ACT-408RL-B 用户手册

版本号：V1.0

瑞强科技

版本修订

版本号	修订日期	描述	审核
V1.0	20201028	创建文档	

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-408RL-B 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，CAN 总 线，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：
  - 标配 256Mbyte NANDFlash，最大支持 1Gbyte
  - 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：
  - 6 路 RS485 通讯，可通过内部路线，选择阻抗匹配电阻。
  - 2 路 RS485/RS232 通讯，分时复用。可根据实际选择使用，内部全隔离保护设计
- CAN：
  - 2 路 CAN 通讯，内置隔离保护设计
- 开关量输出输入：
  - 2 路继电器干节点输出
  - 2 路开关量输入，硬件可配置干/湿节点输入
- 网络：
  - 4 路 10M/100M 自适应工业以太网，标准 RJ45 接口
  - 内部 EMC 保护，隔离保护设计
- 无线功能（可选）：
  - 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
- USB：
  - 内置一个 USB HOST 接口
- 电源：
  - 输入电压：85~265VAC 交流、110~300VDC 直流
  - 单机功耗：< 10W
- 机械特性
  - 外壳金属材质
  - 尺寸：1U
  - 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-408RL-B 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-408RL-B 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

2.2.1 使用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 ubuntu10.04，可直接编译使用

编译命令：arm-linux-gnueabihf-gcc -o target source1.c source2.c

编译链：arm-linux-gnueabihf-4.7.tar.gz

2.2.2 使用自定义 linux 系统

例：使用 ubuntu14.04 或者更高版本的 64 位系统，需要安装 32 位库。安装命令如下：

```
sudo apt-get install ia32-lib
sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 sudo apt-get
install lib32stdc++6
```

安装完后，在/opt 目录下新建目录 arm-a8-cross

```
mkdir -p /opt/arm-a8-cross
```

把编译链拷贝到 ubuntu 的/opt 目录下，解压到刚建立的目录下

```
tar xzf arm-linux-gnueabihf-4.7.tar.gz -C /opt/arm-a8-cross
```

添加临时环境变量或者永久环境变量：

临时：export PATH=$PATH:/opt/arm-a8-cross /bin

永久：echo export PATH=$PATH:/opt/arm-a8-cross /bin >> ~/.bashrc

添加完成后，重新打开终端。查看编译链版本

arm-linux-gnueabihf-gcc -v

如图所示为环境搭建完成：

编译命令：arm-linux-gnueabihf-gcc -o target source1.c source2.c

2.3 管理机登录

2.3.1 调试口（console）登录

硬件接口及线序，详见本说明第三章第 6 项。正常连接后，可以直接输入用户名和密码即可进入系统。

2.3.2 网络登录

默认网口 IP 地址：

编号	设备	IP
1	eth0	192.168.1.177
2	eth1	192.168.2.177
3	eth2	192.168.3.177

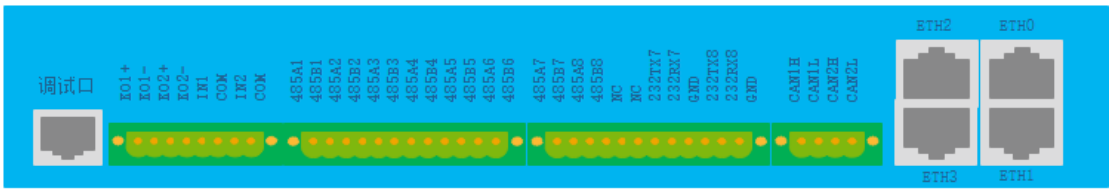
4	eth3	192.168.4.177

用户名及密码

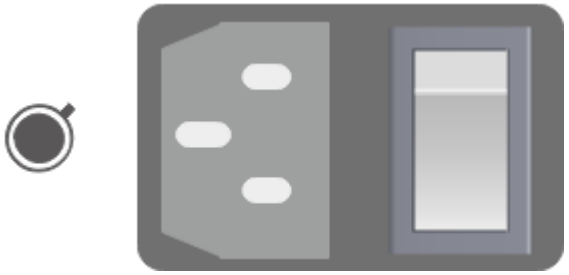
用户名	密码
Root	root

登录方式：可用 telnet 或者 ssh。

三、接口定义

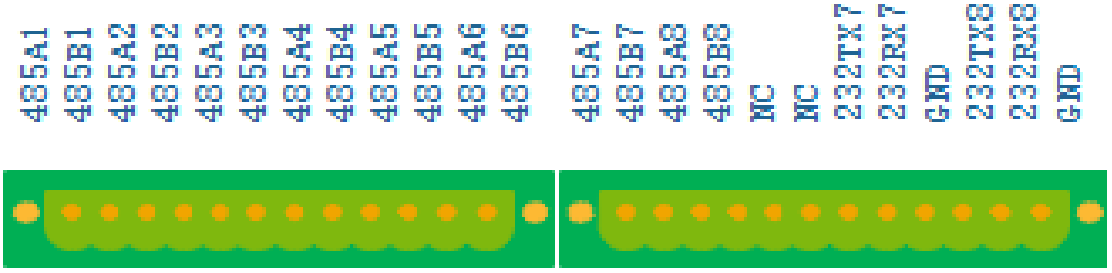


1. 电源接口



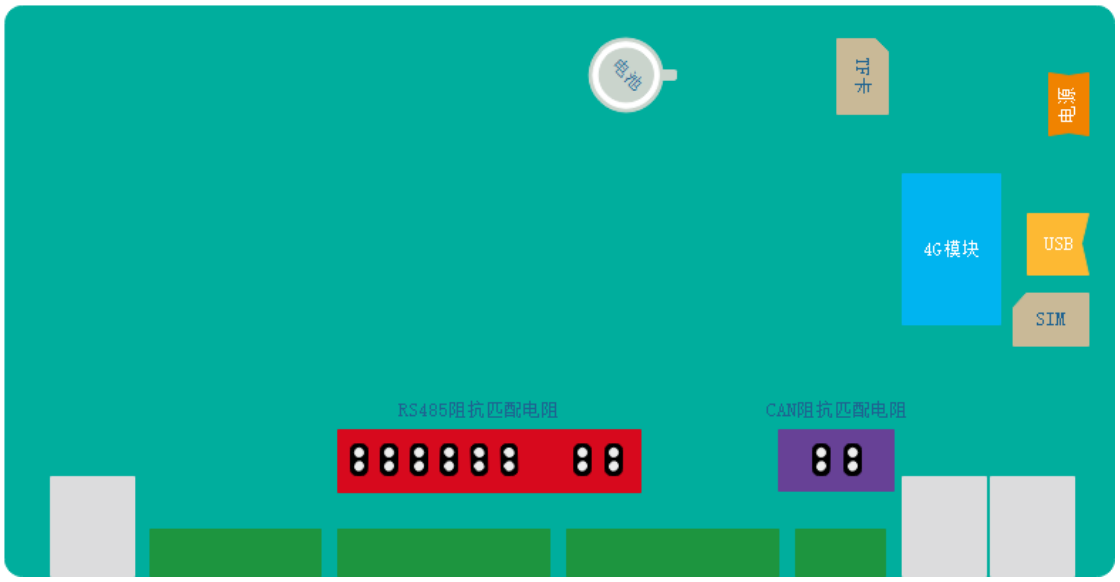
编号	标识符	功能说明
1	AC/DC220V	电源接口，支持交流/直流
2		外壳接地端子（非零线N）

2. RS485 接口

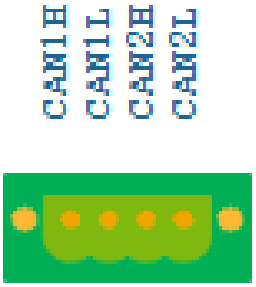


编号	标识符	功能说明
1	485An	第n 通道RS485-A 接口（n=1~6）
2	485Bn	第n 通道RS485-B 接口（n=1~6）
3	232TXn	第n 通道RS232-TX 接口（n=7~8）
4	232RXn	第n 通道RS232-RX 接口（n=7~8）
5	GND	GND，通讯地
6	NC	空端子

注：RS485 阻抗匹配电阻，需通过内部跳线选择。出厂默认为带 120 欧阻抗匹配电阻。如下图，红色框内的为RS485 阻抗匹配电阻跳线。

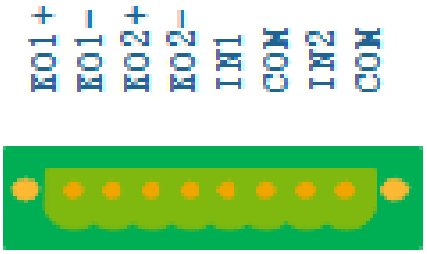


3. CAN 接口



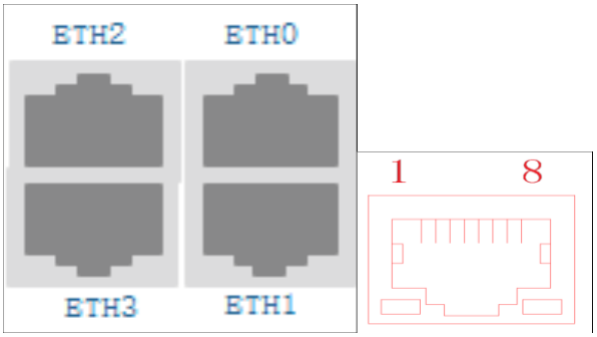
编号	标识符	功能说明
1	CANnH	第n 通道CAN 总线H 端（n=1,2）
2	CANnL	第n 通道CAN 总线L 端（n=1,2）

4. 开关量输出输入



编号	标识符	功能说明
1	Kon+	第n 路继电器干节点输出端A（n=1,2）
2	Kon-	第n 路继电器干节点输出端B（n=1,2）
3	INn	第n 路开关量干节点输入端A（n=1,2）
4	COM	开关量公共输入端

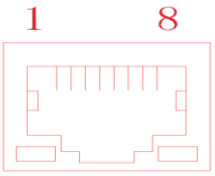
5. 网络接口



网口编号	编号	标识符	功能说明
ETH0/1/2/3	1	E_TX+	以太网ETH_TX+
	2	E_TX-	以太网ETH_TX-
	3	E_RX+	以太网ETH_RX+
	4	NC	未使用
	5	NC	未使用
	6	E_RX-	以太网ETH_RX-
	7	NC	未使用
	8	NC	未使用

IP	0	Eth0	192.168.1.177
	1	Eth1	192.168.2.177
	2	Eth2	192.168.3.177
	3	Eth3	192.168.4.177

6. 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

7. RS485/RS232 驱动接口映射

```
/dev/ttyS1 -> /dev/ttyCH0
/dev/ttyS2 -> /dev/ttyCH1
/dev/ttyS3 -> /dev/ttyCH2
/dev/ttyS4 -> /dev/ttyCH3
/dev/ttyS5 -> /dev/ttyCH4
/dev/ttyS6 -> /dev/ttyCH5
/dev/ttyS7 -> /dev/ttyCH6
/dev/ttyS8 -> /dev/ttyCH7
/dev/ttyS9 -> /dev/ttyO1
```

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	ttySn	对应ttyCH(n-1)，第n 通道RS485/RS232 驱动接口
2	ttyS9	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。

附录：

1、startup.sh 文件内容：

```
#!/bin/sh
ln -sf /dev/ttyCH0 /dev/ttyS
ln -sf /dev/ttyCH1 /dev/ttyS2
ln -sf /dev/ttyCH2 /dev/ttyS3
ln -sf /dev/ttyCH3 /dev/ttyS4
ln -sf /dev/ttyCH4 /dev/ttyS5
ln -sf /dev/ttyCH5 /dev/ttyS6
ln -sf /dev/ttyCH6 /dev/ttyS7
ln -sf /dev/ttyCH7 /dev/ttyS8
 ln -sf /dev/ttyO1 /dev/ttyS9
ip link set can0 type can bitrate 100000
ifconfig can0 up
ip link set can1 type can bitrate 100000
ifconfig can1 up
```

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;
}
```

3.io_out.c 文件内容：

```
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <stdint.h>

#define MAXFILENAME_LEN 100
const char *gpio_dir = "/sys/class/gpio/";

int gpio[2] = {
    115,  /* OUT1 */
    114,  /* OUT2 */
};

int write_sysfs_int(int port, int val)
{
    int ret;
    FILE *sysfsfp;
```

```
    char temp[MAXFILENAME_LEN] = "\0";
    sprintf(temp, "%sgpio%d/value",gpio_dir,port);
    sysfsfp = fopen(temp, "w");
    if (sysfsfp == NULL) {
        printf("failed to open %s\n", temp);
        return -1;
    }
    fprintf(sysfsfp, "%d", val);
    fclose(sysfsfp);
    return 0;
}

int main(int argc, char *argv[ ])
{
    int port,val;
    if(argc<3)
    {
        printf("input erro :\n Useage io_out (port no) (value)\n");
        return 0;
    }
    port = atoi(argv[1]);
    val = atoi(argv[2]);
    if((port<=0) || (port>2)){
        printf("The Port no Only 1-2 is valid\n");
        return -1;
    }
    port--;
    return write_sysfs_int(gpio[port], val);
    return 0;
}
```

4.io_in.c 文件内容：

```
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <stdint.h>
```

```
#define MAXFILENAME_LEN 100
const char *gpio_dir = "/sys/class/gpio/";

int gpio[2] = {
        117,  /* IN1 */
        20,  /* IN2 */
};

int read_sysfs_int(int port, int *val)
{
        int ret;
        FILE *sysfsfp;
        char temp[MAXFILENAME_LEN] = "\0";
        sprintf(temp, "%sgpio%d/value",gpio_dir,port);
        sysfsfp = fopen(temp, "r");
        if (sysfsfp == NULL) {
                printf("failed to open %s\n", temp);
                return -1;
        }
        fscanf(sysfsfp, "%d", val);
        fclose(sysfsfp);
        return 0;
}

int main(int argc, char *argv[ ])
{
        int port,val;
        if(argc<2)
        {
                printf("input erro :\n Useage io_out (port no)\n");
                return 0;
        }
        port = atoi(argv[1]);
        if((port<=0) || (port>2)){
                printf("The Port no Only 1-2 is valid\n");
                return -1;
        }
        port--;
```

```
    if(read_sysfs_int(gpio[port], &val))
        return -1;
    else
        printf("val = %d\n",val);
    return 0;
}
```