Swaiot物端SDK使用指南

简介：

使用“物端对接工具（iot-tools）”可以导出一包代码物端对接代码。您可以将代码移植到您所设置的平台，实现快速物联对接。

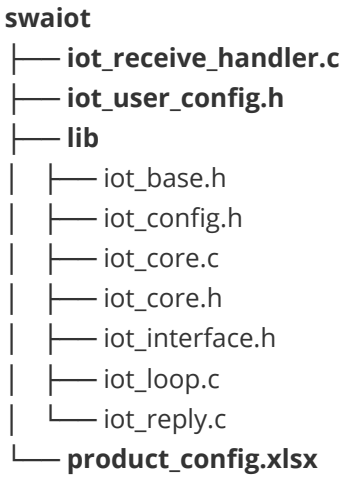
本说明版权属于 深圳创维-RGB电子有限公司，由SWAIOT实验室维护。如有更改，恕不另行通知。20206.16

使用说明：

下面您可以根据如下说明，接入创维SWAIOT物联平台。

文件结构说明

使用“物端对接工具（iot-tools）”导出配置代码后，会在导出文件夹中生成一个名为“swaiot”的文件夹。“swaiot”文件夹的目录结构如下：



备注：

product_config.xlsx文件：设置信息一览表，用于内部对接以及资料归档。

iot_receive_handler.c文件：自动生成的回调函数，用于处理模块或者智慧屏下发的指令。

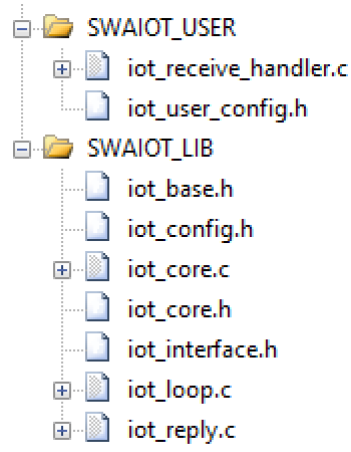
iot_user_config.h文件：自动生成的函数函数接口，内部包含了所有可用的函数接口。

lib文件夹：物联网协议依赖文件夹。需要添加到include路径中，但无需阅读。

请将swaiot文件添加到产品工程中，编译需要将swaiot和swaiot/lib添加到include路径中。

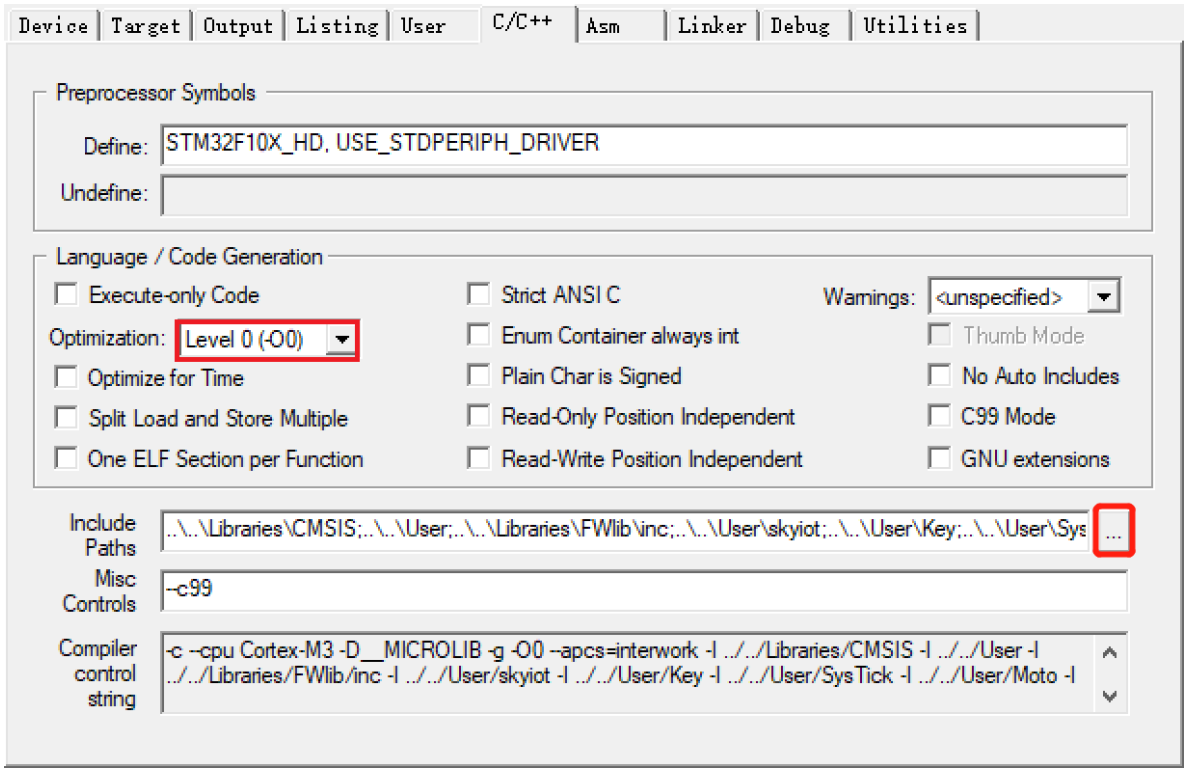
Keil以及MDK移植参考及接口说明

1、将swaiot文件夹添加至工产品原始工程程中。

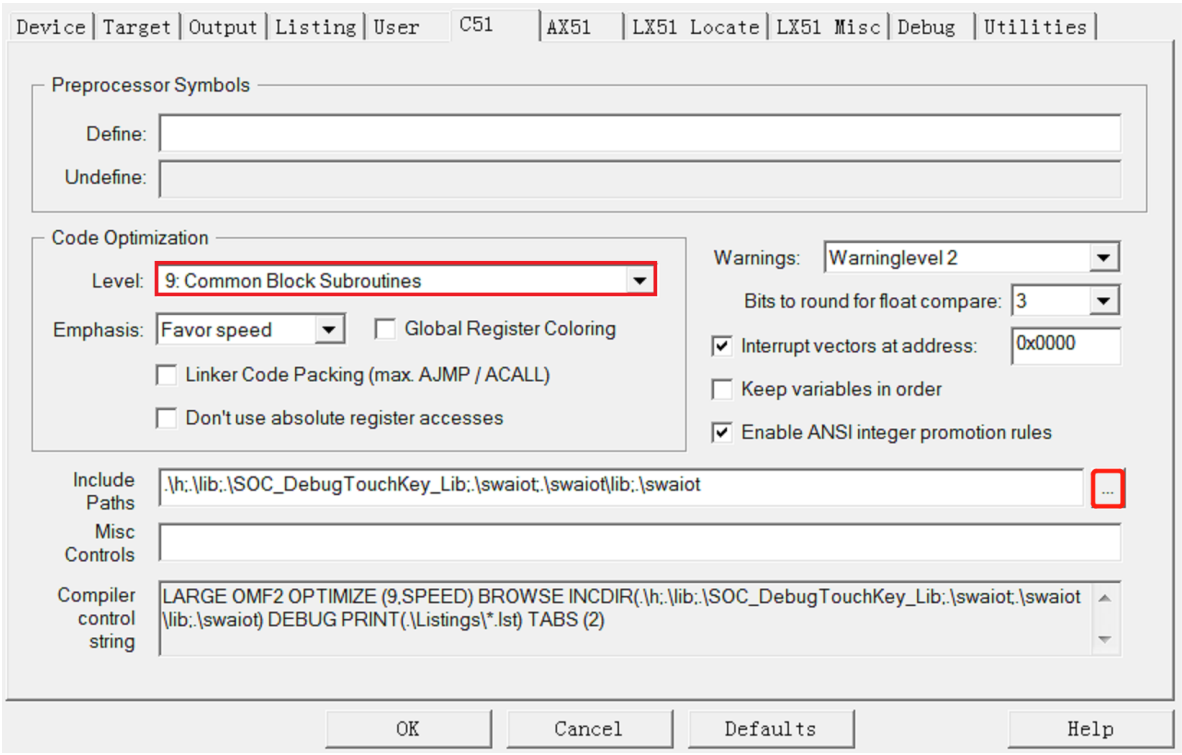


2、将swaiot和swaiot/lib两个文件夹添加至Include Pahts中。根据代码大小设置优化等级

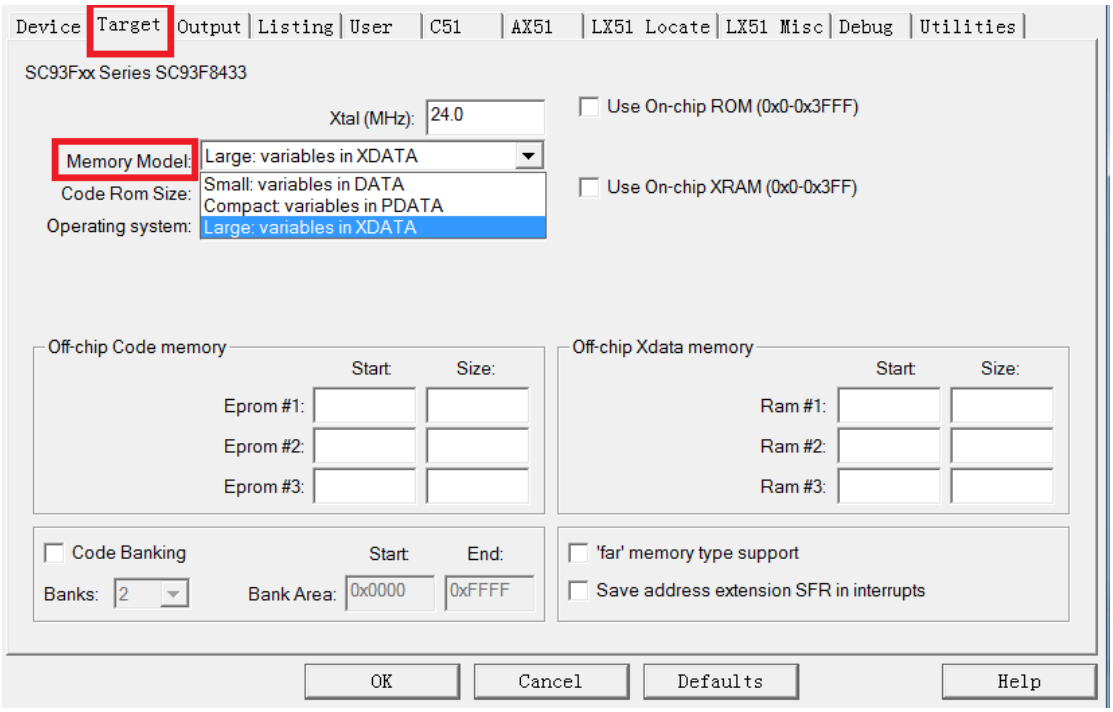
由于SDK中函数使用量较多，因此建议C51代码优化开到最大，其他平台根据Flash和ram的剩余量进适当优化。



（C51平台）



另外对于C51平台存在多种寻址模式可以选择，编译时如果出现DATA溢出，请将内存模式设置为xdata



3、对iot_user_config.h文件的修改及调用，适配串行口发送宏定义。

①、修改如下代码中的宏定义“SEND_TO_NET_CMD”，将宏定义所替换的字符串替换为当前的串口发送函数。

如#define SEND_TO_NET_CMD(dataP,dataSize) uart_write_bytes(dataP, dataSize)

注：函数原型为 void uart_write_bytes(char *dataP, int dataSize)

```
// Add header file of serial port sending function or declaration of serial port
sending function


// Replace "// uart_write_bytes(dataP, dataSize)" with serial port sending
function of your platform
// "Datap" is a char * array, and "datasize" is the array length
/** Define of serial port sending function. "Datap" is a char * array, and
"datasize" is the array length **/
#define SEND_TO_NET_CMD(dataP, dataSize) // uart_write_bytes(dataP, dataSize)
```

②、修改MCU接收中断（也可以是任意处理接收数据的代码节点，不局限于中断），添加接收函数。

在串口数据接收的C文件中引用头文件 iot_user_config.h，并将接收到的数据逐个注如下函数。

注：函数只接收单个byte。如果存在多个，请按顺序逐次调用并传参。

```
/*******************************************************************************
Function....: swaiotPushUartData
Description.: Receive and analyze uart data
Parameters..: byte, enter a byte from uart register
Return......: NONE
*******************************************************************************/
extern void swaiotPushUartData(unsigned char byte);
```

③、在主函数中初始化SDK，并定时调用处理函数。

函数initIotCore：　请在主函数中初始化调用initIotCore函数，函数参数：

version：　字符型指针，指向一个长度为4的char形数组。[1.0.0.0]代表版本：1.0.0.0（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

platformName：字符型指针，指向一个字符串（以\0结尾），代表平台名称。（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

serialNumber：　字符型指针，指向一个长度为32的char形数组。默认全为0。（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

函数swaiotEventHandler_10HZ：请在任意位置以100MS的间隔调用（100ms只是建议值，实际调用中不要求严格的保持100ms）

```
/*******************************************************************************
Function....: initIotCore
Description.: initial firmware version(pointer of 4Bytes buffer) ; MCU's Type
description (string); IOT's serial number(pointer of 32Bytes buffer)
Parameters..: version, firmware version(pointer of 4Bytes buffer)
Parameters..: platformName, String ending with 0
Parameters..: serialNumber, pointer of 32Bytes buffer
Return......: NONE
*******************************************************************************/
extern void initIotCore(char *version, char *platformName, unsigned char
*serialNumber);
```

```
/**
Timing function of automatic parsing
**/
/*****************************************************************************
Function....: swaiotEventHandler_10HZ
Description.: Timing function of automatic parsing, send command to module
Parameters..: NONE
Return......: NONE
*****************************************************************************/
extern void swaiotEventHandler_10HZ(void);
```

④、执行模块功能：（如果使用wifi模块，下述三个接口功能必须实现）

iot_user_config.h中的所有函数都具备了相应的功能，比如调用sendCmdSetModuleConnect()设备将进入配网状态、调用sendCmdSetModuleUnbind()系统进入强制解绑状态、调用sendCmdSetModuleFacTest()模块进入工厂自检模式。（对接智慧屏时，功能还输仅支持获取时间，与属性操作函数，心跳、串码回复灯功能SDK会自动完成）

```
/*****************************************************************************
Function....: sendCmdSetModuleConnect
Description.: set module to connect mode
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*****************************************************************************/
extern void sendCmdSetModuleConnect(void);
/*****************************************************************************
Function....: sendCmdSetModuleFacTest
Description.: set module to factory test mode
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*****************************************************************************/
extern void sendCmdSetModuleFacTest(void);
/*****************************************************************************
Function....: sendCmdSetModuleUnbind
Description.: set module to forced unbind mode
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*****************************************************************************/
extern void sendCmdSetModuleUnbind(void);
```

除此之外，根据您使用iot_tools工具中不同的配置，会生成不同的功能函数，每个函数都有相关的备注提示您其功能。您可以根据个人需求逐个添加调用。

可选扩展功能

可选扩展功能：协议中可选择的扩展功能。注意获取时间是UTC标准时间，请在物端自行解析时区。

☑ 监听模组状态　☑ 支持UTC时间获取　☑ 支持SSID获取　☑ 支持信号强度获取
☑ 支持IP地址获取　☑ 支持MAC地址获取

对应：

```
#ifdef LISTENER_MODULE_STATUS
/*******************************************************************************
Function....: sendCmdGetModuleState
Description.: get module state
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*******************************************************************************/
extern void sendCmdGetModuleState(void);
#endif

#ifdef SUPPORT_GET_UTC
/*******************************************************************************
Function....: sendCmdGetUtc
Description.: get the current UTC time in seconds
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*******************************************************************************/
extern void sendCmdGetUtc(void);
#endif
#ifdef SUPPORT_GET_IP
/*******************************************************************************
Function....: sendCmdGetIp
Description.: get module IP address (Exclusive to WiFi module)
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*******************************************************************************/
extern void sendCmdGetIp(void);
#endif
#ifdef SUPPORT_GET_SSID
/*******************************************************************************
Function....: sendCmdGetIp
Description.: get module SSID (Exclusive to WiFi module)
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*******************************************************************************/
extern void sendCmdGetSsid(void);
#endif
#ifdef SUPPORT_GET_MAC
/*******************************************************************************
Function....: sendCmdGetMac
Description.: get module MAC address (Exclusive to WiFi module)
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*******************************************************************************/
extern void sendCmdGetMac(void);
#endif
#ifdef SUPPORT_GET_RSSI
/*******************************************************************************
Function....: sendCmdGetRssi
Description.: get module RSSI (need special modules)
```

```
Parameters..: NONE
Return......: NONE
Explain.....: NONE
*********************************************************************************/
extern void sendCmdGetRssi(void);
#endif
.....................
```

⑤、属性发送函数（**由于为了更简单的兼容51系芯片，SDK中不使用函数指针进行动态注入**）

根据您使用iot_tools工具中配置的不同的属性，系统会自动生成属性更新函数。函数命名规则为：

extern void sendcmdProp**XxxXxx**(type value)

其中XxxXxx代表属性名被转义成驼峰写法的字符串名称,例如：

对应：

```
#ifdef CMD_MASK_PROD_POW_S
/*********************************************************************************
Function....: sendcmdPropPowS
Description.: update property POW_S
Parameters..: value, property of type char
Return......: NONE
Explain.....: NONE
*********************************************************************************/
extern void sendcmdPropPowS(char value);
#endif
```

您可以直接调用并更新属性（传输是否成功，您可以使用iot_tools中的调试助手进行本地验证）

4、对iot_receive_handler.c文件的接口说明

在iot_receive_handler.c文件中描述了所有接收事件的回调函数，如更新一个属性、模块状态发生变化、更新当前时间等。具体函数说明均可参考源码中所附带的备注。下面对自动生成函数的结构予以说明。

①、模块状态监听函数（智慧屏对接无效）

如果您在iot_tools工具配置了监听模组状态，生成文件中会包含此函数，您可以根据枚举内容来确定模块具体状态。（具体状态有那些，可以参考物联网通信协议指令说明）

```
#ifdef LISTENER_MODULE_STATUS
/*********************************************************************************
Function....: moduleStateChangeHandler
Description.: Callback function for updating module state
Parameters..: ENUM_MODULE_STATE_SKY state : it is an enumeration of states.
Please refer to the iot_base.h
Return......: NONE
*********************************************************************************/
  void moduleStateChangeHandler(ENUM_MODULE_STATE_SKY state){
      // update state
  }
#endif
```

您可以通过此函数的状态反馈来确定WIFI灯或其他状态提示方式的状态。以WIFI模块为例（*智慧屏不支持此功能，不会触发。*），收到：

（WIFI提示灯状态）

STA_STATE_CONFIGING：WIFI提示灯慢速闪烁（5分钟）

STA_STATE_CONNECTED_SERVER：WIFI提示灯常亮

STA_STATE_CONNECTED或STA_STATE_DISCONNECT_SERVER：WIFI提示灯熄灭

（出场检测提示状态）

STA_STATE_IN_FACTORY_MODE：进入工厂自检，需要提示

STA_STATE_FACTORY_OVER：工厂自检成功提示

STA_STATE_FACTORY_FAIL：工厂自检失败提示

```
void moduleStateChangeHandler(ENUM_MODULE_STATE_SKY state){
      // update state
   if(state == STA_STATE_NONE){
     // 无效状态，不做任何处理
   }else if(state == STA_STATE_CONFIGING){
     // 配网或联网中，需要提示wifi状态（WIFI指示灯闪烁5分钟）
   }else if(state == STA_STATE_CONNECTED){
     // 联网成功，可忽略或提示wifi状态（WIFI指示灯快速闪烁）
   }else if(state == STA_STATE_DISCONNECT_SERVER){
     // 连接断开，需要提示关闭WIFI提示灯
   }else if(state == STA_STATE_CONNECTED_SERVER){
     // 服务器连接成功，需要提示wifi状态（WIFI指示灯常量）
   }else if(state == STA_STATE_IN_FACTORY_MODE){
     // 进入模块工厂自检状态，需要提示，以便于工厂厂测人员处理（如全屏闪烁）
   }else if(state == STA_STATE_FACTORY_OVER){
     // 工厂自检成功，需要提示，以便于工厂厂测人员处理（如全屏熄灭，恢复关机状态）
   }else if(state == STA_STATE_FACTORY_FAIL){
     // 工厂自检失败，需要提示，以便于工厂厂测人员处理（如全常亮，或报错。）
   }else if(state == STA_STATE_BACK_MODE){
     // 忽略
   }else if(state == STA_STATE_ABANDON){
     // 忽略
   }else if(state == STA_STATE_FORCE_UNBIND_OVER){
     // 强制解绑成功，可忽略（之后会收到STA_STATE_CONFIGING）
```

```
    }else if(state == STA_STATE_FORCE_UNBIND_FAIL){
      // 强制解绑失败，可忽略
    }else if(state == STA_STATE_PRODUCT_UPGRADE_MODE){
      // 进入升级状态，如使用升级模式，需要提示
    }else if(state == STA_STATE_PRODUCT_UPGRADE_OVER){
      // 升级成功，如使用升级模式，需要提示
    }else if(state == STA_STATE_PRODUCT_UPGRADE_FAIL){
      // 升级失败，如使用升级模式，需要提示
    }
  }
```

枚举ENUM_MODULE_STATE_SKY的原型存放在“swaiot/lib/iot_base.h”中。

②、属性更新函数：

根据您在iot_tools中配置的不同属性类型，生成文件中会包含不同的属性过呢高兴函数。函数在接收到数据时被调用，通过传参name来判断属性名称，并通过value来传递属性值。

代码中会自动生成通过strcmp生成判断结构，你可以在if语句中添加自己需要的操作。

例如设置了一个名为POW_S的8位属性，代码中会自动添加

if(0 == strcmp(name, PROD_POW_S_NAME))

相关状态判断

```
#ifdef SUPPORT_DATA_TYPE_INT8
/*********************************************************************************
Function....: receiveAPropU8
Description.: Callback function for updating 8-bit property
Parameters..: char *name: property name(string)
Parameters..: iot_u8_t value: command value(uint8_t).
Return......: NONE
Explain.....: Distinguishing property names by using strcmp(s1, s2)
*********************************************************************************/
  void receiveAPropU8(char *name, iot_u8_t value){
      // has a uint8_t data
    if(0 == strcmp(name, PROD_POW_S_NAME)){
      // has POW_S
    }
……
  }
#endif
……
  void receiveAPropU16(char *name, iot_u16_t value){
……
  }
……
  void receiveAPropS32(char *name , iot_s32_t value){
……
  }
……
  void receiveAPropF32(char *name , float value){
……
  }
……
  void receiveAPropStr(char *name , char *value_p){
……
  }
```

其中：

void receiveAPropU8(char *name, iot_u8_t value) 代表八位无符号属性的接收回调函数

void receiveAPropU16(char *name, iot_u16_t value) 代表十六位无符号属性的接收回调函数

void receiveAPropS32(char *name , iot_s32_t value) 代表三十二位有符号属性的接收回调函数

void receiveAPropF32(char *name , float value)代表三十二位单浮点属性的接收回调函数

void receiveAPropStr(char *name , char *value_p) 代表字符串属性的接收回调函数

③、其他功能回调函数

根据您使用iot_tools工具中配置的不同的功能，系统会自动生成不同的回调函数。函数具体功能请参照代码中的代码说明。例如下面便是更新时间的回调函数，使用者可以在函数中添加更新时间相关处理代码。

```
#ifdef SUPPORT_GET_UTC
/*******************************************************************************
Function....: moduleTimeHandler
Description.: Callback function for updating time
Parameters..: uint32_t utc:  UTC time in seconds units, which is based on the
time 1970/1/1 8:0:0
Return......: NONE
Explain.....: 2018.01.01 0:0:0(UTC/GMT 0)  is  80 09 49 5A       5A is high, 80
is low
............: If you use Beijing time (GMT + 8), you need to subtract 8*3600
from the original data
*******************************************************************************/
  void moduleTimeHandler(iot_s32_t utc){
      // exp : 2018.01.01 0:0:0(UTC/GMT 0) is  80 09 49 5A
      // If you use Beijing time (GMT + 8), you need to subtract 8*3600 from the
original data
  }
#endif
.....
```