Product Specifications
LoRa Wireless Communication Module

LM-130EVB
LoRaWAN Compliant Evaluation Board

VER: 1.5

GlobalSat WorldCom Corporation

16F., No. 186, Jian 1st Rd, Zhonghe Dist.,

New Taipei City 23553, Taiwan

Tel: 886.2.8226.3799/ Fax: 886.2.8226.3899
service@globalsat.com.tw
www.globalsat.com.tw

USGlobalSat Incorporated

14740 Yorba Court Chino, CA 91710

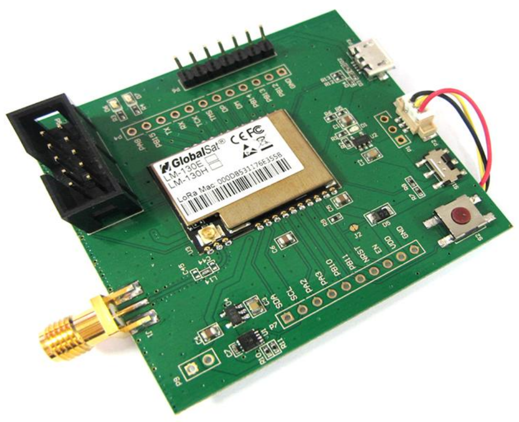
Tel: 888.323.8720 / Fax: 909.597.8532
sales@usglobalsat.com
www.usglobalsat.com

General Information

The LM-130EVB LoRa® Node is a LoRaWAN™ Class A end-node device based on the GlobalSat LM-130 module which is a LoRaWAN™ certified module. The LM-130EVB is a standalone battery powered node with power management, it includes humidity and temperature sensors to generate data, which are transmitted either on a regular schedule (can be configure) or initiated by a button-press. This node provides a convenient platform to quickly demonstrate the long-range and low power consumption capabilities of the modem, as well as interoperability when connected to LoRaWAN™ v1.0 compliant gateways and the infrastructure. The LM-130EVB also provides a standard USB interface for connection to a host computer, providing a bridge to the UART interface of the LM-130 module. As with all LoRaWan^{TM} compliant of products, it can also help developers to develop the applications rapidly, including hardware and software design by using high level ASCII command to control the protocol, before the end product is ready.

The LM-130EVB Contents:

1. LM-130EVB
2. RPSMA antenna
3. UART to USB wire connector



Features

- LoRaWAN^{TM} Compliant Evaluation Board
- RF ISM band, supports 868/ 915 MHz
- Long range transmission (1km to 10km)
- 2-way duplex communication
- Push button (trigger event)
- Configurable payload
- Configurable report interval
- Active report mode
- Built-in humidity/ temperature sensor
- High penetration and strong anti-interference capability

Hardware Specifications

Item	Parameters
LoRa Module	GlobalSat LM-130 LoRaWAN^{TM} module
RF Band	868 MHz / 915 MHz
RF Output Power	Max. 20dBm
Receiving Sensitivity	-132dBm @ 980bps
Dimensions	71 x 55 x 15 mm (not including antenna)
Interface	UART
Battery	Re-chargeable Li-polymer battery 820mAh Over current protection
Sensor	Temperature/ Humidity SHT3x-DIS
LED Indicator	Power on: Green LED on. Power off: Green LED off. Charging: Red LED on. Full battery: Red LED off. Data transmitting: Blue LED blinking with 0.2 seconds interval change. End data transmitting: Blue LED off.
Operation Conditions	Temperature -20℃ ~ 60℃; Humidity 5% ~ 95%
Micro USB	Charging @ 500mA

Item	Parameters
JTAG	For updating boot loader (internal use)
Button	Power switch : On/ Off Push button: Send report

Firmware Behavior

Active report mode: On (Default)

LM-130EVB reports default data to Gateway **by interval automatically**. The Active report mode default set to “On”.

- Configurable report interval. (default = 10 secs)
- Default payload includes “GS130”, battery voltage, temperature and humidity sensor data. Payload is **not** modifiable when active report mode is on.
- The push button is disabled when active report mode set to on.

Active report mode: Off

LM-130EVB will send report with default data to Gateway once the **push button is pressed**.

- Default Payload includes “GS130”, battery voltage, temperature and humidity sensor data.
- To send self-define payload, please refer “AAT2 Tx= [parameter1],[parameter2],[parameter3]“ AT command to send self-define payload.

Configuration

Activation of an end-device can be achieved in two ways, either via “Over-The-Air Activation (OTAA)” when an end-device is deployed or reset, or via “Activation By Personalization (ABP)” in which the two steps of end-device personalization and activation are done as one step.

■ Over-the-Air Activation

For over-the-air activation, end-devices must follow a join procedure prior to participating in data exchanges with the network server. An end-device has to go through a new join procedure every time it has lost the session context information.
The join procedure requires the end-device to be personalized with the following information before its starts the join procedure: a globally unique end-device identifier (DevEUI), the application identifier (AppEUI), and an AES-128 key (AppKey).

■ Activation by Personalization

Under certain circumstances, end-devices can be activated by personalization. Activation by personalization directly ties an end-device to a specific network by-passing the join request join accept procedure.
Activating an end-device by personalization means that the DevAddr and the two session keys NwkSKey and AppSKey are directly stored into the end-device instead of the DevEUI, AppEUI and AppKey. The end-device is equipped with the required information for participating in a specific LoRa network when started. Each device should have a unique set of NwkSKey and AppSKey. Compromising the keys **of one device shouldn‘t compromise the security of the communications of other devices.**

Operation Mode

Bi-directional end-devices (Class A): End-devices of Class A allow for bi-directional communications whereby each end-device's uplink transmission is followed by two short downlink receive windows. The transmission slot scheduled by the end-device is based on its own communication needs with a small variation based on a random time basis (ALOHA-type of protocol). This Class A operation is the lowest power end-device system for applications that only require downlink communication from the server shortly after the end-device has sent an uplink transmission. Downlink communications from the server at any other time will have to wait until the next scheduled uplink.

UART Interface

All of the LM-130 module's settings and commands are transmitted over UART using the ASCII interface. All commands need to be terminated with <CR><LF> and any replies they generate will also be terminated by the same sequence.

The settings for the UART interface are 57600 bps, 8 bits, no parity, 1 Stop bit, no flow control.

AT command

Command	Description
AAT1 UpdateFW	Upgrade the LM-130 module firmware. Response ***ok*** after entering the command.
AAT1 Save	All parameters are saved. Response ***ok*** after parameters are saved.
AAT1 FwVersion	Show up firmware version.
AAT1 Reset	Resets and restarts the LM-130 module. Response ***ok*** after entering the command.
AAT1 SLEEP	Put LM-130 into sleep mode. Input 0xFF by UART to wake up LM-130 to leave sleep mode. Response ***ok*** after entering the command.
AAT1 Restore	Restore the defaults of FW. Response ***ok*** after entering the command.
AAT1 TestMode=[parameter1]	[parameter1]: 0: Disable **(Active Report Mode: Off)** 1: Enable **(Active Report Mode: On)** Response: ***ok*** if value is valid ***invalid_param*** if parameter1 is not valid **This command sets the state of the active report mode for the module.**
AAT1 TestMode=?	Response: 0: disable (Active Report Mode: Off) 1: enable (Active Report Mode: On) **This command will return the state of the active report mode.**

Command	Description
AAT2 DevAddr=[parameter1]	[parameter1]: 4-byte hexadecimal number representing the device address, from 00000000 – FFFFFFFF. Response: ***ok*** if address is valid ***invalid_param*** if parameter1 is not valid This command configures the module with a 4-byte unique network device address [parameter1]. The [parameter1] must be unique to the current network. This must be directly set solely for activation by personalization devices. This parameter must not be set before attempting to join using over-the-air activation because it will be overwritten once the join process is over.
AAT2 DevAddr=?	Response: 4-byte hexadecimal number representing the device address, from 00000000 to FFFFFFFF. This command will return present end-device address of the module.
AAT2 DevEui=[parameter1]	[parameter1]: 8-byte hexadecimal number representing the device EUI. Response: ***ok*** if address is valid ***invalid_param*** if parameter1 is not valid This command sets the globally unique device identifier for the module. The identifier must be set by the host MCU. The module contains a pre-programmed unique EUI and can be retrieved using user provided EUI can be configured using the AAT2 DevEui command.
AAT2 DevEui=?	Response: 8-byte hexadecimal number representing the device EUI. This command returns the globally unique end-device identifier, as set in the module.
AAT2 AppEui=[parameter1]	[parameter1]: 8-byte hexadecimal number representing the application EUI. Response: ***ok*** if address is valid ***invalid_param*** if parameter1 is not valid This command sets the application identifier for the module.
AAT2 AppEui=?	Response: 8-byte hexadecimal number representing the application EUI.

Command	Description
	This command will return the application identifier for the module. The application identifier is a value given to the device by the network.
AAT2 NwkSKey=[parameter1]	[parameter1]: 16-byte hexadecimal number representing the network session key. Response: ***ok*** if address is valid. ***invalid_param*** if parameter1 is not valid. This command sets the network session key for the module. This key is 16 bytes in length, and should be modified with each session between the module and network. The key should remain the same until the communication session between devices is terminated.
AAT2 NwkSKey=?	Response: [parameter1]: 16-byte hexadecimal number representing the network session key. This command sets the network session key for the module.
AAT2 AppSKey=[parameter1]	[parameter1]: 16-byte hexadecimal number representing the application session key. Response: ***ok*** if address is valid. ***invalid_param*** if parameter1 is not valid. This command sets the application session key for the module. This key is unique, created for each occurrence of communication, when the network requests an action taken by the application.
AAT2 AppSKey=?	Response: [parameter1]: 16-byte hexadecimal number representing the application session key. This command sets the application session key for the module.
AAT2 AppKey=[parameter1]	[parameter1]: 16-byte hexadecimal number representing the application key. Response: ***ok*** if address is valid ***invalid_param*** if parameter1 is not valid This command sets the application key for the module. The application

Command	Description
	key is used to identify a grouping over module units which perform the same or similar task.
AAT2 AppKey=?	Response: [parameter1]: 16-byte hexadecimal number representing the application key. This command sets the application key for the module.
AAT2 ADR=[parameter1]	[parameter1]: 0: disable 1: enable Response: ***ok*** if address is valid. ***invalid_param*** if parameter1 is not valid. This command sets if the adaptive data rate (ADR) is to be enabled, or disabled. The server is informed about the status of the module's ADR in every uplink frame it receives from the ADR field in uplink data packet. If ADR is enabled, the server will optimize the data rate and the transmission power of the module based on the information collected from the network.
AAT2 ADR=?	Response: 0: disable 1: enable This command will return the state of the adaptive data rate mechanism.
AAT1 EVK_TxCycle=[parameter1]	[parameter1]: decimal number representing the report interval in seconds, from 1 to 254. This command will only take effect when "TestMode"=1. Response: ***ok*** if parameter1 is valid ***invalid_param*** if parameter1 is not valid This command sets the report interval for the module.
AAT2 JoinMode=[parameter1]	[parameter1]: 0: ABP mode 1: OTAA mode Response:

Command	Description
	ok if address is valid. ***invalid_param*** if parameter1 is not valid. This command informs the ***module activation type.***
AAT2 JoinMode=?	Response: 0: ABP mode 1: OTAA mode This command will return the ***activation type*** of module.
AAT2 reTx=[parameter1]	[parameter1]: decimal number representing the number of retransmissions for an uplink confirmed packet, from 0 to 10. Response: ***ok*** if address is valid. ***invalid_param*** if parameter1 is not valid. This command sets the number of retransmissions to be used for an uplink confirmed packet, if no downlink acknowledgment is received from the server.
AAT2 reTx=?	Response: decimal number representing the number of retransmissions, from 0 to 10. This command will return the currently configured number of retransmissions which are attempted for a confirmed uplink communication when no downlink response has been received.
AAT2 RxDelay1=[parameter1]	[parameter1]:decimal number representing the delay between the transmission and the first reception window in microseconds, from 100000 to 10000000. Response: ***ok*** if address is valid ***invalid_param*** if parameter1 is not valid This command will set the delay between the transmission and the first reception window to the [parameter1] in microseconds. The delay between the transmission and the second Reception window is calculated in software as the delay between the transmission and the first Reception window + 1000000 (us).
AAT2 RxDelay1=?	Response: decimal number representing the interval, in milliseconds, for

Command	Description
	rxdelay1. This command will return the interval, in microseconds, for rxdelay1.
AAT2 Tx=[parameter1], [parameter2], [parameter3]	[parameter1]: decimal number representing the port number, from ***1*** to ***223***. [parameter2]: string representing the uplink payload type, either ***cnf*** or ***uncnf*** (cnf-confirmed, uncnf-unconfirmed) [parameter3]: hexadecimal value. The length of [parameter3] bytes capable of being transmitted are dependent upon the set data rate (please refer to the LoRaWAN^{TM} Specification for further details). Response: this command may reply with two responses. The first response will be received immediately is valid (ok reply received), a second reply will be received after the end of the uplink transmission. Please refer to the the LoRaWAN^{TM} Specification for further details. Response after entering the command: ● ok - If parameters and configurations are valid. ● Invalid_param – if parameters ([parameter1],[parameter2],[parameter3]) are not valid. ● Tx_ok - if uncnf radio tx return. ● Tx_noACK - if cnf radio tx return without ack. ● Tx_ok - if cnf radio tx return with ack ● Rx < parameter1> < parameter2>– if transmission was successful, [parameter1] port number, from 1 to 223; [parameter2] hexadecimal value that was received from the server.