ManThink
门思科技

基于LWS+MPOS的
LORA/LORAWAN模组

快速开发指南

应用层服务器

根据应用需求开发应用层服务器

LoRaWAN 服务器
阿里巴巴LinkWAN

数据通过LoRaWAN基站上行到LoRaWAN Server

LoRaWAN
基站

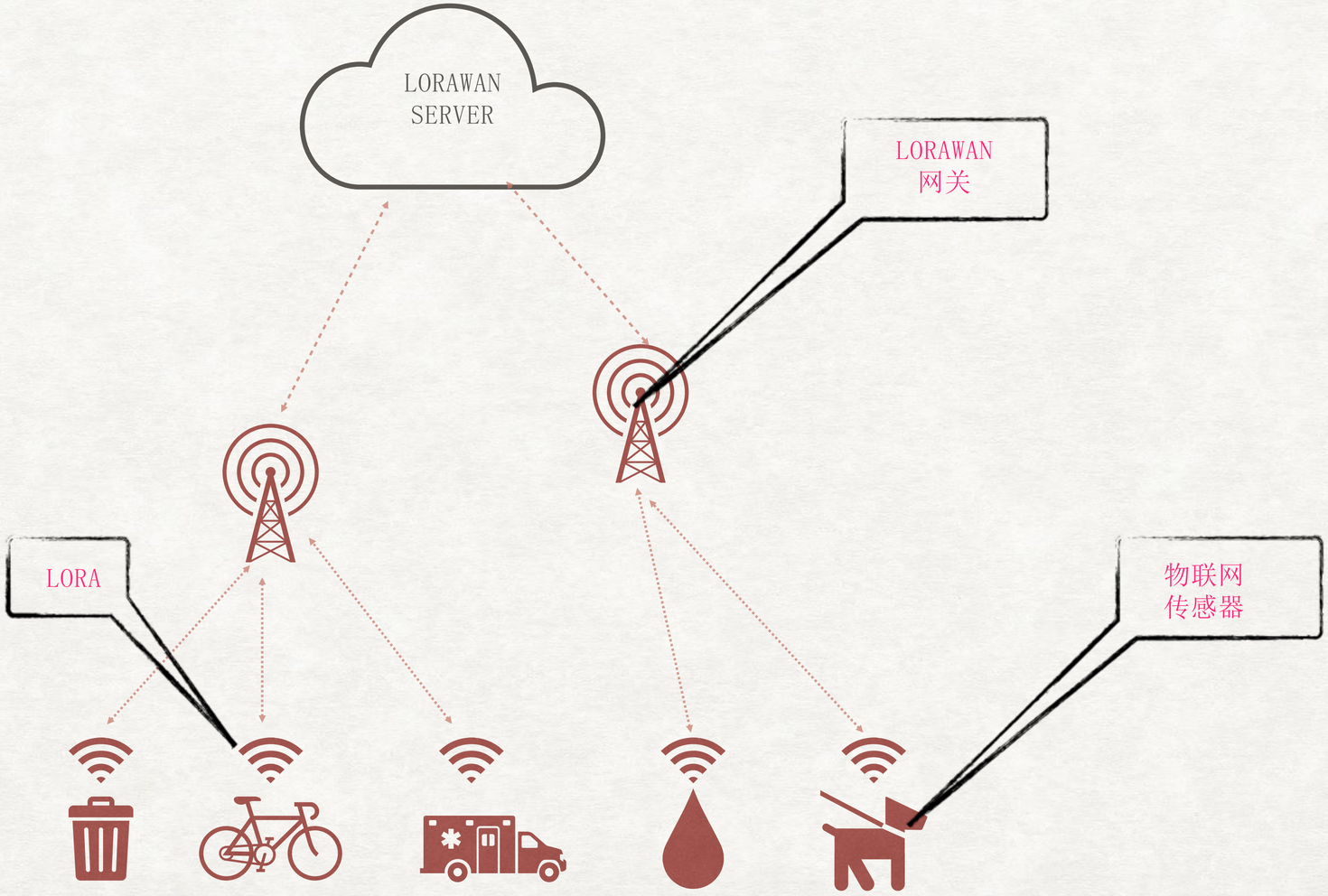
门思科技提供多种型号的基站产品
包括室内版、室外版，半双工/全双工
等多种频点的高性能基站

门思科技
智能设备

门思科技的OM系列模组（OM402/OM411）
支持客户基于模组的二次开发，门思科技提供
LWS协议栈及MP-OS操作系统，可低成本快速
实现物联网应用开发

使用门思科技的LoRa/LoRaWAN模组
或DTU可以快速实现传统设备的
物联网化

DONEC QUIS NUNC

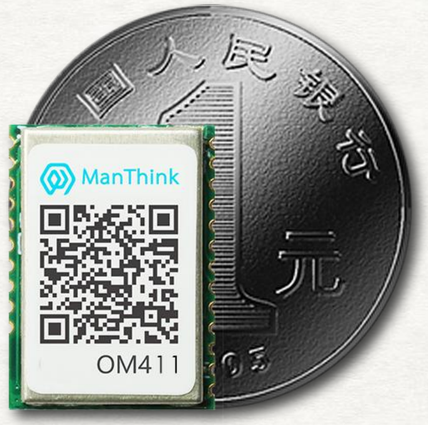


OM402/OM411 硬件特性

超低功耗
- <3uA 休眠电流
- 快速自动休眠 (<1ms)

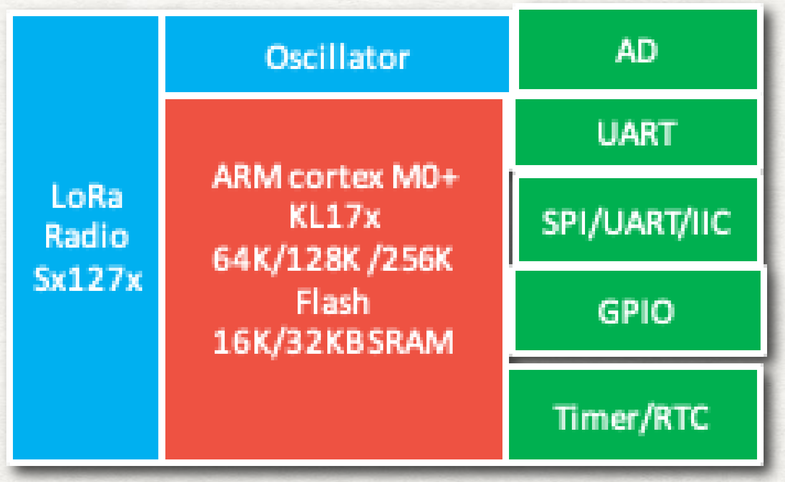
高性能
- -138dBm 接收灵敏度
- 基于Cortex-M0+ MCU

超小尺寸
- 13mmx17.8mm
- LCC 封装



Parameters	
Radio Frequency	410~510MHz(OM402),860~1020MHz(OM802)
Transmission Power	5~20dBm
Receiving Sensitivity	-138dBm@292bps
Harmonic Suppression	≤1GHz：<-36dBm, ＞1GHz：＜-30dBm
Sleep Current(typical)	<3uA
Size	17.8mm x 13.0mm x 2.0mm
MCU	KL17x(32bit cortex-M0+)
Peripherals	SPI/UART/I2C/GPIO/AD
System Memory	64KFlash ,16K SRAM

参数



硬件框图

OM402/OM411 软件特性

Open-System

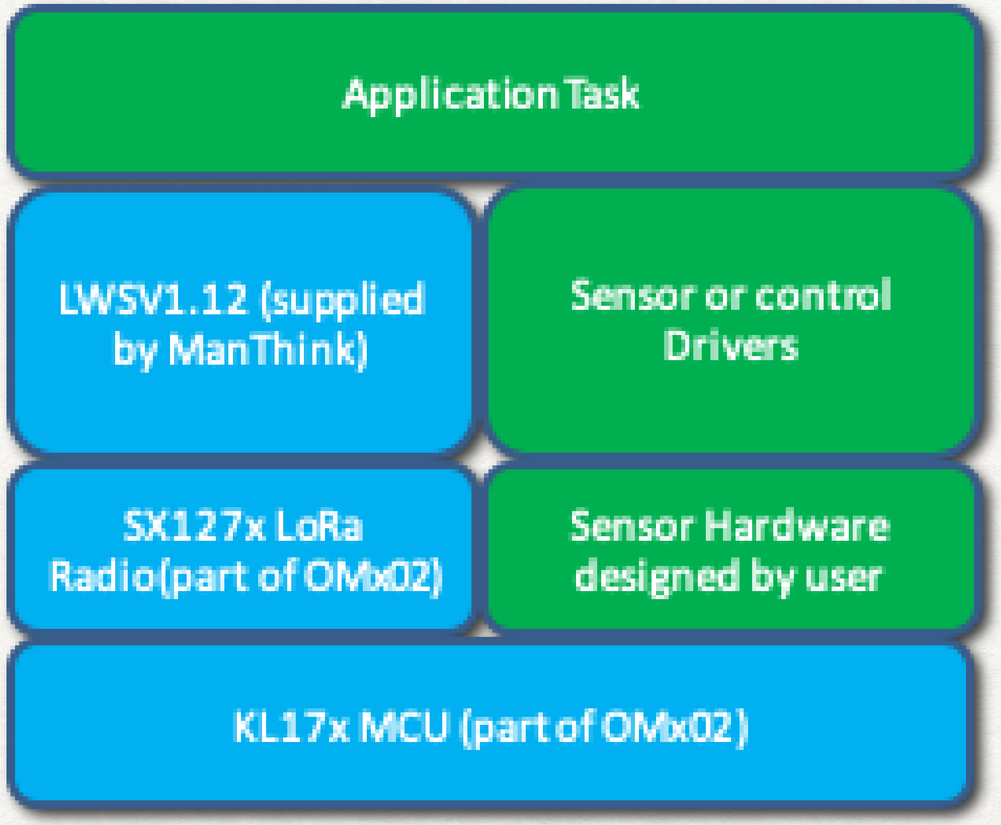
- 开放了精准时间任务系统(Man-Pragnante)
- 自动休眠
- 开放KL17x 的硬件资源用于二次开发
- 提供所有的KL17x 外设驱动
- 提供LoRaWAN的SDK
- 支持透传式LoRaWAN 操作

LoRaWAN 协议

- 支持 Class-A, Class-B and Class-C
- LoRaWAN协议已经通过LoRaWAN认证
- 通过API的方式有利于开发
- 高性能(低丢包率, 多工作模式)

低成本

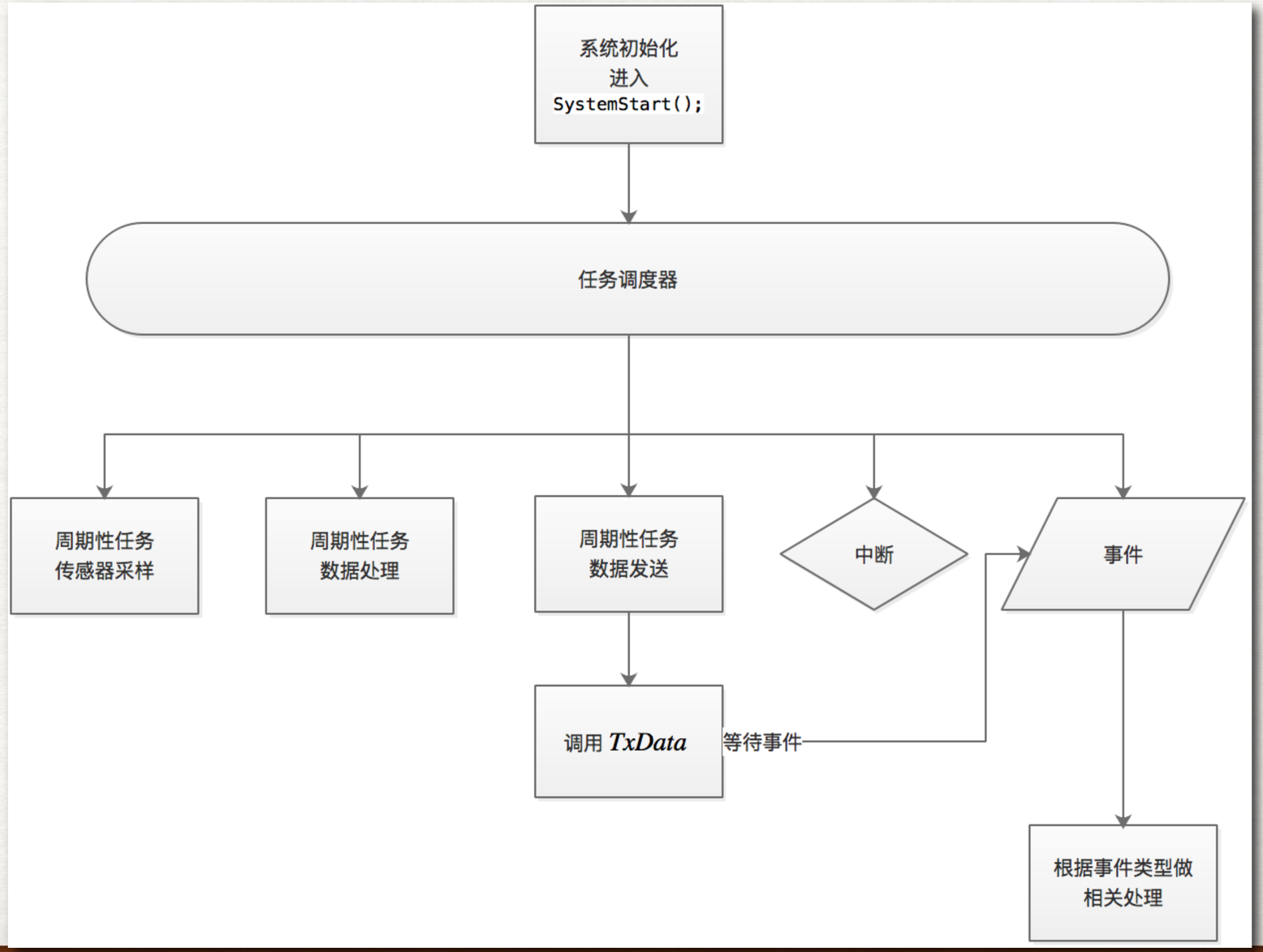
- 不需要额外的MCU
- 缩短产品开发周期



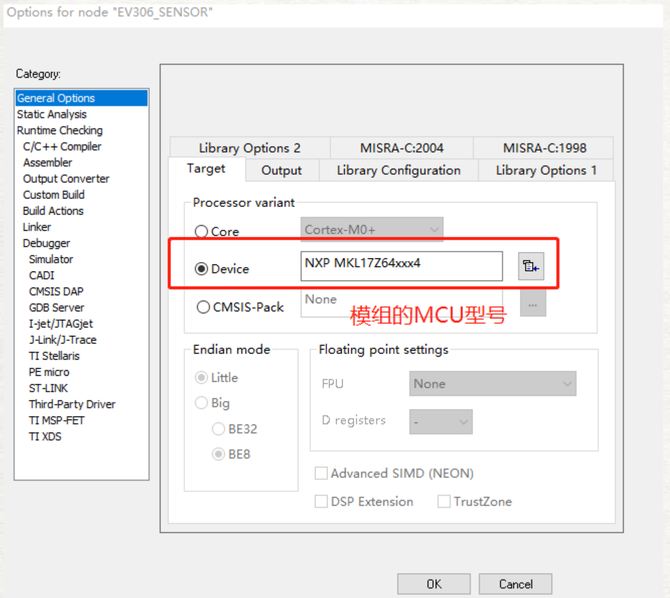
软件架构

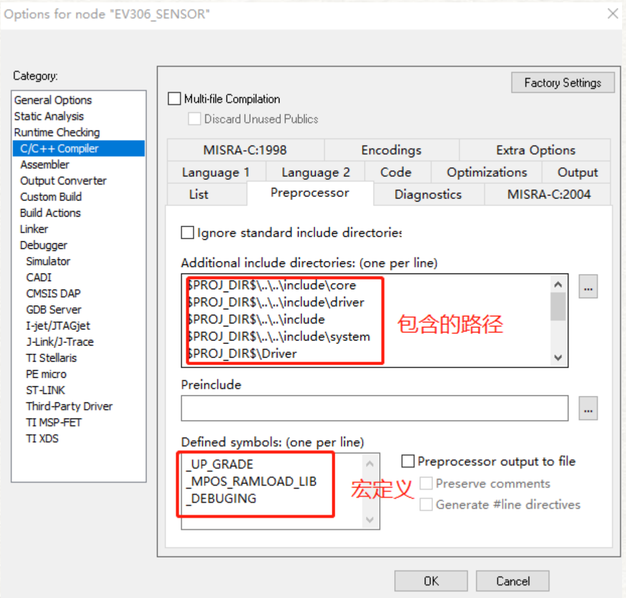
ManThink
门思科技

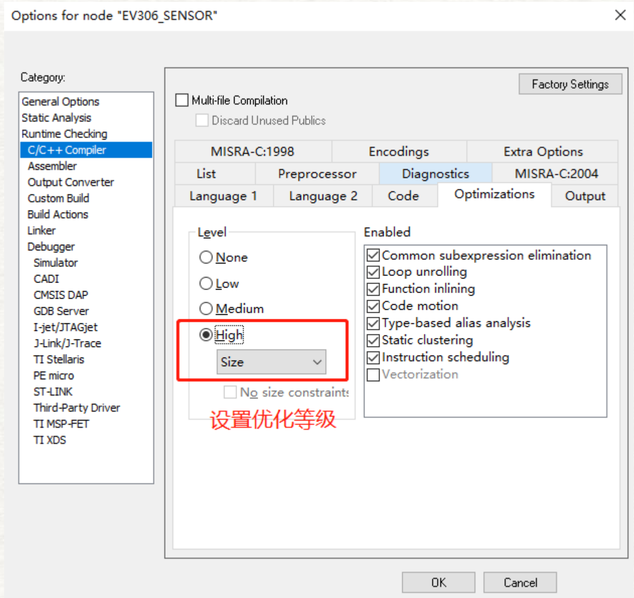
经典物联网传感器数据处理流程

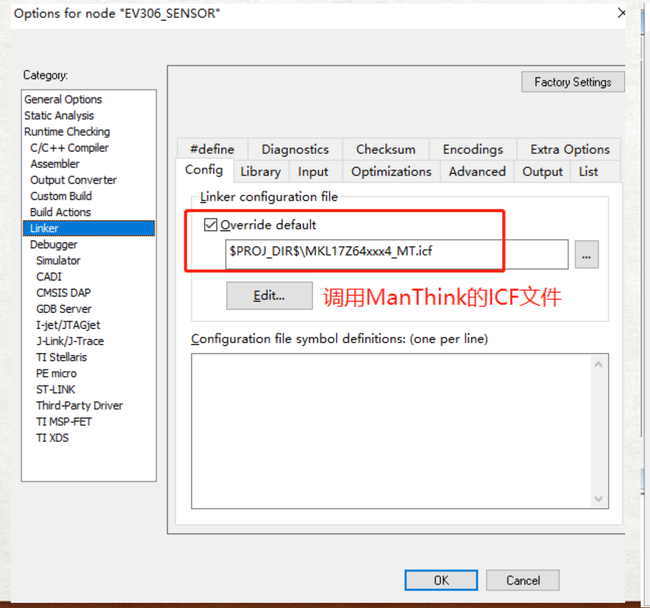


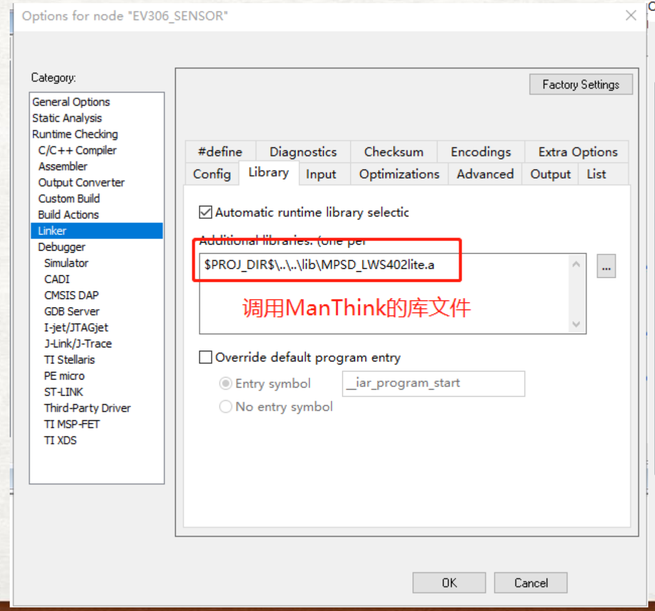
IAR的工程配置

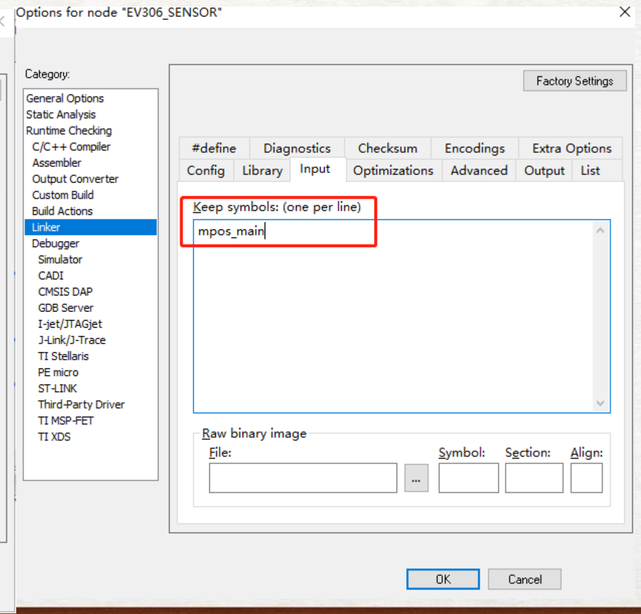

Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Library Options 2
MISRA-C:2004
MISRA-C:1998
Target
Output
Library Configuration
Library Options 1
Processor variant
Core
Cortex-M0+
Device
NXP MKL17Z64xxx4
CMSIS-Pack
None
模组的MCU型号
Endian mode
Little
Big
BE32
BE8
Floating point settings
FPU
None
D registers
Advanced SIMD (NEON)
DSP Extension
TrustZone
OK
Cancel


Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Multi-file Compilation
Discard Unused Publics
MISRA-C:1998
Encodings
Extra Options
Language 1
Language 2
Code
Optimizations
Output
List
Preprocessor
Diagnostics
MISRA-C:2004
Ignore standard include directories
Additional include directories: (one per line)
$PROJ_DIR$\..\..\include\core
$PROJ_DIR$\..\..\include\driver
$PROJ_DIR$\..\..\include
$PROJ_DIR$\..\..\include\system
$PROJ_DIR$\Driver
包含的路径
Preinclude
Defined symbols: (one per line)
_UP_GRADE
_MPOS_RAMLOAD_LIB
_DEBUGING
宏定义
Preprocessor output to file
Preserve comments
Generate #line directives
OK
Cancel


Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Multi-file Compilation
Discard Unused Publics
MISRA-C:1998
Encodings
Extra Options
List
Preprocessor
Diagnostics
MISRA-C:2004
Language 1
Language 2
Code
Optimizations
Output
Level
None
Low
Medium
High
Size
No size constraints
设置优化等级
Enabled
Common subexpression elimination
Loop unrolling
Function inlining
Code motion
Type-based alias analysis
Static clustering
Instruction scheduling
Vectorization
OK
Cancel


Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
#define
Diagnostics
Checksum
Encodings
Extra Options
Config
Library
Input
Optimizations
Advanced
Output
List
Linker configuration file
Override default
$PROJ_DIR$\MKL17Z64xxx4_MT.icf
Edit...
调用ManThink的ICF文件
Configuration file symbol definitions: (one per line)
OK
Cancel


Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
#define
Diagnostics
Checksum
Encodings
Extra Options
Config
Library
Input
Optimizations
Advanced
Output
List
Automatic runtime library selectic
$PROJ_DIR$\..\..\lib\MPSD_LWS402lite.a
调用ManThink的库文件
Override default program entry
Entry symbol
_iar_program_start
No entry symbol
OK
Cancel


Options for node "EV306_SENSOR"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
#define
Diagnostics
Checksum
Encodings
Extra Options
Config
Library
Input
Optimizations
Advanced
Output
List
Keep symbols: (one per line)
mpos_main
Raw binary image
File:
Symbol:
Section:
Align:
OK
Cancel

系统初始化

```
  StackFunction.watchdog=true;    // 是否启用看门狗，debug时需要禁能看门狗
  StackFunction.nosleep=false;    // MPOS提供的模拟休眠功能用于系统debug，release版本产品需要将该参数置为 true
  StackFunction.uart1=true;       //是否使用MPOS提供的UART功能，如果需要自己写UART1的程序，此参数设置为false
  StackFunction.FlashBackup=true;// 设置存储参数到两个flash里面用于备份
  StackFunction.MTprotocol=false; //置为true则使用ManThink定义的串口通信协议格式，帧头为0xFF，0xAA
  mpos_driver.Clock_Init();       // 时钟系统初始化
  mp_userInit=MT_MyInit;          //装载用户自己的初始化函数

void MT_MyInit()
{
   mpos_osfun.SysParaInit(); //系统参数初始化，从flash中的设置参数配置到系统中
   MT_MyHookInit();          //装载hook函数
   MT_MyParaInit();          //初始化用户数据
   MT_MyBoardInit();         //初始化用户的硬件
   MT_LoRaWANParaInitial();  //初始化LoRaWAN参数
   mpos_driver.kickdog();    //踢狗
   mpos_lws.LWS_init();      //LoRaWAN协议栈装载函数
   mpos_driver.kickdog();    //踢狗
   mpos_lws.LoRaWANInit();   //初始化LoRaWAN参数
   MT_SetupSensorTxTask();   //装载周期性任务
   MT_SetupOnceEventTask();  //装载单次执行的任务
}
```

LORAWAN参数初始化

```
RunStatus.Varible.State.Bits.Mode=1;// 设置LoRaWAN的模式 为ClassA
RunStatus.Varible.State.Bits.ADR=1; //启用ADR功能
RunStatus.Varible.State.Bits.OTA=0; //设置LoRaWAN为ABP入网方式
RunStatus.Varible.State.Bits.FDD=1; //模组工作在FDD模式
RunStatus.Varible.DR=0; //初始化模组的通信速率，DR=0 意味SF=12
mpos_lws.paraFWGet (&paraFwReg);  //获取FW的参数
mpos_lws.paraRDGet (&paraRdReg);  //获取频点参数
paraFwReg.SFwRegister.RxWinDelay1=1;  //设置第一个窗口时间为1秒
paraFwReg.SFwRegister.RxWinDelay2=2;  //设置第二个窗口时间间隔为2秒
paraFwReg.SFwRegister.JoinDelay1=5;   //设置OTAA入网的第一个窗口时间间隔为5秒
paraFwReg.SFwRegister.JoinDelay2=6;   //设置OTAA入网的第二个窗口时间间隔为6秒
for(int i=0;i<4;i++)                  //设置4个频段的占空比，根据运营商的需求进行设置
{
    paraFwReg.SFwRegister.DutyBand[i][0]=0;
    paraFwReg.SFwRegister.DutyBand[i][1]=0;
}
mpos_osfun.os_wlsbf8 (paraFwReg.SFwRegister.AppEui, 0x8100000002000001); //设置模组的APPEUI
mpos_osfun.memcpy1 (paraFwReg.SFwRegister.DevKey,AppKey);  //设置模组的DevKey
mpos_lws.paraFWSave (&paraFwReg);                          //保存设置的参数到flash并将新参数装载到协议栈

paraRdReg.SRdRegister.dn2Dr=0;    //设置第2个窗口的DR值
paraRdReg.SRdRegister.Power=22;  //设置射频发送功率，实际功率为设置值-2.当设置为22时，实际功率为20dBm。 OM402 最高支持20dBM，OM411可以支持到22dBm
mpos_osfun.os_wlsbf2 (paraRdReg.SRdRegister.channelMap,0x00FF); //设置频点的channelMap
for(int i=0;i<16;i++)                       //可以设置16个频点
{
  mpos_osfun.os_wlsbf4 (paraRdReg.SRdRegister.Freq[i].SFreq,(470300000+200000*i),(i/2));
  paraRdReg.SRdRegister.Freq[i].DRRange.Bits.HiDr=5;
  paraRdReg.SRdRegister.Freq[i].DRRange.Bits.LoDr=0;
}
mpos_lws.paraRDSave (&paraRdReg);//保存 射频参数到flash并装载到协议栈
```

周期任务

用户可以建立周期性执行任务，通过任务实现业务逻辑。
周期性任务可以设置任务执行次数，执行次数达到后任务自动停止。
用户可以在任务中添加休眠前函数和唤醒函数
用于MCU休眠事件时对硬件的低功耗处理

```
S_periodTask  MT_TaskSensorTx;              //定义一个周期性任务
MT_TaskSensorTx.Task=MT_SensorTx;           // 设置读取传感器信息并发送数据的任务
MT_TaskSensorTx.interval=600000;            //设置执行周期，单位为mS
MT_TaskSensorTx.cycles=0xFFFFFFFF;          //设置循环次数，如果设置为0xFFFFFFFF，则为无限循环任务
mpos_osfun.Task_Setup(&MT_TaskSensorTx);    // 装载该任务到系统中，装载任务后，系统会按照设定的周期调用任务函数。
mpos_osfun.Task_Remove(&MT_TaskSensorTx);   // 从系统中移除该任务
```

LORAWAN API

```
void mpos_lws.LWS_SetClassMode (uint8_t classMode);//设置模组的工作模式，1=ClassA，2=ClassB，0=ClassC 。模式切换的结果通过事件通知到用户
LWERRO_t mpos_lws.JoinReset (LWOP_t mode); //启动重新入网，根据之前设置的模式进行OTAA方式入网或者ABP方式入网
                                           //返回值为立即返回，错误结果可通过错误列表查询
Mode:join模式分为LWOP_JOIN 和LWOP_REJOIN两种，入网结果将通过事件的方式通知用户
LWERRO_t mpos_lws.TxData(uint8_t * txBuffer,u1_t lenth,u1_t port,LWOP_t mode);
                          //返回值为立即返回，错误结果可以通过错误列表查询
                          //发送API的执行结果通过事件通知到用户。
txBuffer: 待发送数据的起始地址
lenth: 发送数据包的长度
port: LoRaWAN的端口号
mode: LWOP_LTC=confirm数据包. LWOP_LTU=unconfirmed 数据包
```

LORAWAN 事件

```
void HookUserEvent( mt_ev_t ev ,u1_t port,u1_t * Buffer, u2_t len)
{
  switch( ev )
  {
  case MT_EV_TXDONE:          break; // 无线数据发送结束
  case MT_EV_JOINED:          break; //入网成功
  case MT_EV_JOIN_FAILED:     break; // 入网失败
  case MT_EV_REJOIN_FAILED:   break; //rejoin失败

  case MT_EV_TXOVER_NOPORT:   break; // 数据包发送成功+没有下行数据
  case MT_EV_TXOVER_NACK:     break; // confirm包发送失败
  case MT_EV_TXOVER_DNW1:     break; // 数据包发送成功并在第一窗口收到数据，接收到的数据端口保存在port，数据保存在Buffer，长度保存在len
  case MT_EV_TXOVER_DNW2:     break; //数据包发送成功并在第二窗口收到数据，接收到的数据端口保存在port，数据保存在Buffer，长度保存在len
  case MT_EV_TXOVER_PING:     break; //收到一包ClassB的下行数据，接收到的数据端口保存在port，数据保存在Buffer，长度保存在len
  default:                    break;
  }
}
```