

**Linkmaxx-embedded™**  
**PROGRAMMER'S GUIDE**

**Version 2.1**

November 2010



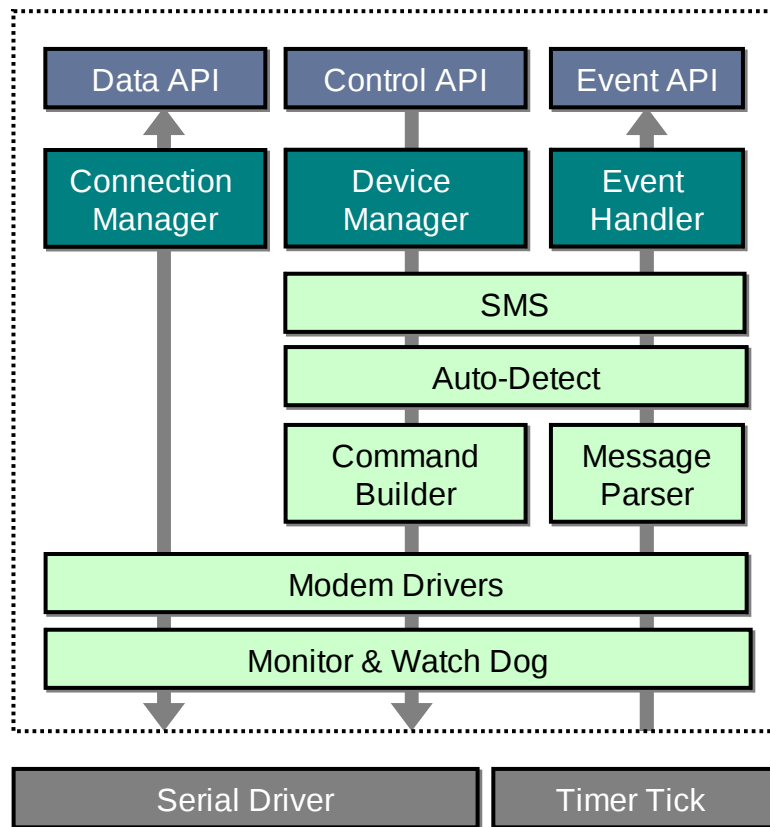
# 1. Overview

Linkmaxx-embedded™ (LME) is a software library designed to handle all communications functions for “AT command” based wireless or wired modems. LME provides a high-level ANSI C interface (API) that replaces the need for AT commands.

LME is provided as an obfuscated ANSI C source code library that is compiled with the embedded application. LME is designed to be always backwards compatible; new functionality is added through ‘extended’ functions, preserving the existing API.

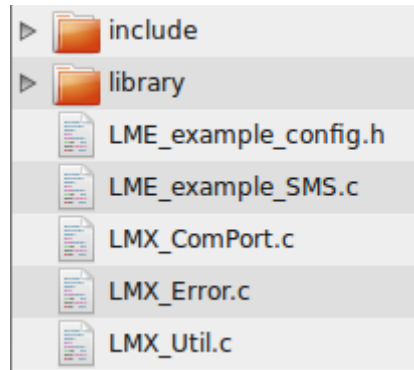
The LME API is ANSI C compatible (C89 or later; ISO/IEC 9899:1999 or later).

LME basic structure is shown below. A serial driver and a timer tick are required for LME to operate.



## 2. Installation

Copy (un-zip if needed) the LME library files to a directory of your choice. LME comes packaged with three sets of files:



### Header Files

<b>include</b>	Directory of header files. Include these files in your project; they are needed by LME
LM_API.h	Primary header file. Include this file in your code.
LM_Config.h	Configuration settings for LME. In this file you can enable/disable modem drivers, code modules, and debug reporting. <i>Note: this is the only file that can be edited, and only the values of the #defines</i>
LM_Connection.h	Data connection functions
LM_Device.h	Modem device functions
LM_Enums.h	Enums used by LME
LM_Service.h	LME utility service functions
LM_State.h	Modem state functions
LM_Structs.h	LME structures
LM_Typedefs.h	LME typedefs, including basic numeric types
LM_Util.h	Utility functions
LMX_Callback.h	Header for external / user supplied call-back functions
LMX_ComPort.h	Header for external / user supplied communication port functions
LMX_Error.h	Header for external / user supplied error reporting functions
LMX_Util.h	Header for external / user supplied misc. util functions

*Note: none of the header files should be edited as the LME library functions depend on them.*

## ***Library Files***

The core LME library consists of an obfuscated source file and two associated header files. Simply include these files as part of your project.

## ***Example Files***

These files can be used for reference, learning or they can be used as part of your project. Any part of these files can be changed, edited or deleted as you wish.

## **3. Run-time Requirements**

The LME code is not designed to be reentrant to ensure proper operation of the modem. It must therefore be run in a single task in a multitasking environment. For the same reason, it should not be invoked from interrupts of any kind. LME comes with example code for porting the code to a number of commonly used operating systems. LME can be used without an operating system, as well. There is also example code available that shows how to port LME to such an environment.

LME is event and callback driven. It will generate events to alert the user of device specific events and make use of callbacks if a requested operation cannot complete immediately. LME still provides mechanisms for polling for device events, in case the host environment does not lend itself well to the use of events.

The LME code requires the host environment to provide a “heartbeat” (by calling `LM_Process()`). This is necessary in order to provide internal timers. The heartbeat provided can be configured, but should not be longer than 100 milliseconds and the accuracy should not exceed 10%. So, a 50 millisecond heart beat should not be less than 45 milliseconds nor should it be greater than 55 milliseconds. If using a hardware timer to generate the heartbeat, remember not to call LME directly from the interrupt handler.

The LME code makes use of dynamically allocated memory. Hence, the host environment must set aside memory for a heap, or add additional space to the existing heap.

## 4. Theory of Operation

LME takes full control of the modem. All access to the modem is ONLY through LME API function calls.

### Code Requirements

LME is designed to be included in, and compiled with an embedded application. LME requires that the `LMX_` functions are implemented to support LME. The most important `LMX_` functions are the serial driver functions and `LM_Process()`. Look in `LMX_ComPort.c` or `LMX_ComPort.h` for the function prototype. `LM_Process()` needs to be called approximately every 100 mS (milliseconds) so that LME's timers will operate properly.

### Automatic Modem Watchdog

LME contains a 'watchdog' function that monitors all registered modem devices for responsiveness and operation. The automatic watchdog checks a modem regularly to ensure that it is operating properly. If a device fails to operate properly, LME will attempt to restart the device automatically.

### Modems Supported

LME supports a large and growing number of modem devices (wireless or wired) that communicates using Hayes style AT commands.

### Multiple *Simultaneous* Modem support

LME supports any number of connected modem devices, all of which can operate simultaneously. (up to a max. of 255 simultaneously connected modems)

## APPENDIX A - Modem State Machine

Modem State Machine

