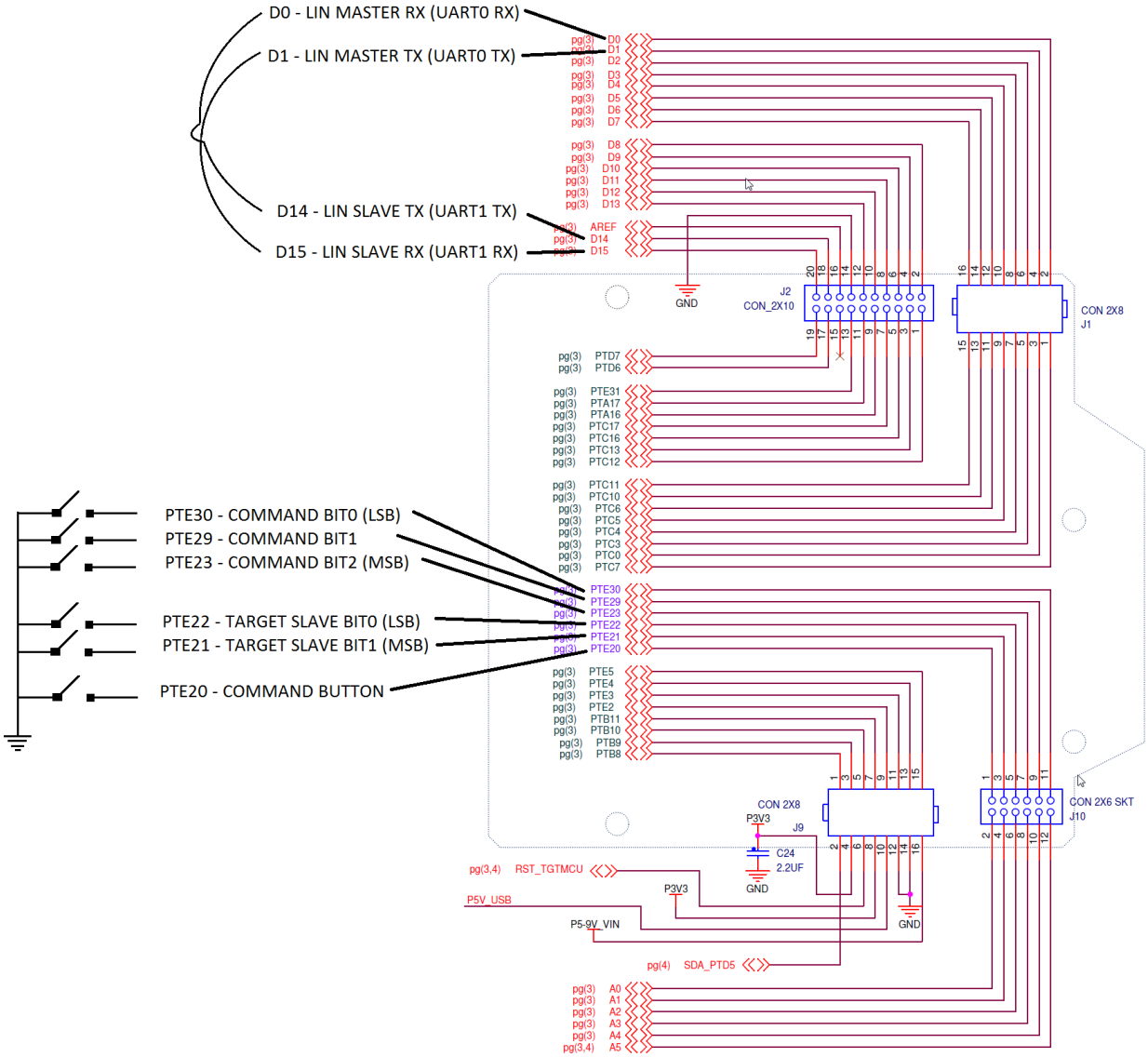

Figure 3: HW I/O mapping