|  |  |
| --- | --- |
| MiCO Document Working Group(MDWG) | Jacky Zhang |
| Track Number: AN0030CN | MXCHIP Co., Ltd |
| Version: 1.0 | May 2016 |
| Category: Application Note |  |

MiCO智能硬件微信直连开发教程（插座版）

**摘要（Abstract）**

本文档描述了微信直连云开发流程，介绍了如何基于MiCO直连微信云实现一个开关控制。

**文档状态（Status of This Document）**

本文档是MiCO Documentation Working Group (MDWG)的产品，用于对MiCO的特性，功能，应用的描述和说明。该文档描述的微信直连云流程仍在公测阶段， 因此软件的功能在正式版本中可能有所变动。

**适用对象（Status of This Document）**

本文档适用于所有MiCOKit开发者、云平台接入需求者，对所有MiCO-IOT物联网智能硬件开发者公开。软件代码平台为MiCOKit-3165，其他平台可以进行代码移植。

**获取更多帮助（More Help）**

MiCO开发团队向您推荐：MiCO开发者学习网站：<http://mico.io/>（至开发者中心），获取更多最新资料。

手机微信“扫一扫”关注：“MiCO总动员”公众号，获取MiCO团队小伙伴最新活动信息。



登录上海庆科官方网站：<http://mxchip.com/>，获取公司最新产品信息。

**版权声明（Copyright Notice）**

Copyright (c) 2016 MDWG Trust and the persons identified as thedocument authors. All rights reserved.

# 微信直连简介

## 微信直连概述

微信团队2016年3月9日正式发布硬件设备直连微信硬件平台的SDK，所谓“直连”指硬件设备直接连接微信提供的服务端（即微信硬件云），厂商无需为硬件搭建专用的远程服务器。“直连”SDK提供设备端到服务端的连接数据通路，同时将提供设备的连接鉴权，局域网发现，固件管理，统计数据上报，状态更新等功能。接入了“直连”SDK的硬件设备，厂商可以使用标准的数据格式进行应用端开发，如公众号内页面可以使用HTML5进行快速开发，或者直接使用微信提供的公众号标准设备面板（如灯控面板） 。

## 微信开放设备标准面板

微信团队2016年3月18日正式发布设备标准面板，使用“微信硬件云标准接入方案”的设备厂商在开发过程中，可以直接调用微信提供的标准操控面板，实现对硬件设备的开关控制、功能设置等操作。微信提供的设备标准面板，进一步降低了硬件厂商的APP及前端开发的门槛，缩短了开发周期，降低了开发成本，厂商无需进行服务端的开发及复杂的原生APP开发即可快速为用户提供设备操控界面。目前微信硬件平台已提供开关控制、灯光亮度调节、空调相关功能调节三类标准面板供厂商及开发者使用，当然你也可以通过HTML5自定义开发公众号设备界面。

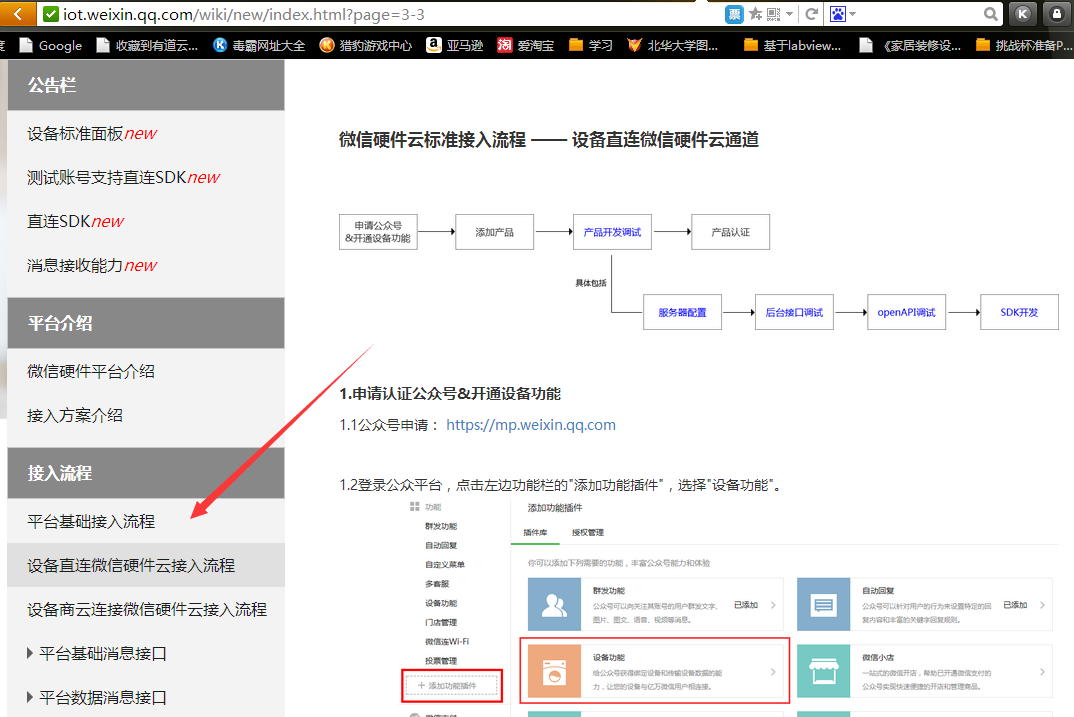


## 微信直连开发准备

1. micokit开发板EMW3165
2. MiCO SDK基于2.4.1版本
3. 已经申请好的微信服务号
4. 详细阅读微信直连SDK开发流程

微信官方在<http://iot.weixin.qq.com/wiki/new/index.html>的微信开发者中心里有微信接入的基本教程和开发工具使用说明，如下图所示，建议开发者在阅读本文档教程前先对微信直连有一个清晰的认识。

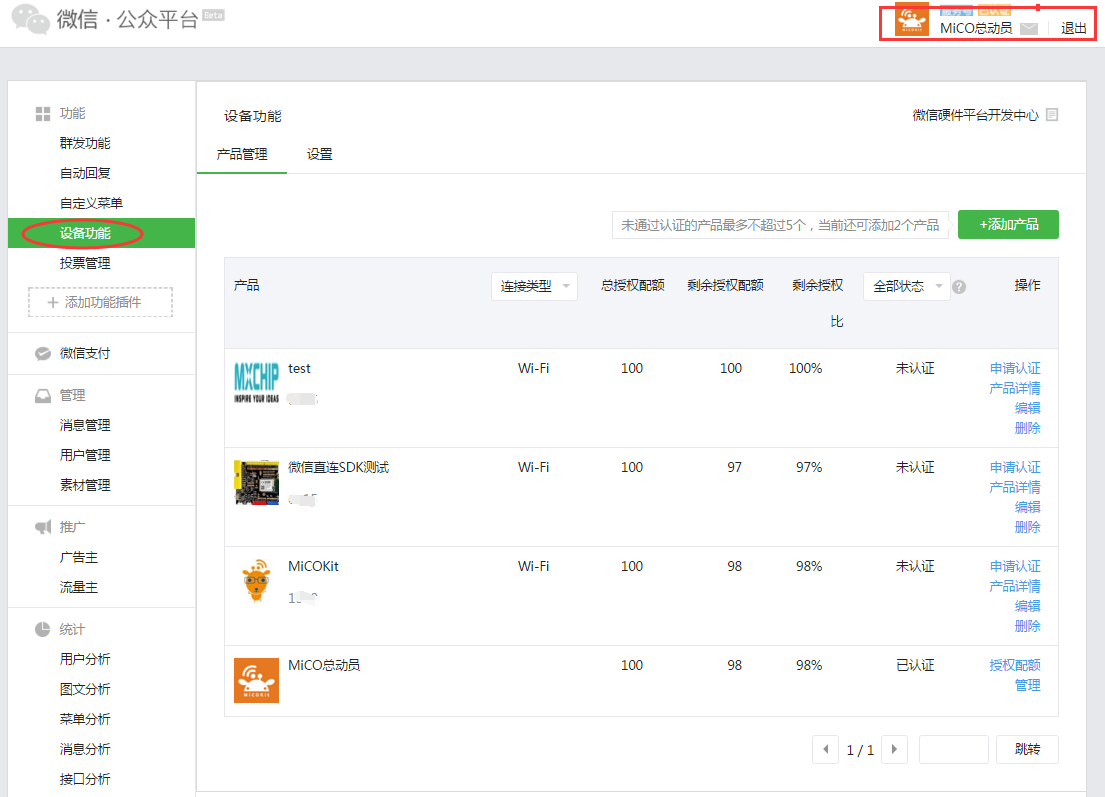




# 微信服务号配置

## 确认微信服务号

需要有已认证的微信服务号并开通设备功能方能开发微信硬件。打开网址<https://mp.weixin.qq.com>。



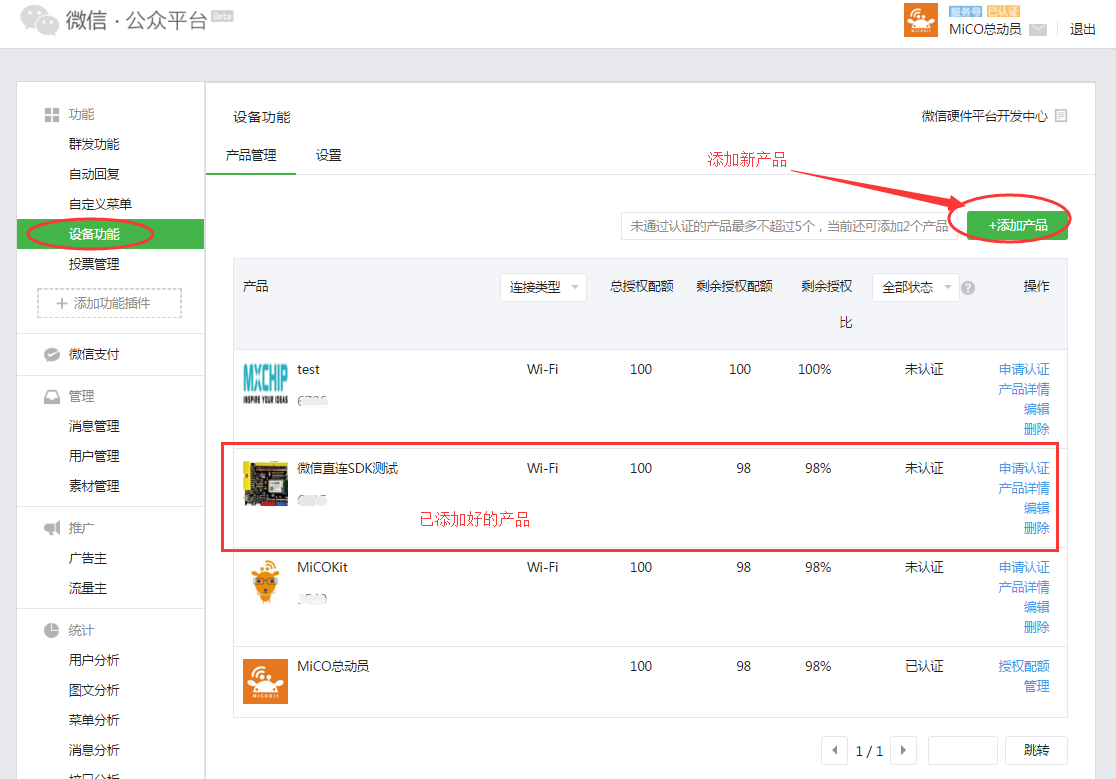
这里需要用到“微信原始ID”、“AppID”、“AppSecrect”，后续微信认证的时候需要用到。





## 添加新产品

在设备功能——>添加产品，填写基础信息，定义产品能力、获取产品编号、生成二维码。添加过程内容较多，请参考下面的图示过程。















## 查看下载二维码

配置成功后可以查看到产品详情，如进入产品阶段需要申请认证扩大配额，具体细节请沟通微信商务。





## 自定义菜单

在公众号关注以后，我们需要在菜单中增加一个子菜单从而能跳转到微信的插座面板进行设备操作。比如，庆科的MiCO总动员账号需要加一个自定义菜单选项。



在跳转网页中填写地址，会跳转到微信插座面板:（请填写自己的appid）

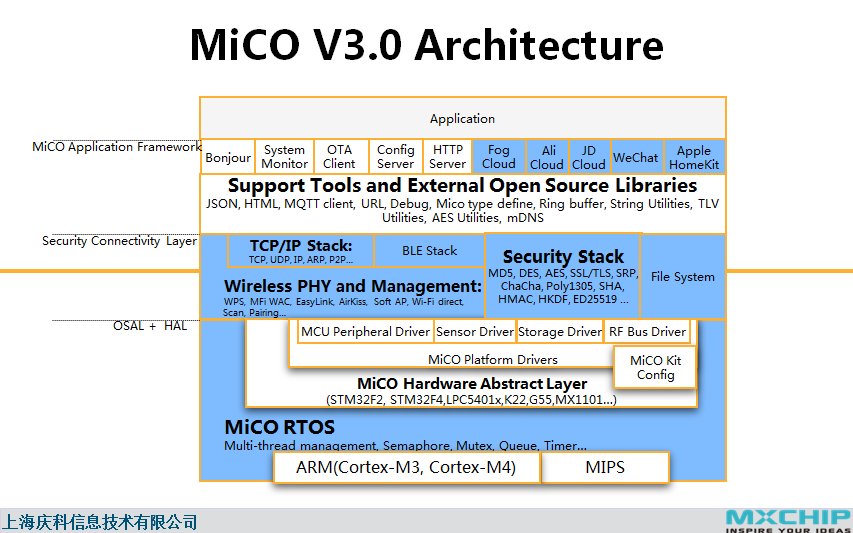
<https://hw.weixin.qq.com/devicectrl/panel/device-list.html?appid=XXXXXXXXXXXXXXXXXX>



# 基于MiCO的微信直连应用

## MiCO应用框架简介

MiCO的应用框架主要包含了：操作系统、WiFi应用层框架，示例程序和典型应用三部分。



最终调用MiCO相关的接口实现功能，它们在MiCO系统中的位置如下：

|  |  |
| --- | --- |
| 目录 | 功能 |
| WiFi应用程序 | |
| Demos/xxxx | 示例程序，用于展示TCP/IP协议栈和应用框架的使用方法，内有基础小例子可供参考 |
| Demos/application | 典型应用，使用WiFi功能的产品级应用范例，wechat\_direct是微信直连例子 |
| WiFi应用框架 | |
| include | 包含MiCO应用头文件 |
| libraries/utilities | 开源工具类代码 |
| libraries\protocols\wechat\direct\_sdk | libwxcloud.a等是由wechat提供的直连SDK，其中airkiss\_porting.c是兼容MiCO的移植文件 |
| libraries\protocols\fog\_cloud | fogcloud是庆科物联云，内置设备绑定及MQTT服务。 |
| MICO\system  MICO\system\easylink | 提供系统服务如存储、配网、局域网发现、远程升级、命令行模式等。其中libairkiss.a是微信提供的配网SDK。 |
| 操作系统 | |
| MICO/core | 操作系统底层实现、射频驱动、TCP/IP协议栈、安全算法等核心功能。 |
| Platform/ Drivers | 操作系统底层驱动实现 |

## MiCO典型应用微信直连流程

为了简化第三方厂商的生产流程，降低接入微信的门槛，微信公众平台提供了新的设备授权接口。在新接口中，deviceid由微信内部生成，作为设备在微信平台的唯一标识，第三方不需要在固件中烧入deviceid。因此设备端只需要准备以下数据作为参数输入即可：

**WEXIN\_original\_ID**，微信服务号原始号，类似：gh\_4201234abcde（注：伪造号不可用）

**WEIXIN\_appid**，微信服务号开发者ID应用ID，类似：wxa12345678abcdefg（注：伪造号不可用）

**WEIXIN\_secret**，微信服务号开发者ID应用密钥，类似：df1234567890eeeeeXXXX7（注：伪造号不可用）

**WEIXIN\_product\_id**，微信服务号中的产品编号，类似：6415（注：伪造号不可用）

注意：代码中请自行修改微信参数

如下是微信认证需要做的三个API请求，

说明：API详细说明请参见微信协议

1. 获取token，接口调用请求说明

http请求方式：GET

https://api.weixin.qq.com/cgi-bin/token?grant\_type=client\_credential&appid=XXXX&secret=XXXX

1. 获取deviceid 和 devicelicence，接口调用请求说明

http请求方式：GET

https://api.weixin.qq.com/device/getqrcode?access\_token=XXXX&product\_id=XXXX

1. 利用deviceid更新设备属性，POST数据最重要的是携带设备的MAC地址

http请求方式：POST

<https://api.weixin.qq.com/device/authorize_device?access_token=XXXX>

POST数据参考如下

|  |
| --- |
| {  "device\_num":"1",  "device\_list":[  {  "id":"gh\_420af5d2de71\_c1b8164a2bcca72c",  "mac":"DOBAE4077ACF",  "connect\_protocol":"4",  "auth\_key":"",  "close\_strategy":"1",  "conn\_strategy":"1",  "crypt\_method":"0",  "auth\_ver":"0",  "manu\_mac\_pos":"-1",  "ser\_mac\_pos":"-2",  "ble\_simple\_protocol": "0"  }  ],  "op\_type":"1"  } |

总的来说，设备端和微信客户端（插座面板）的基本编程流程如下图所示。



## 微信直连应用：wechat\_direct

|  |  |
| --- | --- |
| 文件 | 功能 |
| airkiss\_cloud.c  airkiss\_cloud.h | 开启2个线程，线程1：微信loop线程检测消息响应，线程2：设备汇报状态给微信硬件云。  设置2个回调，回调1：通知响应回调，回调2：事件响应回调 |
| mico\_config.h | 宏定义，配置系统参数 |
| mico\_main.c | MiCO应用主程序，入口名称为application\_start |
| user\_config.h | MiCO系统持久化存储用户数据，定义了一个结构体和存储的微信参数 |
| WeiXinAuth.h  WeiXinAuth.c | 根据WEXIN\_original\_ID 、WEIXIN\_appid 、WEIXIN\_secret、WEIXIN\_product\_id通过https请求获取微信token、deviceid、devicelicence |
| Projects\STM32F4xx\demo\EWARM\demo.eww | IAR工程入口，进入工程修改WeiXinAuth.h文件修改4个参数修改即可实现开发者自己的微信直连应用 |

以下步骤描述了一个基于MiCO的微信直连应用的大致运行流程，以及相关的API的使用，代码并不完整表征一个应用的实现，具体的应用请参考基于MiCO的微信直连源码中包含的示例程序

1. 初始化MiCO系统核心数据，为应用提供虚拟的NVRAM读写空间

g\_mico\_context = mico\_system\_context\_init( sizeof( application\_config\_t) );

g\_app\_context->appConfig = mico\_system\_context\_get\_user\_data( g\_mico\_context );

mico\_system\_init( g\_mico\_context );

2. 设备在airkiss配网以后链路层将打通，已经具备连接能力。

check\_linkstatus( ) ;

3、开始微信认证，为了拿到微信token、deviceid、devicelicence。

start\_weixin\_auth ( );

4、启动微信airkiss discovery 服务，以便局域网能发现设备进行数据交互

err = airkiss\_discovery\_start( WEXIN\_original\_ID, g\_app\_context->appConfig->deviceid);

5、启动微信直连SDK（初始数据和回调函数），微信采用了消息收发引擎，要求SDK进入loop死循环。

void task\_execute\_sdk\_runloop() {

while (0 != airkiss\_cloud\_init((uint8\_t \*)g\_app\_context->appConfig->devicelicence, (uint32\_t)strlen((const char \*)g\_app\_context->appConfig->devicelicence),

&m\_task\_mutex, &m\_malloc\_mutex, heapbuf, sizeof(heapbuf)))

{

airkiss\_cloud\_log("SDK init failed!!!");

mico\_thread\_msleep(1000);

}

//register callback functions

airkiss\_callbacks\_t cbs;

cbs.m\_notifycb = ReceiveNotifyCB;

cbs.m\_respcb = ReceiveResponseCB;

cbs.m\_eventcb = ReceiveEventCB;

airkiss\_regist\_callbacks(&cbs);

//消息机制，类似iOS runloop

uint32\_t sleep\_time;

airkiss\_cloud\_log("Everything is ready!!");

for (;;) {

sleep\_time = airkiss\_cloud\_loop();

mico\_thread\_msleep(sleep\_time);

}

}

6、微信协议定义了标准的JSON数据格式（不可修改），请参阅微信协议。当注册了回调函数以后，插座面板发送的JSON格式数据会被接收，用户请在此进行JSON数据解析操作。

//通知，接收微信消息

void ReceiveNotifyCB(uint32\_t funcid, const uint8\_t\* body, uint32\_t bodylen) {

//字符串转json对象

recv\_json\_object = json\_tokener\_parse((const char\*)body);

//解析微信返回的json数据

}

7、微信下发的数据需要有设备的应答，应答的JSON格式必须遵照微信的协议。

void airkiss\_status\_report\_thread(void \*arg) {

uint32\_t taskid = 0;

for (;;) {

mico\_rtos\_get\_semaphore( &g\_msg\_send\_sem, MICO\_WAIT\_FOREVER );

airkiss\_cloud\_log( "set ok, ack && report current status." );

memset(m\_statusCBmsg, 0, sizeof(m\_statusCBmsg));

if(g\_app\_context->appConfig->power\_switch){

sprintf((char\*)m\_statusCBmsg, "{\"asy\_error\_code\": 0,\"asy\_error\_msg\": \"ok\",\"msg\_id\":%lld,\"msg\_type\":\"%s\",\"services\":{\"operation\_status\":{\"status\":0},\"outlet\":{\"port\_on\_off\":[true,false],\"port\_in\_use\":[true,false]}}}", g\_recv\_msg\_id, g\_recv\_msg\_type);

}

else{

sprintf((char\*)m\_statusCBmsg, "{\"asy\_error\_code\": 0,\"asy\_error\_msg\": \"ok\",\"msg\_id\":%lld,\"msg\_type\":\"%s\",\"services\":{\"operation\_status\":{\"status\":0},\"outlet\":{\"port\_on\_off\":[false,false],\"port\_in\_use\":[true,false]}}}", g\_recv\_msg\_id, g\_recv\_msg\_type);

}

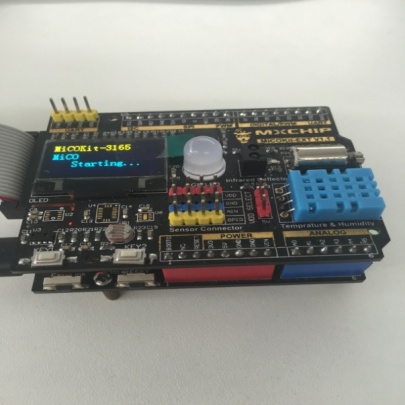
airkiss\_cloud\_log("send status msg: %s", m\_statusCBmsg);

taskid = airkiss\_cloud\_sendmessage(g\_funcid, (uint8\_t \*)m\_statusCBmsg, strlen((const char \*)m\_statusCBmsg));

}

}

8、演示效果请参考：





## 移植微信直连代码到MiCO\_SDK\_2.4.1

到官网mico.io下载MiCO\_SDK\_2.4.1版本SDK，请将版本对应好。

拷贝文件：

1. 拷贝wechat直连SDK到libraries\protocols目录
2. 拷贝wechat\_direct到Demos目录

3、修改json C库代码

在libraries\utilities\json\_c\json\_utils.c文件中需要修改三处以便适应64位整型类型，注意是在函数

int json\_parse\_int64(const char \*buf, int64\_t \*retval)里：

A、int32\_t num64 修改成 int64\_t num64;

B、if (sscanf(buf, "%d", &num64) != 1) 修改成：

if (sscanf(buf, "%lld", &num64) != 1)

C、snprintf(buf\_cmp\_start, sizeof(buf\_cmp), "%d", num64);修改成：

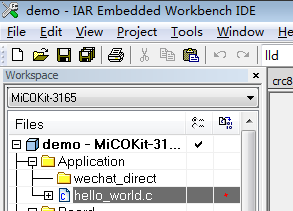
snprintf(buf\_cmp\_start, sizeof(buf\_cmp), "%lld", num64);

4 、说明一下：wechat直连的原始SDK在这个地方下载：

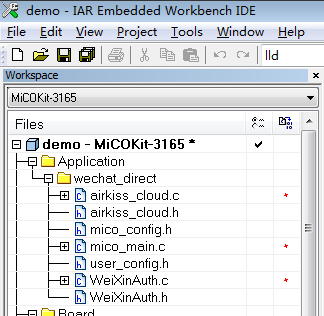


配置工程demo.eww工程：

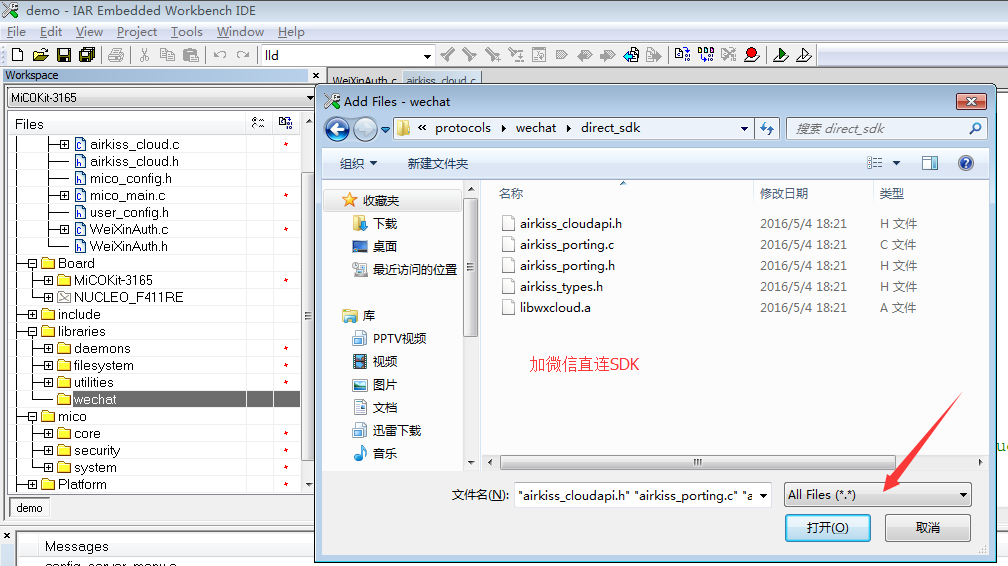
1. 打开IAR工程Projects\STM32F4xx\demo\EWARM\demo.eww
2. 删除hello\_world.c，右键Application->Add->Add Group,命名为 wechat\_direct



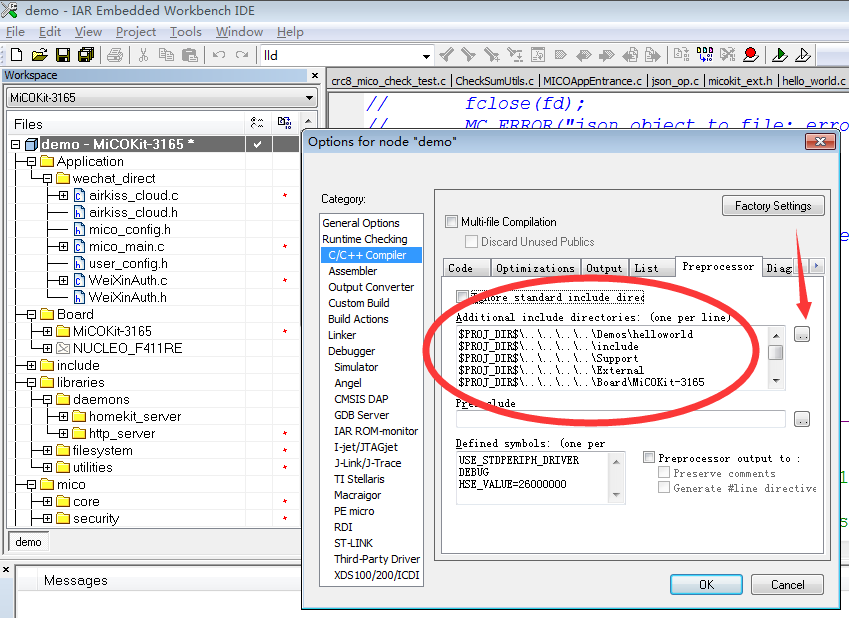
1. 右键点击wechat\_direct，添加应用文件



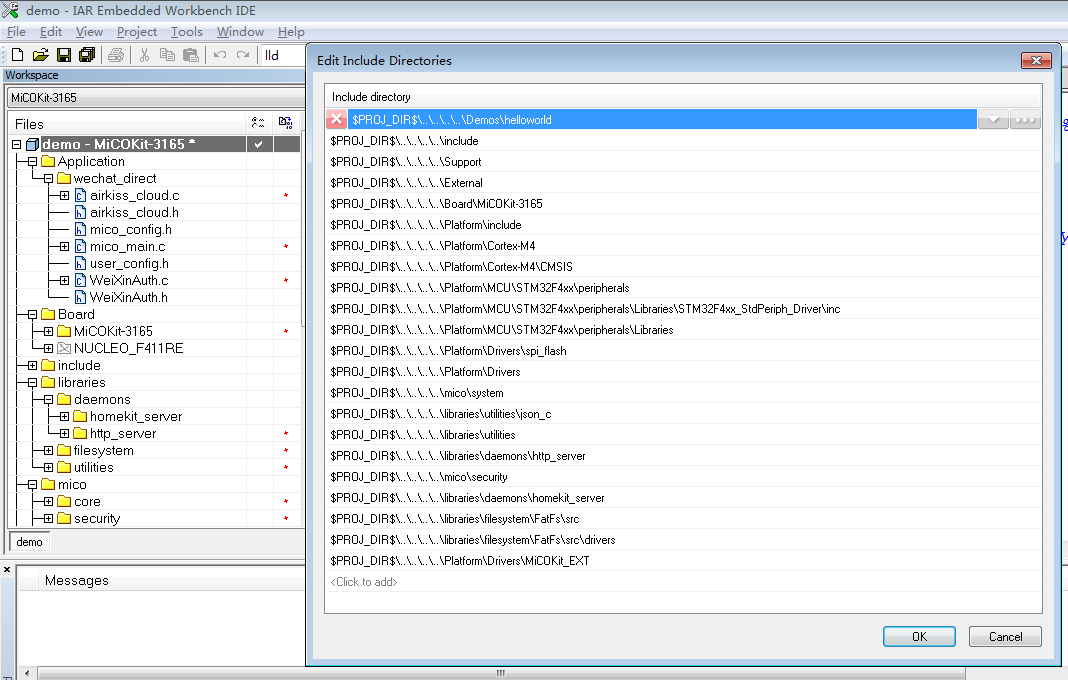
1. 添加微信直连SDK到工程



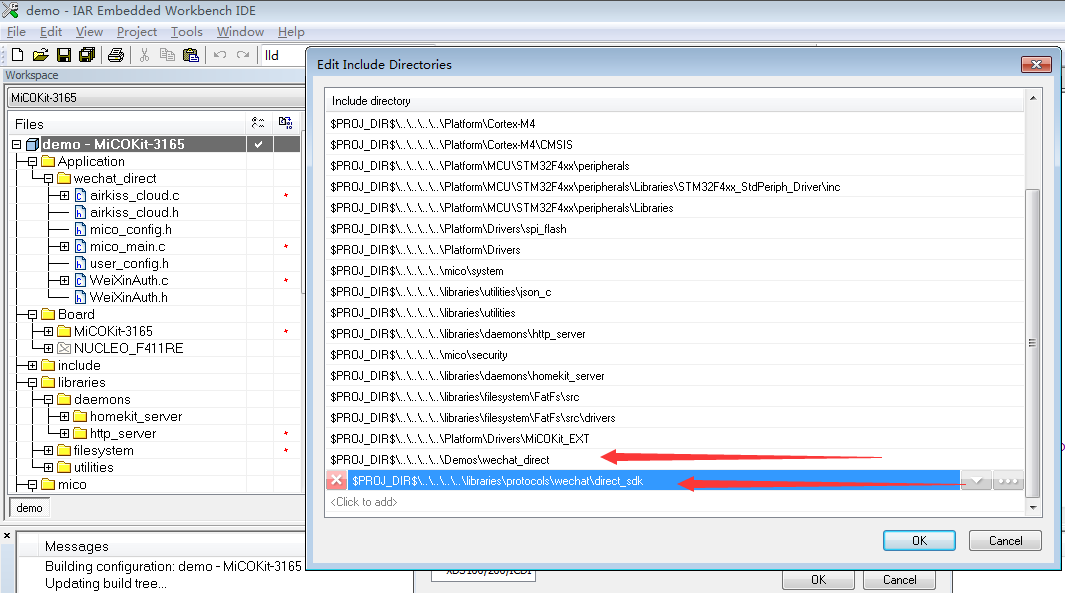
1. 右键点击demo - MiCOKit-3165 Options…配置Preprocessor头文件



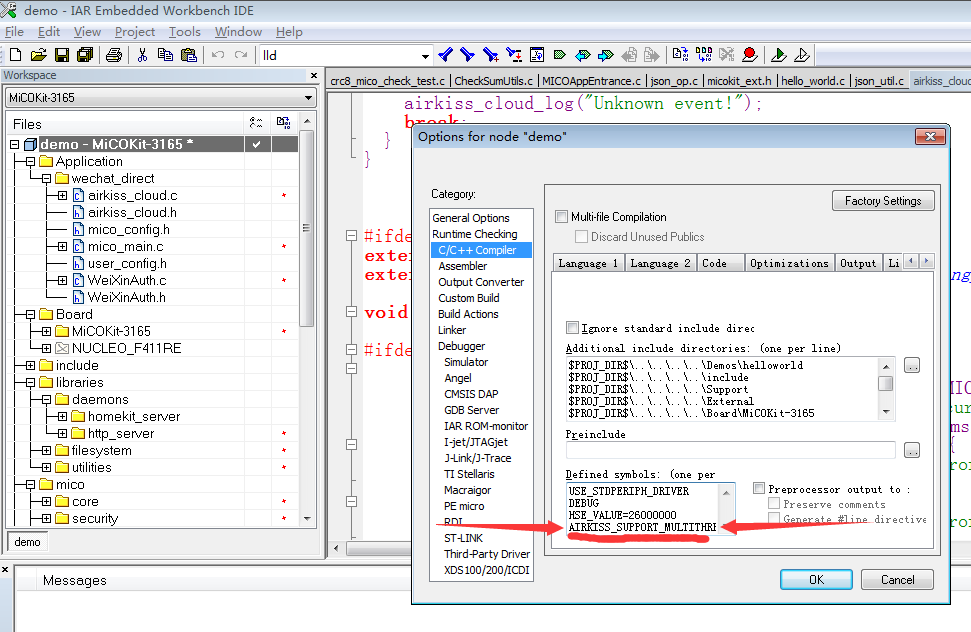
6、修改Preprocessor头文件原本为：



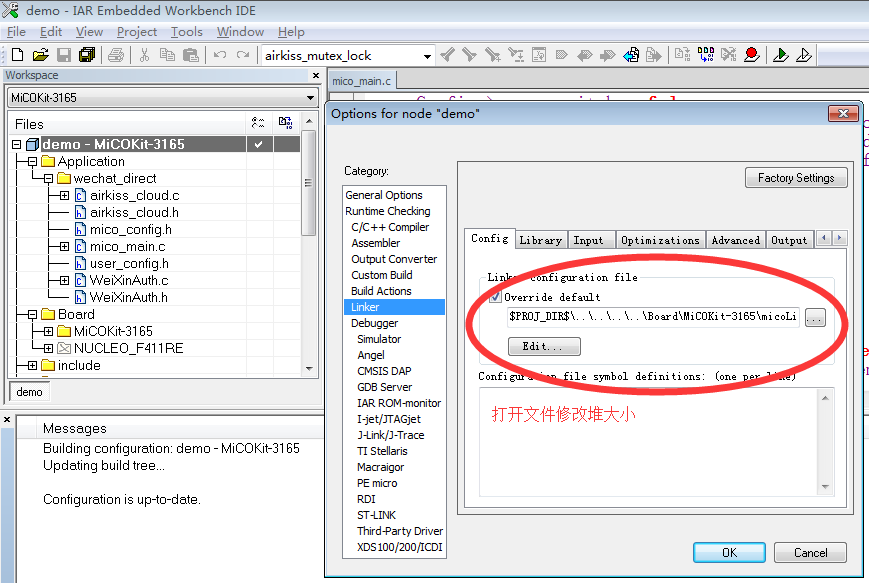
需要删除helloworld路径，修改为wechat\_direct路径：

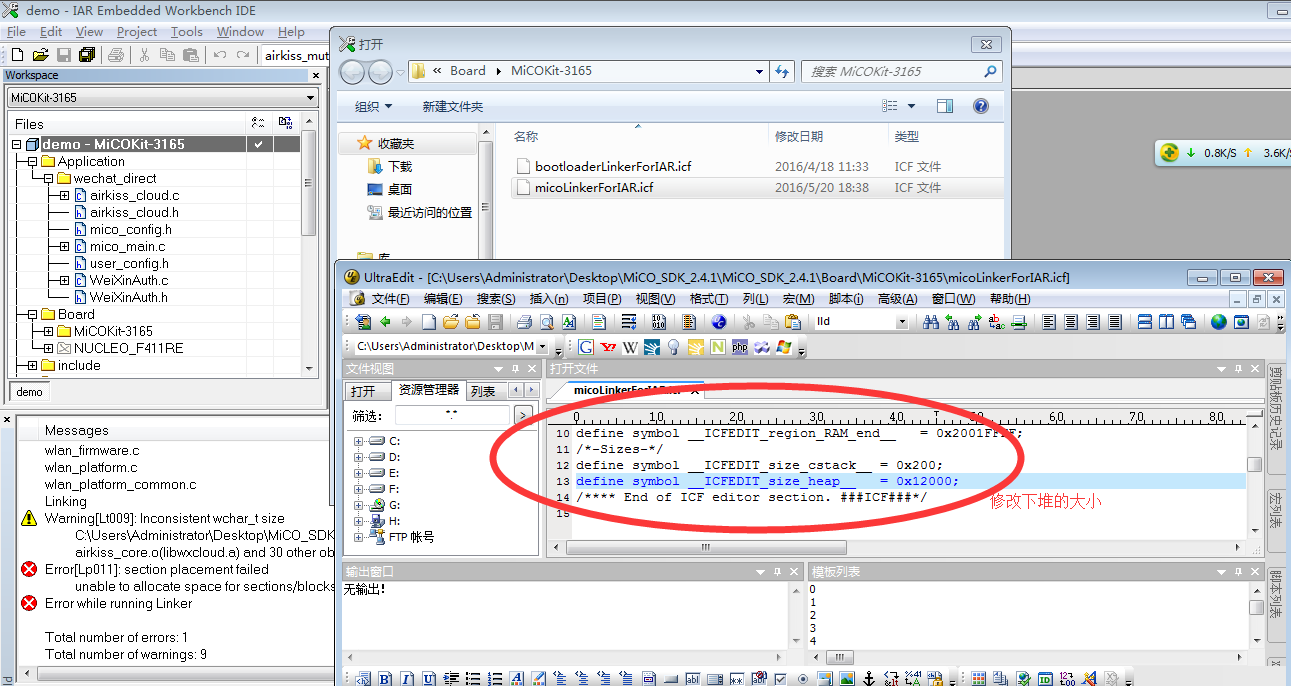


1. 增加一个全局宏定义AIRKISS\_SUPPORT\_MULTITHREAD



1. 最后修改下堆的大小





# 术语表（Glossary）

## AirKiss概述

AirKiss是微信硬件平台为Wi-Fi设备提供的微信配网、局域网发现和局域网通讯的技术。开发者若要实现通过微信客户端对Wi-Fi设备配网、通过微信客户端在局域网发现Wi-Fi设备，或者把微信客户端内的音乐、图片、文件等消息通过局域网发送至Wi-Fi设备，需要在硬件设备中集成相应的AirKiss静态库。

## 微信公众号（服务号、订阅号、企业号）

订阅号：为媒体和个人提供一种新的信息传播方式，构建与读者之前更好的沟通与管理模式。

服务号：给企业和组织提供更强大的业务服务与用户管理能力，帮助企业快速实现全新的公众号服务平台。微信硬件开发需要服务号，庆科的MiCO总动员公众号属于服务号。

企业号：为企业或组织提供移动应用入口，帮助企业建立与员工、上下游供应链及企业应用间的连接。

# 版本更新说明

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 更新内容 | 作者 |
| 2016-5-20 | V1.0 | 1. 初始版本 | Jacky Zhang  Mcx zhao |