



FCU1101 嵌入式控制单元

Embedded Control Unit

Product Manual

Rev. 1.0

2018/09/21

注意事项与维护



1、注意事项

- 请勿带电插拔核心板及外围模块！
- 请遵循所有标注在产品上的警示和指引信息。
- 请保持本产品干燥。如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干。
- 使用中注意本产品的通风散热，避免温度过高造成元器件损坏。
- 请勿在无尘、脏乱的环境中使用或存放本产品。
- 请勿将本产品应用在冷热交替环境中，避免结露损坏元器件。
- 请勿粗暴对待本产品，跌落、敲打或剧烈晃动都可能损坏线路及元器件。
- 请勿使用有机溶剂或腐蚀性液体清洗本产品。
- 请勿自行修理、拆卸本公司产品，如产品出现故障请及时联系本公司进行维修。
- 擅自修改或使用未经授权的配件可能损坏本产品，由此造成的损坏将不予以保修。

2、售后维修

如产品使用过程中出现硬件故障可根据售后服务政策进行维修

服务政策：参见官方网站 www.forlinx.com 售后服务说明

地 址：河北省保定市高开区向阳北大街 2699 号飞凌嵌入式四楼西厅

联 系 人：售后维修部

电 话：0312-3102637 邮编：071000

邮寄须知：建议使用顺丰、圆通或韵达，且不接收任何到付。

技术支持与定制

1、技术支持方式

1.1 电话: 0312-3119192 / 0312-3102619

1.2 论坛: bbs.witech.com.cn

1.3 邮箱:

Linux 技术支持:	linux@forlinx.com
Android 技术支持:	android@forlinx.com
硬件技术支持:	hardware@forlinx.com

1.4 知识库: bbs.witech.com.cn/kb

2、技术支持时间

周一至周五: 上午 9:00—11:30, 下午 13:30—17:00;

公司按照国家法定节假日安排休息, 在此期间无法提供技术支持, 请将问题发送至邮箱或论坛技术支持区, 我们会在工作日尽快给您回复。

3、定制开发服务

我公司提供嵌入式操作系统底层驱动、硬件板卡的有偿定制开发服务, 以缩短您的产品开发周期。

了解定制流程: <http://www.forlinx.com/OEM.htm>

填写需求文档: <http://www.forlinx.com/docs/PR.docx>

发至项目邮箱: project@forlinx.com

资料更新与获取

1、资料的更新

产品相关资料会不断的完善更新，本手册内容亦然如此；当您在使用这些内容时，请确保其为最新状态。

2、更新后如何通知

飞凌嵌入式产品资料更新通知采用微信公众号推送，敬请关注！



订阅号

3、资料如何获取

3.1 网络下载：

请注册并登陆“bbs.witech.com.cn”找到“开发板资料下载”选择对应平台下载；
下载前请阅读《资料下载说明》：<http://bbs.witech.com.cn/thread-67932-1-1.html>。

3.2 光盘：

请联系我公司销售人员购买。

版权声明

本手册版权归保定飞凌嵌入式技术有限公司所有。未经本公司的书面许可，任何单位和个人无权以任何形式复制、传播、转载本手册的任何部分，违者将被追究法律责任。

更新记录

日期	版本	更新内容
2018.09.21	V1.0	用户手册第一版

目录

注意事项与维护	1
技术支持与定制	2
资料更新与获取	3
版权声明	3
更新记录	4
目录	5
第一章 FCU1101 产品介绍	7
1.1 产品简介	7
1.2 应用领域	7
1.3 硬件参数	8
1.4 软件参数	8
第二章 FCU1101 功能介绍	9
2.1 接口示意图	9
2.2 电源供电	9
2.3 用户登录	10
2.3.1 网络登录	10
2.3.2 调试串口	11
2.3.3 查看系统 log	12
2.4 RS485 的测试	12
2.4.1 线序说明	12
2.4.2 软硬件对应关系	13
2.4.3 RS485 的测试方法	13
2.4.4 Modbus 测试	14
2.5 按键信号输入	15
2.5.1 复位按键测试	15
2.5.2 BOOT 按键测试	15
2.6 指示灯控制 (DO)	15
2.7 有线网卡测试	15
2.7.1 软硬件对应关系	15
2.7.2 网卡测试	16
2.7.3 网络 socket 测试	16
2.7.4 网络的相关服务测试	16
2.8 WiFi	17
2.8.1 WiFi 测试	18
2.8.2 AP 测试	20
2.9 4G 模块测试	20
2.9.1 4G 拨号	20
2.9.2 4G 上网测试	21
2.9.3 4G-AP	22
2.9.4 4G 模块断电/上电	22
2.10 ZigBee、Lora 测试	23
2.10.1 ZigBee 模块测试	23

2.10.2 Lora 模块测试	26
2.11 RTC	30
2.12 TF 卡测试.....	30
2.13 看门狗测试	31
2.13.1 看门狗 20s 复位.....	31
2.13.2 5s 定时喂狗.....	31
2.14 数据库测试	31
第三章 系统固件更新	32
3.1 TF 卡更新固件.....	32
3.1.1 制作 TF 卡.....	32
3.1.2 TF 卡更新系统.....	34
3.2 在线更新固件	34
第四章 更新用户程序	35
4.1 设置自启动应用	35
4.2 MQTT 说明	35

第一章 FCU1101 产品介绍

1.1 产品简介

FCU1101 嵌入式控制单元采用 NXP i.MX6UltraLite 处理器开发设计，具有超高效、高性能、成本低等优势，主频高达 528MHZ，内存 256MB,NandFlash 256MB（可扩展 1GB），内部集成 RS485、以太网口、4G、WiFi、ZigBee、Lora 等功能接口和模块，满足不同场合的需求。

产品特点：

- 支持 9~36V 宽电压输入，具备防反接保护、过流保护；
- 支持 1 路 10/100M Ethernet；
- 支持 4 路 RS485，隔离保护设计；
- 支持 WiFi，模块支持 STA 和 AP 双模式；
- 支持全网通 4G 模块；
- 支持 ZigBee/Lora 模块（二选一，选配）；
- 支持外扩存储卡；
- 支持加密功能（选配）；

1.2 应用领域

FCU1101 嵌入式控制单元适用于能源管理系统、光伏逆变、水电气数据采集；生产制造、加工企业生产过程信号采集、数据传输；MES（制造执行系统）信号采集、数据传输；危化场所环境监测，大气环境监测信号采集、数据传输；农业大棚环境监测信号采集、数据传输；养殖环境监测信号采集、数据传输；医药、食品仓储环境监测信号采集、数据传输等领域。



1.3 硬件参数

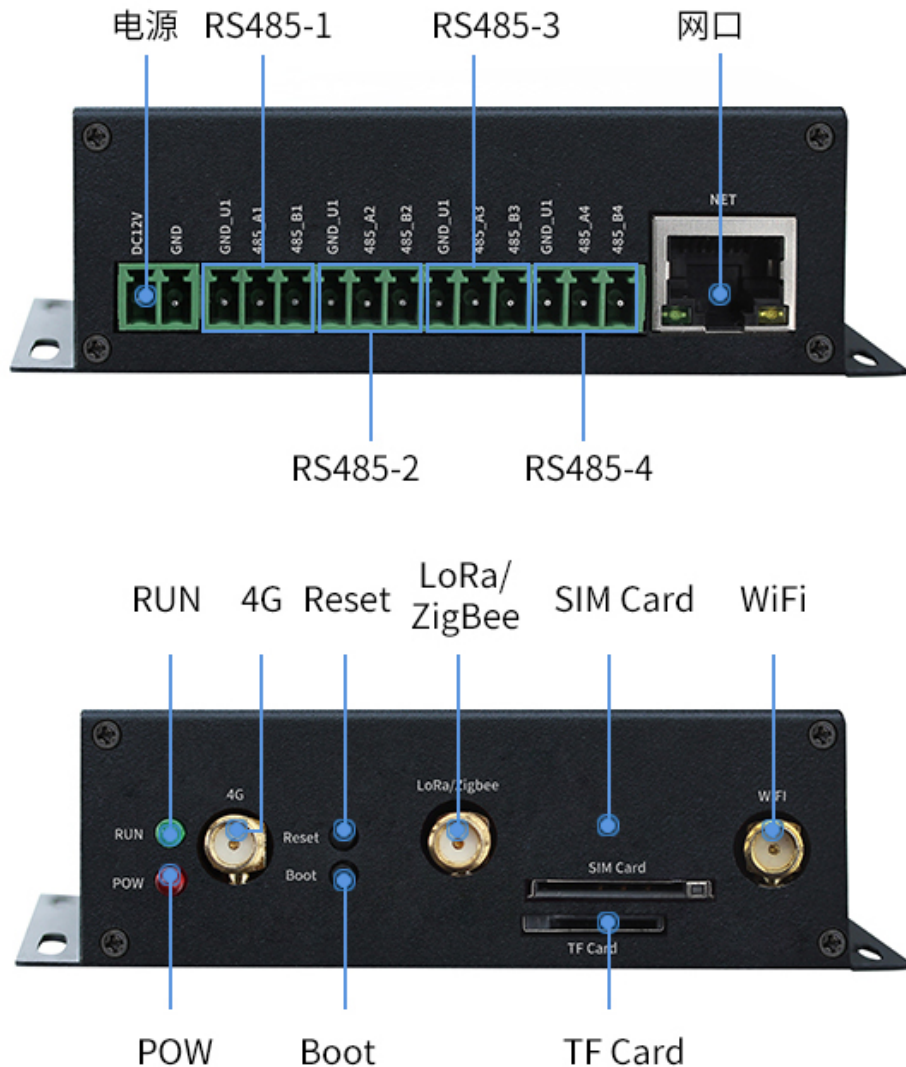
设备	详细描述	
CPU	NXP i.MX6UltraLite ARM Cortex-A7 528MHz	
RAM	DDR3 256MB	
ROM	NAND Flash 256MB（可扩展1GB）	
Ethernet	1路，10/100Mbps	
RS485	4路，1.5KV隔离保护，ESD 4级	
复位按键	1个，用于系统复位	
Boot按键	1个，与复位按键同时使用，用于系统固件更新	
4G	型号：移远EC20（不带GPS和语音功能） 网络制式：中国移动 4G/3G/2G、中国联通 4G/3G/2G、中国电信 4G	
WiFi	支持STA、AP模式	
ZigBee	1路，2.4GHz，型号：WLT2408NZ	二选一
Lora	1路，433MHz，支持20dBm，27dBm，30dBm三种发射功率 型号：E32-TTL-100(433T20DC) 20dBm (默认使用) E32(433T27D) 27dBm E32-TTL-1W(433T30D) 30dBm	
TF Card	支持最高32GB TF卡	
RTC	支持掉电保持时钟，NTP自动对时	
Power IN	主电源输入DC 12V，支持DC9V~36V宽压输入 具备反接保护、过流保护	
工作环境	湿度：5%~95%，无凝露 工作温度：-35℃~70℃（注：WiFi模块工作温度为0℃~70℃） 存储温度：-40℃~85℃	

1.4 软件参数

软件支持	详细描述
操作系统	Linux-3.14.38
文件系统	Yaffs2
GCC	4.6.2
RS485	提供 485 测试示例, 支持 Modbus 协议 (RTU 模式)
网络协议	TCP/IP、UDP、DHCP、TFTP、FTP、Telnet、SSH、Web、HTTP、IPtables、MQTT、SQLITE
以太网	10M/100M 自适应以太网, 支持静态/动态分配 IP 地址, 支持修改 MAC 地址
WiFi	支持 STA、AP 功能
4G 网络	EC20 模块, 支持移动、联通、电信 4G 上网功能
RTC	支持 NTP 自动校时
ZigBee 模块	支持 WLT2408NZ 模块收发数据
Lora 模块	支持 E32-TTL-100 模块收发数据
看门狗	支持复位时间设置

第二章 FCU1101 功能介绍

2.1 接口示意图



2.2 电源供电

本产品支持 9~36V 宽压输入，默认输入为 DC12V，飞凌提供电源适配器为 12V 2A。整机供电后，电源指示灯“POW”亮。

参考图		说明	
 实物图	 示意图	引脚序号	引脚说明
		1	电源正极，DC 12V，支持DC9V~36V宽压输入
		2	电源负极，GND

2.3 用户登录

FCU1101 支持调试串口与网络两种登录系统的控制方法。

2.3.1 网络登录

将 PC 机或笔记本通过网线与 FCU1101 连接，FCU1101 的 eth0 开机默认 IP 地址为 192.168.0.232，将 PC 的 IP 地址设置为 192.168.0.xx 网段，正常上电之后，通过 telnet 或者 SSH 登录，输入命令测试基本功能与应用。

Telnet 登录:

在 Windows 的命令窗口下输入 telnet 192.168.0.232，用户名输入 root，密码为空。注：开始->运行，输入 cmd 回车，即可进入 Windows 的命令窗口

```
freescall login :root
root@ freescall
```

SSH 登录:

因 FCU1101 默认为 root 用户登录，无密码，采用 SSH 登录之前需要先采用 telnet 登录，修改 root 的登录密码。

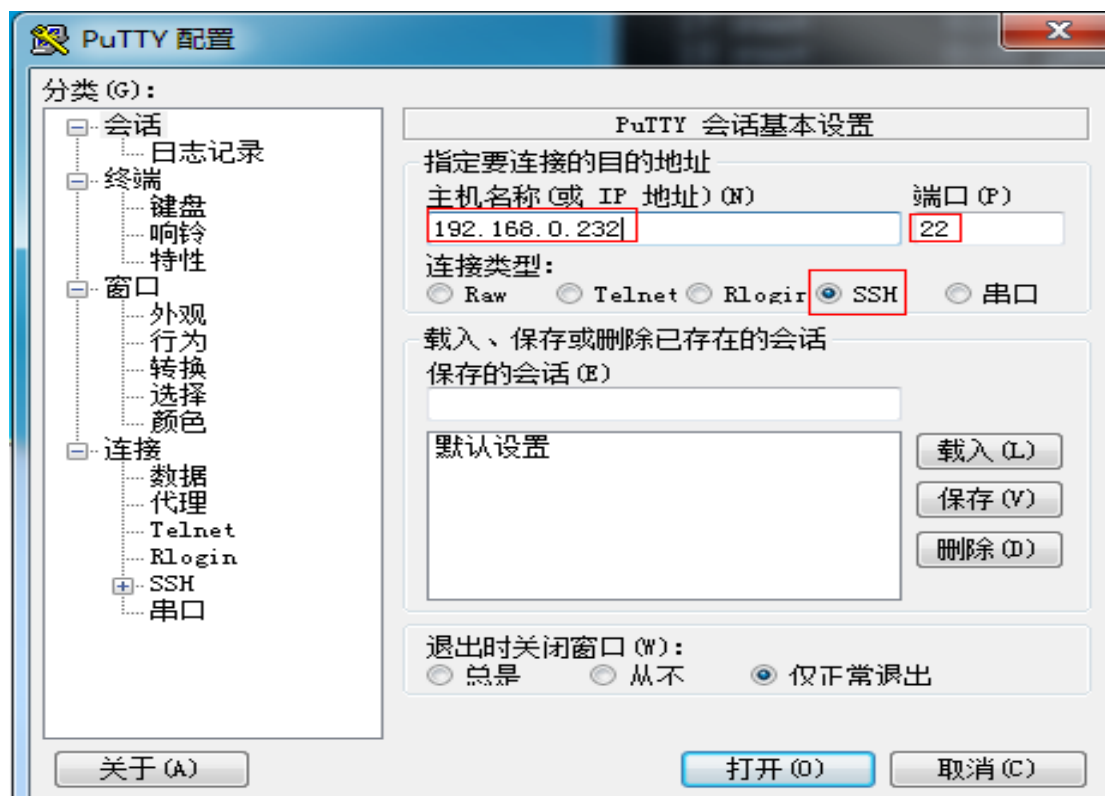
采用 telnet 登录，请参考上节。

```
freescall login :root
root@ freescall
```

修改密码:

```
root@freescall /$ passwd root
Changing password for root
New password:          (输入密码之后，不显示)
Bad password: too weak
Retype password:
Password for root changed by root
密码长度最大为 8 位。
```

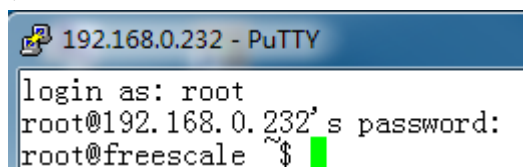
打开 putty.exe(用户资料/工具/putty.exe)，如下设置，点击打开:



第一次登录，会提示如下警告，点击“是”即可。



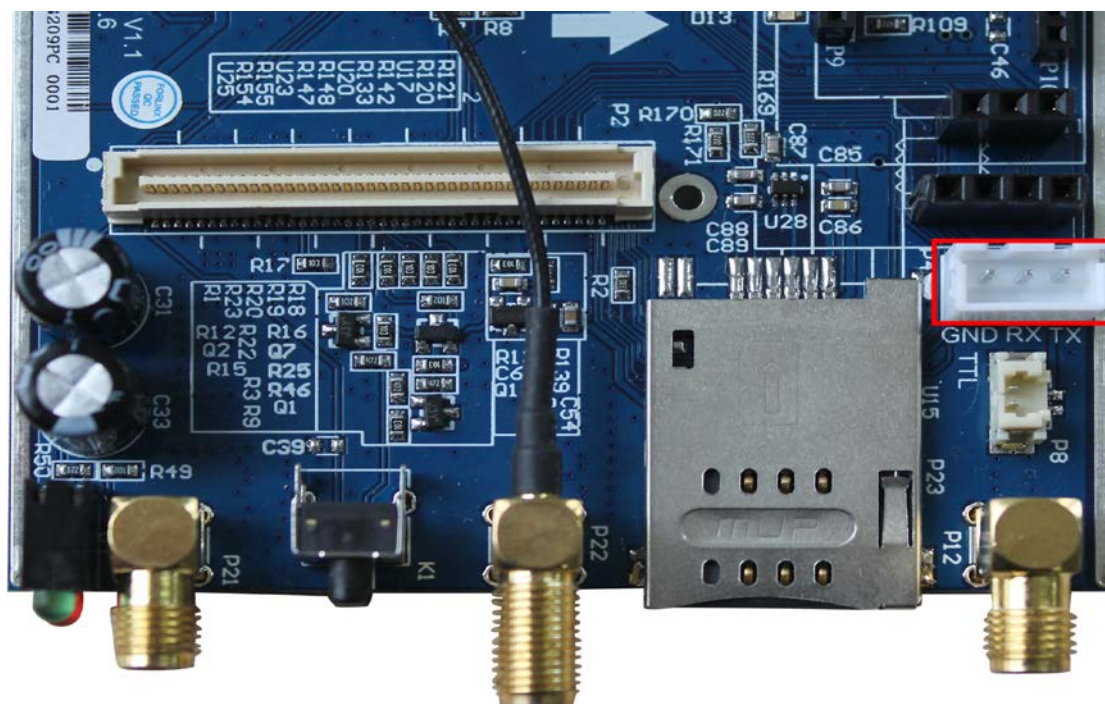
登录时，输入用户名与密码。



2.3.2 调试串口

调试串口需要将外壳去掉，适用于前期研发调试。后期现场调试维护建议使用网络登录方式。

TTL 调试串口位置如下图：



2.3.3 查看系统 log

系统启动之后，可在 `/var/log/messages` 中查看内核启动的信息或者使用 `dmesg` 命令。

采用 `dmesg`:

```
root@freescall /$dmesg | less
[ 0.000000] Booting Linux on physical CPU 0x0
.....
```

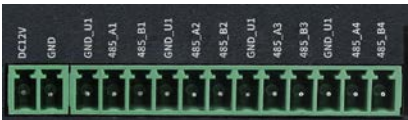

查看 `/var/log/messages`

```
root@freescall ~$ cat /var/log/messages
Jun 25 10:21:59 freescall syslog.info syslogd started: BusyBox v1.23.1
Jun 25 10:21:59 freescall user.notice kernel: klogd started: BusyBox v1.23.1 (2017-11-11
14:18:59 CST)
Jun 25 10:21:59 freescall user.info kernel: [ 0.000000] Booting Linux on physical CPU
0x0
.....
```

2.4 RS485 的测试

2.4.1 线序说明

参考图	说明
-----	----

实物图		引脚	引脚说明
示意图		GND_U1	RS485 隔离数字地
		485_A1	RS485_1 Data+
		485_B1	RS485_1 Data-
		GND_U1	RS485 隔离数字地
		485_A2	RS485_2 Data+
		485_B2	RS485_2 Data-
		GND_U1	RS485 隔离数字地
		485_A3	RS485_3 Data+
		485_B3	RS485_3 Data-
		GND_U1	RS485 隔离数字地
		485_A4	RS485_4 Data+
		485_B4	RS485_4 Data-

2.4.2 软硬件对应关系

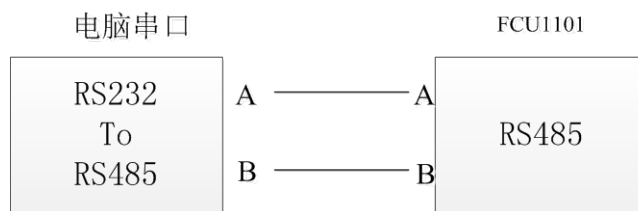
设备节点名称	硬件标识名称
/dev/ttymx1	RS485_1
/dev/ttymx2	RS485_2
/dev/ttymx3	RS485_3
/dev/ttymx4	RS485_4

2.4.3 RS485 的测试方法

本测试方法以和电脑通讯为例，如用户有专用 RS485 接口设备可直接连接测试；

本测试方法以 RS485_1 为例，如用户需要测试其他 RS485 接口，只需修改对应接口的设备节点名称即可；

将 RS232 转 RS485 模块和 FCU1101 的 RS485_1 接口连接，连接方式如下所示：



输入如下命令：

```
root@freescale /$485-test /dev/ttymx1 9600
```

RS485 会以 9600 波特率向 PC 机发送 “112233445566778899”，同时设置 PC 端串口助手软件定时发送 “abc123”。调试信息如下：

```
root@freescale /$ 485-test /dev/ttymx1 9600
Welcome to TTYtest! Press Ctrl + 'c' to stop.
/dev/ttymx1,creat thread 1994724464 sucess
/dev/ttymx1,creat thread 1986335856 success
```

```
sendTotal= 9 num = 1 send = □"3DUfw 达 (112233445566778899 的 ASCII 进制
```

数)

```
recvTotal= 6 num = 1 recv = abc123
616263313233
```

PC 端的串口助手软件（用户资料/工具/sscom32.exe），发送与接收数据如下：



2.4.4 Modbus 测试

Modbus 的源码和编译方法参考用户资料/Linux/应用/test/Modbus。本操作说明只针对串口（即 RTU 模式）进行说明。

1. 准备两套 FCU1101，将 A-A，B-B 相连接，本次试验用的是 ttymx1；
2. 分别与计算机通过网口连接好，启动调试端口，接好电源后启动 Linux 系统。
3. 在作为服务器端的模块上通过网络调试端口启动服务：
进入系统后，在 /forlinx 目录下输入命令运行程序：./unit-test-server rtu
等待客户端程序进行连接。
4. 在作为客户端的板上通过网络调试端口连接服务器端：
进入系统后，在 /forlinx 目录下输入命令行运行程序：./unit-test-client rtu
5. 则可以在网络调试端口中看到数据传送接收信息：

```

root@freescale /forlinx$ ./unit-test-server rtu
Opening /dev/ttymxcl at 115200 bauds (N, 8, 1)
Waiting for a indication...
<11><05><01><30><FF><00><8F><59>
[11][05][01][30][FF][00][8F][59]
Waiting for a indication...
<11><01><01><30><00><01><FE><A9>
[11][01][01][01][94][88]
Waiting for a indication...
<11><0F><01><30><00><25><05><CD><6B><B2><0E><1B><CC><C1>
>
[11][0F][01][30][00][25][97][73]
Waiting for a indication...
<11><01><01><30><00><25><FE><B2>
[11][01][05][CD][6B][B2][0E][1B][45][E6]
Waiting for a indication...
<11><02><01><C4><00><16><BB><55>
[11][02][03][AC][DB][35][20][18]
Waiting for a indication...
<11><06><01><6B><12><34><F6><0D>
[11][06][01][6B][12][34][F6][0D]
Waiting for a indication...
<11><03><01><6B><00><01><F6><BA>
[11][03][02][12][34][74][F0]
Waiting for a indication...
<11><10><01><6B><00><03><06><02><2B><00><01><00><64><DB>
><78>

root@freescale /forlinx$ ./unit-test-client rtu
Opening /dev/ttymxcl at 115200 bauds (N, 8, 1)
** UNIT TESTING **
1/1 No response timeout modification on connect: OK

TEST WRITE/READ:
[11][05][01][30][FF][00][8F][59]
Waiting for a confirmation...
<11><05><01><30><FF><00><8F><59>
1/2 modbus_write_bit: OK
[11][01][01][30][00][01][FE][A9]
Waiting for a confirmation...
<11><01><01><01><94><88>
2/2 modbus_read_bits: OK
OK
[11][0F][01][30][00][25][05][CD][6B][B2][0E][1B][CC][C1]
]
Waiting for a confirmation...
<11><0F><01><30><00><25><97><73>
1/2 modbus_write_bits: OK
[11][01][01][30][00][25][FE][B2]
Waiting for a confirmation...
<11><01><05><CD><6B><B2><0E><1B><45><E6>
2/2 modbus_read_bits: OK
OK
OK
OK
OK

```

2.5 按键信号输入

复位为系统复位按键。Boot 为系统烧写按键，不可自定义，如下图：



2.5.1 复位按键测试

按 Reset 按钮，系统立即重启。同时 RUN 灯，在启动过程中一直不亮，启动完成之后，RUN 灯作为心跳灯闪烁。

2.5.2 BOOT 按键测试

Boot 与 Reset 配合实现固件更新，具体使用方法参考“系统固件更新”章节中的“TF 卡更新系统”部分。

2.6 指示灯控制（DO）

FCU1101 有 2 个 LED 指示灯，分别为电源和系统运行，如下图：



电源指示灯为系统供电指示，上电自动点亮，掉电自动熄灭；

RUN 灯有两种指示：

- （1）烧写指示：烧写过程中 RUN 灯一直不亮，烧写完成之后 RUN 灯闪烁（每隔 1s 亮灭一次）。
- （2）系统启动指示：系统启动时，RUN 灯一直不亮。系统启动完成后，RUN 灯作为心跳灯。

2.7 有线网卡测试

FCU1101 支持 1 路 10M/100M 自适应以太网，为 RJ45 接口。

2.7.1 软硬件对应关系

设备节点名称	硬件标识名称
eth0	eth0

2.7.2 网卡测试

将 PC 与 FCU1101 的网口使用网线连接，并设定 PC 机 IP 为固定 IP（此处以 192.168.0.72 为例），FCU1101 的 IP 地址为 192.168.0.232，默认系统启动后，已经打开网络。

在 Windows 的命令窗口下输入 ping 192.168.0.232，注：开始->运行，输入 cmd 回车，即可进入 Windows 的命令窗口。

```
C:\Users\Administrator>ping -n 3 192.168.0.232

正在 Ping 192.168.0.232 具有 32 字节的数据:
来自 192.168.0.232 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.0.232 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.0.232 的回复: 字节=32 时间<1ms TTL=64

192.168.0.232 的 Ping 统计信息:
    数据包: 已发送 = 3, 已接收 = 3, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 0ms, 平均 = 0ms
```

PC 与 FCU1101 之间可 ping 通，使用 Telnet 或 SSH 登录成功，即说明网卡工作正常。

2.7.3 网络 socket 测试

参考用户资料/Linux /socknet/ socket 测试.pdf。

2.7.4 网络的相关服务测试

2.7.4.1 Telnet 服务

FCU1101 在/etc/inetd.conf 脚本文件中已经启动了 telnet 服务，设置好 IP 地址后就可以作为一台 telnet 服务器了。

在 Windows 的命令窗口下输入 telnet 192.168.0.232，用户名输入 root，密码为空。注：开始->运行，输入 cmd 回车，即可进入 Windows 的命令窗口

```
Welcome to Freescale Semiconductor Embedded Linux Environment

freescale login: root
root@freescale ~$
```

2.7.4.2 Web 服务

FCU1101 移植了一个默认 webserver: boa。boa webserver 是一个小巧高效的 web 服务器，可运行在 Unix 或 Linux 平台，支持 CGI，源代码开放；是一个非常适合于嵌入式系统的单任务 http 服务器。

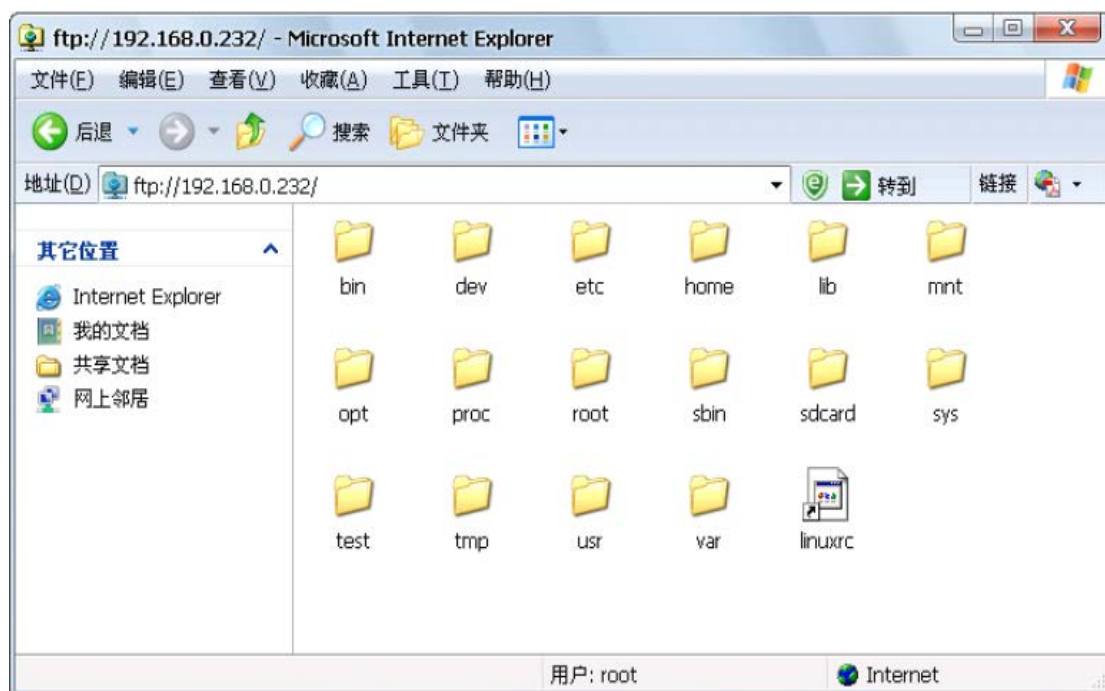
系统启动时已经自动启动了 boa 服务，在 IE 中输入 FCU1101 的 IP 地址即可浏览开发板 webserver 中的网页。下图是在 IE 中浏览的截图：



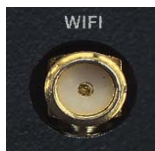
2.7.4.3 FTP

系统启动时已经自动启动了 vsftpd 服务，可在电脑上用 ftp 软件访问，用户名 root。

下图是在 PC 机上用 IE 访问 ftp 的截图：



2.8 WiFi



系统启动之后，默认 WiFi 模块已使能，驱动已加载。

2.8.1 WiFi 测试

2.8.1.1 查看驱动是否加载

```
root@freescale /$ lsmod
8723bu 1280657 0 - Live 0x7f004000
```

注：出现 8723bu 即模块已经加载。无 8723bu，则模块没被加载，检查内核版本与文件系统中模块版本是否一致。

2.8.1.2 配置 WiFi

```
root@freescale ~$ cd /forlinx/
root@freescale /forlinx$ ./WiFi.sh -i 8723 -s ciscosbWiFi -p abcdefghijklmn
```

说明：-s 表示 WiFi 热点的名称 -p 表示密码，如果没有密码请输入 -p NONE。路由器采用 wpa 加密方式。

连接打印内容如下：

```
WiFi 8723
ssid ciscosbWiFi
pasw abcdefghijklmn
[ 796.232774] RTL871X: module exit start
[ 796.236566] usbcore: deregistering interface driver rtl8723bu
[ 796.244647] RTL871X: rtw_ndev_uninit(wlan0)
[ 796.291328] RTL871X: rtw_cmd_thread: DriverStopped(1) SurpriseRemoved(0) break
at line 564
[ 796.326230] RTL871X: rtw_dev_unload: driver not in IPS
[ 796.367965] RTL871X: module exit success
[ 796.504334] RTL871X: module init start
[ 796.508124] RTL871X: rtl8723bu v4.3.16_14189.20150519_BTCOEX20150119-5844
[ 796.519696] RTL871X: build time: Jun 7 2018 08:37:20
[ 796.524782] RTL871X: rtl8723bu BT-Coex version = BTCOEX20150119-5844
[ 796.685775] RTL871X: rtw_ndev_init(wlan0)
[ 796.711882] usbcore: registered new interface driver rtl8723bu
[ 796.717792] RTL871X: module init ret=0
[ 800.991534] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
rfkill: Cannot open RFKILL contro[ 802.327088] RTL871X: set bssid:00:00:00:00:00:00
l device
ioctl[SIOCSIWAP]: Operation not permitted
wlan0: Trying to associate with b[ 803.937118] RTL871X: set ssid [ciscosbWiFi]
fw_state=0x00000008
c:67:1c:41:2e:3c (SSID='ciscosbWiFi' freq=2437 MHz)
[ 803.947778] RTL871X: set bssid:bc:67:1c:41:2e:3c
[ 803.992148] RTL871X: start auth
```

```
[ 803.998736] RTL871X: auth success, start assoc
[ 804.008267] RTL871X: assoc success
[ 804.012081] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
wlan0: Associated with bc:67:1c:41:2e:3c
[ 804.032130] RTL871X: send eapol packet
[ 804.056264] RTL871X: send eapol packet
[ 804.060826] RTL871X: set pairwise key camid:4, addr:bc:67:1c:41:2e:3c, kid:0,
type:AES
wlan0: WPA: Key negotiation completed with bc:67:1c:41:2e:3c [PTK[ 804.073162]
RTL871X: set group key camid:5, addr:bc:67:1c:41:2e:3c, kid:2, type:AES
=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to bc:67:1c:41:2e:3c completed (auth)
[id=0 id_str=]
udhcpc (v1.23.1) started
Sending discover...
Sending select for 192.168.1.111...
Lease of 192.168.1.111 obtained, lease time 86400
Deleting routers
adding dns 222.222.202.202
adding dns 222.222.222.222
Finshed!
```

2.8.1.3 测试 WiFi

测试 WiFi, ping 域名或者 IP。

```
root@freescale /forlinx$ ping www.baidu.com -c 3 -I wlan0
PING www.baidu.com (220.181.112.244): 56 data bytes
64 bytes from 220.181.112.244: seq=0 ttl=51 time=21.816 ms
64 bytes from 220.181.112.244: seq=1 ttl=51 time=36.377 ms
64 bytes from 220.181.112.244: seq=2 ttl=51 time=43.212 ms

--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 21.816/33.801/43.212 ms
```

2.8.1.4 模块断电/上电

模块断电：

```
root@freescale /$ echo 1 > /sys/class/gpio/gpio117/value
root@freescale /$ [ 190.448553] usb 1-1: USB disconnect, device number 2
[ 190.493067] RTL871X: rtw_ndev_uninit(wlan0)
```

模块上电：

```
root@freescale /$ echo 0 > /sys/class/gpio/gpio117/value
[ 228.328712] usb 1-1: new high-speed USB device number 3 using ci_hdrc
[ 228.705151] RTL871X: rtw_ndev_init(wlan0)
```

2.8.1.5 卸载模块

在实际使用时，有可能会需要卸载模块，卸载模块之前，需要先 kill -9 \$(pidof wpa_supplicant) kill -9 \$(pidof udhcpc)，再卸载模块，否则会打印一些异常信息。

```
root@freescale /forlinx$ rmmod 8723bu
[ 1238.641278] RTL871X: module exit start
[ 1238.645068] usbcore: deregistering interface driver rtl8723bu
[ 1238.655545] RTL871X: indicate disassoc
[ 1238.660020] RTL871X: rtw_cmd_thread: DriverStopped(1) SurpriseRemoved(0) break
at line 564
[ 1238.669813] RTL871X: rtw_ndev_uninit(wlan0)
[ 1238.732788] RTL871X: rtw_dev_unload: driver not in IPS
[ 1238.774330] RTL871X: module exit success
```

2.8.2 AP 测试

默认生成的 WiFi 热点的 IP 地址段为 192.168.0.100-192.168.0.120，默认网关为 192.168.0.10，采用 WPA-PSK 加密方式。用户可通过修改/etc/udhcpd.conf 修改地址段与网关等配置信息。

通过配置文件/etc/hostapd.conf 查看或者修改热点名称 FORLINX 和 WPA 密码 12345678。

2.8.2.1 以太网-AP

1. 设置以太网 IP，配置网络防火墙：

注：本例为以太网连接路由器说明，配置完以太网后，需要测试是否可以连接外网，如果可以连接外网（方法参考“有线网卡”章节），请继续按照操作步骤执行，如果不可以请检查以太网或者路由器连接是否正常。

```
udhcpc -i eth0 /*自动分配 IP */
echo 1 > /proc/sys/net/ipv4/ip_forward /* 打开 IP 转发 */
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE /*设置转发规则*/
```

2. 设置 WiFi 的模式与 IP

```
确保模块 8723bu 已经加载。
ifconfig wlan0 up /*打开 WiFi*/
ifconfig wlan0 192.168.0.10 netmask 255.255.255.0 /*设置 IP 与子网掩码*/
ifconfig wlan0 promisc /*设置 wlan0 为混杂模式 */
```

3. 开启 AP

```
udhcpd /etc/udhcpd.conf & /*WiFi 地址、网关等配置信息*/
/home/hostapd -d /etc/hostapd.conf & /* 加密方式、用户名、密码等设置 */
```

4. 手机等移动终端可以通过 WiFi 连接到 FCU1101 的 AP 热点，访问外网。

2.8.2.2 4G-AP

参考“4G 模块测试”章节中“4G-AP”部分。

2.9 4G 模块测试



4G 天线



SIM 卡插槽

开机之后，默认 4G 模块已供电，驱动已经加载。

2.9.1 4G 拨号

保证模块正常供电之后，拨号，动态分配 IP：


```
root@freescale ~$ cd /forlinx/
root@freescale /forlinx$ ./quectel-CM &
[01-01_00:17:44:133] WCDMA&LTE_QConnectManager_Linux&Android_V1.1.34
[01-01_00:17:44:135] ./quectel-CM profile[1] = (null)/(null)/(null)/0, pincode = (null)
[01-01_00:17:44:136] Find /sys/bus/usb/devices/2-1 idVendor=2c7c idProduct=0125
[01-01_00:17:44:136] Find /sys/bus/usb/devices/2-1:1.4/net/eth2
[01-01_00:17:44:136] Find usbnet_adapter = eth2
[01-01_00:17:44:136] Find /sys/bus/usb/devices/2-1:1.4/GobiQMI/qcqm2
[01-01_00:17:44:137] Find qmichannel = /dev/qcqm2
[01-01_00:17:44:162] Get clientWDS = 7
[01-01_00:17:44:194] Get clientDMS = 8
[01-01_00:17:44:226] Get clientNAS = 9
[01-01_00:17:44:258] Get clientUIM = 10
[01-01_00:17:44:290] Get clientWDA = 11
[01-01_00:17:44:322] requestBaseBandVersion EC20CEHCR06A02M1G
[01-01_00:17:44:418] requestGetSIMStatus SIMStatus: SIM_READY
[01-01_00:17:44:450] requestGetProfile[1] cmnet///0
[01-01_00:17:44:482] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached,
DataCap: LTE
[01-01_00:17:44:514] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[01-01_00:17:44:578] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached,
DataCap: LTE
[01-01_00:17:44:609] requestSetupDataCall WdsConnectionIPv4Handle: 0xe17ea3c0
[01-01_00:17:44:705] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[01-01_00:17:44:738] ifconfig eth2 up
[01-01_00:17:44:772] busybox udhcpc -f -n -q -t 5 -i eth2
[01-01_00:17:44:789] udhcpc (v1.23.1) started
[01-01_00:17:44:833] Sending discover...
[01-01_00:17:44:903] Sending select for 10.47.148.8...
[01-01_00:17:44:983] Lease of 10.47.148.8 obtained, lease time 7200
[01-01_00:17:45:017] Deleting routers
[01-01_00:17:45:071] adding dns 111.11.1.3
[01-01_00:17:45:107] adding dns 111.11.11.3
```

2.9.2 4G 上网测试

测试是否连接外网:

```
root@freescale /forlinx$ ping www.baidu.com -I eth2 -c 3
PING www.baidu.com (111.13.100.92): 56 data bytes
64 bytes from 111.13.100.92: seq=0 ttl=53 time=62.803 ms
64 bytes from 111.13.100.92: seq=1 ttl=53 time=58.642 ms
64 bytes from 111.13.100.92: seq=2 ttl=53 time=60.662 ms

--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
```

```
round-trip min/avg/max = 58.642/60.702/62.803 ms
```

测试 ping 百度，发送接收 3 包数据，丢包率为 0，4G 可上外网。如果 4G 模块不能连接外网，请检查 4G 拨号设置或 SIM 卡是否欠费。

2.9.3 4G-AP

1. 4G 模块拨号成功并分配 IP，可连接外网。设置转发规则：

```
root@freescale ~$ cd /forlinx/
root@freescale /forlinx$ ./quectel-CM & /*拨号*/
echo 1 > /proc/sys/net/ipv4/ip_forward /* 打开 IP 转发 */
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE /*eth2 为 4G 模块识别出的网卡，设置转发规则 */
```

2. 设置 WiFi 的模式与 IP

```
确保模块 8723bu 已经加载。
ifconfig wlan0 up /*打开 WiFi*/
ifconfig wlan0 192.168.0.10 netmask 255.255.255.0 /*设置 IP 与子网掩码*/
ifconfig wlan0 promisc /*设置 wlan0 为混杂模式 */
```

3. 开启 AP

```
udhcpd /etc/udhcpd.conf & /*WiFi 地址、网关等配置信息*/
/home/hostapd -d /etc/hostapd.conf & /* 加密方式、用户名、密码等设置，此时用户名为 FORLINX，密码为 12345678 */
```

4. 手机等移动终端可以通过 WiFi 连接到 FCU1101 的 AP 热点，访问外网。

2.9.4 4G 模块断电/上电

```
root@freescale ~$ echo 129 > /sys/class/gpio/export
root@freescale ~$ echo out > /sys/class/gpio/gpio129/direction
```

- 4G 模块断电：

```
root@freescale ~$ echo 0 > /sys/class/gpio/gpio129/value
[ 519.297047] usb 2-1: USB disconnect, device number 2
[ 519.311230] option1 ttyUSB0: GSM modem (1-port) converter now disconnected from ttyUSB0
[ 519.329489] option 2-1:1.0: device disconnected
[ 519.348923] option1 ttyUSB1: GSM modem (1-port) converter now disconnected from ttyUSB1
[ 519.369374] option 2-1:1.1: device disconnected
[ 519.380716] option1 ttyUSB2: GSM modem (1-port) converter now disconnected from ttyUSB2
[ 519.400360] option 2-1:1.2: device disconnected
[ 519.419736] option1 ttyUSB3: GSM modem (1-port) converter now disconnected from ttyUSB3
[ 519.440144] option 2-1:1.3: device disconnected
[ 519.450022] GobiNet 2-1:1.4 eth2: unregister 'GobiNet' usb-ci_hdrc.1-1, GobiNet Ethernet Device
```

- 4G 模块上电：

```
root@freescale ~$ echo 1 > /sys/class/gpio/gpio129/value
```

```
[ 534.608359] usb 2-1: new high-speed USB device number 3 using ci_hdrc
[ 534.783720] option 2-1:1.0: GSM modem (1-port) converter detected
[ 534.808343] transfer_flags is URB_ZERO_PACKET
[ 534.812739] transfer_flags is URB_ZERO_PACKET
[ 534.817112] transfer_flags is URB_ZERO_PACKET
[ 534.833529] transfer_flags is URB_ZERO_PACKET
[ 534.841520] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB0
[ 534.857396] option 2-1:1.1: GSM modem (1-port) converter detected
[ 534.877793] transfer_flags is URB_ZERO_PACKET
[ 534.886631] transfer_flags is URB_ZERO_PACKET
[ 534.891532] transfer_flags is URB_ZERO_PACKET
[ 534.895919] transfer_flags is URB_ZERO_PACKET
[ 534.903716] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 534.919950] option 2-1:1.2: GSM modem (1-port) converter detected
[ 534.940672] transfer_flags is URB_ZERO_PACKET
[ 534.945068] transfer_flags is URB_ZERO_PACKET
[ 534.953994] transfer_flags is URB_ZERO_PACKET
[ 534.959072] transfer_flags is URB_ZERO_PACKET
[ 534.966414] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB2
[ 534.982713] option 2-1:1.3: GSM modem (1-port) converter detected
[ 535.003152] transfer_flags is URB_ZERO_PACKET
[ 535.007551] transfer_flags is URB_ZERO_PACKET
[ 535.016521] transfer_flags is URB_ZERO_PACKET
[ 535.021546] transfer_flags is URB_ZERO_PACKET
[ 535.029368] usb 2-1: GSM modem (1-port) converter now attached to ttyUSB3
[ 535.055274] GobiNet 2-1:1.4 eth2: register 'GobiNet' at usb-ci_hdrc.1-1, GobiNet
Ethernet Device, 22:98:32:24:39:78
[ 535.090661] creating qcqmi2
```

2.10 ZigBee、Lora 测试



Lora/ZigBee 天线

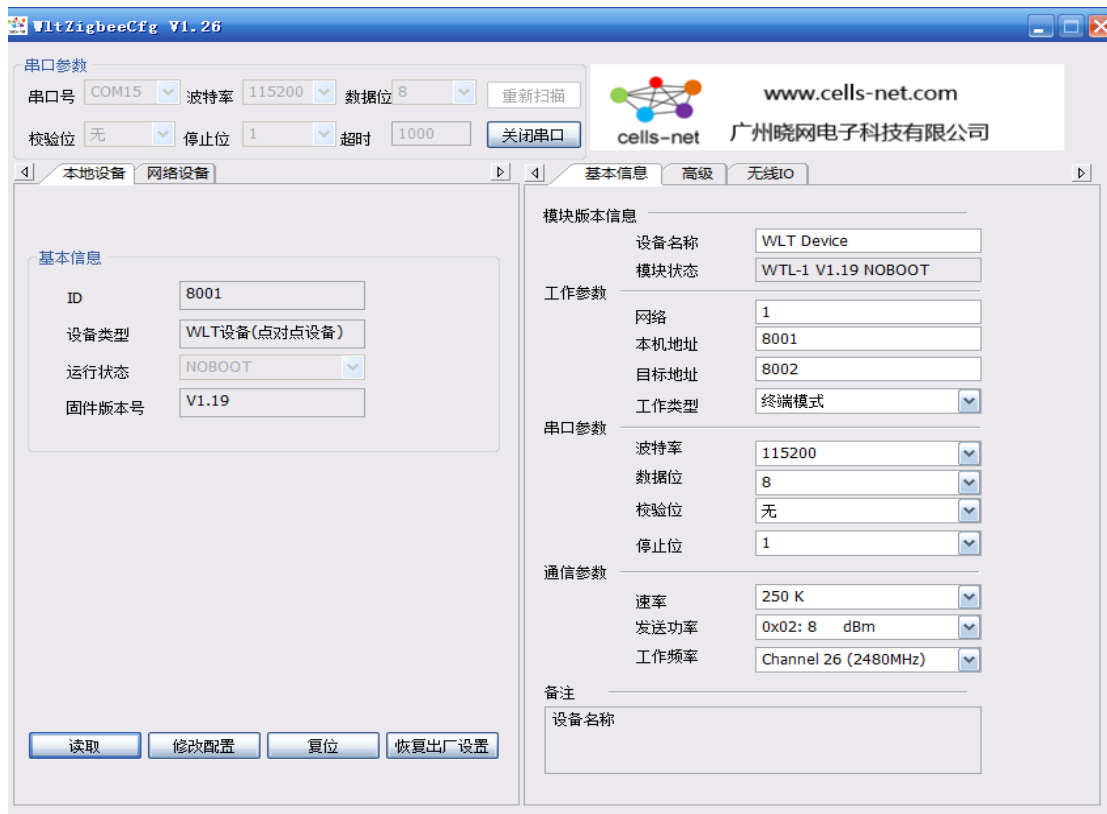
2.10.1 ZigBee 模块测试

系统启动之后，模块默认已正常供电。

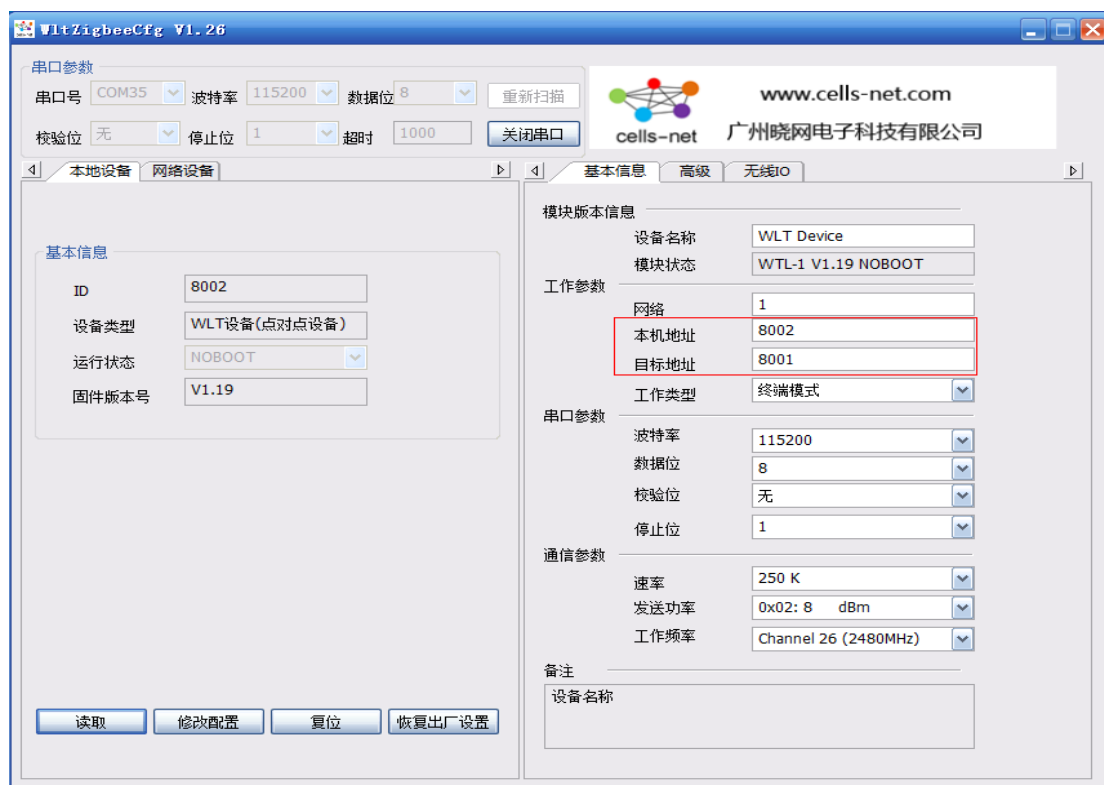
2.10.1.1 查看模块参数

默认已经通过 PC 配置好参数。本例采用点对点方式，源地址与目的地址对应一致。

ZigBee 的参数设置，以及主机模式的设置，参考用户资料/Linux/应用/ZigBee 中内容进行修改测试，此处不再多做介绍。



FCU1101 参数



模块 A 参数

2.10.1.2 测试

FCU1101 与模块 A 通讯测试，其中模块 A 经 USB 转 TTL 接到电脑，在 FCU1101 上运行如下命令：

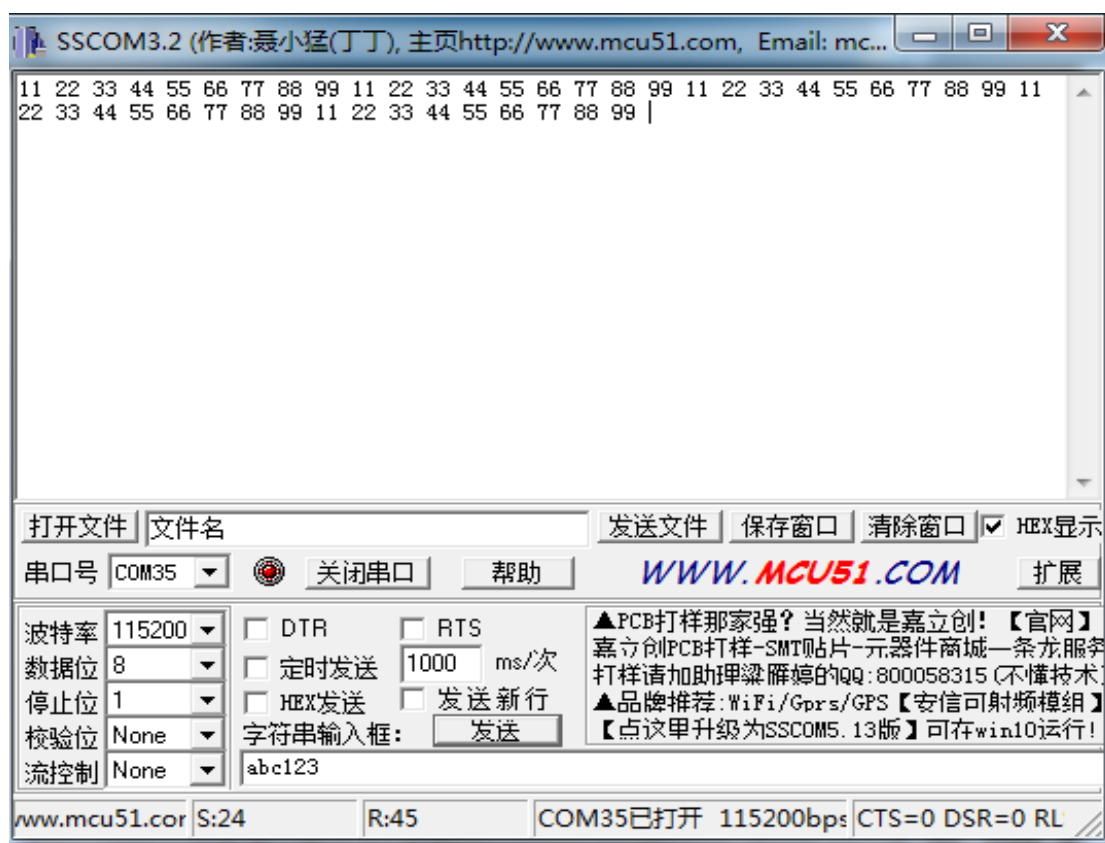
```
root@freescale ~$ 485-test /dev/ttymx5 115200
```

FCU1101 不断的向模块 A 发送 “112233445566778899”，同时模块 A 发送 “abc123”，测试调试信息如下：

```
Welcome to TTYtest! Press Ctrl + 'c' to stop.

/dev/ttymxc5, creat thread 1994724464 sucess
/dev/ttymxc5, creat thread 1986335856 sucess
sendTotal= 9 num = 1 send = "3DUfw
sendTotal= 18 num = 2 send = "3DUfw
recvTotal= 6 num = 1 recv = abc123
616263313233
sendTotal= 27 num = 3 send = "3DUfw
recvTotal= 12 num = 2 recv = abc123
616263313233
sendTotal= 36 num = 4 send = "3DUfw
recvTotal= 18 num = 3 recv = abc123
616263313233
recvTotal= 24 num = 4 recv = abc123
616263313233
sendTotal= 45 num = 5 send = "3DUfw
^C
/dev/ttymxc5, Send: 5 ,Receive: 4
```

同时在电脑上打开串口助手，参数设置如下：



串口助手接收到 ZigBee 发出的数据，同时调试端口能接收到串口助手发送的数据。

2.10.1.3 模块断电/上电

```
root@freescale ~$gpio-test out 0 1 #模块断电
root@freescale ~$gpio-test out 0 0 #模块上电
```

2.10.2 Lora 模块测试

系统启动之后，模块默认已正常供电，为一般模式，Lora 具体参数如下图所示：



Lora 模块默认参数

2.10.2.1 模式控制

将 Lora 的 M0 M1 AUX 设置为高电平，进入休眠模式，可配置参数。M0 M1 设置为低电平，AUX 设置为低电平，进入一般模式，无线打开，传输数据。

➤ 休眠模式

1. 将 M0、M1、AUX 设置为高电平。

```
echo 132 > /sys/class/gpio/export /*M0 导入到用户空间 */
echo out > /sys/class/gpio/gpio132/direction /*M0 设置为输出，并输出默认低电平*/
echo 135 > /sys/class/gpio/export /*M1 导入到用户空间 */
echo out > /sys/class/gpio/gpio135/direction /*M1 设置为输出，并输出默认低电平*/

echo 1 > /sys/class/gpio/gpio132/value /*M0 输出高电平*/
echo 1 > /sys/class/gpio/gpio135/value /*M1 输出高电平*/

echo 136 > /sys/class/gpio/export /*AUX 导入到用户空间 */
echo out > /sys/class/gpio/gpio136/direction /*AUX 设置为输出，并输出默认低电平*/
echo 1 > /sys/class/gpio/gpio136/value /*AUX 输出高电平*/
```

2. 写入参数

```
root@freescale ~$cd /forlinx
root@freescale /forlinx$./lora-write /dev/ttymx5 0xc0 0x00 0x0 0x1a 0x17 0x44
test[0] is c0
test[1] is 0
test[2] is 0
test[3] is 1a
test[4] is 17
test[5] is 44
```

```
Welcome to lora write data test!  
send data=====
```

其中 0xc0 0x00 0x0 0x1a 0x17 0x44 参数的含义，参考用户资料/Linux /应用 /lora-EC32-TTL-100/E32_Usermanual_CN_1.40.pdf 中的“参数设置指令”章节。

3. 读取参数

```
root@freescale ~$cd /forlinx  
root@freescale /forlinx$ ./lora-read /dev/ttymx5 9600  
root@freescale ~$ ./lora-read /dev/ttymx5 9600  
Welcome to TTYtest! Press Ctrl + 'c' to stop.  
.....  
Read Test Data finished,Read:c0  
Read Test Data finished,Read:0  
Read Test Data finished,Read:0  
Read Test Data finished,Read:1a  
Read Test Data finished,Read:17  
Read Test Data finished,Read:44  
ADDH is 0  
ADDL is 0  
val is 0  
parity is 8N1  
baud speed is 9600  
air rate is 1.2k  
channel is 23, is 433  
transparent transfer mode  
TXD AUX push-pull output, RXD pullup input  
wireless wakeup time is 250ms  
open FEC error correction  
transmit power is 20Bm
```

➤ 一般模式

M0 M1 设置为低电平，AUX 设置为低电平，进入一般模式，无线打开，传输数据。

M0、M1、AUX 对应 gpio 导入用户空间，如果已经导入，跳过此步骤。

```
echo 132 > /sys/class/gpio/export /*M0 导入到用户空间 */  
echo out > /sys/class/gpio/gpio132/direction /*M0 设置为输出，并输出默认低电平*/  
echo 135 > /sys/class/gpio/export /*M1 导入到用户空间 */  
echo out > /sys/class/gpio/gpio135/direction /*M1 设置为输出，并输出默认低电平*/  
echo 136 > /sys/class/gpio/export /*AUX 导入到用户空间 */  
echo out > /sys/class/gpio/gpio136/direction /*AUX 设置为输出，并输出默认低电平*/
```

将 M0、M1、AUX 输出低电平：

```
echo 0 > /sys/class/gpio/gpio132/value /*M0 输出低电平*/  
echo 0> /sys/class/gpio/gpio135/value /*M1 输出低电平*/  
echo 0 > /sys/class/gpio/gpio136/value /*AUX 输出低电平*/
```

2.10.2.3 测试

FCU1101 与模块 A 的模块地址与频率信道必须为相同。采用透传方式进行测试说明。

RF Setting V3.45



成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.

中文

English

模块型号: E32
版本: 4.3
当前频率: 433.0MHz
当前参数: 0x0, 0x0, 0x1a, 0x17, 0x44

COM3

关闭串口

查看支持型号

读取参数

写入参数

恢复出厂设置

波特率: 9600bps

奇偶校验: 8N1

空中速率: 2.4Kbps

发射功率: 30dBm

前向纠错: 打开

传输方式: 透传

唤醒时间: 250ms

IO 驱动: 推挽

模块地址: 0

频率信道: 23

本软件所著作权归成都亿佰特电子科技有限公司所有

官方网站: www.cdebyte.com

FCU1101 的参数

RF Setting V3.45



成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.

中文

English

模块型号: E32
版本: 4.3
当前频率: 433.0MHz
当前参数: 0x0, 0x0, 0x1a, 0x17, 0x44

COM3

关闭串口

查看支持型号

读取参数

写入参数

恢复出厂设置

波特率: 9600bps

奇偶校验: 8N1

空中速率: 2.4Kbps

发射功率: 30dBm

前向纠错: 打开

传输方式: 透传

唤醒时间: 250ms

IO 驱动: 推挽

模块地址: 0

频率信道: 23

本软件所著作权归成都亿佰特电子科技有限公司所有

官方网站: www.cdebyte.com

模块 A 的参数

FCU1101 与模块 A 通讯进行测试，其中模块 A 经 USB 转 TTL 接到电脑上，在 FCU1101 运行如下命令：

```
root@freescale ~$ 485-test /dev/ttymx5 9600
```

FCU1101 不断的向模块 A 发送“112233445566778899”，同时模块 A 发送“abc123”，测试调试信息如下：


```

root@freescale ~$
root@freescale ~$ 485-test /dev/ttymx5 9600

Welcome to TTYtest! Press Ctrl + 'c' to stop.

/dev/ttymx5, creat thread 1994876016 sucess
/dev/ttymx5, creat thread 1986487408 sucess
sendTotal= 9 num = 1 send = "3DUfw
sendTotal= 18 num = 2 send = "3DUfw
sendTotal= 27 num = 3 send = "3DUfw
sendTotal= 36 num = 4 send = "3DUfw
recvTotal= 6 num = 1 recv = abc123
616263313233
sendTotal= 45 num = 5 send = "3DUfw
recvTotal= 12 num = 2 recv = abc123
616263313233
sendTotal= 54 num = 6 send = "3DUfw
recvTotal= 18 num = 3 recv = abc123
616263313233
sendTotal= 63 num = 7 send = "3DUfw
sendTotal= 72 num = 8 send = "3DUfw
^C
/dev/ttymx5, Send: 8 ,Receive: 3
root@freescale ~$

```

同时电脑端的串口助手，参数配置如下图。



串口助手接收到 Lora 发出的数据，同时调试端口能接收到串口助手发送的数据。

2.10.2.1 模块断电/上电

```

root@freescale ~$ gpio-test out 0 1 #模块断电
root@freescale ~$ gpio-test out 0 0 #模块上电

```

2.11 RTC

RTC测试，主要通过使用date和hwclock 工具设置软、硬件时间，测试当操作系统重启的时候，软件时钟读取RTC时钟是否同步。默认FCU1101已经安装了纽扣电池。

1. 查看系统时间

```
root@freescale ~$ date -u
```

2. 设置时间如下命令

```
root@freescale ~$ date -s 2018.09.11-15:01:00
Tue Sep 11 15:01:00 HKT 2018
```

3. 查看硬件时间

```
root@freescale ~$ hwclock -w
```

hwclock -w — 将系统时间设置到时钟芯片里面，此时rtc 就可以使用了。如果没有这步，下次启动时，系统时间是不会更新的。

4. 断电重启系统，此时，软、硬件时间已经同步，说明RTC工作正常。

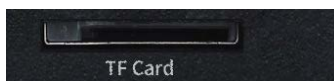
```
root@freescale ~$ date -u
Tue Sep 11 15:02:05 UTC 2018
```

5. 网络对时

```
root@freescale ~$ ntpdate cn.pool.ntp.org
4 Aug 09:25:24 ntpdate[957]: step time server 182.92.12.11 offset 1533345862.993804
sec
root@freescale ~$ date -u
Sat Aug 4 09:25:30 HKT 2018
```

注意：/etc/localtime 采用的是 Hong_Kong 的时区，/etc/timezone 为 Asia/Hong_Kong

2.12 TF 卡测试



TF 卡插槽

插上 TF 卡后系统会自动将其挂载到/media 目录。同时终端会打印关于 TF 卡的信息，由于存在很多种卡，显示的信息可能会有差别。

以 8G TF 卡为例，插入 TF 卡打印信息如下：

```
root@freescale /$ [ 6110.464467] mmc0: host does not support reading read-only switch. assuming write-enable.
[ 6110.478041] mmc0: Problem setting current limit!
[ 6110.598051] mmc0: new ultra high speed DDR50 SDHC card at address aaaa
[ 6110.618925] mmcblk0: mmc0:aaaa SL08G 7.40 GiB
[ 6110.656638] mmcblk0: p1
[ 6110.836429] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

查看 TF 卡中的文件，命令如下：

```
root@freescale /$ ls -l /media/mmcblk0p1/
drwxr-xr-x  2 root  root    4096 Oct 28  2016 bin
drwxr-xr-x  2 root  root    4096 Nov  7  2016 system
```

例如拷贝 TF 卡中的内容 bin 到 forlinx 路径下：

```
root@freescale /$cp /media/mmcblk0p1/bin /forlinx/  
root@freescale /$ sync
```

注意：拷贝完之后，sync 同步，否则拷贝不成功。

2.13 看门狗测试

FCU1101 默认看门狗不开启，如果需要开启，请参考“设置自启动应用”章节操作。

开启看门狗打开后，默认不操作 60s 之后重启系统。看门狗应用中可设置的最小复位时间为 2s。

如果执行定时喂狗，则超时时间到，系统不重启。

2.13.1 看门狗 20s 复位

打开看门狗，设置超时时间。

```
root@freescale /$ wdttest /dev/watchdog settimeout 20 &
```

定时时间 20s 到了之后，系统重启。

2.13.2 5s 定时喂狗

打开看门狗，设置超时时间，并定时喂狗。

```
root@freescale /$ wdttest /dev/watchdog keepalive 5 &
```

5s 周期性喂狗，超时时间到了，系统不重启。

2.14 数据库测试

```
root@freescale ~$ sqlite3
```

调试终端信息：

```
SQLite version 3.7.14.1 2012-10-04 19:37:12  
Enter ".help" for instructions  
Enter SQL statements terminated with a ";"  
/*创建数据库*/  
sqlite> create table tbl1 (one varchar(10), two smallint);  
/*数据库中插入 */  
/*hello!|10      */  
/*goodbye|20     */  
sqlite> insert into tbl1 values('hello!',10);  
sqlite> insert into tbl1 values('goodbye', 20);  
/*读取数据库中内容 */  
sqlite> select * from tbl1;  
显示数据库中的内容如下：  
hello!|10  
goodbye|20  
sqlite> .quit
```

从上面可看出，创建、读写数据库正常使用。

第三章 系统固件更新

本产品更新固件方法有：TF 卡更新固件，在线更新固件。

3.1 TF 卡更新固件

3.1.1 制作 TF 卡

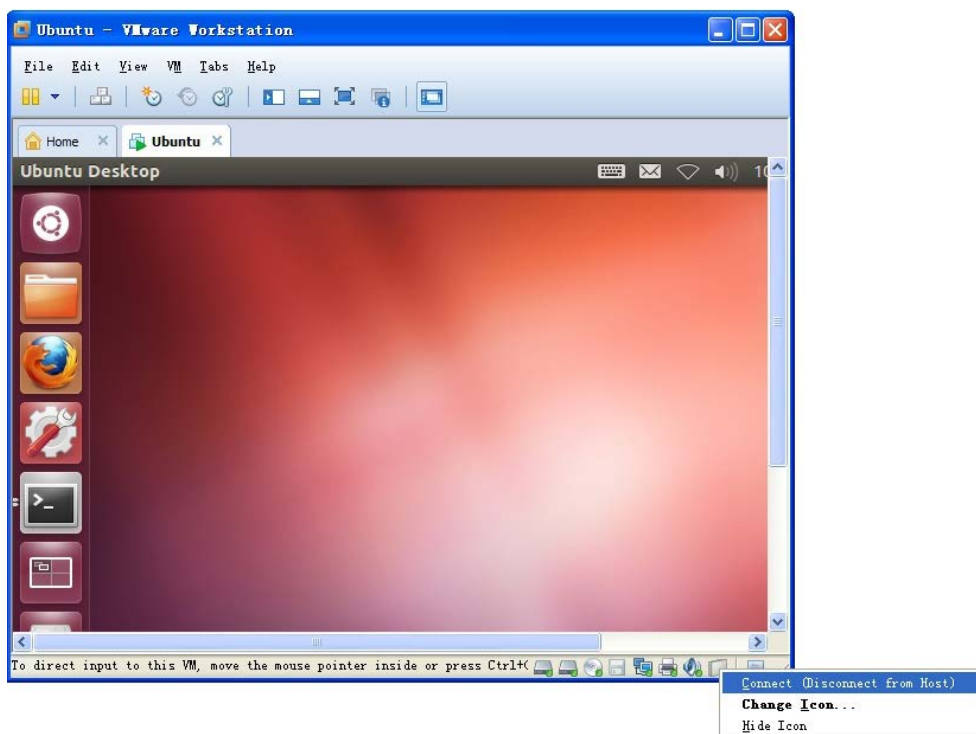
以 createSdcard.tar.bz2 为例说明，将 createSdcard.tar.bz2 拷贝到 ubuntu 系统的任一目录，假设为/home/forlinx/work。

📁 tools\createSdcard.tar.bz2。

步骤 1：解压 createSdcard.tar.bz2，命令如下。

```
cd /home/forlinx/work/  
tar xvf createSdcard.tar.bz2
```

步骤 2：使用 USB 读卡器把 TF 卡插入到电脑的 USB 端口（VMware 虚拟机用户如果 U 盘没有被虚拟机识别，可以使用如下方式将 U 盘连接到虚拟机）。



步骤 3：进入 createSdcard 目录

```
cd /home/forlinx/work/createSdcard
```

执行脚本：

```
./createSdCard.sh
```

执行上述命令后，终端会列出电脑的硬盘或优盘，对应选择自己的 TF 卡，然后回车。

注意：判定自己的 U 盘是 sda/sdb/sdc 可以根据容量进行判断，比如自己的 U 盘容量为 4G，则其 size 为 3872256 字节 ≈ 4G，建议用户执行此操作时不要同时插入多个优盘，以免混淆。

这里以我们的操作为例：

选择 1，回车

```
#####
This script will create a bootable SD card from custom or pre-built binaries.

The script must be run with root permissions and from the bin directory of
the SDK

Example:
$ sudo ./create-sdcard.sh

Formatting can be skipped if the SD card is already formatted and
partitioned properly.

#####

Available Drives to write images to:

# major minor size name
1: 8 16 3872256 sdb

Enter Device Number: 
```

格式化，选择y，回车，完成格式化。

```
# major minor size name
1: 8 16 3872256 sdb

Enter Device Number: 1

sdb was selected

Checking the device is unmounted
unmounted /dev/sdb1

sdb1 sdb2 sdb3
3775275

#####

Detected device has 1 partitions already

Re-partitioning will allow the choice of 1 partitions

#####

Would you like to re-partition the drive anyways [y/n] : 
```

格式化完成，如图：

```
untar update.tar.bz2 to boot partition
bin/
bin/zImage
bin/u-boot.imx
bin/imx6ul-14x14-evk.dtb
bin/ramdisk.img.u
bin/imx6ul-14x14-evk.dts
system/
system/rootfs.tar.bz2
system/zImage
system/u-boot.imx
system/imx6ul-14x14-evk.dtb
system/logo.bmp
Buring th u-boot.imx to sdcard
129+0 records in
129+0 records out
132096 bytes (132 kB) copied, 0.284852 s, 464 kB/s
315+0 records in
315+0 records out
322560 bytes (323 kB) copied, 1.49111 s, 216 kB/s

Syncing....

Un-mount the partitions

Remove created temp directories

Operation Finished
```

卡制作完成后可以看到 **boot** 分区包含 **sdrun** 和 **target** 两个目录。**sdrun** 文件夹内容用于引导系统烧写，无需修改；**target** 目录内容会烧写到 **flash** 芯片。如果需要替换镜像文件，只需将 **target** 目录中的对应文件替换掉，并保持同样的命名，之后再重新进行系统烧写。

3.1.2 TF 卡更新系统

将上一节中制作好的 TF 卡插入 FCU1101 的 TF 卡槽中，给设备上电，按住 **boot** 键的同时按下 **reset** 键，松开 **reset** 后再抬起 **boot** 键。TF 卡中新的固件会自动更新到 FCU1101 中。更新时间较长，系统更新完成后，“**RUN**”指示灯闪烁，更新固件成功。

FCU1101 重新上电或者复位，即可正常使用。

3.2 在线更新固件

有些用户不需要整个系统进行更新，只需要更新内核或者设备树时，可采用在线更新固件。

将待更新的 **dtb**、**zImage** 文件复制到文件系统的某个目录下（本文示例为 **/root**），然后擦除掉对应分区中的旧固件，再分别使用 **nandwrite** 命令将新的固件写到对应的分区中。参考命令如下所示：

➤ 更新内核：

```
root@freescale ~$ flash_erase /dev/mtd3 0 0 /*擦除内核分区*/
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
root@freescale ~$ nandwrite -p /dev/mtd3 /root/zImage /*重新写入内核*/
Writing at 0x00000000
Writing at 0x00020000
Writing at 0x00040000
.....
Writing at 0x00580000
Writing at 0x005a0000
Writing at 0x005c0000
Writing at 0x005e0000
```

➤ 更新 dtb:

```
root@freescale ~$ flash_erase /dev/mtd2 0 0 /*擦出设备树分区*/
Erasing 128 Kibyte @ 2e0000 -- 100 % complete
root@freescale ~$ nandwrite -p /dev/mtd2
/root/imx6ul-14x14-evk-gpmi-256m-fcu1101.dtb
/*重新写入设备树*/
Writing at 0x00000000
```

➤ 更新 uboot:

kobs-ng 是 NXP 官方提供的烧写 **uboot** 到 **nandflash** 中的工具。将新的 **u-boot.imx** 复制到文件系统的某个目录下（本文示例为 **/root**），将原来的 **u-boot.imx** 擦除，将新的 **u-boot.imx** 烧写进去。

```
root@freescale ~$ flash_erase /dev/mtd0 0 0
root@freescale ~$ kobs-ng init -x /root/u-boot.imx
root@freescale ~$ sync
```

第四章 更新用户程序

用户可通过 TF 卡或者网络的 FTP 更新用户程序到文件系统。可参考“应用”文件夹进行设计。

4.1 设置自启动应用

开机自启动脚本为/etc/rc.d/rc.local，可以将开机需要进行设置的命令可以在此脚本中加入。假设开机时自动执行4G拨号脚本/forlinx/quectel-CM &，在/etc/rc.d/rc.local中

```
#start boa server
/sbin/boa
/usr/sbin/dropbear 后加入
/forlinx/quectel-CM &
```

4.2 MQTT 说明

FCU1101 支持 MQTT 协议，2 种方法：一种是采用阿里云的物联网套件，另一种采用 eclipse 的 mosquitto 做 MQTT 的服务器或者客户端。

详细参考：

用户资料/Linux/应用/阿里云设备端 C-SDK/阿里云设备端 C-SDK 移植记录.pdf

用户资料/Linux /应用/mosquito/mqtt 协议移植总结.pdf