课程目录

1. STM32WL 简介
2. STM32WL 硬件简介
3. STM32WL 软件简介
4. LoRa和LoRaWAN介绍
5. STM32WL LoRa 例程介绍
6. STM32WL 使用STM32 CubeMX 创建LoRa 节点应用
7. STM32WL LoRa RF 测试
8. STM32WL 安全特性介绍
9. STM32WL FUOTA 应用设计

life.augmented

STM32WL 软件简介

David Liu

课程目录

1 STM32WL 软件包

2 STM32WL 软件架构

3 STM32WL LoRa 软件架构

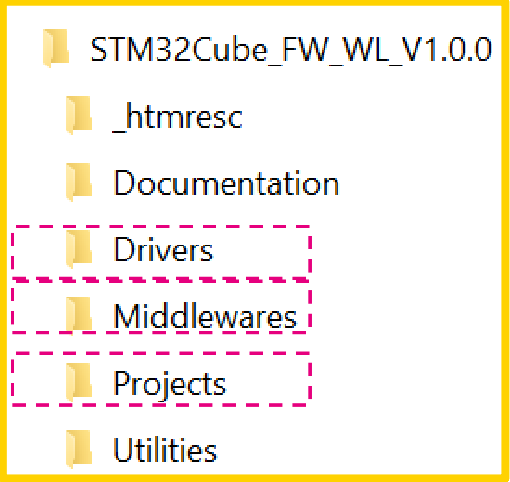
4 LoRaWAN 协议层介绍

5 LoRa SubGHz层介绍

life.augmented

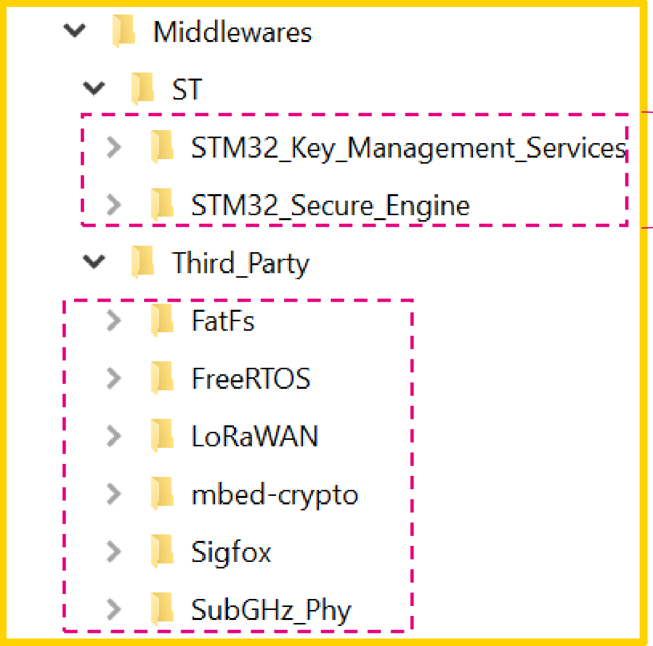
STM32WL 软件包

STM32Cube_FW_WL_V1.0.0



```
STM32Cube_FW_WL_V1.0.0
    _htmresc
    Documentation
    Drivers
    Middlewares
    Projects
    Utilities
```

Middleware

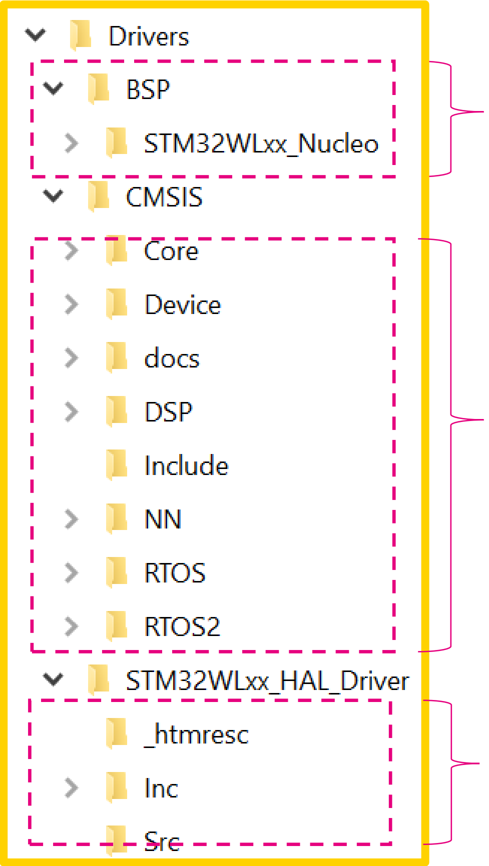


```
v Middlewares
    v ST
        > STM32_Key_Management_Services
        > STM32_Secure_Engine
    v Third_Party
        > FatFs
        > FreeRTOS
        > LoRaWAN
        > mbed-crypto
        > Sigfox
        > SubGHz_Phy
```

Middleware
- KMS
- SE

Third party
- FreeRTOS
- FatFs
- LoRaWAN
- Mbed-crypto
- Sigfox
- SubGHz_Phy

Driver



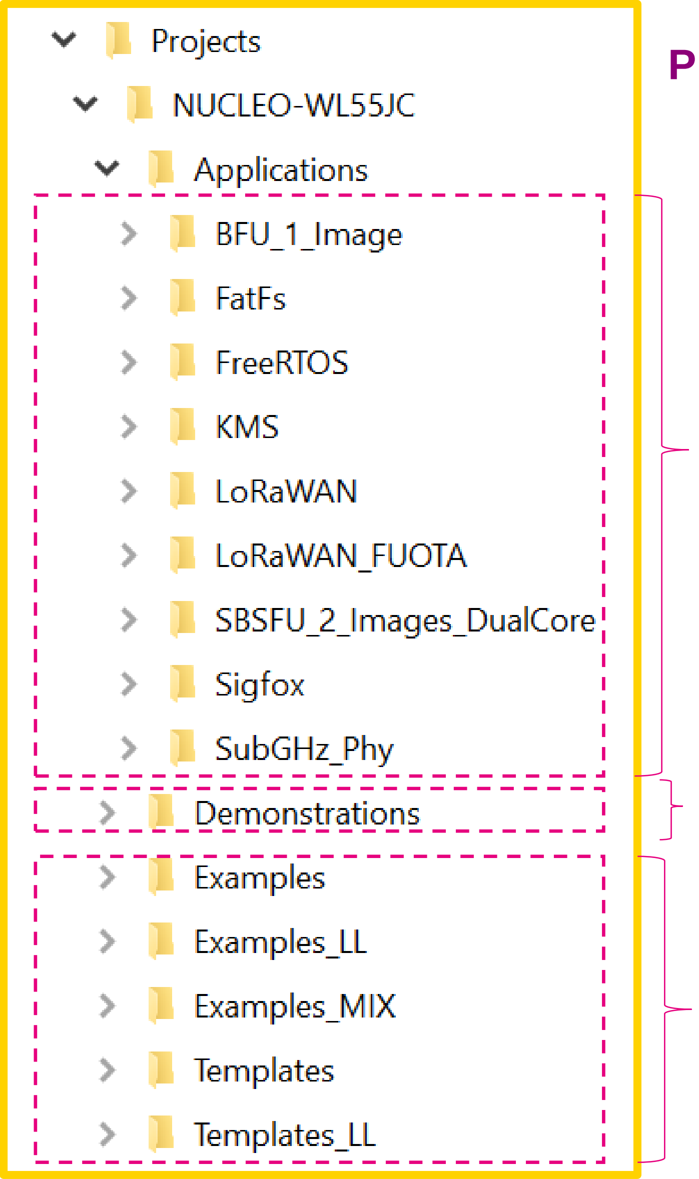
```
v Drivers
    v BSP
        > STM32WLxx_Nucleo
    v CMSIS
        > Core
        > Device
        > docs
        > DSP
          Include
        > NN
        > RTOS
        > RTOS2
    v STM32WLxx_HAL_Driver
          _htmresc
        > Inc
          Src
```

BSP

CMSIS

HAL Driver

Projects



```
v Projects
    v NUCLEO-WL55JC
        v Applications
            > BFU_1_Image
            > FatFs
            > FreeRTOS
            > KMS
            > LoRaWAN
            > LoRaWAN_FUOTA
            > SBSFU_2_Images_DualCore
            > Sigfox
            > SubGHz_Phy
        > Demonstrations
        > Examples
        > Examples_LL
        > Examples_MIX
        > Templates
        > Templates_LL
```

STM32WL Application
- LoRa
- Sigfox
- SubGHz
- SBSFU
- KMS
- FUOTA
- FreeRTOS

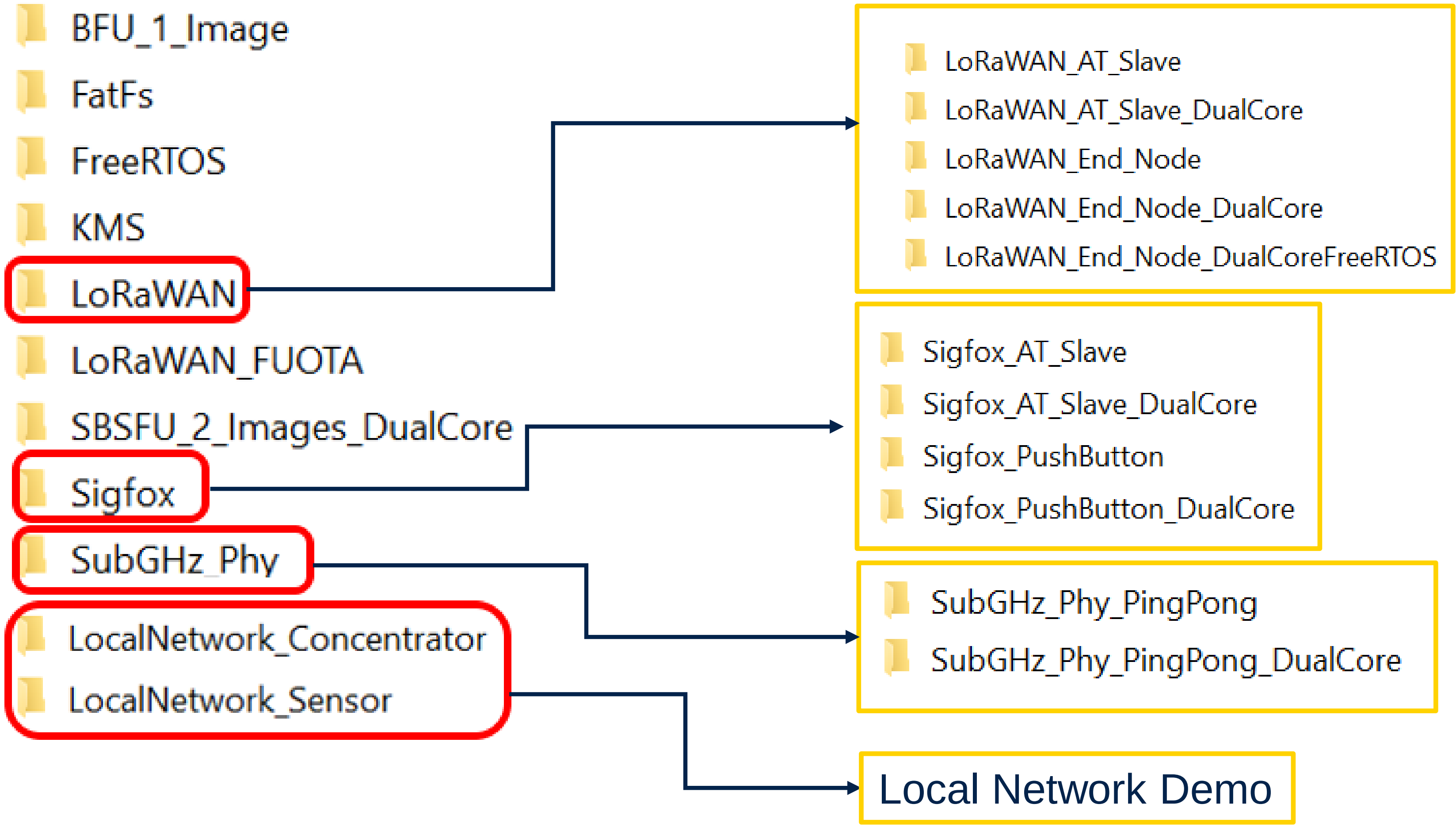
Local Network Demo

Peripheral Examples

下载链接：Here

STM32WL 无线Demo

- STM32Cube_FW_WL_V1.0.0



- BFU_1_Image
- FatFs
- FreeRTOS
- KMS
- LoRaWAN
- LoRaWAN_FUOTA
- SBSFU_2_Images_DualCore
- Sigfox
- SubGHz_Phy
- LocalNetwork_Concentrator
- LocalNetwork_Sensor

LoRaWAN:

- LoRaWAN_AT_Slave
- LoRaWAN_AT_Slave_DualCore
- LoRaWAN_End_Node
- LoRaWAN_End_Node_DualCore
- LoRaWAN_End_Node_DualCoreFreeRTOS

Sigfox:

- Sigfox_AT_Slave
- Sigfox_AT_Slave_DualCore
- Sigfox_PushButton
- Sigfox_PushButton_DualCore

SubGHz_Phy:

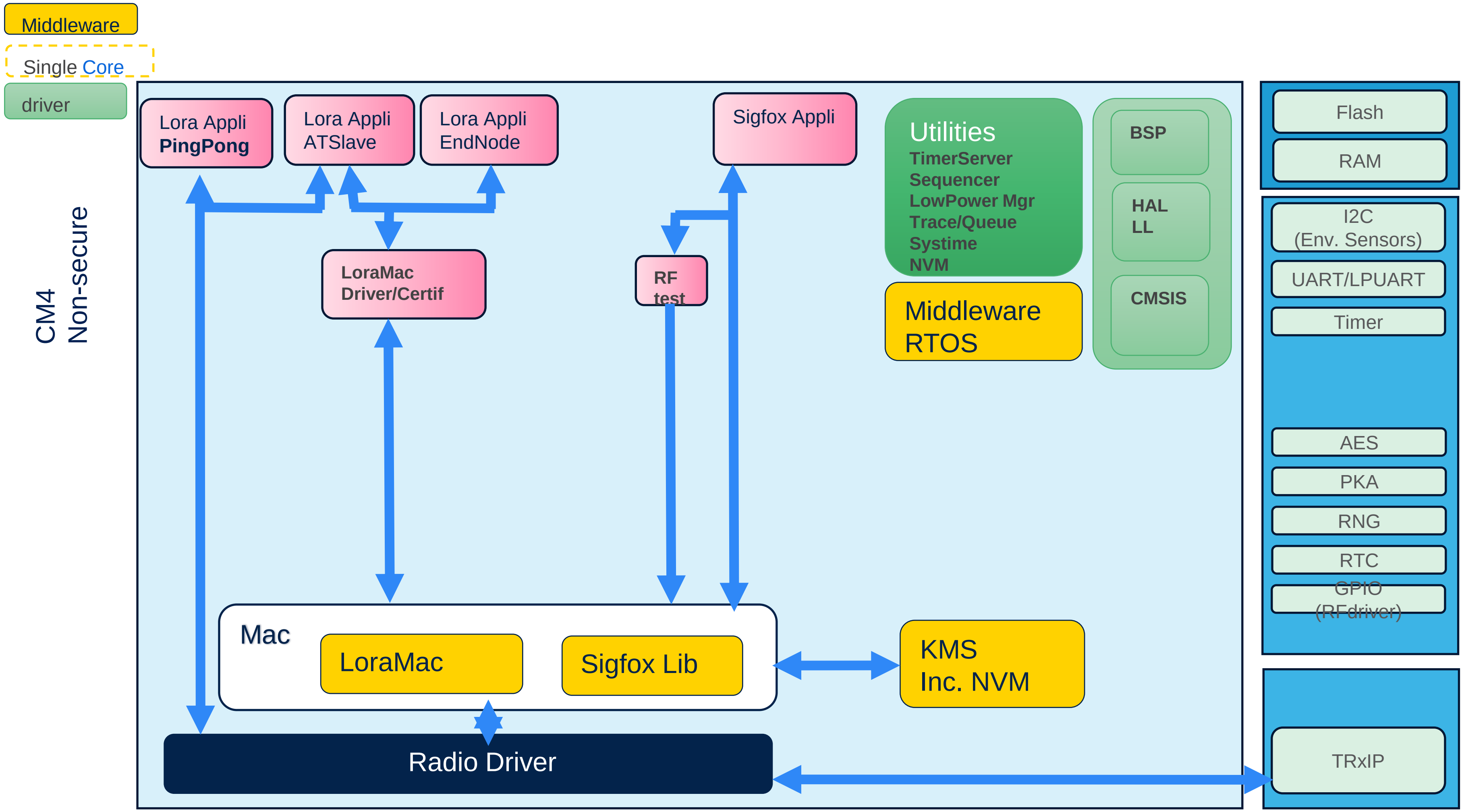
- SubGHz_Phy_PingPong
- SubGHz_Phy_PingPong_DualCore

LocalNetwork_Concentrator, LocalNetwork_Sensor:

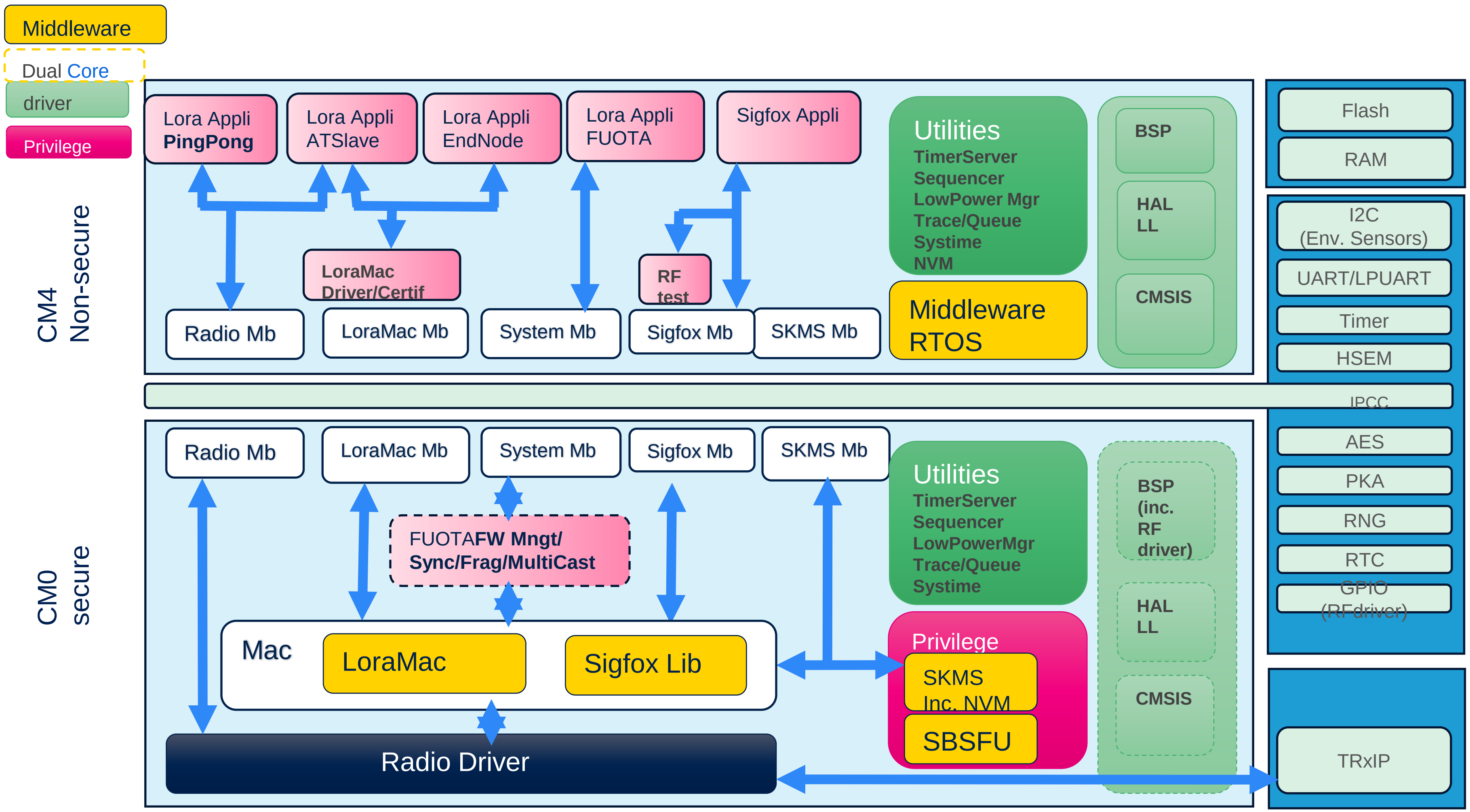
Local Network Demo

STM32WL 软件架构

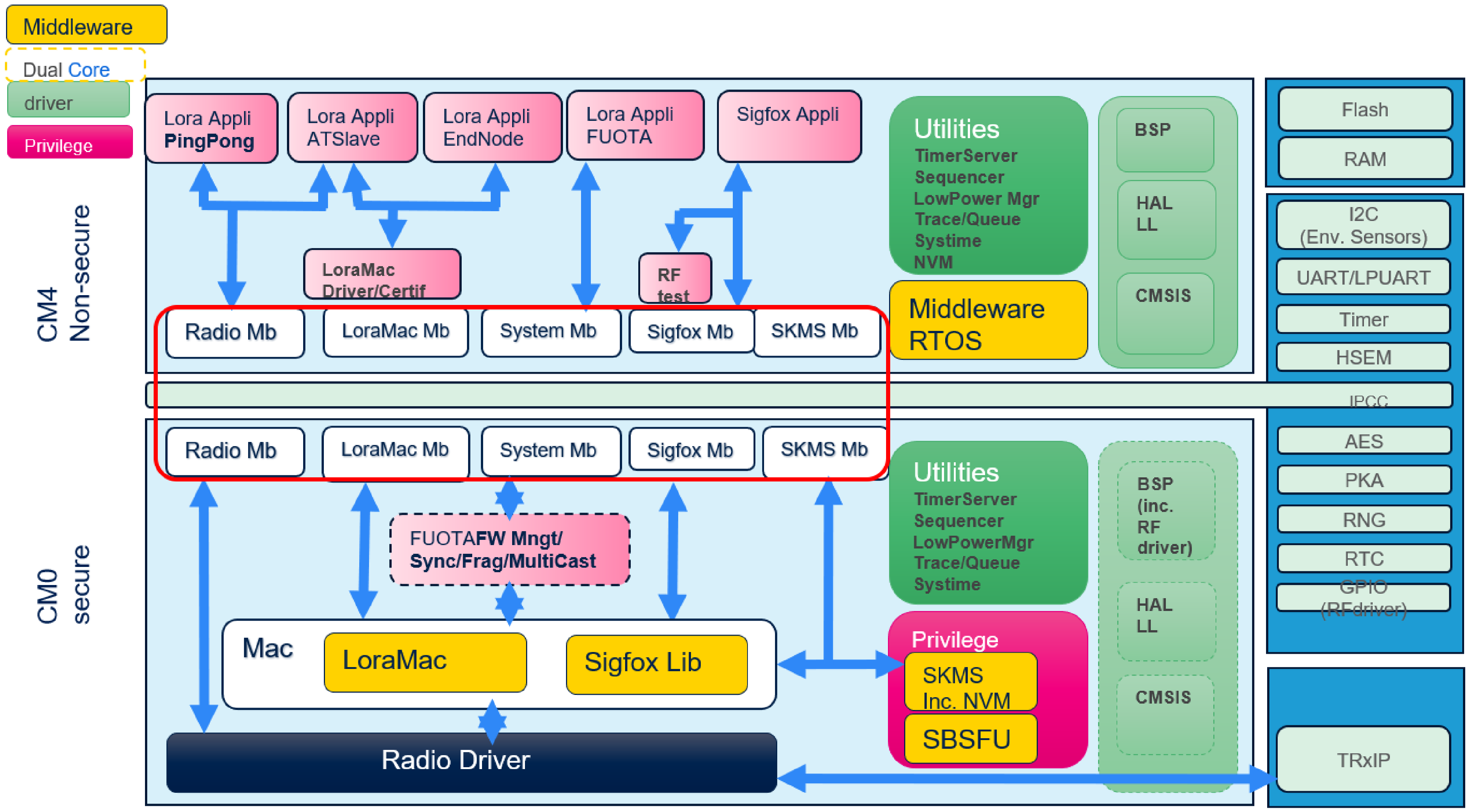
STM32WL 软件架构(单核)



STM32WL 软件架构(双核)

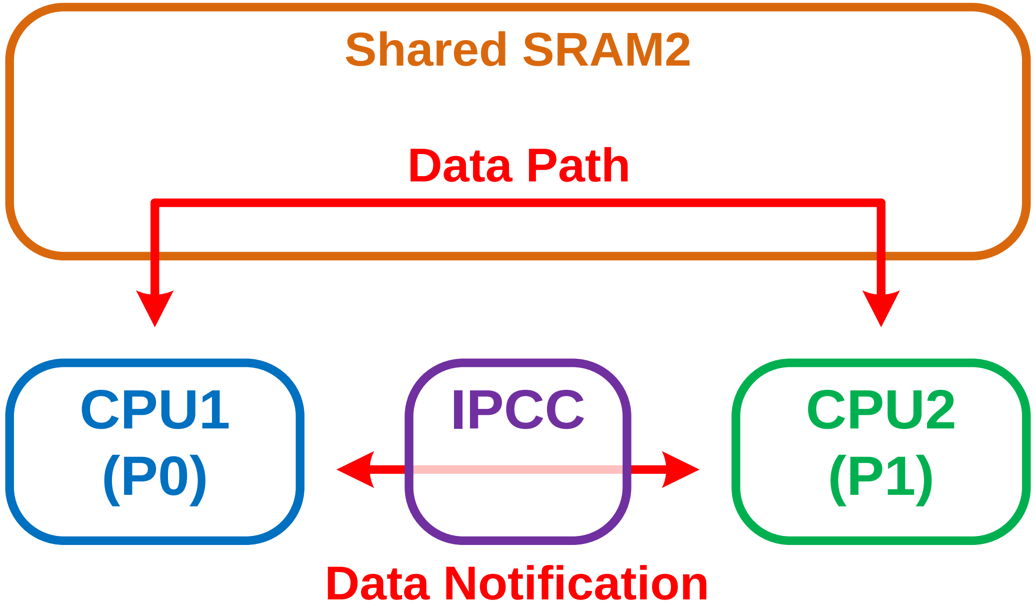


STM32WL Mailbox概览



什么是Mailbox

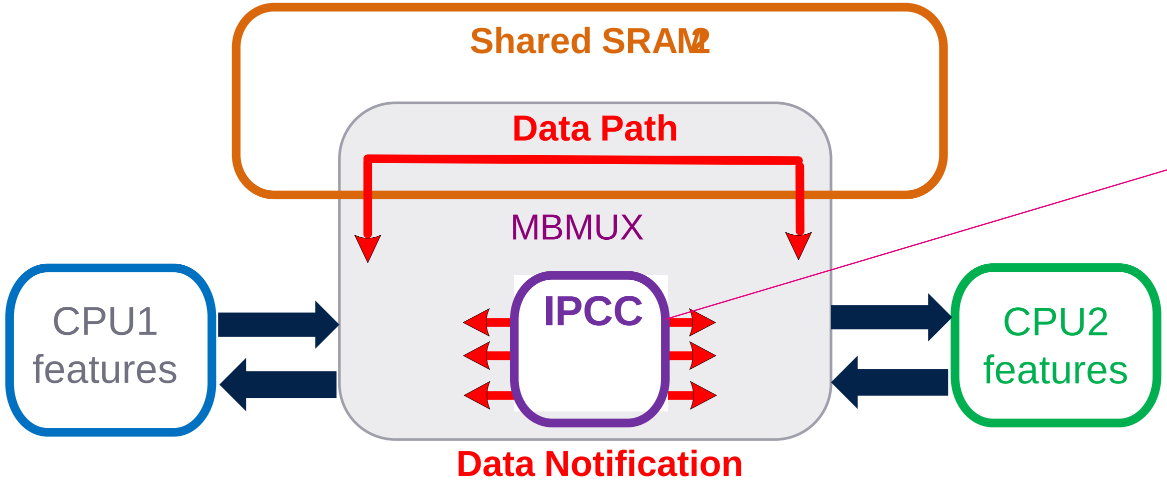
- **Mailbox**是一项SW服务，实现了一种在两个处理器之间交换数据的方式。 它建立在两种资源之上：
  - **IPCC**：这是一个HW IP，其唯一目的是触发到远程CPU的中断并从远程CPU接收中断
  - **SRAM2**：RAM2_SH1两个CPU都可以读取/写入共享存储器。 它用于存储所有缓冲区，这些缓冲区将包含两个CPU之间要交换的数据。

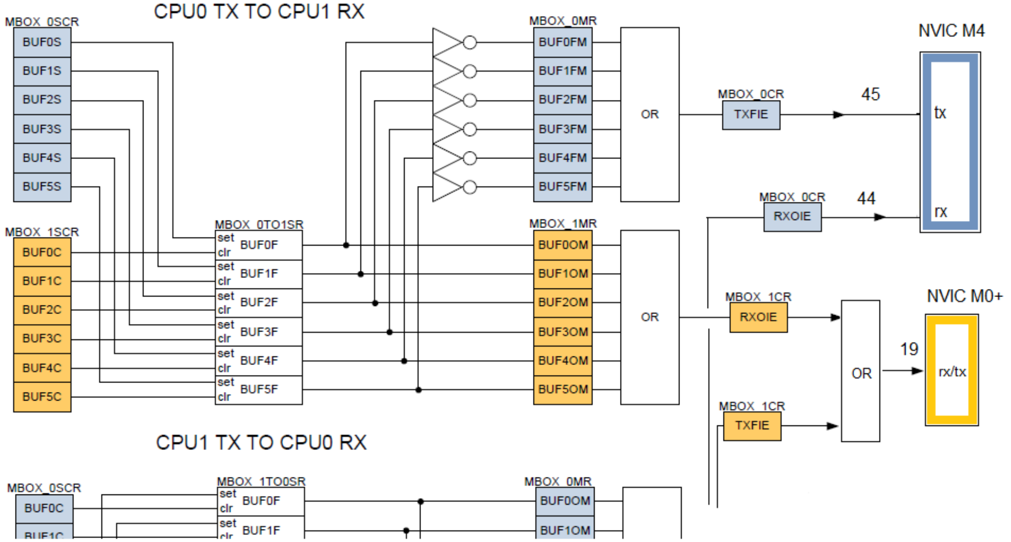


什么是MBMUX?

- MBMUX是Mailbox软件服务的一层，它将IPCC通道irq与共享内存缓冲区结合在一起，以允许两个CPU之间进行消息交换

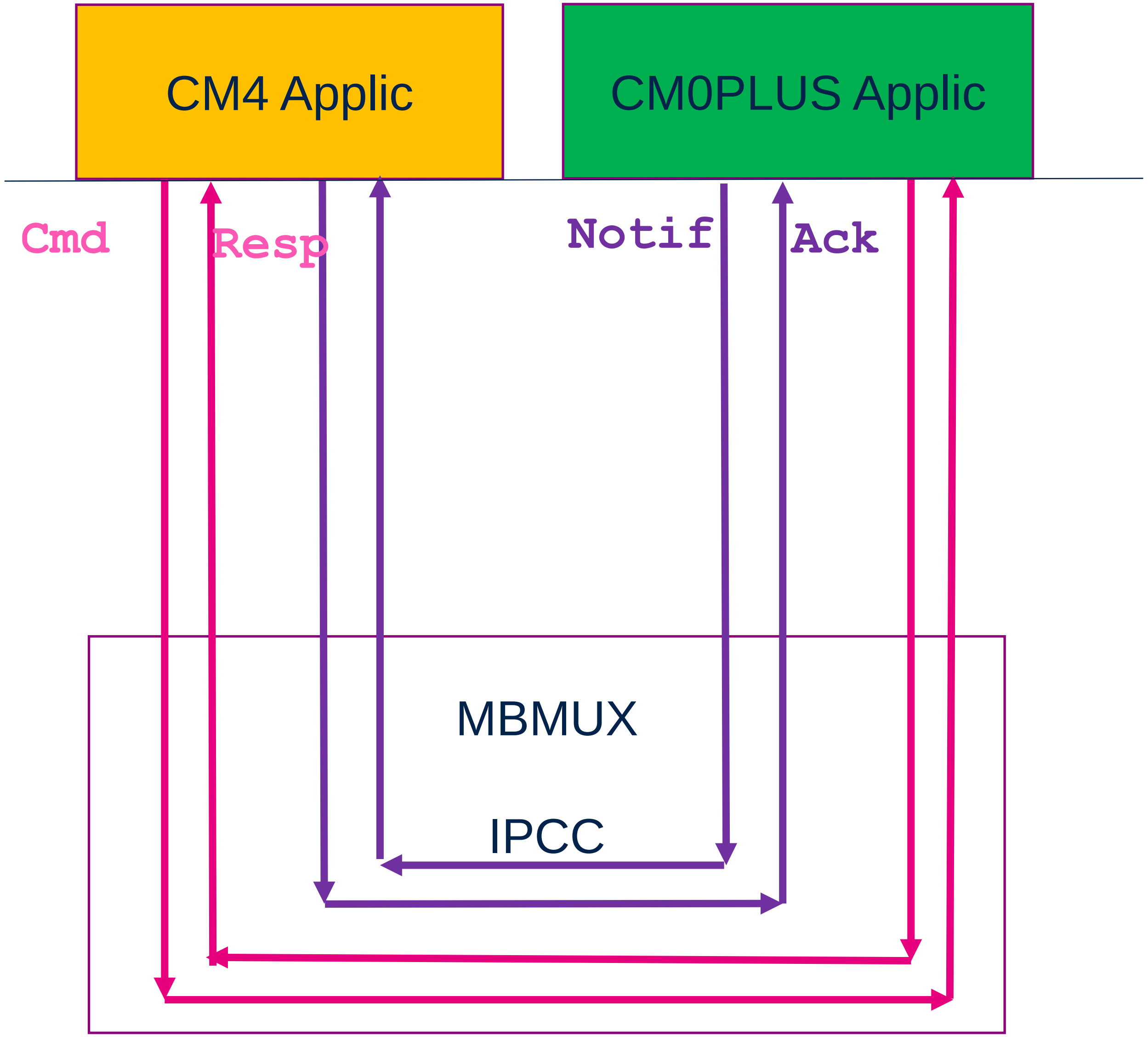
- MBMUX还是一个多路复用器，允许在2x6 IPCC通道上映射功能（Lora，Sigfox，Radio，Mwbus， Kms（SKS），Trace等）





STM32WL Mailbox和MBMUX 工作模式

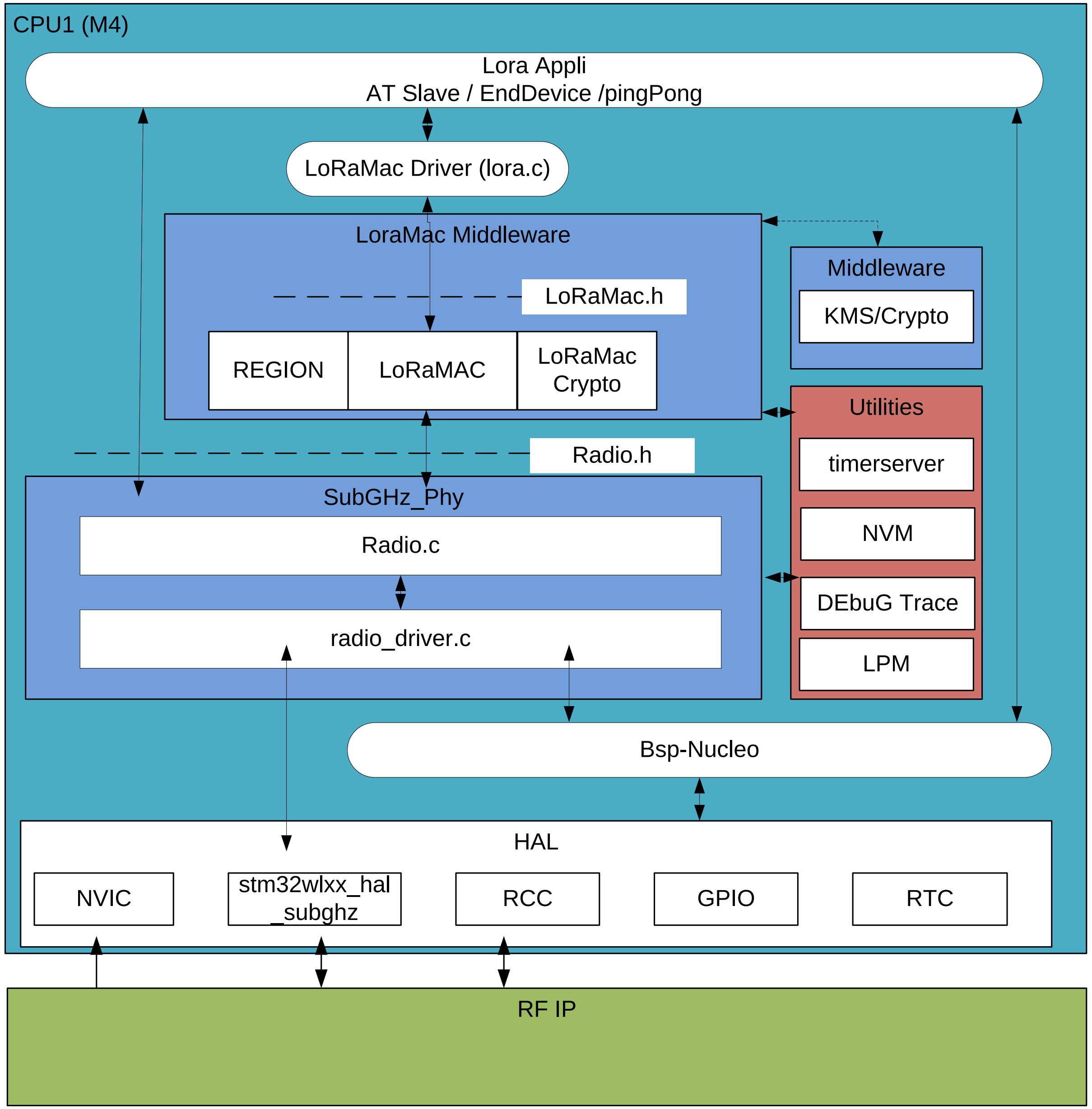
- Cmd: CM4 发送给CM0PLUS 的命令.
  - 用于调用在CM0PLUS上实现的函数
- Resp: CM0PLUS 给CM4响应.
  - 告诉CM4函数已经被执行，并返回可用的返回值
- Notif: 由CM0PLUS 发给CM4的通知.
  - 用于调用在CM4上实现的函数
- Ack: CM4 给 CM0PLUS的应答.
  - 告诉CM0PLUS函数已经被执行，并返回可用的返回值



STM32WL LoRa 软件架构

LoRa软件架构

- **Application:**
  - Ping-Pong 应用
  - End-Node 应用
  - AT-Slave 应用
- **Middleware**
  - LoRaWAN 包含LoRa Mac 层
  - SubGHz_Phy 包含Phy 层
  - KMS
    - 密钥管理系统
- **Utilities**
  - 低功耗管理器. 调度器 .定时服务器
- **BSP**
  - RF 开关, 板级配置

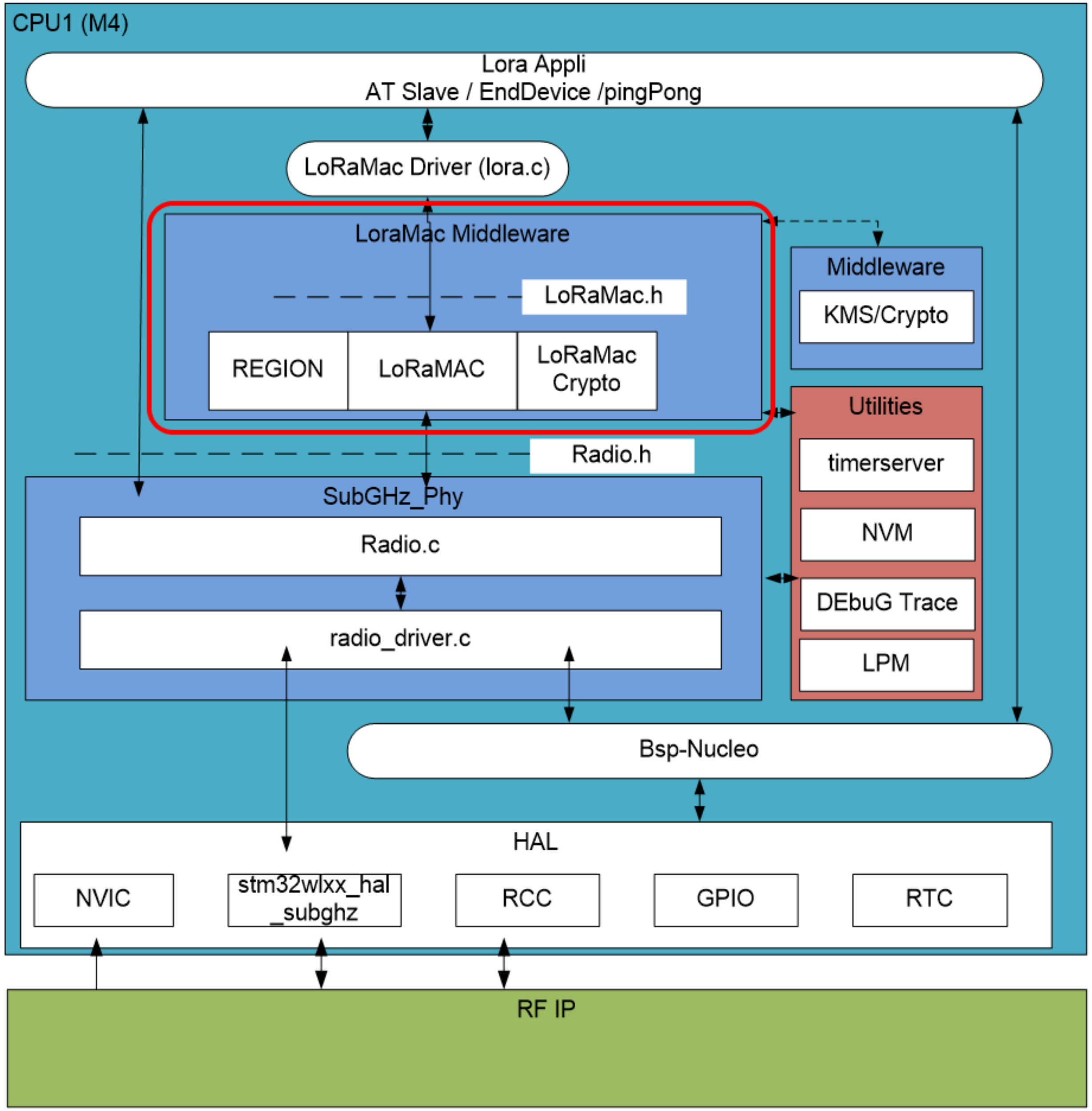


life.augmented

LoRaWAN 协议层介绍

LoRaWAN

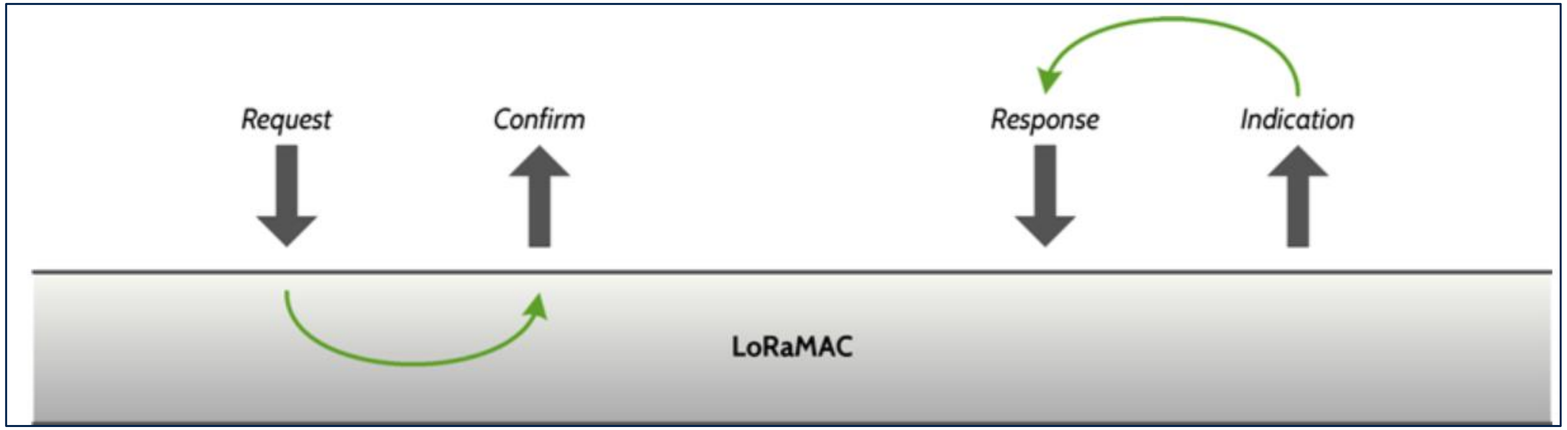
- LoRaWAN
  - **LoRaMAC层**
  - SubGHz_Phy 层



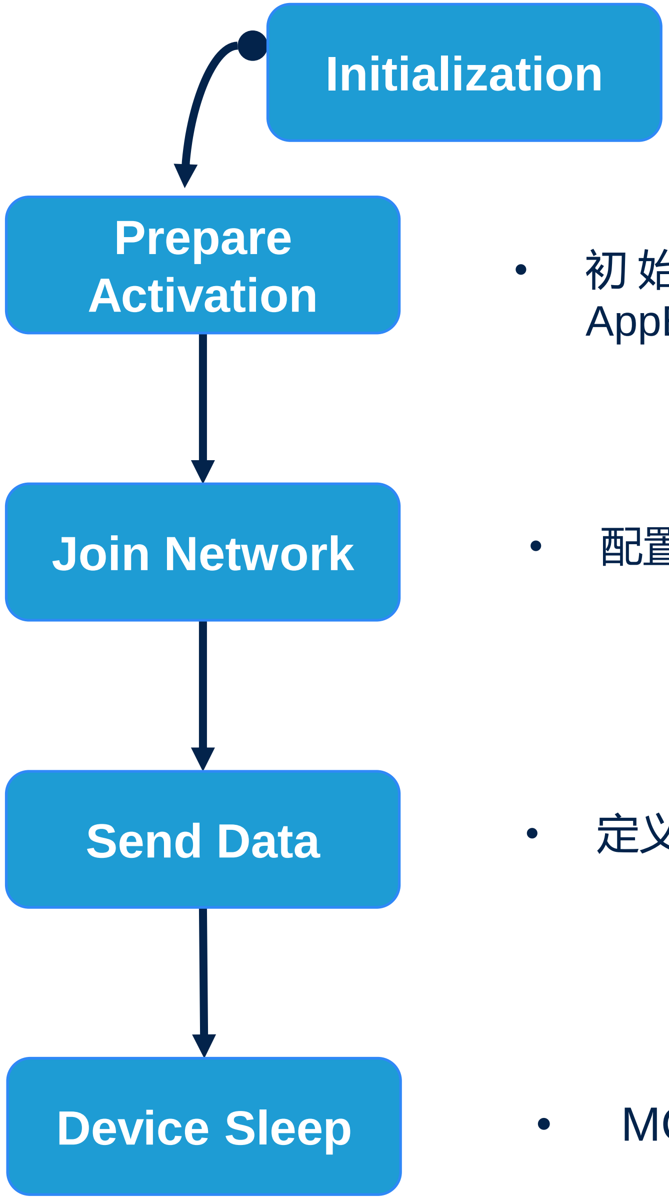
LoRaMAC 工作模式

LoRaMAC层提供MCPS（MAC公共部分子层）服务，MLME（MAC层管理实体）服务和MIB（MAC信息库）。该概念遵循请求确认和指示响应体系结构。

- **MCPS** 用于数据收发
- **MLME** 用于管理LoRaWAN 网络
- **MIB** 负责存储重要的运行时信息，并保存LoRaMAC层的配置



LoRaWAN 工作流程



Initialization

Prepare Activation

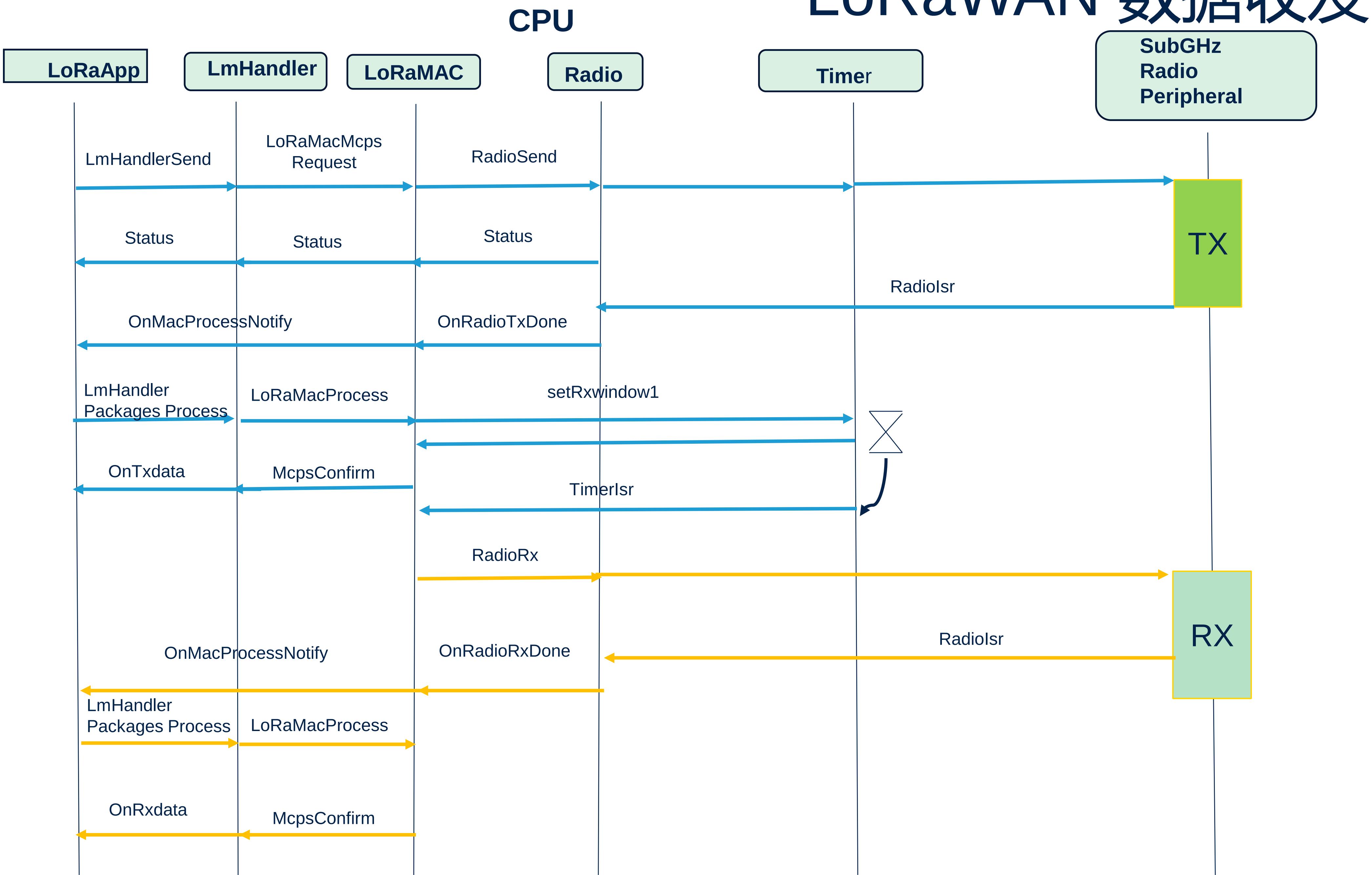
Join Network

Send Data

Device Sleep

- 初始化，定义回调函数
- 初始化LoRaWAN 设备的 unique ID, DevEUI, AppEUI, Region, ADR, DutyCycle.(**MIB**) 参数
- 配置入网方式OTAA o或者 ABP(**MLME**)
- 定义数据的格式, 端口, 数据缓存,速率(**MCPS**)
- MCU进入低功耗

LoRaWAN 数据收发流程



life.augmented

初始化示例代码

```
static void McpsConfirm( McpsConfirm_t *mcpsConfirm )
{
// Implementation of the MCPS-Confirm primitive
}
static void McpsIndication( McpsIndication_t *mcpsIndication )
{
// Implementation of the MCPS-Indication primitive
}
static void MlmeConfirm( MlmeConfirm_t *mlmeConfirm )
{
// Implementation of the MLME-Confirm primitive
}
static void MlmeIndication( MlmeIndication_t *mlmeIndication )
{
// Implementation of the MLME-Indication primitive
}
static void OnMacProcessNotify( void )
{
// Mac notification. Process run function
}
LoRaMacPrimitives_t LoRaMacPrimitives;
LoRaMacCallback_t LoRaMacCallbacks;
LoRaMacStatus_t Status;
```

```
int main( void )
{
LoRaMacPrimitives.MacMcpsConfirm = McpsConfirm;
LoRaMacPrimitives.MacMcpsIndication = McpsIndication;
LoRaMacPrimitives.MacMlmeConfirm = MlmeConfirm;
LoRaMacPrimitives.MacMlmeIndication = MlmeIndication;
LoRaMacCallbacks.GetBatteryLevel = BoardGetBatteryLevel;
LoRaMacCallbacks.GetTemperatureLevel = NULL; // apply
board specific temperature reading
LoRaMacCallbacks.NvmContextChange = NvmCtxMgmtEvent;
LoRaMacCallbacks.MacProcessNotify = OnMacProcessNotify;
// Initialization for the region EU868
Status = LoRaMacInitialization( &LoRaMacPrimitives,
&LoRaMacCallbacks, LORAMAC_REGION_EU868 );
if( Status == LORAMAC_STATUS_OK )
{
// Initialization successful
}
}
```

入网操作示例代码

```
MlmeReq_t mlmeReq;
LoRaMacStatus_t status;
MibRequestConfirm_t mibReq;
uint8_t devEui[] = LORAWAN_DEVICE_EUI;
uint8_t joinEui[] = LORAWAN_JOIN_EUI;
// This comment is a placeholder for the initialization of the LoRaMAC layer. Set dev eui
mibReq.Type = MIB_DEV_EUI;
mibReq.Param.DevEui = devEui;
LoRaMacMibSetRequestConfirm( &mibReq );
// Set join eui
mibReq.Type = MIB_JOIN_EUI;
mibReq.Param.JoinEui = joinEui;
LoRaMacMibSetRequestConfirm( &mibReq );
// Setup the request type
mlmeReq.Type = MLME_JOIN;
// Fill the join parameters
mlmeReq.Req.Join.Datarate = DR_0;
status = LoRaMacMlmeRequest( &mlmeReq );
if( status == LORAMAC_STATUS_OK )
{
// Join request was send successfully
}
```

数据发送示例代码

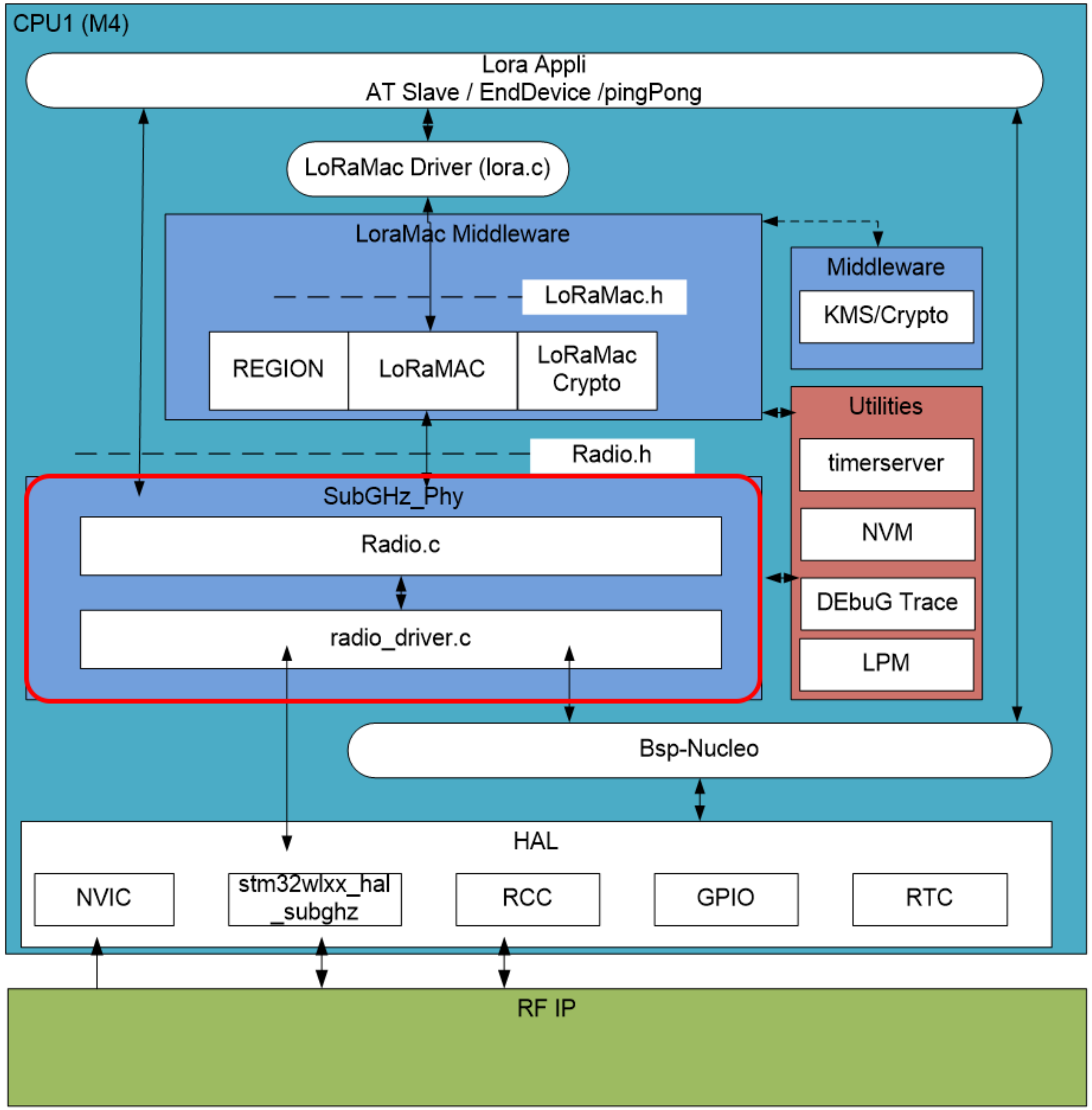
```
if( IsTxConfirmed == false )
{
mcpsReq.Type = MCPS_UNCONFIRMED;
mcpsReq.Req.Unconfirmed.fPort = AppPort;
mcpsReq.Req.Unconfirmed.fBuffer = AppDataBuffer;
mcpsReq.Req.Unconfirmed.fBufferSize = AppDataSize;
mcpsReq.Req.Unconfirmed.Datarate =
LORAWAN_DEFAULT_DATARATE;
}
else
{
mcpsReq.Type = MCPS_CONFIRMED;
mcpsReq.Req.Confirmed.fPort = AppPort;
mcpsReq.Req.Confirmed.fBuffer = AppDataBuffer;
mcpsReq.Req.Confirmed.fBufferSize = AppDataSize;
mcpsReq.Req.Confirmed.Datarate = LORAWAN_DEFAULT_DATARATE;
}
```

```
// Update global variable
AppData.MsgType = ( mcpsReq.Type == MCPS_CONFIRMED )
? LORAMAC_HANDLER_CONFIRMED_MSG :
LORAMAC_HANDLER_UNCONFIRMED_MSG;
AppData.Port = mcpsReq.Req.Unconfirmed.fPort;
AppData.Buffer = mcpsReq.Req.Unconfirmed.fBuffer;
AppData.BufferSize = mcpsReq.Req.Unconfirmed.fBufferSize;
LoRaMacStatus_t status;
// Data Send
status = LoRaMacMcpsRequest( &mcpsReq );
```

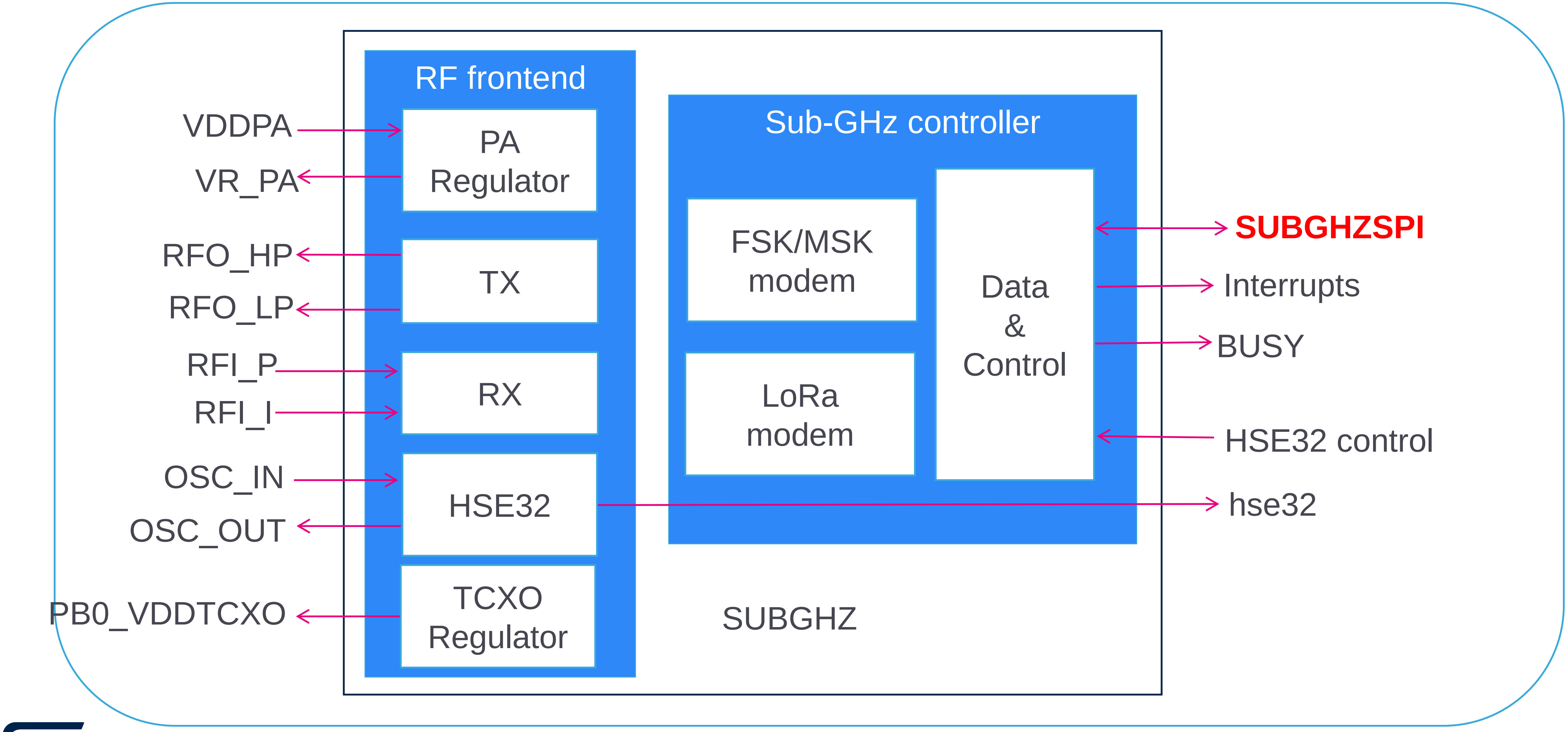
LoRa SubGHz层介绍

SubGHz_Phy层

- Middleware
  - LoRaWAN 包含Mac 层
  - **SubGHz_Phy 包含Phy 层**



SubGHz 内部结构



life.augmented

SubGHz 概要

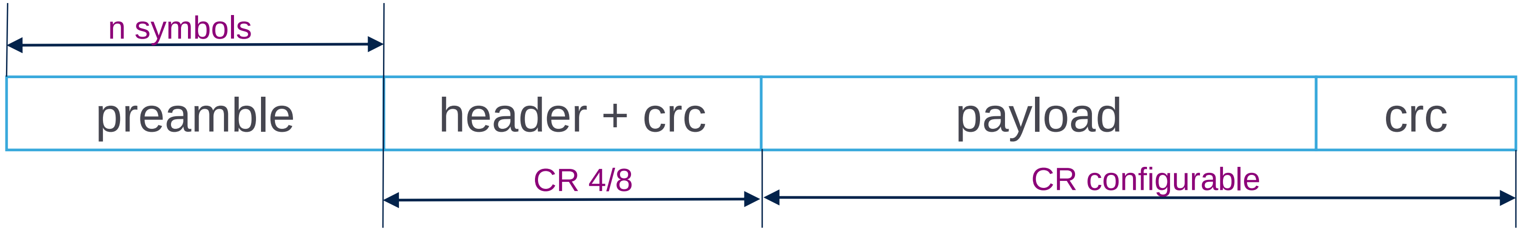
- Sub-GHz 主要特征:
  - ISM 频率范围150 – 960 MHz
  - 调制模式和速率:
    - LoRa™, 速率从 0.013 到 17.4 kbps
    - (G)FSK, 速率 从0.6 t到 300 kbps
    - (G)MSK, 速率 从0.1 到10 kbps
    - BPSK,速率从100 bps 到600 bps
  - 规范
    - ETSI EN 300 220, EN 300 113, EN 301 166
    - FCC CFR 47 part 15, 24, 90, 101
    - ARIB STD-T30, T67, T108
  - 协议标准:
    - LoRaWAN™, Sigfox, ….
    - Proprietary protocols

应用优势

- 多协议支持
- 超低功耗
- 功率输出高达+22 dBm
- 自动校准

LoRa™ 调制

- 数据帧
  - 带有报头和可变长度有效载荷的显式包
  - 没有报头和固定长度有效载荷的隐式数据包

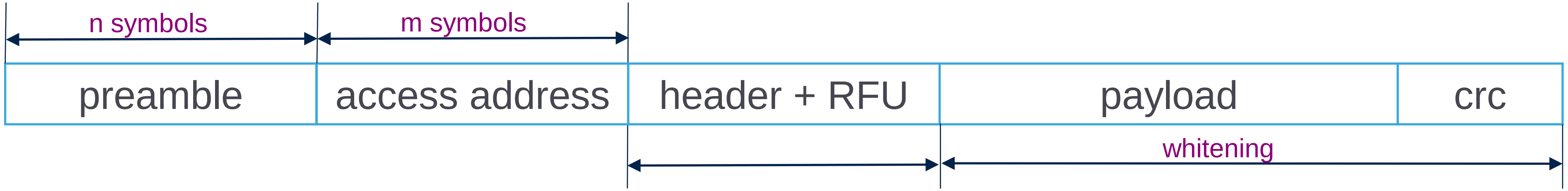


- 调制解调器配置：
  - 调制带宽BW7.81 kHz至500 kHz
  - 扩展因子SF 32个码片/符号）最多4096个码片/符号）
  - 编码率CR 4/4至4/8
- 通道活动检测

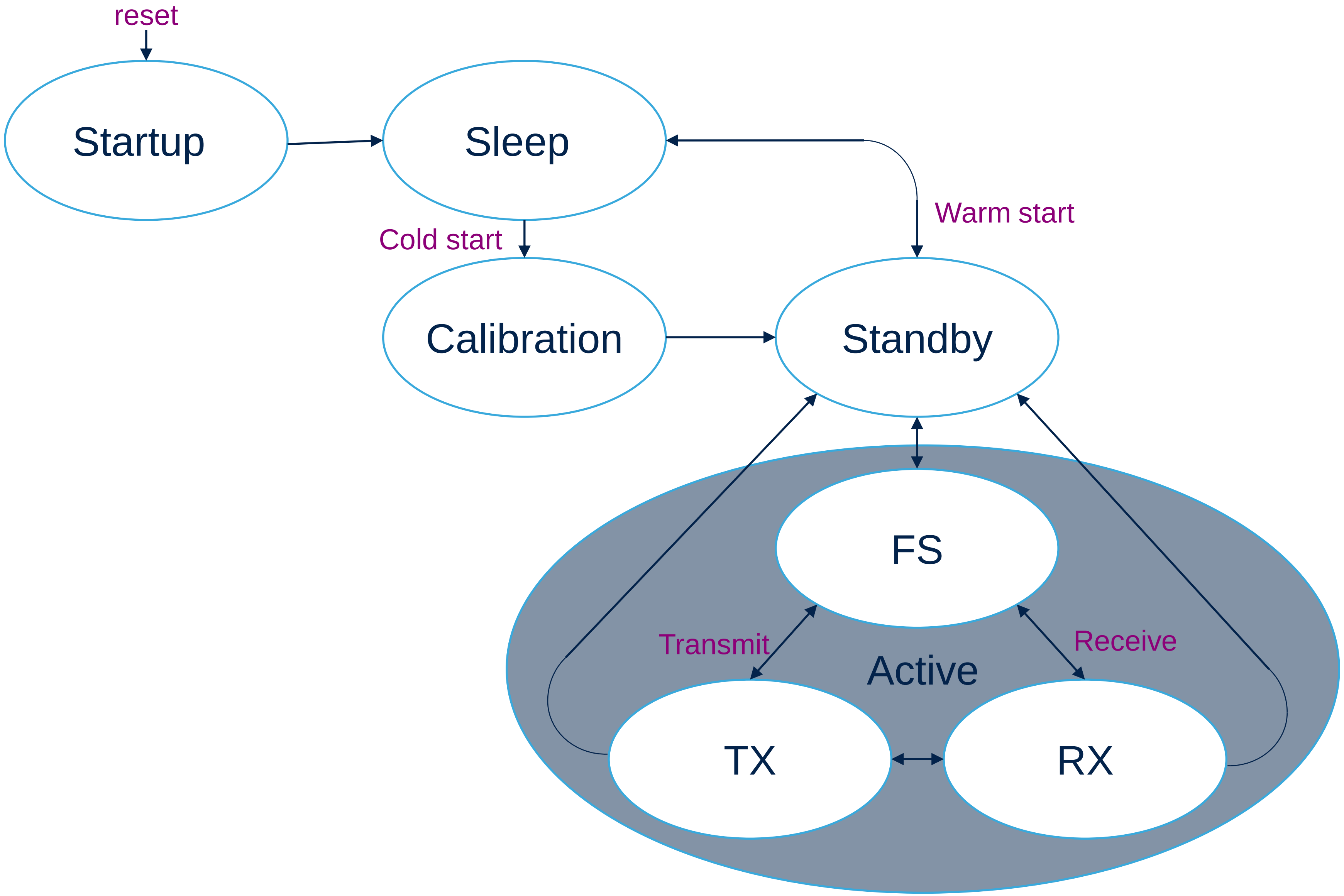
FSK/MSK调制

数据帧

- FSK和MSK调制解调器一起使用
- NRZ编码
- 可编程前导码长度
- 长度可选的访问地址
- 具有报头和可变长度有效载荷的可变长度数据包
- 不含报头和固定长度有效载荷的固定长度数据包
- 可选的白化（9位LFSR x9 + x5 + x1，可编程初始化值）
- 有效载荷CRC（可编程多项式，初始化值，取反和长度）

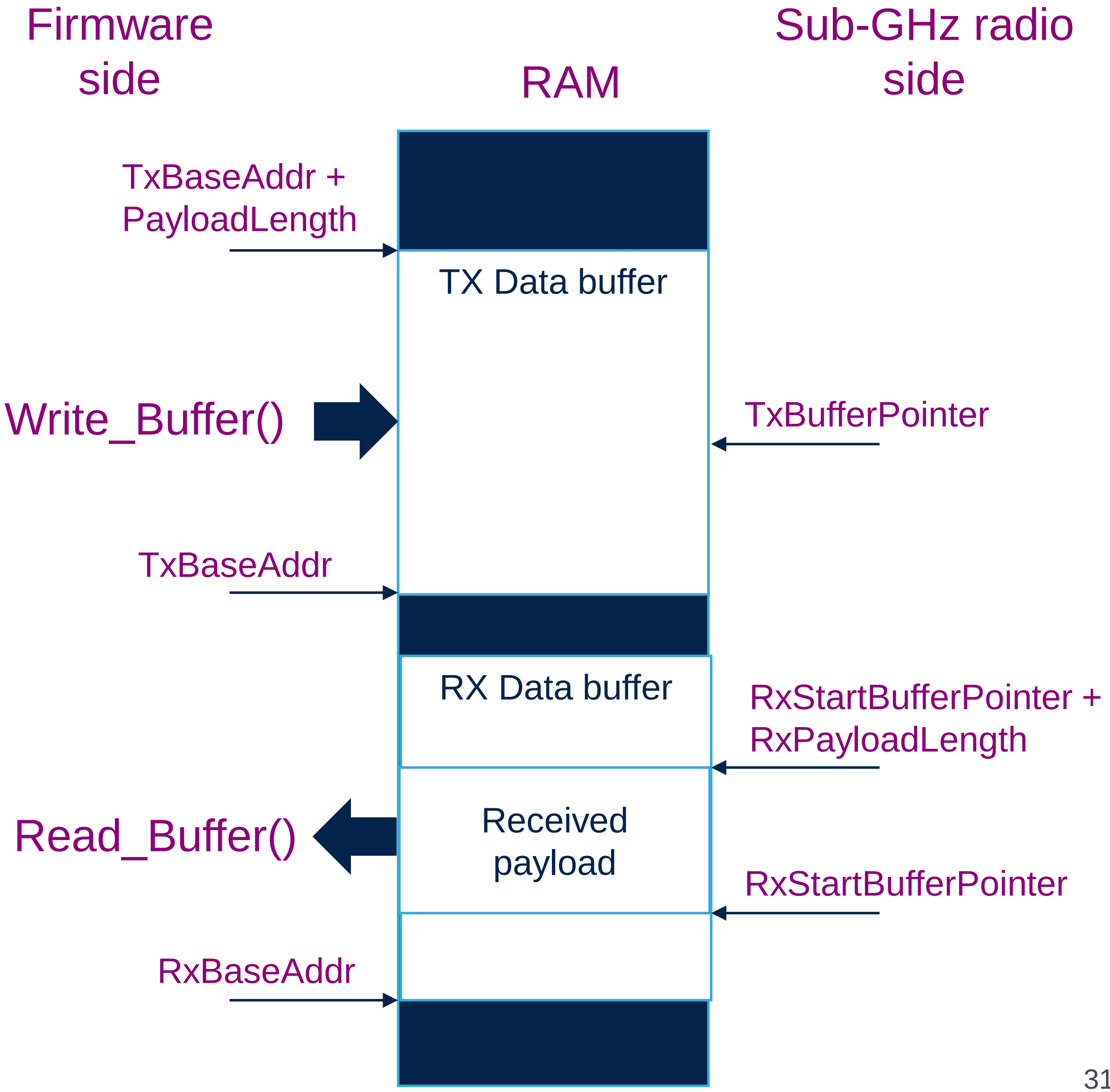


Sub-GHz radio 工作模式

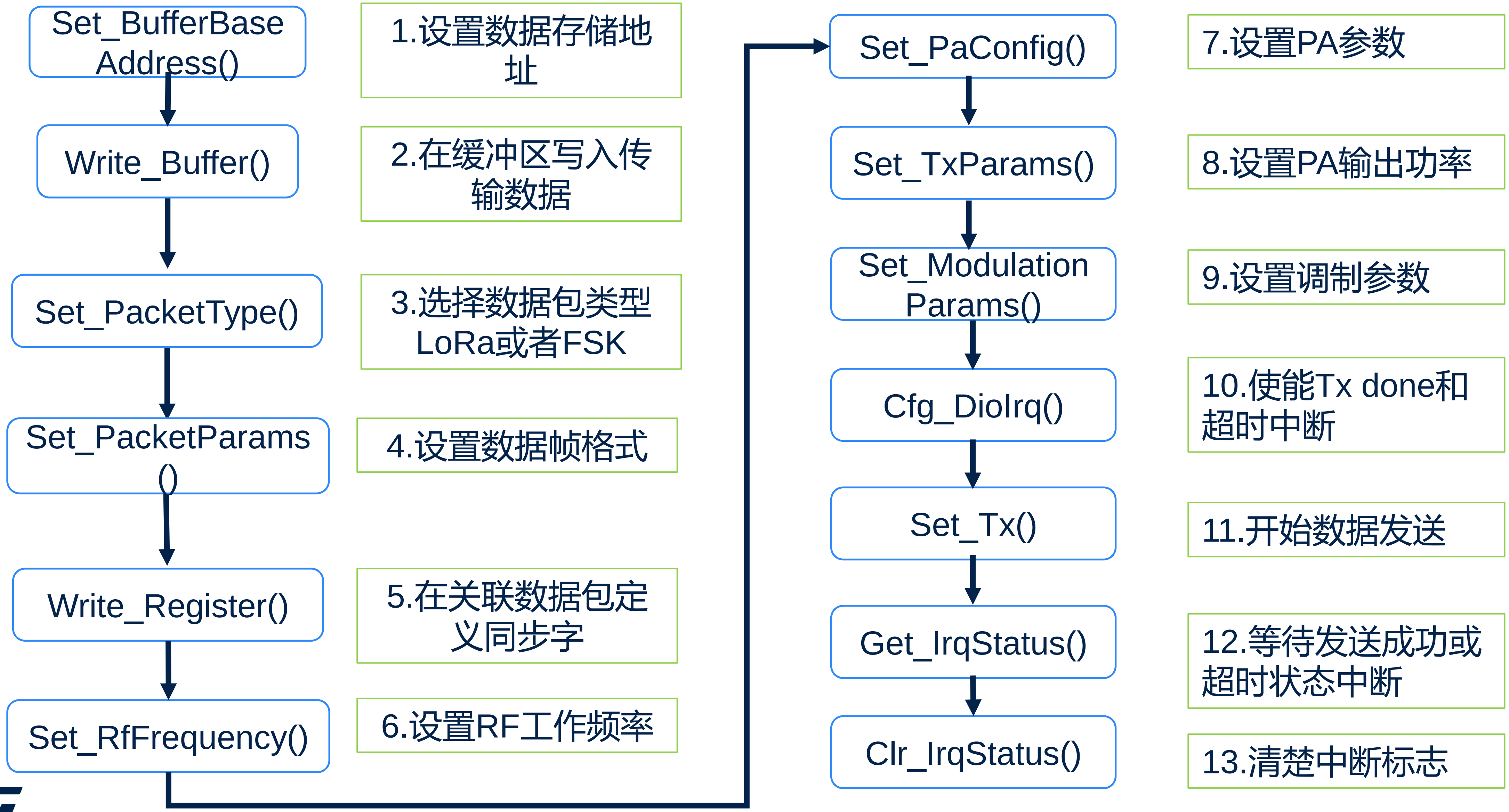


SubGHz 的数据缓冲区

- 256 字节的 RAM
- **TX 数据缓冲区**
  - 由软件写入，硬件读取
  - 参数:
    - TxBaseAddr,
    - TxBufferPointer
    - PayloadLength
- **RX 数据缓冲区**
  - 由硬件写入，软件读取
  - 参数:
    - RxBaseAddr,
    - RxStartBufferPointer
    - RxPayloadLength

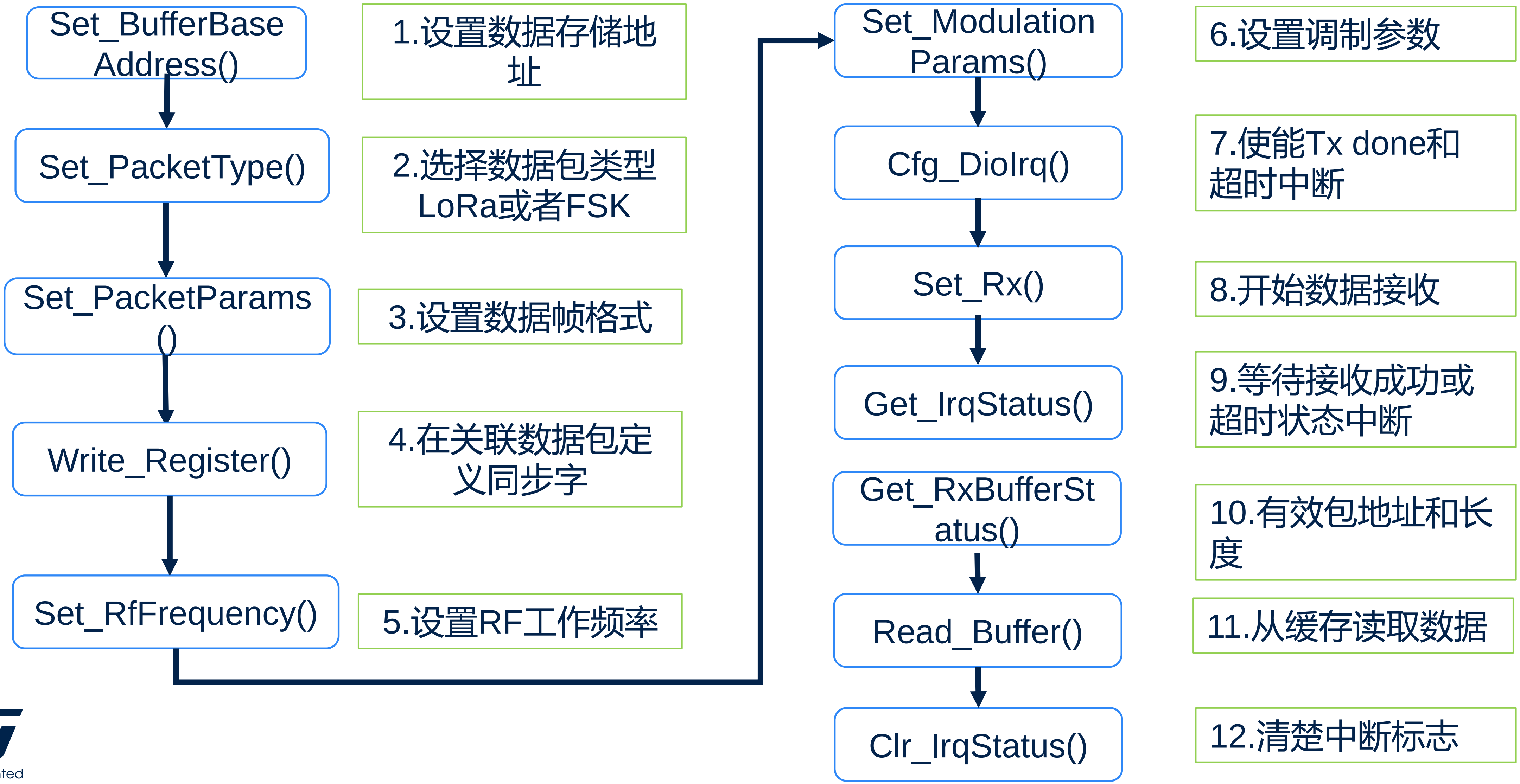


LoRa™, (G)FSK, (G)MSK 发送操作顺序



Set_BufferBase Address()	1.设置数据存储地址
Write_Buffer()	2.在缓冲区写入传输数据
Set_PacketType()	3.选择数据包类型LoRa或者FSK
Set_PacketParams ()	4.设置数据帧格式
Write_Register()	5.在关联数据包定义同步字
Set_RfFrequency()	6.设置RF工作频率
Set_PaConfig()	7.设置PA参数
Set_TxParams()	8.设置PA输出功率
Set_Modulation Params()	9.设置调制参数
Cfg_DioIrq()	10.使能Tx done和超时中断
Set_Tx()	11.开始数据发送
Get_IrqStatus()	12.等待发送成功或超时状态中断
Clr_IrqStatus()	13.清楚中断标志

LoRa™, (G)FSK接收操作顺序



Set_BufferBase Address()

1.设置数据存储地址

Set_PacketType()

2.选择数据包类型LoRa或者FSK

Set_PacketParams()

3.设置数据帧格式

Write_Register()

4.在关联数据包定义同步字

Set_RfFrequency()

5.设置RF工作频率

Set_Modulation Params()

6.设置调制参数

Cfg_DioIrq()

7.使能Tx done和超时中断

Set_Rx()

8.开始数据接收

Get_IrqStatus()

9.等待接收成功或超时状态中断

Get_RxBufferStatus()

10.有效包地址和长度

Read_Buffer()

11.从缓存读取数据

Clr_IrqStatus()

12.清楚中断标志

life.augmented

使用射频接口发送数据示例代码

```
// Radio initialization
RadioEvents.TxDone = OnTxDone;
RadioEvents.RxDone = OnRxDone;
RadioEvents.TxTimeout = OnTxTimeout;
RadioEvents.RxTimeout = OnRxTimeout;
RadioEvents.RxError = OnRxError;
Radio.Init( &RadioEvents );

// Radio Tx Configuation in Lora mode
Radio.SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                   LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                   LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                   true, 0, 0, LORA_IQ_INVERSION_ON, 3000 );

// Radio Set Rf frequency
Radio.SetChannel( RF_FREQUENCY );

// Radio send a buffer
Radio.Send( Buffer, BufferSize );
```

life.augmented

使用射频接口接收数据示例代码

```
// Radio initialization
RadioEvents.TxDone = OnTxDone;
RadioEvents.RxDone = OnRxDone;
RadioEvents.TxTimeout = OnTxTimeout;
RadioEvents.RxTimeout = OnRxTimeout;
RadioEvents.RxError = OnRxError;
Radio.Init( &RadioEvents );

// Radio Rx Configuation in FSK mode
Radio.SetRxConfig( MODEM_FSK, FSK_BANDWIDTH, FSK_DATARATE,
         0, FSK_AFC_BANDWIDTH, FSK_PREAMBLE_LENGTH,
         0, FSK_FIX_LENGTH_PAYLOAD_ON, 0, true,
         0, 0,false, true );

// Radio Set Rf frequency
Radio.SetChannel( RF_FREQUENCY );

// Radio set in Rx mode a buffer
Radio.Rx( RX_TIMEOUT_VALUE );
```

释放您的创造力

- /STM32
- @ST_World
- community.st.com
- www.st.com/STM32WL
- wiki.st.com/stm32mcu
- github.com/意法半导体
- STM32无线模块 – 视频播放列表
- STM32WL博客文章

STM32

life.augmented

Thank you

life.augmented