

Interrupt Driven EUSART Based LIN-Master Library Module

1. Introduction.....	2
2. Module Features	2
3. List of Component Modules	3
4. Using the Library Module in a Project	3
5. List of Shared Parameters	4
<i>Shared Macros</i>	4
6. Macros	5
8. Error and Status Flags	9

1. Introduction

The goal of the EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) Based Interrupt Driven LIN (Local Interconnect Network) Master Module is to supply the designer with a library containing all necessary code to quickly develop an application running on top of a LIN-Master Protocol.

Designed to encapsulate implementation details and having in mind future upgrades, it contains several number of macros that Initialize, Read, Write and Check the status of messages transferred in a LIN network.

The library is based on Interrupt-driven reception/transmission buffer that provides the benefit of parallel processing. The C code is both linkable and relocatable, allowing easy integration into existing application code.

2. Module Features

- Supports user-defined baud rate, maximum message size, error detection and sleep/wake-up bus control.
- Incorporates interrupt-driven transmission and reception, allowing users to run other tasks in the foreground.
- Simple macro based Interface, which enables easy read, write and communications management.
- Supports for PIC18 family devices containing EUSART module.

3. List of Component Modules

ELINMInt.ex.txt	This is main test file developed to demonstrate the usage of the library functions.
ELINMInt.C	LIN C-code source file. Contains the variables and functions. <u>One needs include this file in their project.</u>
ELINMInt.H	This file contains the prototype of the functions, definition of macros, unions, structures, constants and definition of external references. <u>One needs include this file in all source files of it's project who access LIN.</u>
ELINMInt.def	This file contains the set-up definition of all parameters necessary to customize the driver.

4. Using the Library Module in a Project

Please follow below steps to use this library module in your project.

1. Use Application Maestro to configure your code as required.
2. At the Generate Files step, save the output to the directory where your code source files reside.
3. Launch MPLAB, and open the project's workspace.
4. Verify that the Microchip language tool suite is selected (*Project>Select Language Toolsuite*).
5. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select ELINMInt.C and click **OK**.
6. Now right-click on the "Linker Scripts" node and select "Add Files". Add the appropriate linker file (.lkr) for the project's target microcontroller.
7. Add any other files that the project may require. Save and close the project.
8. In the source files (C) that interact with the module add an include directive at the head of the code listing to include ELINMInt.H. By doing so, all files required to make the generated code work in your project will be included by reference when you build the project.
9. To use the module in your application, invoke the macros as needed.

5. List of Shared Parameters

Shared Macros

Usage	Name	Brief Description
Initialization	mELINMIntInitialize	Initializes EUSART and Driver's variables
Transmission	mELINMIntTXBufferAvailable	Checks for available transmission buffers
Transmission	mELINMIntGetTXPointer	Sets a tag and returns a buffer data pointer
Transmission	mELINMIntSendMessage	Requests the transmission of a message
Transmission	mELINMIntTXStatus	Return the status of a sent message
Transmission	mELINMIntMessageSent	Checks for end of a message transmission
Transmission	mELINMIntTXErrorDetected	Checks for transmission errors
Transmission	mELINMIntTXErrorTag	Returns the tag of a message with error
Transmission	mELINMIntTXErrorCode	Returns the error code of a transmission
Reception	mELINMIntRXBufferAvailable	Checks for available reception buffers
Reception	mELINMIntReceiveMessage(tag,i,s)	Requests a slave to send a message
Reception	mELINMIntMessageReceived	Checks for end of reception process
Reception	mELINMIntGetMessageTag	Returns the tag of a received message
Reception	mELINMIntGetRXPointer	Returns a pointer to a received message
Reception	mELINMIntRXMessageAvailable	Checks for any message arrival
Reception	mELINMIntRXStatus	Return the status of a received message
Reception	mELINMIntRXErrorDetected	Checks for errors in reception
Reception	mELINMIntRXErrorTag	Returns the error tag
Reception	mELINMIntRXErrorCode	Returns the error code of a reception
Bus Control	mELINMIntCheckWakeUPReceived	Signals that a slave issued a wake-up
Bus Control	mELINMIntSendWakeUPSignal	Sends a wake-up signal to the slaves
Bus Control	mELINMIntSleepTimeOut	Signals when bus IDLE time exceeded sleep time-out

6. Macros

Macro	mELINMIntInitialize()
Overview	This macro initializes the driver -> EUSART and LIN Driver Variables
Input	None
Output	0=Initialization OK Non-Zero=Initialization Error Code
Side Effects	
Stack Requirement	
Macro	mELINMIntTXBufferAvailable()
Overview	This macro checks if there is a transmission buffer available. The application must call this macro before trying to initiate any transmission
Input	None
Output	1=There is an available buffer to be used to transmit a message 0=No buffer is currently available for transmission
Side Effects	
Stack Requirement	
Macro	mELINMIntGetTXPointer(tag)
Overview	This macro returns a pointer to the available transmission buffer.
Input	tag – The tag of a message. This is an identification of the message to be saved and used by the LIN protocol to inform the application of an eventual error that the transmission of a specific message suffered. In the event of an error being detected the user can access the tag of the message with error and with this tag read the error code.
Output	(BYTE *) - A byte type data pointer to the transmission buffer. The application shall load the message to be transmitted using this pointer.
Side Effects	
Stack Requirement	
Macro	mELINMIntSendMessage(tag,i,s)
Overview	This macro requests the transmission of a message through LIN
Input	tag – The tag that identifies a message, previously defined by the application when calling the mELINMIntGetTXPointer macro. i – The ID of the message, ranging from 0x00 to 0x3F. Bits 6 and 7 of the ID will be filled with parity bits and their original content ignored. s – The size of the message, limited to 8 for all standard messages and to ELINMINT_MAX_MESSAGE_SIZE in the case of an Extended Frame (ID=0x3E or ID=0x3F)
Output	None
Side Effects	
Stack Requirement	
Macro	mELINMIntTXStatus(tag)
Overview	This macro checks the status of a message already transmitted
Input	tag – This byte contains a message tag, which is an identification of the message that was sent
Output	Error Code according to the error table
Side Effects	
Stack Requirement	

Macro	mELINMIntMessageSent(tag)
Overview	This macro checks if a message defined by tag was already sent.
Input	tag – This byte contains a message tag, which is an identification of the message and through it the driver can track a specific message
Output	1 = Message Already Sent 0 = Message yet NOT sent
Side Effects	
Stack Requirement	

Macro	mELINMIntTXErrorDetected()
Overview	This macro flags if an error was detected in the transmission of a message.
Input	None
Output	1 = Error Detected 0 = No Error Detected
Side Effects	
Stack Requirement	

Macro	mELINMIntTXErrorTag()
Overview	This macro returns the tag of the message that presented an error
Input	None
Output	A byte with the tag of the message with error
Side Effects	
Stack Requirement	

Macro	mELINMIntTXErrorCode(tag)
Overview	This macro returns in one byte the error associated with a message tag – the identification of the message where an error was detected
Input	
Output	The error code, defined according to the erro table – See Error and Status Flags
Side Effects	
Stack Requirement	

Macro	mELINMIntRXBufferAvailable()
Overview	This macro flags if there is a reception buffer available.
Input	None
Output	0 = No buffer available 1 = Buffer Available
Side Effects	
Stack Requirement	

Macro	mELINMIntReceiveMessage(tag)
Overview	This macro requests a message to be sent from a slave.
Input	tag – The tag defined by the application to be associated with the incoming message requested i – The ID of the requested message s – The size of the message in bytes
Output	None
Side Effects	
Stack Requirement	

Macro	mELINMIntMessageReceived(tag)
Overview	This macro checks if a message was received
Input	tag – The tag defined by the application to be associated with the incoming message requested
Output	1=Message Received 0=Message not Received yet
Side Effects	
Stack Requirement	

Macro	mELINMIntGetMessageTag()
Overview	This macro returns the tag (identification) of a message that was received.
Input	None
Output	The tag (identification) of the received message.
Side Effects	
Stack Requirement	

Macro	mELINMIntGetRXPointer(tag)
Overview	This macro returns a pointer to the received message
Input	tag – The tag identifying the message .
Output	(BYTE *) - A byte type data pointer to the reception buffer. The application can read the message using this pointer
Side Effects	
Stack Requirement	

Macro	mELINMIntRXMessageAvailable()
Overview	This macro checks for the reception of a message
Input	None
Output	1 = Message Received 0 = No Message received
Side Effects	
Stack Requirement	

Macro	mELINMIntRXStatus(tag)
Overview	This macro checks the status of a received message
Input	tag – the identification of the message
Output	The error code, defined according to the error table
Side Effects	
Stack Requirement	

Macro	mELINMIntRXErrorDetected()
Overview	This macro checks for reception errors
Input	None
Output	1 = Error Detect 0 = No Error Detect
Side Effects	
Stack Requirement	

Macro	mELINMIntRXErrorTag()
-------	-----------------------

Overview	This macro returns the tag of the message that presented an error
Input	None
Output	tag – The identification of the message that was received with error
Side Effects	
Stack Requirement	
Macro	mELINMIntRXErrorCode(tag)
Overview	This macro returns the code identifying the error detected in the reception of the message identified by tag
Input	tag – The identification of the message received with error
Output	The error code, defined according to the error table
Side Effects	
Stack Requirement	
Macro	mELINMIntCheckWakeUPReceived()
Overview	This macro flags the reception of a Wake-Up signal from the slave
Input	None
Output	1=Wake-Up Received 0=NO Wake-Up Received
Side Effects	
Stack Requirement	
Macro	mELINMIntSendWakeUPSignal()
Overview	This macro sends a Wake-Up signal to the slaves and resets the sleep timeout condition
Input	None
Output	None
Side Effects	
Stack Requirement	
Macro	mELINMIntSleepTimeOut()
Overview	This macro detects the Time-Out of the Bus
Input	None
Output	1=Time-Out 0=NO Time-Out
Side Effects	
Stack Requirement	

7. Error and Status Flags

The LIN driver may signal the following errors after transmissions or receptions:

#define	Definition
ELINMINT_NO_ERROR	No error was detected
ELINMINT_THMIN_ERROR	Header Time Too Short
ELINMINT_THMAX_ERROR	Header Time Too Long
ELINMINT_TFMIN_ERROR	Frame Time Too Short
ELINMINT_TFMAX_ERROR	Frame Time Too Long
ELINMINT_CHECKSUM_ERROR	Checksum Incorrect
ELINMINT_DATA_ERROR	Received and Transmitted Bytes Don't Match
ELINMINT_FRAMING_ERROR	Framing Error

The mELINMIntInitialize macro may return the following errors:

NOT DEFINED IN VERSION 1.0