NPLink Mote SDK
Developer Manual
开发手册

NPLink Mote SDK
Developer Manual

日期	版本号	说明
2015.11.03	V0.0.1	创建文档
2015.12.20	V0.0.2	根据最新的 SDK 修改文档
2016.01.30	V0.0.3	增加 ADR 配置参数、增加发送失败消息
2016.03.26	V0.0.4	增加低功耗设置、工作模式设置 API；增加 MAC 的参数设置。

一、概述

本文档适用于使用 NPLink-Mote-SDK 进行应用开发的研发人员，以及 NPLink 的测试人员。

相关术语：

API	Application Programming Interface
OSAL	Operating System (OS) Abstraction Layer
LoRaWAN	Long Range Wireless Area Network
MAC	Medium Access Control
OTAA	Over-the-air Activation
ABP	Activation by Personalization
RO	Read Only
RW	Read and Write

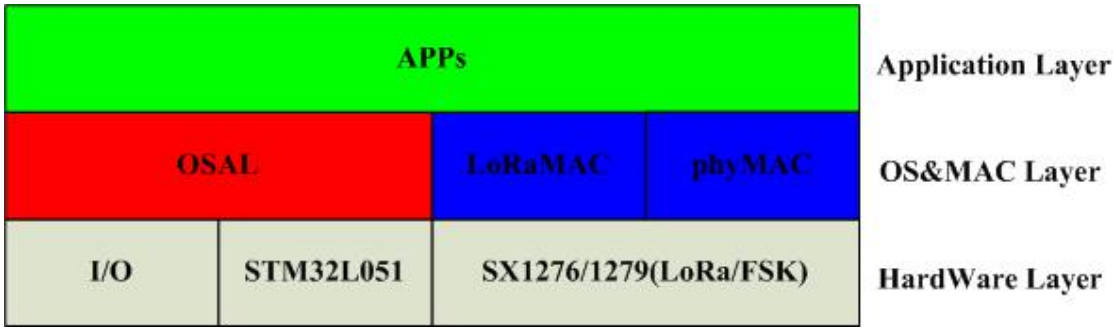
二、代码框架

NPLINK-Mote-SDK 的整体代码体系结构如图 2.1 所示，整体可分为 3 层，分别为：

第 1 层：硬件层，主要包括外设 IO 的驱动、STM32L051 的驱动库文件、以及通信芯片 SX1276/79 的驱动。

第 2 层：OSAL 及 MAC 层，实现了 OSAL 的管理及 MAC 的核心代码，LoRa MAC 以 lib 的形式提供服务。

第 3 层：应用层，包括了自带的 APP task 及用户可自定义的业务逻辑 task。



（图 2.1 NPLINK-Mote-SDK 的代码体系结构）

NPLINK-Mote-SDK 整个工程包含 5 个主模块，主模块下包含若干个功能函数，具体描述如下：

模块名	包含函数	作用及功能
startup	startup_stm32l051xx.s	设置初始堆栈指针（SP）； 初始程序计数器（PC）为复位向量，并在执行 main 函数前初始化系统时钟； 设置向量表入口为异常事件的入口地址；
hal/drivers	system_stm32l0xx.c	STM32l0xx 微控制器专用系统文件
	stm32l0xx_hal.c	STM32l0xx 芯片标准外设库驱动源文件
	stm32l0xx_hal_adc_ex.c	
	stm32l0xx_hal_cortex.c	
	stm32l0xx_hal_gpio.c	
	stm32l0xx_hal_pwr.c	
	stm32l0xx_hal_pwr_ex.c	
	stm32l0xx_hal_rcc.c	
	stm32l0xx_hal_rcc_ex.c	
	stm32l0xx_hal_spi.c	
	stm32l0xx_hal_tim.c	
	stm32l0xx_hal_tim_ex.c	

	stm32l0xx_hal_uart.c	
	stm32l0xx_hal_uart_ex.c	
	stm32l0xx_hal_usart.c	
	stm32l0xx_hal_dma.c	
	stm32l0xx_hal_dac.c	
hal/board	spi_board.c	**包含** SPI1 **通信功能的初始化、应用及读写的实现，主要用于** MCU **与** SX1276/9 **之间的通信**
	oled_board.c	**用于模块自带** oled **液晶显示屏的初始化以及字符串显示的实现**
	led_board.c	LED **驱动程序**
	key_board.c	**按键驱动程序**
	gpio_board.c.	IO **外部中断的分配及中断处理函数的实现，主要用于** MCU **与** SX **芯片的** IO **中断处理**
	rtc_board.c	STM32l0xx **的** RTC **实现，当系统以低功耗方式运行时，系统需要依赖** RTC **进行定时和唤醒。**
	uart_board.c	STM32l0xx **的串口** 1 **实现。**
	time-board.c	STM32l0xx **的** TIMER 2 **实现，当系统未以低功耗方式运行时，** MAC **依赖**

		TIMER 2 进行延时。
	sx1276-board.c	定义与 SX1276 相连接的 IO 并初始化相应的中断，同时对 SX1276 Radio、天线等属性做相应的初始化
	timer.c	实现 MAC 层使用的定时器，用于延时操作，属于逻辑定时器，硬件使用 RTC 或 TIMER 2 定时器实现。
	delay.c	一个简单的 ms 级延时实现
	board.c	目标板通用功能的实现，主要是对目标板电压进行检测。
	utilities.c	辅助函数的实现
hal/radio	sx1276.c	对 SX1276/9 芯片进行初始化、基本配置以及应用的、功能的实现
osal	osal.c	osal 操作标准函数的定义
	osal_memory.c	内存（堆）分配系统
	osal_mutex.c	Mutex 的创建及相应的操作
	osal_tick.c	SysTick_Configuration
	osal_timer.c	osal 定时器的相关操作，包含任务开启、轮寻、结束等操作 osal 定时器做出的相应处理

	osal_app.c	所有任务 ID 的分配及其任务的初始化
mac	LoraMac_osal.h	LoRaWAN MAC 任务头文件
	LoRaMacUsr.h	LoRaWAN MAC 用户接口头文件
	NPLink-Mote-Mac.lib	LoRa MAC 层实现的 lib 文件，是实现无线收发和协议的核心代码。
app	hal_osal.c	硬件抽象层任务，实现硬件的初始化及相关事件处理
	app_osal.c	应用层任务，可与 MAC 层实现数据的交互
	stm32l0xx_hal_it.c	STM32l0xx 芯片中断处理文件
	stm32l0xx_hal_msp.c	STM32l0xx 芯片外设驱动文件
	mian.c	主函数，工程入口点

三、快速使用指南

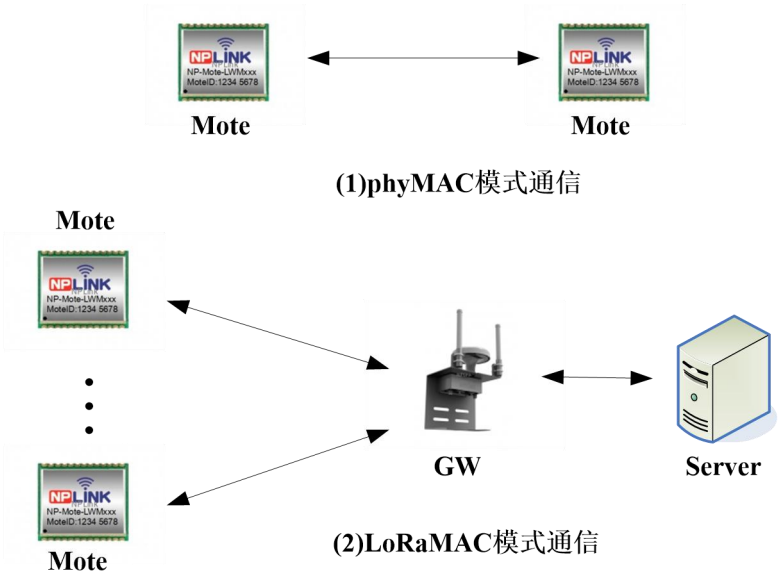
3.1 编译器宏定义

NPLink-Mote-SDK 使用了几个编译器宏来开启/关闭一些功能，见下表：

宏名	作用
USE_LOW_POWER_MODE	低功耗开启或关闭，开启后，HAL 中的串口、LED、KEY、OLED 等外设将不会初始化，并会启用 RTC 定时器来辅助休眠时长的定义。

USE_DEBUG	**串口调试信息开启或关闭，开启后，将会通过 Mote 的串口 1 输出辅助调试信息。**

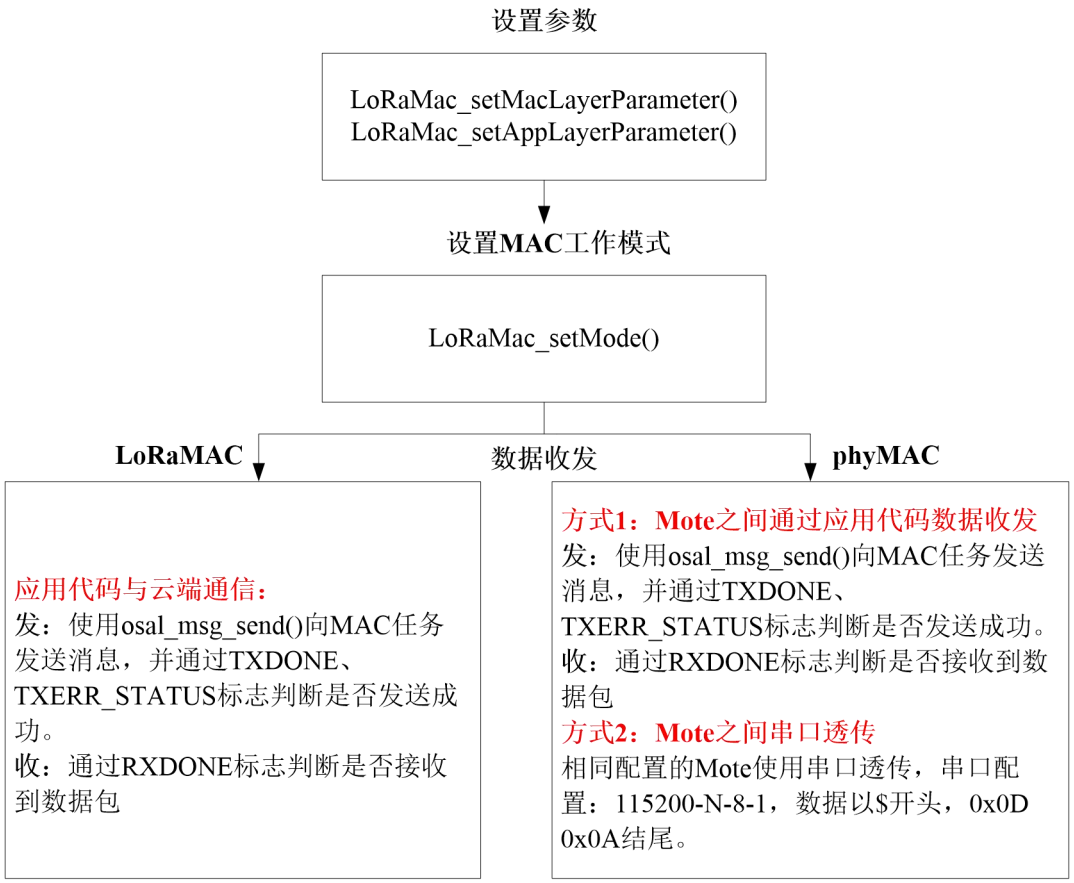
3.2 数据收发



（图 3.1　NPLink-Mote-SDK 的 2 种 MAC 通信模式）

NPLink-Mote-SDK 提供 2 种 MAC 通讯接入模式，即 LoRaWAN MAC（简称 LoRaMAC）和 phyMAC，phyMAC 实现 Mote 到 Mote 的双向通信；LoRaMAC 实现 Mote 到网关再到 Server 的上下行通信，见图 3.1 所示。在这 2 种 MAC 工作模式下，物理层均提供 2 种调制方式：低速率的 LoRa 调制方式和高速率的 FSK 调制方式。

为了实现数据收发，应该遵循下述基本步骤进行开发：



四、OSAL

关于 OSAL **的** API **接口，详见文档《**NPLink OSAL API manual**》的描述。**

简单的说，当用户需要定义一个任务时，需撰写实现 2 **个函数：初始化函数及事件处理函数，并将这** 2 **个函数添加到** OSAL **的相应位置即可(** osal_app.c **中)，示例如下：**

```
void osalInitTasks( void )
{
  u8 taskID = 0;
  osal_memset( tasksEvents, 0, (sizeof( u16 ) * tasksCnt) );
  HardWare_Init(taskID++);
  LoRaMAC_Init(taskID++);
  APP_Init(taskID++);
}
```

```
const pTaskEventHandlerFn tasksArr[] =
{
  HardWare_ProcessEvent,
  LoRaMAC_ProcessEvent,
  APP_ProcessEvent,
};
```

注意：添加新的 task **后，初始化函数及事件处理函数，在以上** 2 **处的位置应该严格对应。如** APP_Init **和** APP_ProcessEvent **均位于最后（即第** 3 **个），由于** HardWare **及** LoRaMAC **任务为基础任务，建议可将新的任务跟在它们后面。**

五、数据收发及参数 API

在 APP **任务中，可调用** MAC **的服务器进行参数的设定以及进行数据的收发，在数据收发之前，应进行相应的参数设定，如果未设定参数，则** MAC **层以默认参数运行。**

5.1 **参数设置及获取** API

NPLink-Mote-SDK **提供** APP **层面及** MAC **层相关的参数配置和获取** API**，用户可通过这些** API **函数方便地对工作参数进行设定。**

5.1.1 LoRaMac_setAppLayerParameter()

说明：此函数可设置一些 LoRaWAN MAC **的** APP **相关运行参数，支持的参数列表如表** 5.1 **所示。**

注意事项：无。

原型：u8 LoRaMac_setAppLayerParameter(void* pdata_in, u32 parameterIDs);

参数：

void* pdata_in　**指向参数的存储空间，应传入** LoRaMacAppPara_t **类型指针。**

u32 parameterIDs　**参数** ID **号，支持的** ID **号如表** 5.1 **所示，参数可采用“按位”方式传入，即可以同时设定多个参数，当同时设定多个参数时，将设定的** ID **号“按位或”方式传入即可。**

返回值：返回处理的状态，如表 5.2 **所示。**

表 5.1 NPLink-Mote LoRaWAN MAC **支持的** APP **参数** ID **号(仅** LoRaMAC **需要)**

参数名	**参数值**	**默认值**	**读/写**	**含义**
PARAMETER_DEV_ADDR	1 << 0		RO	NPLink Mote **的设备地址** DevAddr
PARAMETER_DEV_EUI	1 << 1	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00	RW	LoRaWAN DevEUI **值**
PARAMETER_APP_EUI	1 << 2	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00	RW	LoRaWAN AppEUI **值**
PARAMETER_APP_KEY	1 << 3	0x2B,0x7E,0x15,0x16,0x28,0xAE,0xD2,0xA6,0xAB,0xF7,0x15,0x88,0x09,0xCF,0x4F,0x3C	RW	LoRaWAN AppKey，**当使用** over-the-air activation **时使用。**
PARAMETER_NWK_SKEY	1 << 4	0x2B,0x7E,0x15,0x16,0x28,0xAE,0xD2,0	RW	LoRaWAN NwkSkey，**当** activation by

		xA6,0xAB,0xF7,0x15,0x88,0x09,0xCF,0x4F,0x3C		personalization **时使用。**
PARAMETER_APP_SKEY	1 << 5	0x2B,0x7E,0x15,0x16,0x28,0xAE,0xD2,0xA6,0xAB,0xF7,0x15,0x88,0x09,0xCF,0x4F,0x3C	RW	LoRaWAN AppSkey，**当** activation by personalization **时使用。**

注： 1）Mote **加入（激活）**LoRaWAN **网络有两种方式：**OTAA（over-the-air activation）、ABP（activation by personalization）。

2）**节点成功加入网络后，应该具有** 3 **个信息：设备地址（**DevAddr**）、应用** ID **号（**AppEUI**）、网络会话密钥（**NwkSkey**）、应用会话密钥（**AppSkey**）。**

3）**当采用** ABP **方式加入网络时，**DevAddr、NwkSkey、AppSkey **直接在** Mote **中配置生成。**

4）**当采用** OTAA **加入网络时，**Mote **使用** AppKey、AppEUI、DevEUI **三者信息构成请求包，通过命令向服务器请求** DevAddr、NwkSkey、AppSkey，**服务器计算出三者后下发给** Mote。

5）**当前只支持** ABP **方式加入。**

表 5.2 MAC **支持的操作状态**

状态	**值**	**含义**
LORAMAC_USR_SUCCESS	0	**操作成功**
LORAMAC_USR_INVALID_PARAMETER	1	**不支持的参数**

LORAMAC_USR_FAILURE	0xFF	**操作失败**

5.1.2 LoRaMac_getAppLayerParameter()

说明：此函数可获取一些 LoRaWAN MAC **的** APP **相关运行参数，支持的参数列表如表** 5.1 **所示。**

注意事项：无。

原 型 ： u8 LoRaMac_getAppLayerParameter(void* pdata_out, u32 parameterIDs);

参数：

void* pdata_out **指向参数的存储空间，应传入** LoRaMacAppPara_t **类型指针。**

u32 parameterIDs **参数** ID **号，支持的** ID **号如表** 5.1 **所示，参数可采用“按位”方式传入，即可以同时设定多个参数，当同时设定多个参数时，将设定的** ID **号“按位或”方式传入即可。**

返回值：返回处理的状态，如表 5.2 **所示。**

5.1.3 LoRaMac_setMacLayerParameter()

说明：此函数可设置一些 LoRaWAN MAC **的** MAC/PHY **相关运行参数，支持的参数列表如表** 5.3 **所示。**

注意事项：无。

原型：u8 LoRaMac_setMacLayerParameter(void* pdata_in, u32 parameterIDs);

参数：

void* pdata_in　**指向参数的存储空间，应传入** LoRaMacMacPara_t **类型指针。**

u32 parameterIDs　**参数** ID **号，支持的** ID **号如表** 5.3 **所示，参数可采用“按位”方式传入，即可以同时设定多个参数，当同时设定多个参数时，将设定的** ID **号“按位或”方式传入即可。**

返回值：返回处理的状态，如表 5.2 **所示。**

表 5.3 NPLink-Mote LoRaWAN MAC **支持的** MAC/PHY **参数** ID **号**

参数名	**参数值**	**默认值**	**读/写**	**含义**
PARAMETER_BANDS	1 << 0	{1,TX_POWER_14_DBM, 0, 0}	RW	LoRaWAN **使用的频点，当前支持** 1 **个频点。（仅** LoRa MAC **需要配置）**
PARAMETER_CHANNELS	1 << 1	{779500000,{((DR_5<<4)|DR_0)},0}, {779700000,{((DR_5<<4)|DR_0)},0}, {779900000,{((DR_5<<4)|DR_0)},0}	RW	LoRaWAN **在频点上使用的信道，当前支持最多** 16 **个信道。默认设置了** 3 **个信道。（仅** LoRa MAC **需要配置）**
PARAMETER_DATARATE	1 << 2	DR_7	RW	**设置发送速率（仅** LoRa MAC **需要配置）**

PARAMETER_ADR_SWITCH	1 << 3	TRUE	RW	ADR 使能或去使能（仅 LoRa MAC 需要配置）
PARAMETER_PHY_FREQUENCY	1 << 4	779500000Hz	RW	PHY MAC 工作模式下的频点（仅 phyMAC 需要配置）
PARAMETER_PHY_SPREADING_FACTOR	1 << 5	12	RW	PHY MAC 工作模式下，LORA 调制方式的扩频因子（仅 phyMAC 需要配置）
PARAMETER_PHY_MODULATION_MODE	1 << 6	LORA	RW	PHY MAC 工作模式下，调制方式的选择，可选 LORA 调制或 FSK 调制。（仅 phyMAC 需要配置）

频点的设定，采用结构体数组的形式，每个成员为一个频点数据结构体，每个频点参数的结构体定义如下：

```
typedef struct
{
    uint16_t DCycle; //频点占空比
    int8_t TxMaxPower; //最大发射功率
    uint64_t LastTxDoneTime;
    uint64_t TimeOff;
}PACKED Band_t;
```

示例： Band = { DutyCycle, TxMaxPower, LastTxDoneTime, TimeOff } = { 1 , TX_POWER_14_DBM, 0, 0 }

表 5.4 NPLink-Mote LoRaWAN MAC **支持的发射功率**

参数名	**定义值**	**含义**
TX_POWER_20_DBM	0	20dBm
TX_POWER_14_DBM	1	14dBm
TX_POWER_11_DBM	2	11dBm
TX_POWER_08_DBM	3	08dBm
TX_POWER_05_DBM	4	05dBm
TX_POWER_02_DBM	5	02dBm

信道的设定，采用结构体数组的形式，每个数组成员为一个信道参数，每个信道参数的结构体定义如下：

```
typedef struct
{
    uint32_t Frequency; //频率(Hz)
    int8_t DrRangeValue; //数据速率范围(最大值 | 最小值)
    uint8_t Band;          // 频点索引
}PACKED ChannelParams_t;
```

示例： Channel = { Frequency [Hz], { ((DrMax << 4) | DrMin) }, Band } = { 779500000, { ((DR_5 << 4) | DR_0) }, 0 }

定义一个信道，频率为 779500000，**速率从** DR_0 **到** DR_5，**隶属频点索引** 0。

表 5.5 NPLink-Mote LoRaWAN MAC **支持的发送速率**

参数名	定义值	含义(扩频因子-带宽)
DR_0	0	SF12 - BW125
DR_1	1	SF11 - BW125
DR_2	2	SF10 - BW125
DR_3	3	SF9 - BW125
DR_4	4	SF8 - BW125
DR_5	5	SF7 - BW125
DR_6	6	SF7 - BW250
DR_7	7	FSK

5.1.4 LoRaMac_getMacLayerParameter()

说明：此函数可获取一些 LoRaWAN MAC **的** MAC/PHY **相关运行参数，支持的参数列表如表** 4.3 **所示。**

注意事项：无。

原 型 ： u8 LoRaMac_getMacLayerParameter(void* pdata_out, u32 parameterIDs);

参数：

void* pdata_out **指向参数的存储空间，应传入** LoRaMacMacPara_t **类型指针。**

u32 parameterIDs **参数** ID **号，支持的** ID **号如表** 4.3 **所示，参数可采用“按位”方式传入，即可以同时设定多个参数，当同时设定多个参数时，将设定的** ID **号“按位或”方式传入即可。**

返回值：返回处理的状态，如表 4.2 **所示。**

5.1.5 LoRaMac_setMode()

说明：此函数用于设置 MAC **层工作模式。**

注意事项：无。

原型：u8 LoRaMac_setMode(u8 mode);

参数：

u8 mode　MAC **层的工作模式，**MODE_LORAMAC -- LORA MAC **方式工作，**MODE_PHY -- phy MAC **方式工作。**

返回值：返回处理的状态，如表 4.2 **所示。**

5.1.6 LoRaMac_setlowPowerMode()

说明：此函数用于设置 radio **部分低功耗,使能后，**radio **部分将关闭晶振，并进入** sleep **状态。**

注意事项：无。

原型：void LoRaMac_setlowPowerMode(u8 enable);

参数：

u8 enable **用于设置低功耗使能与否,**TRUE -- **使能低功耗，**FALSE -- **去使能低功耗** 。

返回值：无 。

5.2 APP **数据收发** API

在收发数据之前，需要先通过 LoRaMac_setMode 函数进行工作模式的选择（LORA MAC 或 PHY MAC），并事先设定好在当前工作模式下的工作参数。对于 LORA MAC 工作模式，Mote 的参数应该跟 GWM 一致；对于 PHY MAC 工作模式，两个 Mote 应该设置成一样的参数，才能进行数据收发。

5.2.1 发送无线数据

在 NPLink-Mote-SDK 中，收发无线数据通过 OSAL 的消息进行，当有数据需要发送时，通过给 LoRaMAC TASK 发送一个消息，示例如下：

```
pMsgSend = (loraMAC_msg_t*)osal_msg_allocate(18);
if(pMsgSend != NULL)
{
    osal_memset(pMsgSend,0,17);
    pMsgSend->msgID = TXREQUEST; //消息类型
    pMsgSend->msgLen = 16;//数据长度
    for(u8 dataCount = 0; dataCount < 16; dataCount++)
    {
        pMsgSend->msgData[dataCount] = dataCount;
    }
    osal_msg_send(LoraMAC_taskID,(u8*)pMsgSend);  // 向 LoRaWAN MAC 任务发送消息，LoRaWAN MAC 将发送此数据包
}
```

在 LoRaWAN MACAPP 之间传递的消息结构体定义如下：

```
typedef struct loraMAC_msg
{
    uint8    msgID;//消息 ID 号
    uint8    msgLen;//消息 msgData 的长度
    int8_t msgRxRssi;//接收数据包的信号强度
    int8_t msgRxSnr;//接收数据包的信噪比
    uint16 frame_no ;//发送 or 接收数据的序列号
    int8_t tx_packet_status;//发送数据包的状态
    uint8    reserve;//预留
    uint8 msgData[70];//消息体 payload，最大为 70 字节
}loraMAC_msg_t;
```

5.2.2 接收无线数据

在 NPLink-Mote-SDK **中，通过系统消息的方式来通告数据的接收及处理的反馈（如加入网络、无线包发送完成等），每种“回发的消息”均具有一个唯一的“消息类型”号。目前，支持的消息类型号如下表** 5.6 **所示。**

表 5.6 NPLink-Mote LoRaWAN MAC **支持的消息类型列表**

消息类型定义	**定义值**	**含义**
TXDONE	1	**发送完成**
RXDONE	2	**接收完成**
TXREQUEST	3	**发送请求**

TXERR_STATUS	4	**发送失败**

代码示例如下：

```
if(events & SYS_EVENT_MSG) //系统消息
{//循环接收消息
  while(NULL!=(pMsgRecieve=(loraMAC_msg_t*)osal_msg_receive(APP_taskID)))
  { //pMsgRecieve[0]为消息类型
     switch(pMsgRecieve->msgID)
     {
        //发送完成
        case TXDONE :
        //进行处理，延时继续发送
        osal_start_timerEx(APP_taskID,APP_PERIOD_SEND, 5000);
         break;

        //接收完成
        case RXDONE:
          memset( Rx_buf,0,32);
          sprintf( Rx_buf,"RXLEN:%d",pMsgRecieve->msgLen);
          OLED_ShowString( 0,16, (u8*)Rx_buf,16 );
           for(u8 dataCount = 0; dataCount < pMsgRecieve->msgLen; dataCount++)
           {
              sprintf(pRx_buf,"DATA:%d",pMsgRecieve->msgData[dataCount]);
               pRx_buf += 7;
```

```
  }
 OLED_ShowString( 0,32, (u8*)Rx_buf,16 );
 OLED_Refresh_Gram();
 break;

  default://未知的消息类型，不处理
    break;
  }
 osal_msg_deallocate((u8*)pMsgRecieve); //释放消息空间
  }
return (events ^ SYS_EVENT_MSG);
}
```

注意：目前，LoRaWAN MAC **任务只支持向** APP_taskID **发送“回发的消息”。**