# ACT-204DL 系列用户手册

**版本号：V3.0**

## 版本修订

| 版本号 | 修订日期 | 描述 | 审核 |
| --- | --- | --- | --- |
| V1.0 | 20160907 | 创建文档 | |
| V2.0 | 20170929 | 修改文档 | |
| V3.0 | 20171218 | 修改文档 | |

## 特别说明

本公司保留在未通知用户的情况下，对产品、文档、服务等内容进行修改、更正等其他一切变更权利。

# 目录

# 一、产品概述

ACT-204DL 是一款基于精简指令集（RISC）架构高性能的 32 位 MPU 的嵌入式计算机。该 CPU 是以 ARM Cortex-A8 为核心的系统级单芯片，内置 NEON 单指令流多数流（SIMD）协处理，带有错误校正码（ECC）的 256KB L2 缓存，最高支持 1GHz 的频率。系统提供 RS458/RS232 通讯，有线网络通讯，同时也提供无线 GPRS 通讯，具有体积小、功耗低、效率高等特点，适用于电力集中器、HMI、工业控制、网关等场合。

# 二、产品特性

## 1. 硬件特性

- AM355x CPU：
  - 32bit ARM Corte-A8 架构，主频 800MHz，1.6MIPS/MHz，最高主频 1GHz
  - 32KB I-cache，32KB D-cache，NeonSIMD 协处理器
- 内存：
  - 512Mbyte DDR3、64KB 专用 RAM
- FLASH：
  - 256Mbyte NANDFlash，最大支持 8Gbyte
  - 支持 NAND、NOR、SRAM 等 FLASH
- 加密：
  - 支持 PRNG/DES/3DES/AES/SHA/HMAC 加密，最高 256 位加密模式
- 看门狗：
  - 内置 WDT，溢出时间小于 60 秒，支持空闲唤醒和掉电唤醒
- RTC:
  - 高精度实时时钟，内置供电电池
- 调试口：

- 1 路串口为系统 console 口。波特率：115200，数据位：8，停止位：1，校验位：none，流控：无

- RS485/RS232：
  - 4 路 RS485/RS232 通讯端口，可根据实际选择使用，内部全隔离保护设计
- 网络：
  - 2 路 10M/100M 自适应工业以太网，标准 RJ45 接口
  - 15KV TVS 保护，内部全隔离保护设计
- 无线功能：
  - 射频波段 800/900/1800/1900MHz（可选 2/3/4G）
  - 可选 WIFI：可连接 AP，也可做 AP
  - 1 个 SIM 卡接口，1 个天线接口
  - 传输速度：达到相应功能的标准速度
- SD CARD：
  - 内置一个 SD/MMC 卡接口
- 电源：
  - 输入电压：9~36VDC，推荐使用 24VDC/1.5A
  - 单机功耗：< 4W
- 机械特性
  - 外壳金属材质
  - 尺寸：138mm*120mm*55mm
  - 防护等级：IP63
- 工作环境
  - 工作温度：-40℃~+85℃
  - 工作湿度：5% ~ 95%

## 2. 软件特性

### 2.1 系统特性

ACT-204DL 预装基于 TI AM335x 的 Linux 操作系统，版本为 3.2.0。满足 POSIX 标准或类 UNIX 平台的应用程序。针对系统特有的硬件设备，内核提供了简单、易用的驱动接口，可加速用户的应用程序开发。

ACT-204DL 系统的软件系统共分为 3 部分，分别为 Bootloader、linux 内核和 rootfs。Bootloader 是遵循 GPL 条款的开放源码项目，UBoot 主要是引导内核的启动，支持 NFS 挂载、NAND Flash 启动；linux 内核是整个操作系统的最底层，负责整个硬件的驱动，以及提供各种系统所需的核心功能；rootfs 是用于明确磁盘或分区上的文件的方法和数据结构，即在磁盘上组织文件的方法。

### 2.2 环境配置

本公司提供的虚拟机系统：

用户名：work

密　码：123456

编译环境：本公司提供的虚拟机系统 ubuntu 10.04，可直接编译使用

编译命令：arm-linux-gnueabihf-gcc -o filename filename.c

编译链：本公司提供的 arm-linux-gnueabihf-4.7.tar.gz

非本公司提供的编译环境下，把编译链拷贝到 PC 的 LINUX 系统下，解压编译链后，把根目录下的 bin 目录添加到系统的环境变量即可。

添加环境变量：export　PATH=$PATH:/xxx/bin

如解压到/opt/arm-linux-gnu 目录下，则添加环境变量为：

export　PATH=$PATH:/ opt/arm-linux-gnu /bin

### 2.3 管理机登录

IP：

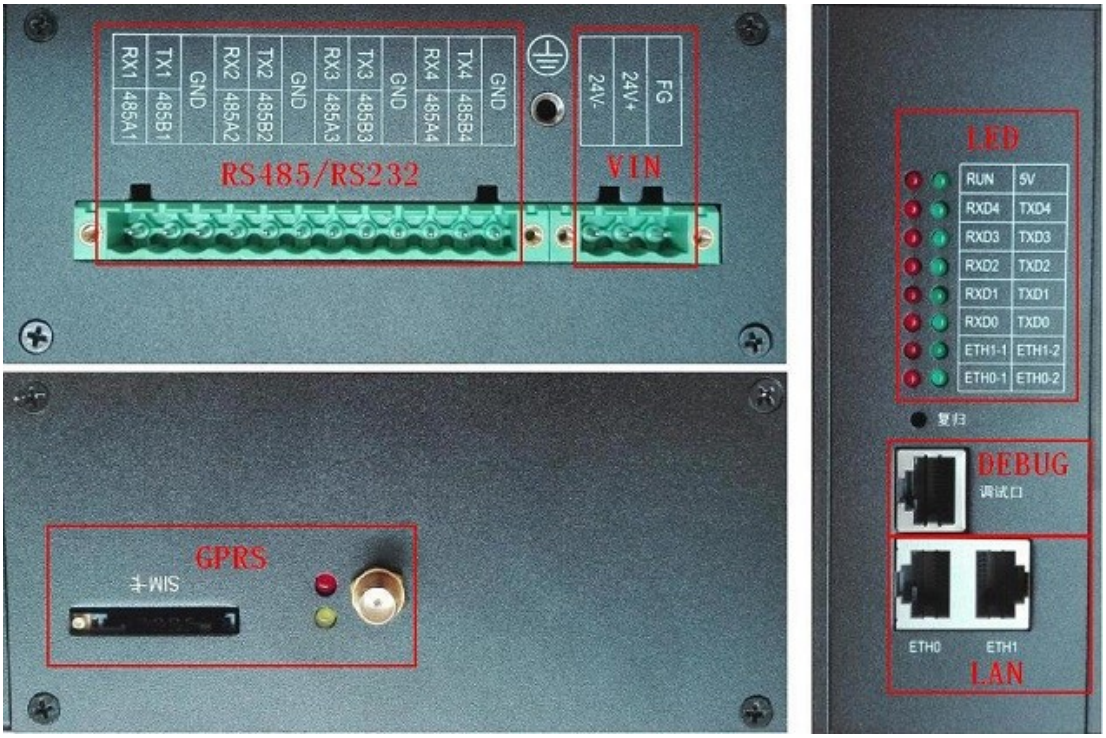
eth0：192.168.1.177

eth1：192.168.2.177

用户名：root

密码　：root

## 三、接口定义




### 1. 电源接口

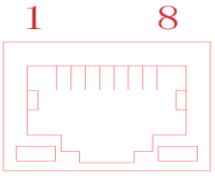
| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | 24V- | 系统电源地 |
| 2 | 24V+ | 系统电源正极 |
| 3 | FG | 屏蔽地、保护地，可不接 |

### 2. RS485 接口

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | RX1/485A1 | 第一通道 RS485 端口 A |
| 2 | TX1/485B1 | 第一通道 RS485 端口 B |
| 3 | GND | GND，系统通讯地 |
| 4 | RX2/485A2 | 第二通道 RS485 端口 A |

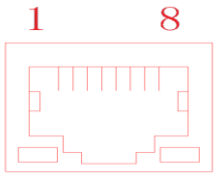
| | | |
|---|---|---|
| 5 | TX2/485B2 | 第二通道 RS485 端口 B |
| 6 | GND | GND，系统通讯地 |
| | RX3/485A3 | 第三通道 RS485 端口 A |
| | TX3/485B3 | 第三通道 RS485 端口 B |
| | GND | GND，系统通讯地 |
| | RX4/485A4 | 第四通道 RS485 端口 A |
| | TX4/485B4 | 第四通道 RS485 端口 B |
| | GND | GND，系统通讯地 |

### 3. 网络接口




| 网口编号 | 编号 | 标识符 | 功能说明 |
|---|---|---|---|
| ETH0/1 | 1 | E0/1_TX+ | 以太网 ETH0/1_TX+ |
| | 2 | E0/1_TX- | 以太网 ETH0/1_TX- |
| | 3 | E0/1_RX+ | 以太网 ETH0/1_RX+ |
| | 4 | NC | 未使用 |
| | 5 | NC | 未使用 |
| | 6 | E0/1_RX- | 以太网 ETH0/1_RX- |
| | 7 | NC | 未使用 |
| | 8 | NC | 未使用 |
| IP | Eth0 | Eth0 | 192.168.1.177 |
| | Eth1 | Eth1 | 192.168.2.177 |

## 4. 调试接口




| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | TX | RS232 调试串口 TX |
| 2 | RX | RS232 调试串口 RX |
| 3 | GND | 系统通讯地 |
| 4-8 | NC | 未使用 |

调试口配置：波特率：115200，数据位：8，停止位：1，校验位：none，流控：无

## 5. 指示灯

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1/2 | RUN/5V | 运行状态指示灯/电源指示灯 |
| 3/4 | RX4/TX4 | 第四通道 RS485/RS232 通讯指示灯 |
| 5/6 | RX3/TX3 | 第三通道 RS485/RS232 通讯指示灯 |
| 7/8 | RX2/TX2 | 第二通道 RS485/RS232 通讯指示灯 |
| 9/10 | RX1/TX1 | 第一通道 RS485/RS232 通讯指示灯 |
| 11/12 | RX0/TX0 | 调试串口 RS232 通讯指示灯 |
| 13/14 | ETH1-1/ETH1-2 | ETH1 状态指示灯 |
| 15/16 | ETH0-1/ETH0-2 | ETH0 状态指示灯 |

### 6. 复归

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | 复归 | 备用，用户可自行编程使用 |

### 7. SD 卡接口

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | SD Card | SD 存储卡接口（内置接口，如需使用，需打开设备） |

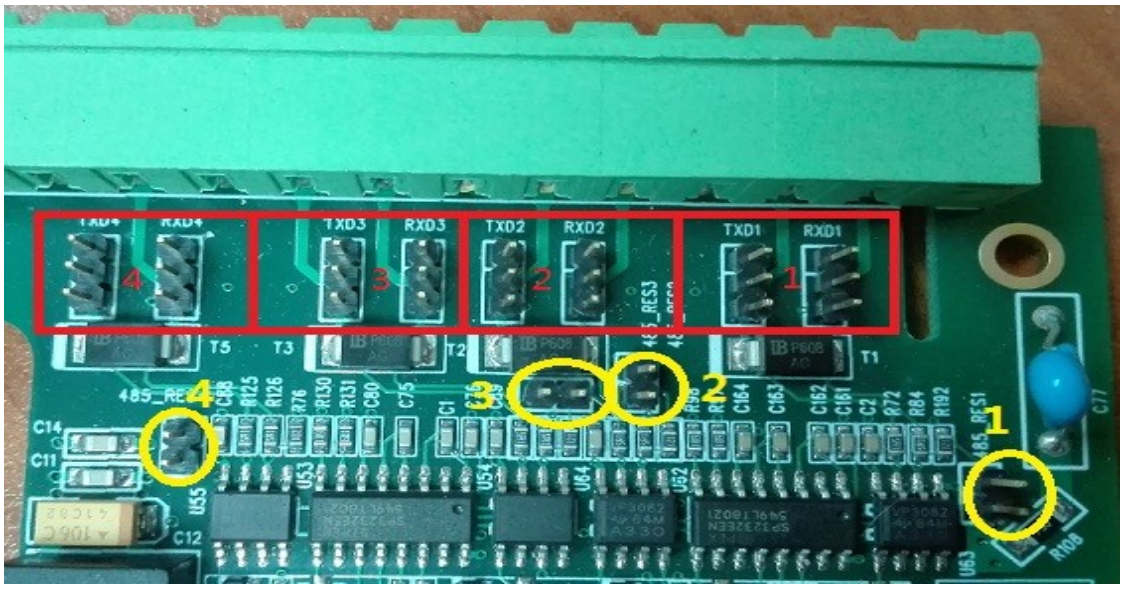
### 8. SIM 卡接口

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | SIM 卡 | 2G/3G/4G 的 SIM 卡接口，支持移动、联通卡 |

### 9. 天线接口及指示灯

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | GPRS | 2G/3G/4G 的 SMA 天线接口 |
| 2 | 红灯 | 电源指示灯 |
| 3 | 红/绿/黄 | 2G/3G/4G 网络状态指示灯 |

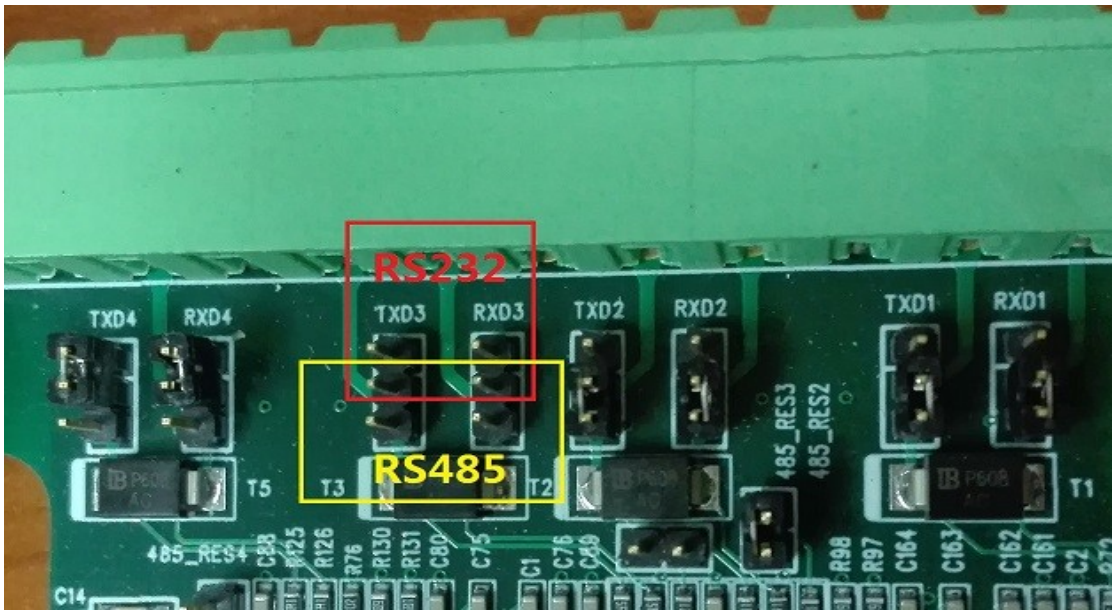
### 10. RS485/RS232 功能选择




如上图，红色部分为 RS485/RS232 功能选择；黄色部分为 RS485 终端电阻选择功能。

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | TXD1/RXD1 | 第一组串口 RS485/RS232 功能选择 |
| 2 | TXD2/RXD2 | 第二组串口 RS485/RS232 功能选择 |
| 3 | TXD3/RXD3 | 第三组串口 RS485/RS232 功能选择 |
| 4 | TXD4/RXD4 | 第四组串口 RS485/RS232 功能选择 |

**10.1RS485/RS232 功能选择。**

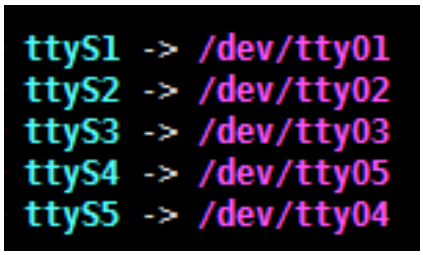


如上图所示，短接红色框中的两个插针，则为 RS232 功能，短接黄色框中的两个插针，则为 RS485 功能。

### 10.2RS485 终端电阻选择

如选择的是 RS485 功能，如果需要终端匹配电阻，则把终端电阻选择功能的插针短接，如不需要，则断开。

## 11. RS485/RS232 驱动接口




驱动接口可以在管理机的/dev 目录下查看。

| 编号 | 标识符 | 功能说明 |
|---|---|---|
| 1 | ttyS1 | 第一组 RS485/RS232 串口 |
| 2 | ttyS2 | 第二组 RS485/RS232 串口 |
| 3 | ttyS3 | 第三组 RS485/RS232 串口 |
| 4 | ttyS4 | 第四组 RS485/RS232 串口 |
| 5 | ttyS5 | GPRS 通讯串口（2G 模块才会使用） |

## 四、驱动实例

在系统的/program 目录下有相应的脚本文件，可以进行一些简单的测试。其中要确保 startup.sh 文件里，端口映射的正确的。文件内容见附录。

脚本及说明：

4G-connect-chat：4G 拨号需要的脚本

at_u 或 at_u.u：4G 模块 AT 指令脚本，可用于查看信号、中心号码等信息

dial-on.sh：拨号脚本，在给 gprs 上电开机后，即可使用此脚本进行拨号

gprs-on.sh：GPRS 上电开机

gprs-off.sh：GPRS 关机断电

serial_3 或 serial：有此名称的为串口测试脚本。

watchdog：看门狗测试程序

startup.sh：开机启动脚本，如果有需要开机启动的程序，可添加到此文件内，注意程序路径的问题。

### 1. RS485/RS232 接口驱动的使用

RS485 和 RS232 是可选的，通过改成接口内部的跳线实现硬件上的转换。

| 编号 | 通讯接口 | 设备文件 |
|---|---|---|
| 1 | RS485/RS232-1 | /dev/ttyS1 |
| 2 | RS485/RS232-2 | /dev/ttyS2 |
| 3 | RS485/RS232-3 | /dev/ttyS3 |
| 4 | RS485/RS232-4 | /dev/ttyS4 |

### 1.1 RS485/RS232 发送、接收数据

使用方法：

```
cd program
./serial n        // n = 1 ,2 ,3,4
```

此时串口 n 会有数据输出，同时也能接收外面传的进来的数据附录：serial.c 文件内容。

## 2. GPRS 操作

GPRS 操作的相应文件，请查看附件文件。

2G：对应的设备文件为 ttyS5

3G/4G：对应的设备文件为 ttyUSB3

### 2.1 GPRS 的 AT 指令使用

使用例子：2G：at_c　5　AT

3G/4G：at_u　2　AT

### 2.2 GPRS 开机拨号

参考附件：GPRS 使用说明

# 附录：

## 1. startup.sh 文件内容：

```
#!/bin/sh
ln -sf /dev/ttyO1 /dev/ttyS1
ln -sf /dev/ttyO2 /dev/ttyS2
ln -sf /dev/ttyO3 /dev/ttyS3
ln -sf /dev/ttyO5 /dev/ttyS4
ln -sf /dev/ttyO4 /dev/ttyS5
```

## 2. serial.c 文件内容：

```
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
```

```
#define max_buffer_size     100     /* buffer size */
/*****************************************/
int fd1;
int flag_close;

int open_serial(int k,int *fd)
{
      int sfd = -1;
      char str[100];
      sprintf(str,"/dev/ttyS%d",k);
      printf("open %s\n",str);

      sfd = open(str,O_RDWR|O_NOCTTY| O_NONBLOCK);
      if(sfd == -1){
            perror(str);
            return -1;
      }
      else{
            *fd = sfd;
            return 0;
      }
}
/*********************************************************************/
int main(int argc, char *argv[ ] )
{
      time_t tNow,tOld;
      int port;
```

```
    char
sbuf[]={"12345678901234567890123456789012345678901234567890\n"};/*
固定发送的数据  */
    char sbufrec[256]={0};
    int sfd,retv,i,ncount=0,mcount = 0;
    struct termios opt;
    int length=sizeof(sbuf);
    /*****************************************************************/
    if(argc < 2)
    {
        printf("input erro :serial <1~4> \n");
        return 0;
    }
    port = atoi(argv[1]);
    open_serial(port,&fd1);
    /*****************************************************************/
    printf("ready for sending data...\n");
    tcgetattr(fd1,&opt);
    cfmakeraw(&opt);
    /*****************************************************************/
    cfsetispeed(&opt,B9600); /*设置波特率为 9600bps*/
    cfsetospeed(&opt,B9600);
    /*****************************************************************/
    tcsetattr(fd1,TCSANOW,&opt);

    while(mcount < 5)
    {
        retv=write(fd1,sbuf,length);    /*  发送数据  */
```

```
if(retv==-1){
        //perror("write");
        printf("write error .....\n");
}
else{
        printf("the number of char sent is %d\n",retv);
}
ncount=0;
printf("ready for receiving data...\n");

time(&tOld);
tNow=tOld;
ncount = 0;
while(((tNow-tOld) < 2))            /*  设置接收超时  */
{
        time(&tNow);
        retv=read(fd2,&sbufrec[0],1);
        if(retv==-1){
                //perror("read");
                //printf("error read \n");
                //printf("tOld=%d;tNow=%d\n",tOld,tNow);
        }
        else{
                printf("%02x ",sbufrec[0]);
                ncount+=1;
        }
}
mcount+=1;
```

```
            printf("\n");
        }
        flag_close = close(fd1);
        if(flag_close == -1)     /*关闭口端口*/
            printf("Close the Device1 failur!\n");
        return 0;
}
```