USR-EG828 技术手册

目录

1. 引言

本文主要对 USR-EG828 产品的使用过程中的相关硬件和软件接口进行说明，方便用户在拿到产品后，更快的对产品进行适配操作。USR-EG828 内置标准 linux ubuntu 20.04 系统。

2. 硬件接口描述

2.1. Linux GCC 编译器下载

命令	接口
apt-get update	更新 Linux 软件源
apt-get install gcc	下载并安装 GCC 编译器

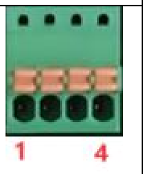
*如果使用交叉编译器，需要去特定网站下载

2.2. 串口

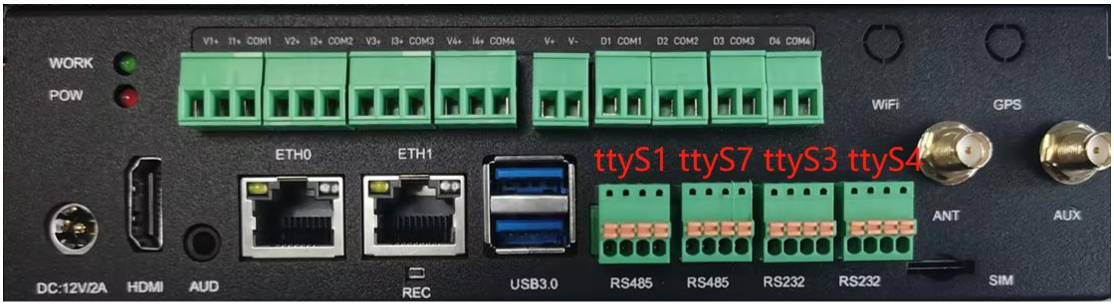
EG828 串口驱动标识说明：

接口	驱动标识
RS485-1	ttyS1
RS485-2	ttyS7
RS232-1	ttyS3
RS232-1	ttyS4

RS232/RS485 接口(RS485/RS232) 座子间距 2.0MM

序号	定义	属性	描述	
1	3.3V	输出	3.3V 电压输出	
2	TX/A	输出	发送（TX/A）	
3	RX/B	输入	接收（RX/B）	
4	GND	地线	地线	

接口标注图片：



Demo 函数示例：

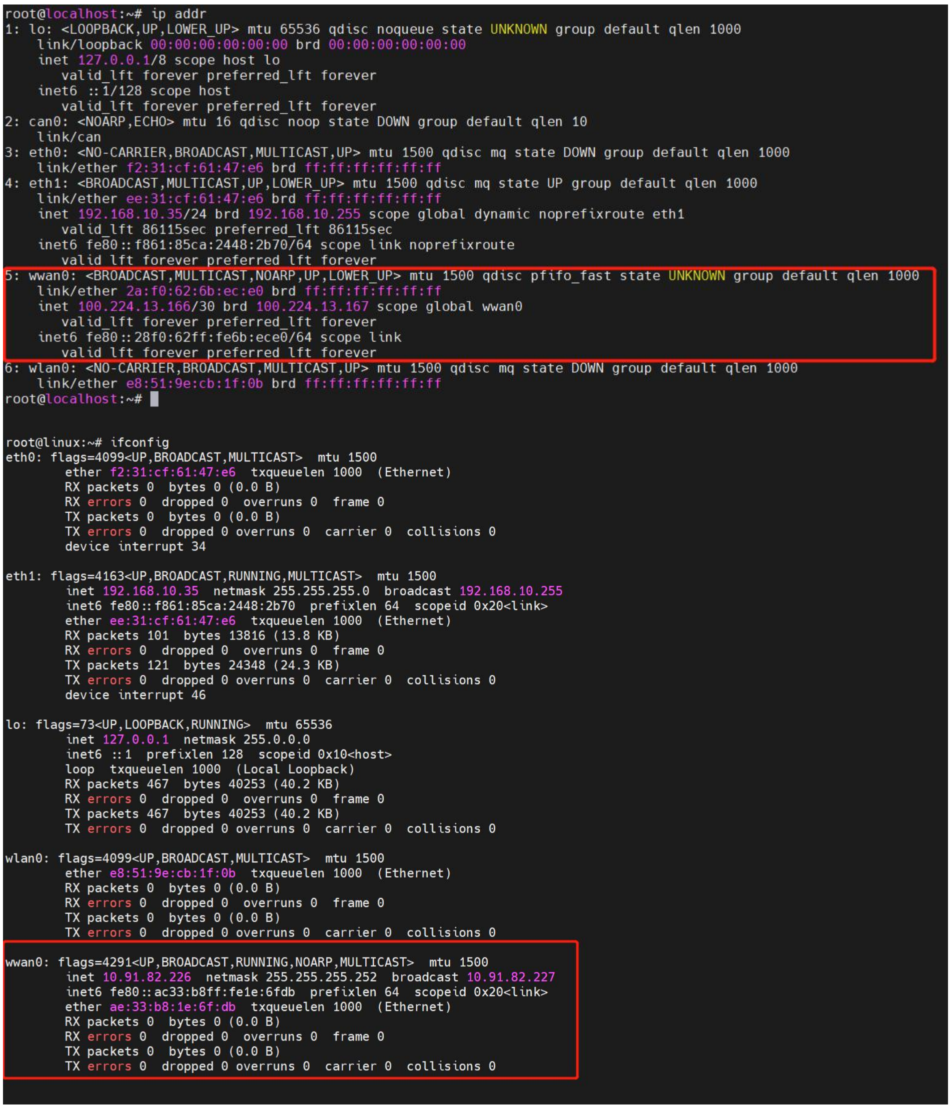
```
int UART_INIT_ttyS1(void)
{
    int serial_port =0;
    serial_port = open("/dev/ttyS1", O_RDWR);
    if (serial_port < 0) {
        printf("Error %i from open: %s\n", errno, strerror(errno));
        return 0;
    }


    // Create new termios struct, we call it 'tty' for convention
    struct termios tty;
    // Read in existing settings, and handle any error
    if(tcgetattr(serial_port, &tty) != 0) {
        printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
        return 0;
    }

    tty.c_cflag &= ~PARENB; // Clear parity bit, disabling parity (most common)
    tty.c_cflag &= ~CSTOPB; // Clear stop field, only one stop bit used in communication (most common)
    tty.c_cflag &= ~CSIZE; // Clear all bits that set the data size
    tty.c_cflag |= CS8; // 8 bits per byte (most common)
    tty.c_cflag &= ~CRTSCTS; // Disable RTS/CTS hardware flow control (most common)
    tty.c_cflag |= CREAD | CLOCAL; // Turn on READ & ignore ctrl lines (CLOCAL = 1)

    tty.c_lflag &= ~ICANON;
    tty.c_lflag &= ~ECHO; // Disable echo
    tty.c_lflag &= ~ECHOE; // Disable erasure
    tty.c_lflag &= ~ECHONL; // Disable new-line echo
    tty.c_lflag &= ~ISIG; // Disable interpretation of INTR, QUIT and SUSP
    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // Turn off s/w flow ctrl
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL); // Disable any special handling of received bytes

    tty.c_oflag &= ~OPOST; // Prevent special interpretation of output bytes (e.g. newline chars)
    tty.c_oflag &= ~ONLCR; // Prevent conversion of newline to carriage return/line feed
    tty.c_cc[VTIME] = 10;    // Wait for up to 1s (10 deciseconds), returning as soon as any data is received.
    tty.c_cc[VMIN] = 0;

    // Set in/out baud rate to be 9600
    cfsetispeed(&tty, B9600);
    cfsetospeed(&tty, B9600);

    // Save tty settings, also checking for error
    if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
        return 0;
    }
    return serial_port;
}
```

2.3. 蜂窝网络

EG828 内置 4G 蜂窝模组，可以直接上网，可以在上电后登录桌面直接查看网络参数，也可以通过 linux 命令进行查询，常用查询命令如下：

命令	接口
ifconfig	wwan

ip addr	wwan

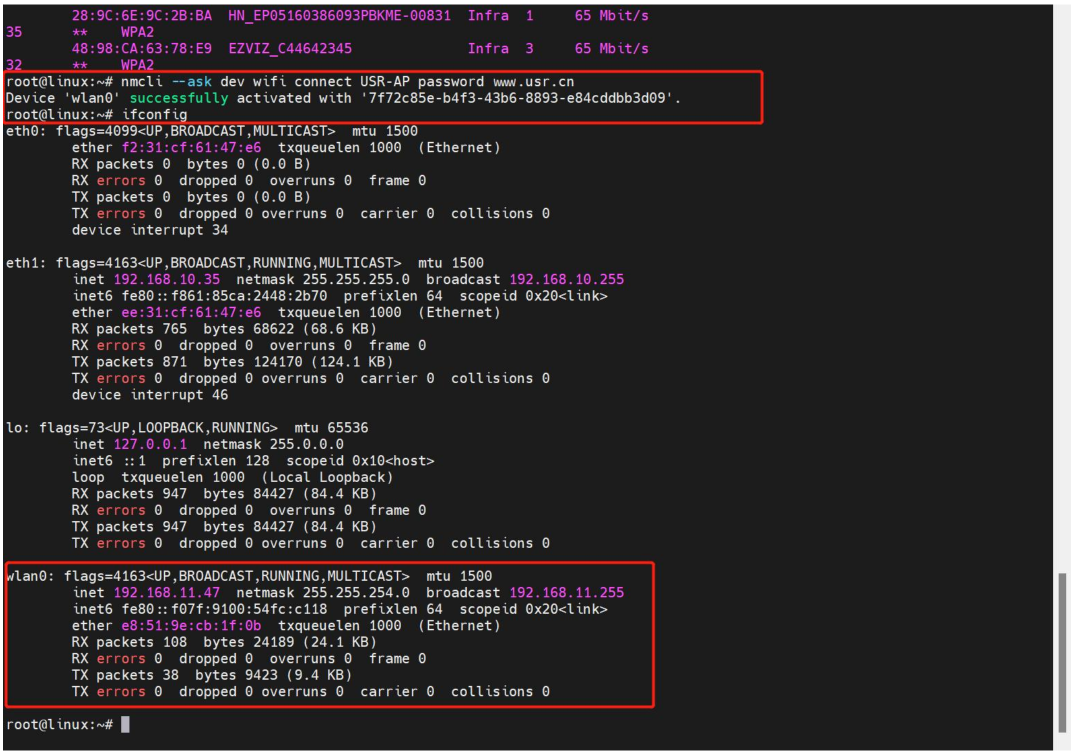


```
root@localhost:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN group default qlen 10
    link/can
3: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether f2:31:cf:61:47:e6 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether ee:31:cf:61:47:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.35/24 brd 192.168.10.255 scope global dynamic noprefixroute eth1
       valid_lft 86115sec preferred_lft 86115sec
    inet6 fe80::f861:85ca:2448:2b70/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: wwan0: <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 2a:f0:62:6b:ec:e0 brd ff:ff:ff:ff:ff:ff
    inet 100.224.13.166/30 brd 100.224.13.167 scope global wwan0
       valid_lft forever preferred_lft forever
    inet6 fe80::28f0:62ff:fe6b:ece0/64 scope link
       valid_lft forever preferred_lft forever
6: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether e8:51:9e:cb:1f:0b brd ff:ff:ff:ff:ff:ff
root@localhost:~#


root@linux:~# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether f2:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 34

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.35  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::f861:85ca:2448:2b70  prefixlen 64  scopeid 0x20<link>
        ether ee:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 101  bytes 13816 (13.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 121  bytes 24348 (24.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 46

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 467  bytes 40253 (40.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 467  bytes 40253 (40.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e8:51:9e:cb:1f:0b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wwan0: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.91.82.226  netmask 255.255.255.252  broadcast 10.91.82.227
        inet6 fe80::ac33:b8ff:fe1e:6fdb  prefixlen 64  scopeid 0x20<link>
        ether ae:33:b8:1e:6f:db  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2.4. WIFI

EG828 内置 WiFi 模组，可以直接进行 WiFi 连接，可以在上电后登录桌面直接进行 WiFi 连接操作，也可以通过 linux 命令进行连接操作，常用查询命令如下：

命令	**功能**
nmcli dev wifi list	搜索 WiFi 热点
nmcli --ask dev wifi connect <SSID> password <password>	连接到指定 WiFi 热点
ifconfig	查询网络状态（WLAN0）
nmcli device disconnect wlan0	断开 WiFi
nmcli connection delete id <SSID>	清除 WiFi 信息



```
        28:9C:6E:9C:2B:BA  HN_EP05160386093PBKME-00831  Infra  1     65 Mbit/s
35      **    WPA2
        48:98:CA:63:78:E9  EZVIZ_C44642345              Infra  3     65 Mbit/s
32      **    WPA2
root@linux:~# nmcli --ask dev wifi connect USR-AP password www.usr.cn
Device 'wlan0' successfully activated with '7f72c85e-b4f3-43b6-8893-e84cddbb3d09'.
root@linux:~# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether f2:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 34

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.35  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::f861:85ca:2448:2b70  prefixlen 64  scopeid 0x20<link>
        ether ee:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 765  bytes 68622 (68.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 871  bytes 124170 (124.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 46

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 947  bytes 84427 (84.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 947  bytes 84427 (84.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.11.47  netmask 255.255.254.0  broadcast 192.168.11.255
        inet6 fe80::f07f:9100:54fc:c118  prefixlen 64  scopeid 0x20<link>
        ether e8:51:9e:cb:1f:0b  txqueuelen 1000  (Ethernet)
        RX packets 108  bytes 24189 (24.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 9423 (9.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux:~#
```

2.5. GPS

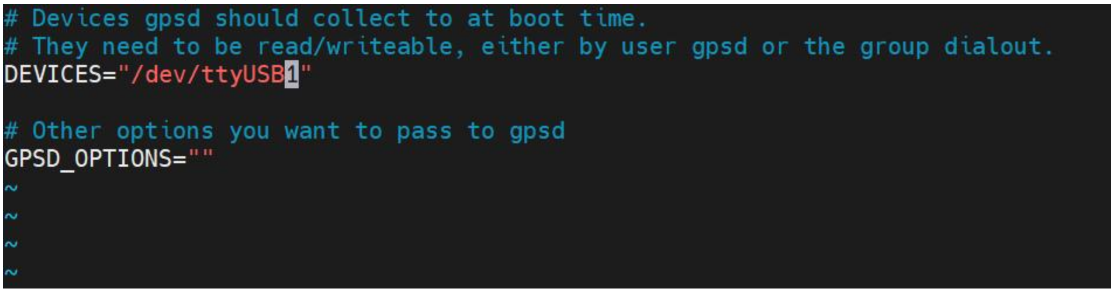
EG828-GL 版本内置 GPS 定位功能，可以通过如下指令进行配置开启并查看获取信息：

命令	**功能**
apt-get install gpsd gpsd-clients	安装 GPSD Client
vim /etc/default/gpsd	修改 GPS 信息接口
echo -ne ″at+qgps=1\r\n″ > /dev/ttyUSB2	开启 GPS 功能
cgps -s	查看定位信息

具体操作步骤如下：

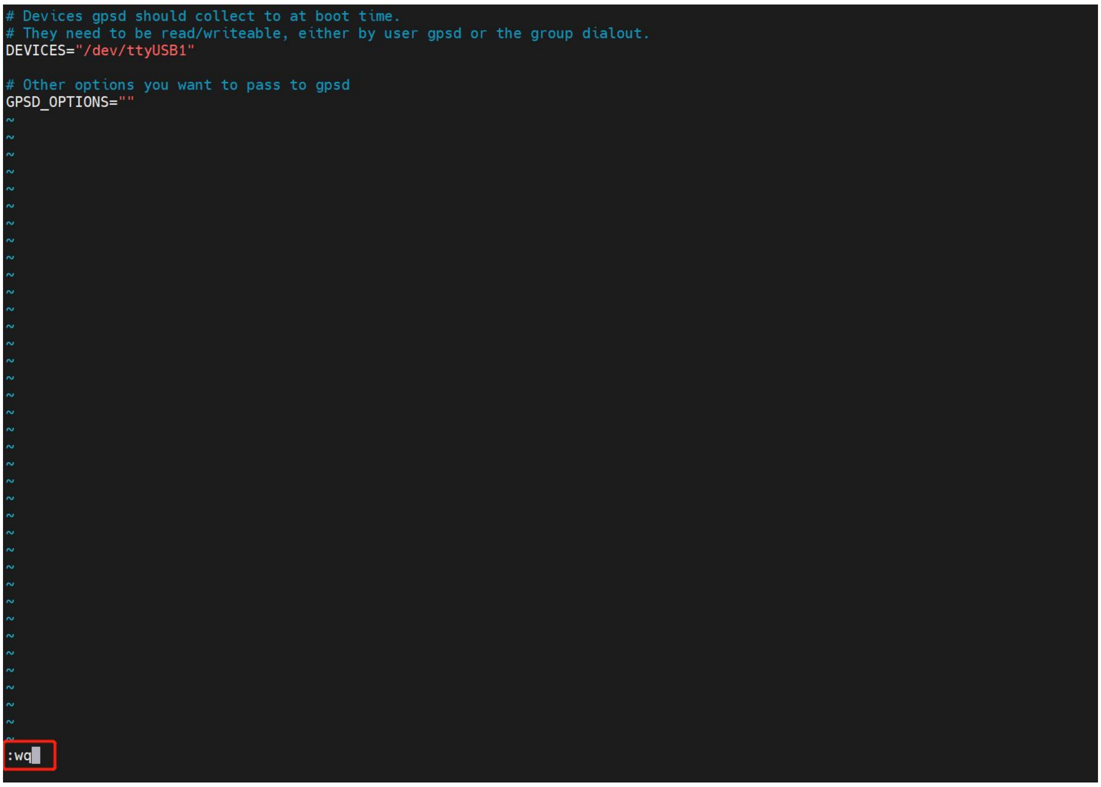
1、 安装 GPSD Client：apt-get install gpsd gpsd-clients

2、 修改 GPS 信息接口：vim /etc/default/gpsd 进入到 gpsd 文件，然后输入：“i”，此时文档进入输入模式，然后修改信息接口为 USB1，具体信息如下图：



```
# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB1"

# Other options you want to pass to gpsd
GPSD_OPTIONS=""
~
~
~
~
```

3、 修改完成后，按住 ctrl+x 进行保存，最好多按两次，然后在使用 esc 键进行退出输入模式，输入 :wq 进行文件保存后，退出到命令界面。

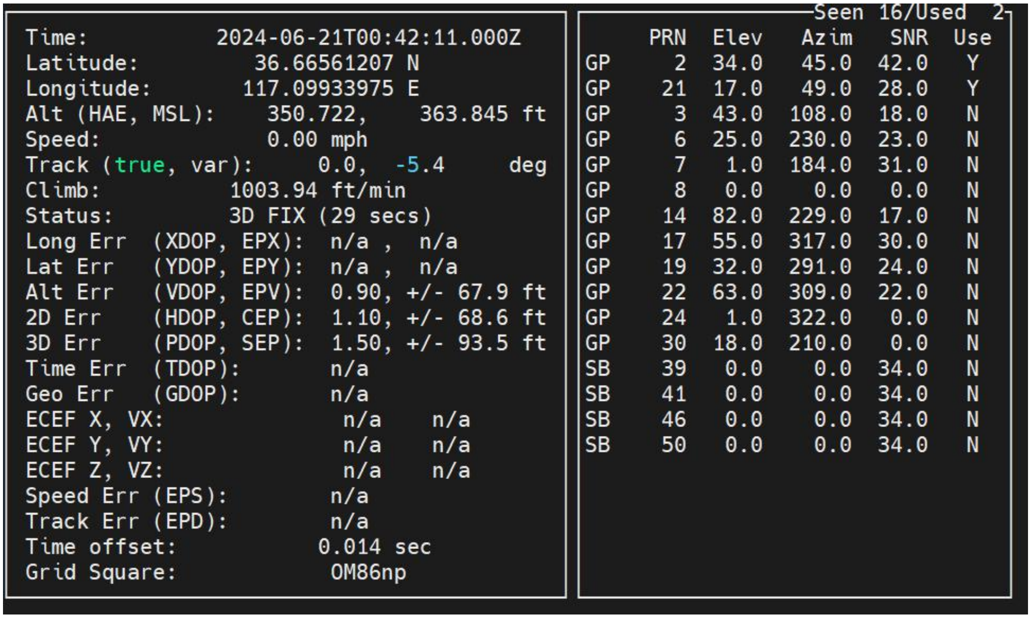


```
# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB1"

# Other options you want to pass to gpsd
GPSD_OPTIONS=""
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

4、 开启 GPS 前首先确认命令通道通畅，执行 cat /dev/ttyUSB2 命令，检测通道是否通畅。如果收到命令回显，则表示正常，如下图，此时退出到命令模式。

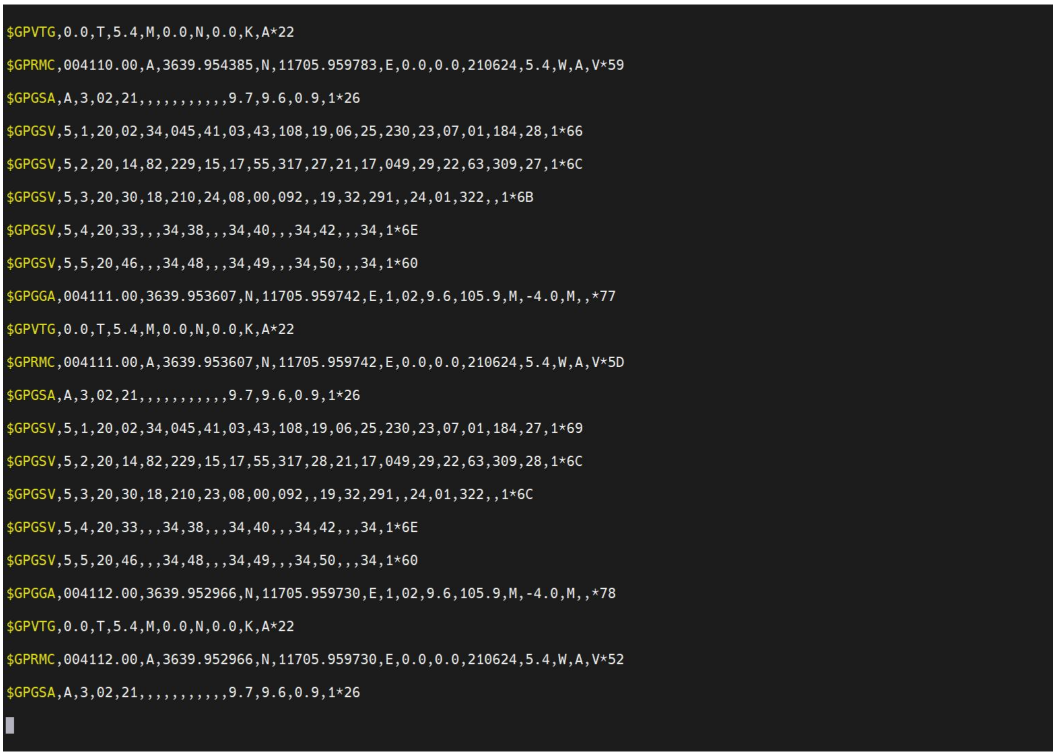
5、 执行 echo -ne "at+qgps=1\r\n" > /dev/ttyUSB2 命令，开启 GPS 功能。

6、 执行 cgps -s 后进入 GPS 信息展示界面，等待一会将出现 GPS 定位信息。



```
                                                      Seen 16/Used  2
Time:          2024-06-21T00:42:11.000Z       PRN  Elev   Azim   SNR  Use
Latitude:         36.66561207 N            GP    2  34.0   45.0  42.0   Y
Longitude:       117.09933975 E            GP   21  17.0   49.0  28.0   Y
Alt (HAE, MSL):    350.722,    363.845 ft  GP    3  43.0  108.0  18.0   N
Speed:             0.00 mph                GP    6  25.0  230.0  23.0   N
Track (true, var):     0.0,  -5.4     deg  GP    7   1.0  184.0  31.0   N
Climb:           1003.94 ft/min            GP    8   0.0    0.0   0.0   N
Status:          3D FIX (29 secs)          GP   14  82.0  229.0  17.0   N
Long Err  (XDOP, EPX):  n/a ,  n/a         GP   17  55.0  317.0  30.0   N
Lat Err   (YDOP, EPY):  n/a ,  n/a         GP   19  32.0  291.0  24.0   N
Alt Err   (VDOP, EPV):  0.90, +/- 67.9 ft  GP   22  63.0  309.0  22.0   N
2D Err    (HDOP, CEP):  1.10, +/- 68.6 ft  GP   24   1.0  322.0   0.0   N
3D Err    (PDOP, SEP):  1.50, +/- 93.5 ft  GP   30  18.0  210.0   0.0   N
Time Err  (TDOP):       n/a                SB   39   0.0    0.0  34.0   N
Geo Err   (GDOP):       n/a                SB   41   0.0    0.0  34.0   N
ECEF X, VX:              n/a    n/a        SB   46   0.0    0.0  34.0   N
ECEF Y, VY:              n/a    n/a        SB   50   0.0    0.0  34.0   N
ECEF Z, VZ:              n/a    n/a
Speed Err (EPS):        n/a
Track Err (EPD):        n/a
Time offset:          0.014 sec
Grid Square:            OM86np
```

7、 也可以通过执行 cat /dev/ttyUSB1 来查看 GPS 的原始数据，也可以通过 USB1 的接口驱动来获取数据并进行处理。



```
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004110.00,A,3639.954385,N,11705.959783,E,0.0,0.0,210624,5.4,W,A,V*59
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
$GPGSV,5,1,20,02,34,045,41,03,43,108,19,06,25,230,23,07,01,184,28,1*66
$GPGSV,5,2,20,14,82,229,15,17,55,317,27,21,17,049,29,22,63,309,27,1*6C
$GPGSV,5,3,20,30,18,210,24,08,00,092,,19,32,291,,24,01,322,,1*6B
$GPGSV,5,4,20,33,,,34,38,,,34,40,,,34,42,,,34,1*6E
$GPGSV,5,5,20,46,,,34,48,,,34,49,,,34,50,,,34,1*60
$GPGGA,004111.00,3639.953607,N,11705.959742,E,1,02,9.6,105.9,M,-4.0,M,,*77
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004111.00,A,3639.953607,N,11705.959742,E,0.0,0.0,210624,5.4,W,A,V*5D
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
$GPGSV,5,1,20,02,34,045,41,03,43,108,19,06,25,230,23,07,01,184,27,1*69
$GPGSV,5,2,20,14,82,229,15,17,55,317,28,21,17,049,29,22,63,309,28,1*6C
$GPGSV,5,3,20,30,18,210,23,08,00,092,,19,32,291,,24,01,322,,1*6C
$GPGSV,5,4,20,33,,,34,38,,,34,40,,,34,42,,,34,1*6E
$GPGSV,5,5,20,46,,,34,48,,,34,49,,,34,50,,,34,1*60
$GPGGA,004112.00,3639.952966,N,11705.959730,E,1,02,9.6,105.9,M,-4.0,M,,*78
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004112.00,A,3639.952966,N,11705.959730,E,0.0,0.0,210624,5.4,W,A,V*52
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
█
```

2.6. IO

EG828 内置 IO 接口，支持 DI，DO 和 AI 接口，默认出厂 2DO，4DI，4AI 配置，其中每个 AI 均支持电压和电流的检测。IO 接口的控制是通过串口实现的，串口标识 ttyS8，串口初始化实现可参考 2.2 章节。默认串口参数如下:

参数	默认值（不可修改）
波特率	921600
校验位	NONE
数据位	8
停止位	1

IO 接口通过串口控制时，使用的是标准的 Modbus RTU 协议，Modbus 指令码如下表：

指令码	说明	操作	操作数量
01 H	读线圈状态	位操作	单个或多个
02 H	读离散输入状态（只能读到 0 和 1）	位操作	单个或多个
03 H	读保持寄存器	字操作	单个或多个
04 H	读输入寄存器	字操作	单个或多个
05 H	写单个线圈	位操作	单个
06 H	写单个保持寄存器	字操作	单个
0F H	写多个线圈	位操作	多个
10 H	写多个保持寄存器	字操作	多个

IO 接口从机地址为 0x01，寄存器列表如下：

IO 接口	寄存器类型	寄存器地址（16 进制）	功能码	数据类型	说明
DI	1x	0000 ~ 0003	02	Bit	
DO	0x	0000 ~ 0001	01/05/0F	Bit	ON：0xFF00 OFF：0x0000
AI	3x	0000 ~ 0006	04	32 Bit Unsigned（AB CD）	
AI	3x	0010 ~ 0016	04	32 Bit Unsigned	

				（AB CD）	

测试 demo 如下，主要实现 DO 接口 1s 翻转功能。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <stdbool.h>

#define SAVLE_ADDR 0X01
#define DO1_ADDR 0X0000
#define DO2_ADDR 0X0001
#define DO3_ADDR 0X0002
#define DO4_ADDR 0X0003

// CRC16
unsigned short crc16(const unsigned char* data, unsigned short length) {
    unsigned short crc = 0xFFFF;
    for (int pos = 0; pos < length; pos++) {
        crc ^= (unsigned short)data[pos];         // XOR byte into least sig. byte of crc

        for (int i = 8; i != 0; i--) { // Loop over each bit
            if ((crc & 0x0001) != 0) { // If the LSB is set
                crc >>= 1; // Shift right and XOR 0xA001
                crc ^= 0xA001;
            } else           // Else LSB is not set
                crc >>= 1;    // Just shift right
        }
    }
    // Note, this CRC calculation is reversed endian to some implementations
    return crc;
}

// IO uart init --ttyS8
int UART_INIT_ttyS8(void)
{
    int serial_port =0;
    serial_port = open("/dev/ttyS8", O_RDWR);
    if (serial_port < 0) {
        printf("Error %i from open: %s\n", errno, strerror(errno));
        return 0;
    }
```

```c

    // Create new termios struct, we call it 'tty' for convention
    struct termios tty;

    // Read in existing settings, and handle any error
    if(tcgetattr(serial_port, &tty) != 0) {
        printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
        return 0;
    }

    tty.c_cflag &= ~PARENB; // Clear parity bit, disabling parity (most common)
    tty.c_cflag &= ~CSTOPB; // Clear stop field, only one stop bit used in communication (most common)
    tty.c_cflag &= ~CSIZE; // Clear all bits that set the data size
    tty.c_cflag |= CS8; // 8 bits per byte (most common)
    tty.c_cflag &= ~CRTSCTS; // Disable RTS/CTS hardware flow control (most common)
    tty.c_cflag |= CREAD | CLOCAL; // Turn on READ & ignore ctrl lines (CLOCAL = 1)

    tty.c_lflag &= ~ICANON;
    tty.c_lflag &= ~ECHO; // Disable echo
    tty.c_lflag &= ~ECHOE; // Disable erasure
    tty.c_lflag &= ~ECHONL; // Disable new-line echo
    tty.c_lflag &= ~ISIG; // Disable interpretation of INTR, QUIT and SUSP
    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // Turn off s/w flow ctrl
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL); // Disable any special handling of received bytes

    tty.c_oflag &= ~OPOST; // Prevent special interpretation of output bytes (e.g. newline chars)
    tty.c_oflag &= ~ONLCR; // Prevent conversion of newline to carriage return/line feed
    tty.c_cc[VTIME] = 10;     // Wait for up to 1s (10 deciseconds), returning as soon as any data is received.
    tty.c_cc[VMIN] = 0;

    // Set in/out baud rate to be 9600
    cfsetispeed(&tty, B921600);
    cfsetospeed(&tty, B921600);

    // Save tty settings, also checking for error
    if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
        return 0;
    }
    return serial_port;
}

```

```c
int main()
{
    int serial_ttyS8 = UART_INIT_ttyS8();//uart init
    int res = 0;
    unsigned char IO_sta = 0;

    while (1)
    {
        // Write to serial port
        unsigned char msg[8] = {0x00};
        msg[0] = SAVLE_ADDR;
        msg[1] = 0x0f;//cmd id
        msg[2] = (DO1_ADDR >> 8) & 0xff;
        msg[3] = DO1_ADDR & 0xff;
        msg[4] = 0x00;
        msg[5] = 0x02;
        IO_sta = ~IO_sta;
        printf("IO_sta is %d\n", IO_sta);

        if(IO_sta == 0)
        {
            //DO OPEN
            msg[6] = 0x01;
            msg[7] = 0x00;
        }
        else
        {
            //DO CLOSE
            msg[6] = 0x01;
            msg[7] = 0x03;
        }
        // crc16
        unsigned short crc = crc16(msg, sizeof(msg));
        unsigned char fullMessage[10];
        for (int i = 0; i < sizeof(msg); ++i)
        {
             fullMessage[i] = msg[i];
        }
        fullMessage[8] = crc & 0xFF; // CRC high
        fullMessage[9] = (crc >> 8) & 0xFF; // CRC  low
        write(serial_ttyS8, fullMessage, sizeof(fullMessage));
        sleep(1); // Delay for 1 second
    }

    // Close the serial port
    close(serial_ttyS8);
    return 0;
}
```

3. Ubuntu 系统

3.1. Ubuntu 版本升级

当前产品配置为 Ubuntu 20.04 版本，如果想要使用最新版本的 Ubuntu,可以通过命令进行版本升级。常用命令如下：(非 root 权限， 需要增加 sudo 命令)

命令	**功能**
apt-get update	更新源
apt-get upgrade	更新已安装软件包
apt-get dist-upgrade	处理依赖关系
apt-get install update-manager-core	安装升级工具
do-release-upgrade	升级
lsb_release -a	版本查询