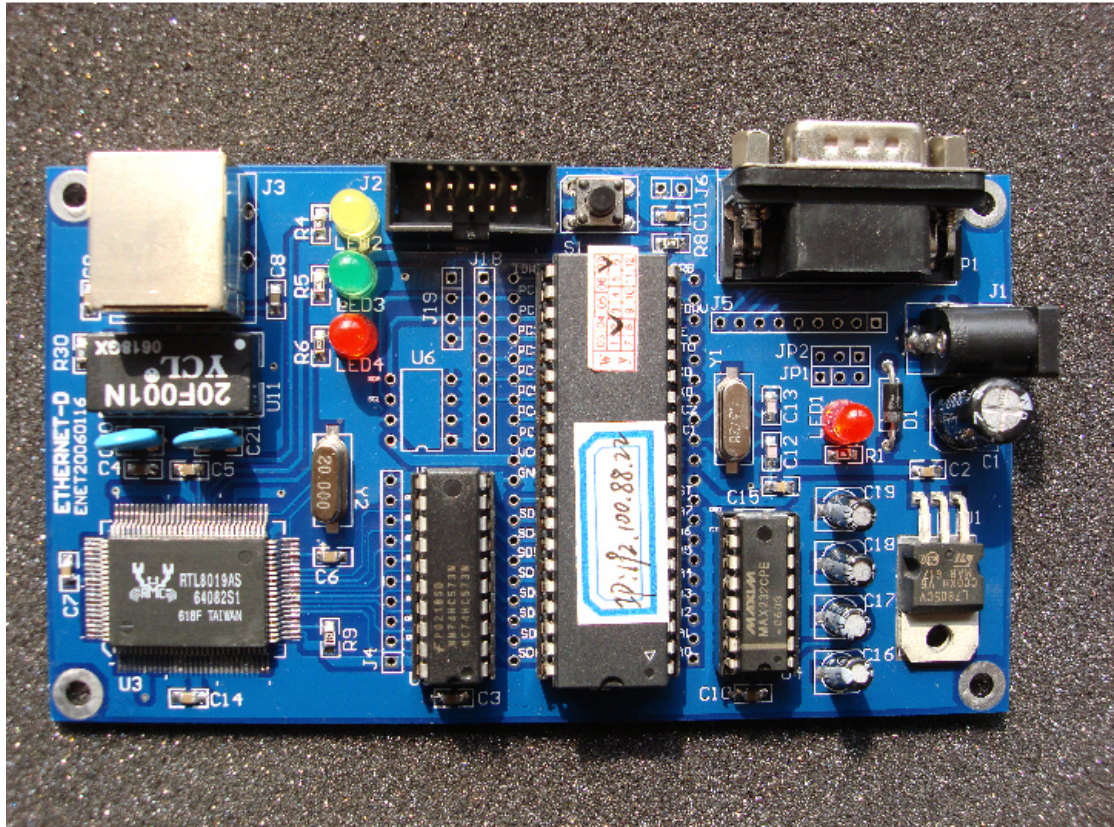


# ETHERNET-D 以太网开发板范例入门手册



作者: MIRROROK (李刚)

版本: v1.0

时间: 2007-2-12

QQ: 4641452

MAIL: lee\_gang123@hotmail.com

ETHERNET-D 开发板是一款基于 AVR 控制器 MEGA32L 和 REALTEK 公司的网络芯片 RTL8019AS 而设计的一款入门级 10M 以太网开发板.本开发板主要资源如下：

		flash	ram	eeeprom
Avr	Mega32l	32KB	2KB	1KB
net	Rtl8019as		16KB	

## 一、硬件介绍：

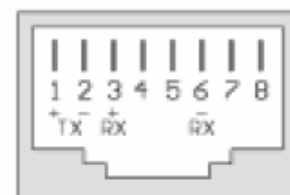


### 1.1.电源输入

本开发板采用 9V 直流供电，使用时外部需要配置一个 9v 直流电源使用！

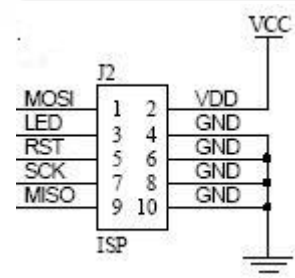
### 1.2.10M 以太网接口：

采用标准的 rj45 接口，采用非屏蔽网络线与网络连接



### 1.3.ISP 接口：

试验板上是下图，原理图如右图,LED 在开发板上没有使用



### 1.4.串口：

采用 DB9 公头 PCB 座子，

TXD PIN2

RXD PIN3

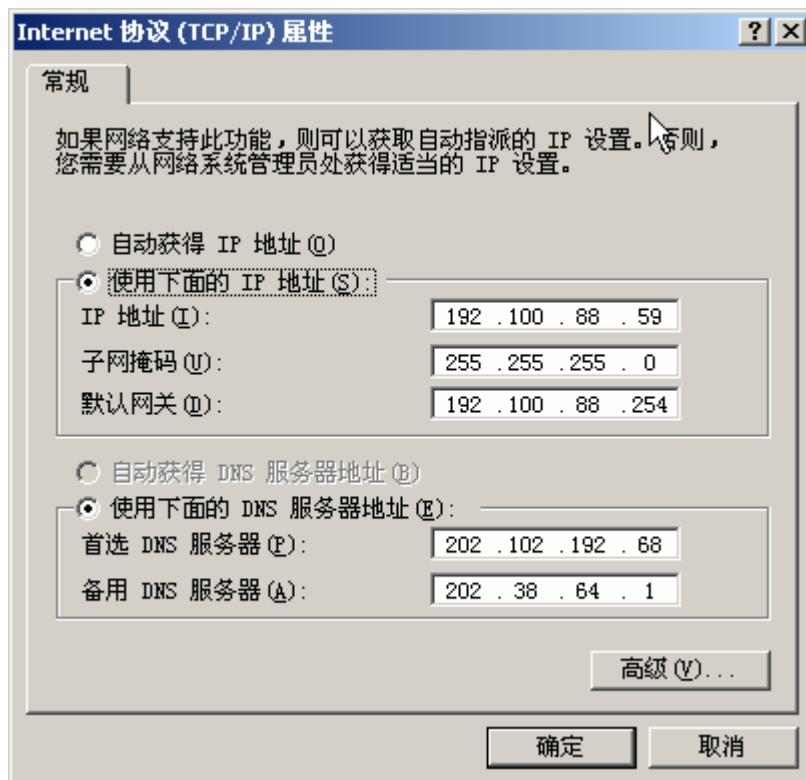
GND PIN5

## 二、开发板测试：

开发板已经预装了 RS232TOTCP 程序，板子 IP 地址为 192.100.88.22，默认网关 192.100.88.254，子网掩码 255.255.255.0，远程连接地址 192.100.88.21，端口 1234 方式 TCP。

**问题：如何修改ip 地址、本地网关、子网掩码**

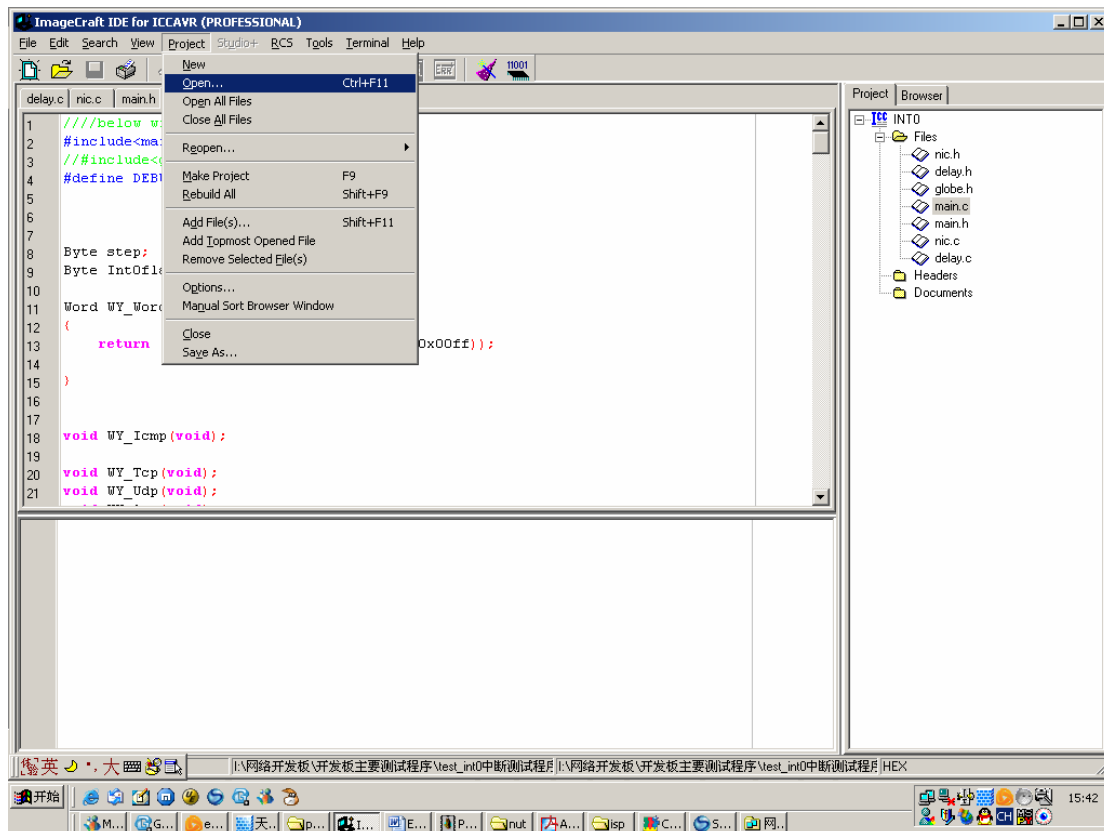
类似的 window 里面 察看本地连接属性。参考下图如果你进行修改。



程序中的修改：

以 RS232TOTCP 为例，

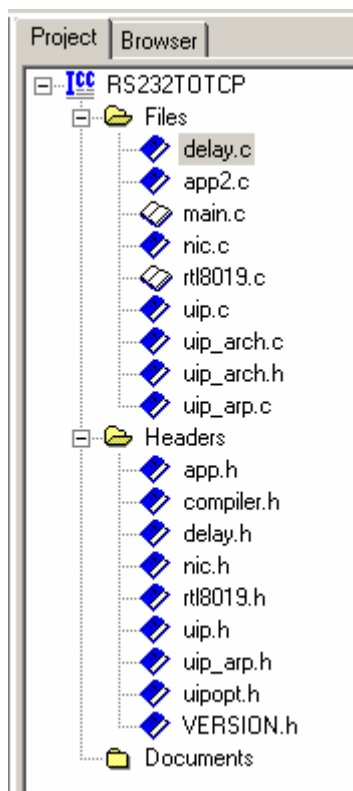
我们启动 iccavr，



选取 project -> open



选取 RS232TOTCP.PRJ，此时项目文件调入 如下图：



选取 uiptopt.h 双击打开文件，  
找到如下代码

```
#define UIP_IPADDR0 192
/**< The first octet of the IP address of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_IPADDR1 100
/**< The second octet of the IP address of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
```

```

#define UIP_IPADDR2    88

/**< The third octet of the IP address of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_IPADDR3    22

/**< The fourth octet of the IP address of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */

#define UIP_NETMASK0    255
/**< The first octet of the netmask of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_NETMASK1    255
/**< The second octet of the netmask of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_NETMASK2    255
/**< The third octet of the netmask of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_NETMASK3    0

/**< The fourth octet of the netmask of
      this uIP node, if UIP_FIXEDADDR is
      1. \hideinitializer */

#define UIP_DRIPADDR0  192

/**< The first octet of the IP address of
      the default router, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_DRIPADDR1  100
/
/**< The second octet of the IP address of
      the default router, if UIP_FIXEDADDR is
      1. \hideinitializer */
#define UIP_DRIPADDR2  88

/**< The third octet of the IP address of
      the default router, if UIP_FIXEDADDR is
      1. \hideinitializer */

```

```
#define UIP_DRIPADDR3 254
```

①如何修改 ip 地址:

```
#define UIP_IPADDR0    192    定义了最高字节
#define UIP_IPADDR1    100    定义了次高字节
#define UIP_IPADDR2     88    定义了次低字节
#define UIP_IPADDR3     22    定义了最低字节
```

如果你要修改为 192.168.0.25  
则修改如下:

```
#define UIP_IPADDR0    192    定义了最高字节
#define UIP_IPADDR1    168    定义了次高字节
#define UIP_IPADDR2     0     定义了次低字节
#define UIP_IPADDR3    25     定义了最低字节
```

②如何修改子网掩码:

```
#define UIP_NETMASK0    255    定义了最高字节
#define UIP_NETMASK1    255    定义了次高字节
#define UIP_NETMASK2    255    定义了次低字节
#define UIP_NETMASK3     0     定义了最低字节
```

一般不必修改, 如果需要请参考修改 ip 地址进行修改

③如何修改网关:

```
#define UIP_DRIPADDR0  192    定义了最高字节
#define UIP_DRIPADDR1  100    定义了次高字节
#define UIP_DRIPADDR2   88    定义了次低字节
#define UIP_DRIPADDR3  254    定义了最低字节
```

如果你要修改为 192.168.0.1  
则修改如下:

```
#define UIP_DRIPADDR0  192    定义了最高字节
#define UIP_DRIPADDR1  168    定义了次高字节
#define UIP_DRIPADDR2   0     定义了次低字节
#define UIP_DRIPADDR3   1     定义了最低字节
```

④如何修改连接的目标地址

打开 app2.c

找到函数 example1\_init ()

此函数包含, 如下语句:

```
uiplib_ipaddr(ipaddr, 192, 100, 88, 59);
```

```
//设置要连接的目标 ip 地址
uiplib_connect(ipaddr, HTONS(1234));
//连接的目标 ip 地址的端口 1234

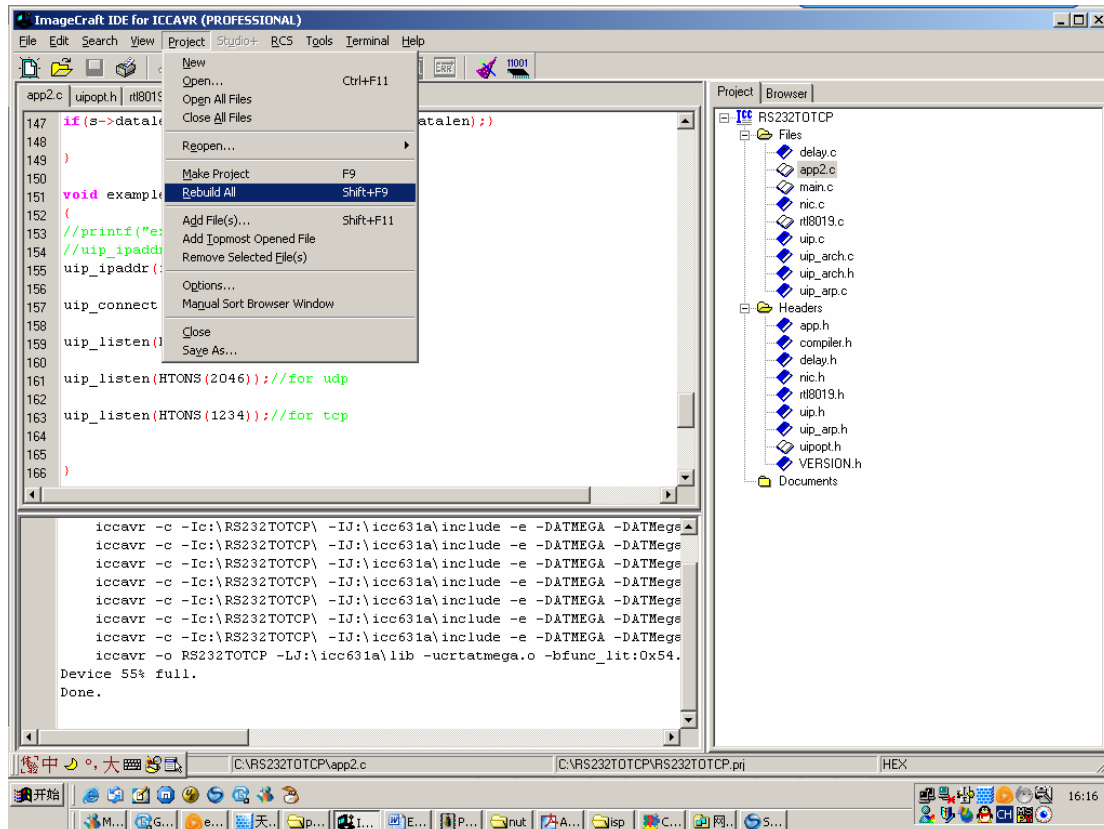
uiplib_listen(HTONS(80));
//侦听端口 80 http 协议适用的
uiplib_listen(HTONS(2046)); //for udp
//侦听端口 2046 udp 应用使用

uiplib_listen(HTONS(1234)); //for tcp
//侦听端口 1234 tcp 应用的端口
```

如果你需要与目标地址 192.168.0.55 （你的 pc 的 ip 地址），端口采用 9090，则修改如下：

```
uiplib_ipaddr(ipaddr, 192, 168, 0, 55);
//设置要连接的目标 ip 地址
uiplib_connect(ipaddr, HTONS(9090));
//连接的目标 ip 地址的端口 9090
****下面 2 句可以不用修改****
uiplib_listen(HTONS(80));
//侦听端口 80 http 协议适用的
uiplib_listen(HTONS(2046)); //for udp
//侦听端口 2046 udp 应用使用
//修改侦听的端口
uiplib_listen(HTONS(9090)); //for tcp
//侦听端口 9090 tcp 应用的端口
这样就修改好了，然后选取 Project-Rerebuild all
```





Iccavr 的下部输出窗口就出现 下面图片

```
J:\icc631a\bin\imakev -f RS232TOTCP.mak
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -c -Ic:\RS232TOTCP\ -IJ:\icc631a\include -e -DATMEGA -DATMega32 -l -Wf-intenum -Mavr_enhanced
iccavr -o RS232TOTCP -LJ:\icc631a\lib -ucrtatmega.o -bfunc_lit:0x54.0x8000 -dram_end:0x85f -bdata:0x6C
Device 55% full.
Done.
```

如果有错误, 请 修改程序, 没有的话, 就可以进行下一步了, 程序下载!!

程序下载需要 isp 下载器 开发板目前没有提供, 可以自己做一个, 图片参考 <http://www.lancos.com/e2p/betterSTK200-mini.gif>, 成本很低的, 效果还不错!!

Ponyprog 的具体流程请参考我的教程

[WORD版本的教程 1-4](#); (word版本)

[以太网教程第 5 集](#) (pdf 版本)

程序下载后, 就可以开始软件的测试了!!!

## 2.1.arp/icmp 测试

Arp 协议和 icmp 协议是一起测试的, 打开 window 启动菜单 [运行](#), 输入你设置的 ip 地址, 如 ping 192.168.0.22 -t, 下面范例演示 在 ip 地址为 192.100.88.59 的计算机上 ping 192.100.88.22 -t





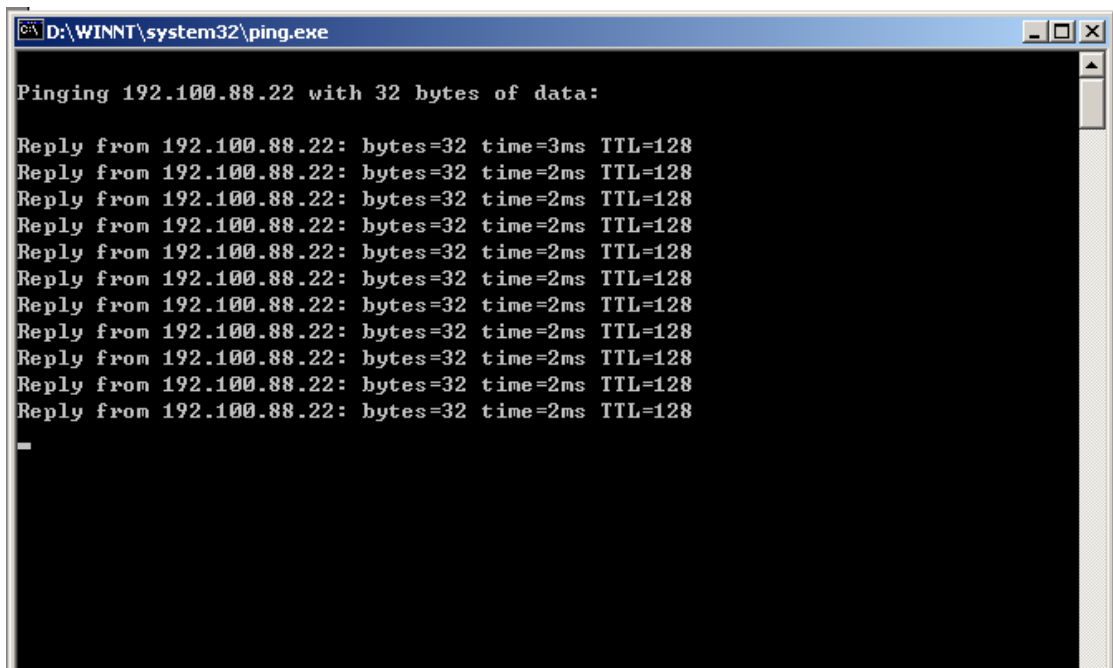
出现下图，并且板子上 led3 绿灯和 led2 黄灯闪烁则说明协议成功！

其中：

Led4 红灯表示网络数据报有冲突

led3 绿灯表示网络数据报接收

led2 黄灯表示网络数据报发送



## Icmp-echoRequest 数据包 iris 网络抓包

The screenshot displays the IRIS v4.0 network capture interface. The left sidebar contains navigation buttons: Capture, Decode, Guard, Filters, and Logs. The main window is titled 'Capture' and is divided into three panes. The left pane shows the 'Packet Decoder' tree, which is expanded to show the details of a selected packet. The middle pane is a table of captured packets, and the right pane shows the raw packet data in hexadecimal and ASCII.

**Packet Decoder Tree:**

- Frame (74 bytes)
  - MAC header (Ethernet II)
    - Destination Address: 6D:69:72:72:6F:6B unknown
    - Source Address: 00:05:5D:E3:99:F9 D-Link System
    - Type: 08-00 DoD IP
  - IPv4 header
    - Version: 4
    - Header length: 5 (20 bytes)
    - Type of service: 0
      - ...0... = Minimize Delay: Normal delay
      - ...0... = Maximize Throughput: Normal throughput
      - ...0... = Maximize Reliability: Normal reliability
      - ...0... = Minimize Monetary Cost: Normal monetary cost
    - Total length: 60 bytes (Correct)
    - Identification: 38234
    - Flags
      - Fragment Offset: 0
      - Time to Live: 128 hops
      - Protocol: 1 ICMP
      - Checksum: 0x744C (Correct)
      - Source IP Address: 192.100.88.59
      - Destination IP Address: 192.100.88.22
      - IP Options = None
  - ICMP header (Echo request)
    - Type: 8 (Echo request)
    - Code: 0
    - Checksum: 0x4359
    - Identifier: 1024
    - Sequence Number: 1539
    - Data: 32 bytes

**Packet List Table:**

No.	Time (h:m:s:ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr.
1.	16:57:18:093	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	ICMP->Echo_Request	ligang
1.	16:57:18:109	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	ICMP->Echo_Reply	192.100.88.22
1.	16:57:18:125	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->15000	ligang
1.	16:57:19:109	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	ICMP->Echo_Request	ligang
1.	16:57:19:109	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	ICMP->Echo_Reply	192.100.88.22
1.	16:57:19:437	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	TCP->MSN Messenger	207.46.100.100

**Raw Packet Data (Hex/ASCII):**

```
0000 6D 69 72 72 6F 6B 00 05 5D E3 99 F9 08 00 45 00 mirrok...].....
0010 00 3C 95 5A 00 00 80 01 74 4C C0 64 58 3B C0 64 .<.2....tL.dX;
0020 58 16 08 00 43 59 04 00 06 03 61 62 63 64 65 66 X...CY....abcd
0030 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 ghijklmnopqrst
0040 77 61 62 63 64 65 66 67 68 69 wabcdefghijklmnop
```

**Status Bar:** No filter | CPU: 0% | 190/2000 | IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9 | D-Link DFE-530TX PCI Fast Ethernet Adapter

## Icmp-echoReply 数据包 iris 网络抓包

IRIS v4.0

File View Capture Decode Filters Tools Help

Iris

Capture

Decode

Guard

Filters

Logs

Statistics

Links

Help

Did you know... You can discover all hosts from your subnet (including the network adapter vendor) by using the Autodiscovery option from Address Book.

its:D:\Program Files\Iris\hints.chm::ht No filter CPU: 0% 190/2000 IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9 D-Link DFE-530TX PCI Fast Ethernet Adapter

**Capture**

Packet Decoder

**Frame (74 bytes)**

**MAC header (Ethernet II)**

- Destination Address: 00:05:5D:E3:99:F9 D-Link S
- Source Address: 6D:69:72:72:6F:6B unknown
- Type: 08-00 DoD IP

**IPv4 header**

- Version: 4
- Header length: 5 (20 bytes)
- Type of service: 0
  - 0000 00 05 5D E3 99 F9 6D 69 72 72 6F 6B 08 00 45 00 ...]...mirrok..
  - 0010 00 3C 95 5A 00 00 80 01 74 4C C0 64 58 16 C0 64 ...<.Z....tL.dX.
  - 0020 58 3B 00 00 4B 59 04 00 06 03 61 62 63 64 65 66 X;..KY....abcd
  - 0030 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 ghijklmnopqrst
  - 0040 77 61 62 63 64 65 66 67 68 69 wabcedefghi
- Total length: 60 bytes (Correct)
- Identification: 38234
- Flags
  - Fragment Offset: 0
  - Time to Live: 128 hops
  - Protocol: 1 ICMP
  - Checksum: 0x744C (Correct)
  - Source IP Address: 192.100.88.22
  - Destination IP Address: 192.100.88.59
  - IP Options = None
- ICMP header (Echo reply)**
  - Type: 0 (Echo reply)
  - Code: 0
  - Checksum: 0x4B59
  - Identifier: 1024
  - Sequence Number: 1539
  - Data: 32 bytes

No.	Time (h:m:s:ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr.
1.	16:57:18:093	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	ICMP->Echo_Request	ligang
1.	16:57:18:109	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	ICMP->Echo_Reply	192.100.88.59
1.	16:57:18:125	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->15000	ligang
1.	16:57:19:109	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	ICMP->Echo_Request	ligang
1.	16:57:19:109	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	ICMP->Echo_Reply	192.100.88.59
1.	16:57:19:437	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	TCP->MSN Messenger	207.46.100.100

Arp 测试 在系统菜单下运行 cmd 输入 arp -d 在 isir 中可以抓包到 这个是 pc 发送的 arp 包

The screenshot displays the IRIS v4.0 interface. The left sidebar contains icons for Capture, Decode, Guard, Filters, and Logs. The main window is titled 'Capture' and shows a 'Packet Decoder' view of a selected packet. The packet details are as follows:

- Frame (42 bytes)**
  - MAC header (Ethernet II)**
    - Destination Address: FF:FF:FF:FF:FF:FF Br
    - Source Address: 00:05:5D:E3:99:F9 D-Link
    - Type: 08-06 ARP
  - Address Resolution Protocol (Request)**
    - Hardware Type: 1 Ethernet [10Mb]
    - Protocol Type: 08-00 DoD IP
    - Length of Hardware Address: 6
    - Length of Protocol Address: 4
    - Operation Code: 1 (Request)
    - Sender's Hardware Address: 00:05:5D:E3:99:F9
    - Sender's IP Address: 192.100.88.59
    - Target's Hardware Address: 00:00:00:00:00:00
    - Target's IP Address: 192.100.88.22
    - Frame Padding: 0 bytes

The packet list on the right shows the following entries:

No.	Time (h:m:s:ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src
1	17:3:7:734	00:E0:18:FF:69:5F	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	TLL
2	17:3:13:109	00:05:5D:E3:99:F9	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	ligang
3	17:3:13:109	6D:69:72:72:6F:68	00:05:5D:E3:99:F9	ARP	ARP->Reply	192.100.88.22
4	17:3:13:453	00:05:5D:E3:99:F9	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	ligang
5	17:3:13:453	00:10:F3:04:72:18	00:05:5D:E3:99:F9	ARP	ARP->Reply	192.100.88.25
6	17:3:18:609	00:10:F3:04:72:18	00:05:5D:E3:99:F9	ARP	ARP->Request	192.100.88.25
7	17:3:18:609	00:05:5D:E3:99:F9	00:10:F3:04:72:18	ARP	ARP->Reply	ligang

At the bottom, a hex dump shows the raw packet data:

```
0000 FF FF FF FF FF FF 00 05 5D E3 99 F9 08 06 00 01 .....].....
0010 08 00 06 04 00 01 00 05 5D E3 99 F9 C0 64 58 3B .....]....dX;
0020 00 00 00 00 00 00 C0 64 58 16 .....dX.
```

The status bar at the bottom indicates: Filter: Untitled, CPU: 0%, 7/2000, IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9, D-Link DFE-530TX PCI Fast Ethernet Adapter.

Arp 测试 这个是开发板返回的 arp 包

The screenshot displays the IRIS v4.0 network analysis interface. The main window is titled 'Capture' and shows a list of captured packets. The selected packet (No. 3) is an ARP->Reply from 192.100.88.22 to 192.100.88.25. The packet details pane on the left shows the MAC header (Ethernet II) and the Address Resolution Protocol (Reply) section. The packet details pane on the right shows the raw packet data in hexadecimal and ASCII.

**Packet List:**

No.	Time (h:m:s:ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src
1	17:3:7:734	00:E0:18:FF:69:5F	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	TLL
2	17:3:13:109	00:05:5D:E3:99:F9	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	ligang
3	17:3:13:109	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	ARP	ARP->Reply	192.100.88.22
4	17:3:13:453	00:05:5D:E3:99:F9	FF:FF:FF:FF:FF:FF	ARP	ARP->Request	ligang
5	17:3:13:453	00:10:F3:04:72:18	00:05:5D:E3:99:F9	ARP	ARP->Reply	192.100.88.25
6	17:3:18:609	00:10:F3:04:72:18	00:05:5D:E3:99:F9	ARP	ARP->Request	192.100.88.25
7	17:3:18:609	00:05:5D:E3:99:F9	00:10:F3:04:72:18	ARP	ARP->Reply	ligang

**Packet Details (Frame 60 bytes):**

- MAC header (Ethernet II)
  - Destination Address: 00:05:5D:E3:99:F9 D-
  - Source Address: 6D:69:72:72:6F:6B unknow
  - Type: 08-06 ARP
- Address Resolution Protocol (Reply)
  - Hardware Type: 1 Ethernet [10Mb]
  - Protocol Type: 08-00 DoD IP
  - Length of Hardware Address: 6
  - Length of Protocol Address: 4
  - Operation Code: 2 (Reply)
  - Sender's Hardware Address: 6D:69:72:72:6F:6B
  - Sender's IP Address: 192.100.88.22
  - Target's Hardware Address: 00:05:5D:E3:99:F9
  - Target's IP Address: 192.100.88.59
  - Frame Padding: 18 bytes

**Raw Packet Data:**

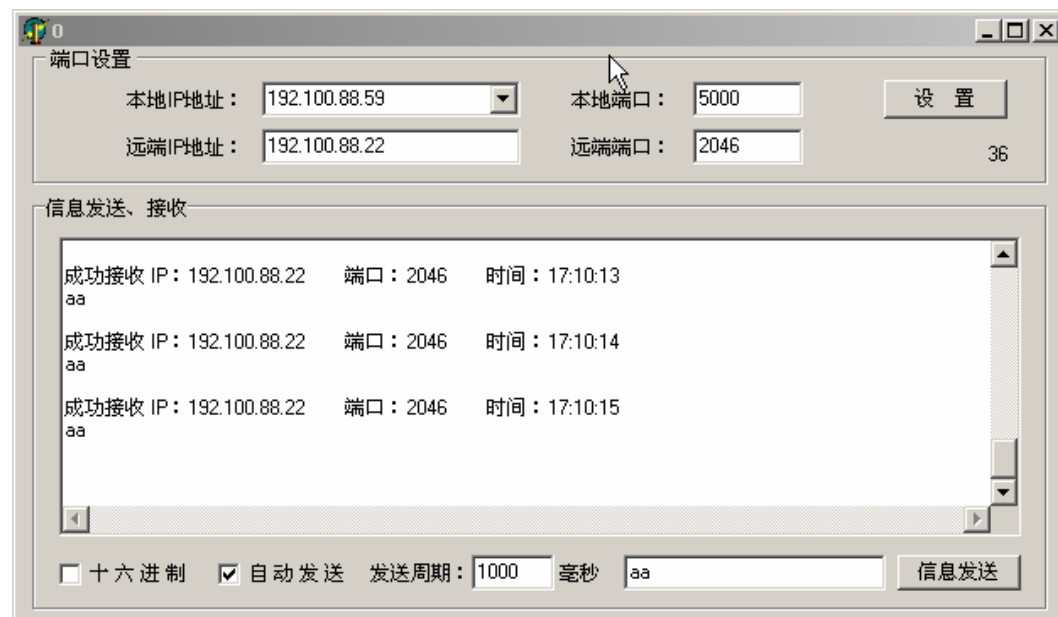
```
0000 00 05 5D E3 99 F9 6D 69 72 72 6F 6B 08 06 00 01 ...]...mirrok...
0010 08 00 06 04 00 02 6D 69 72 72 6F 6B C0 64 58 16 .....mirrok.dX.
0020 00 05 5D E3 99 F9 C0 64 58 3B 61 62 63 64 65 66 ..]....dX;abcdef
0030 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72                ghijklmnopqr
```

这样 arp 协议也测试通过了!!!!

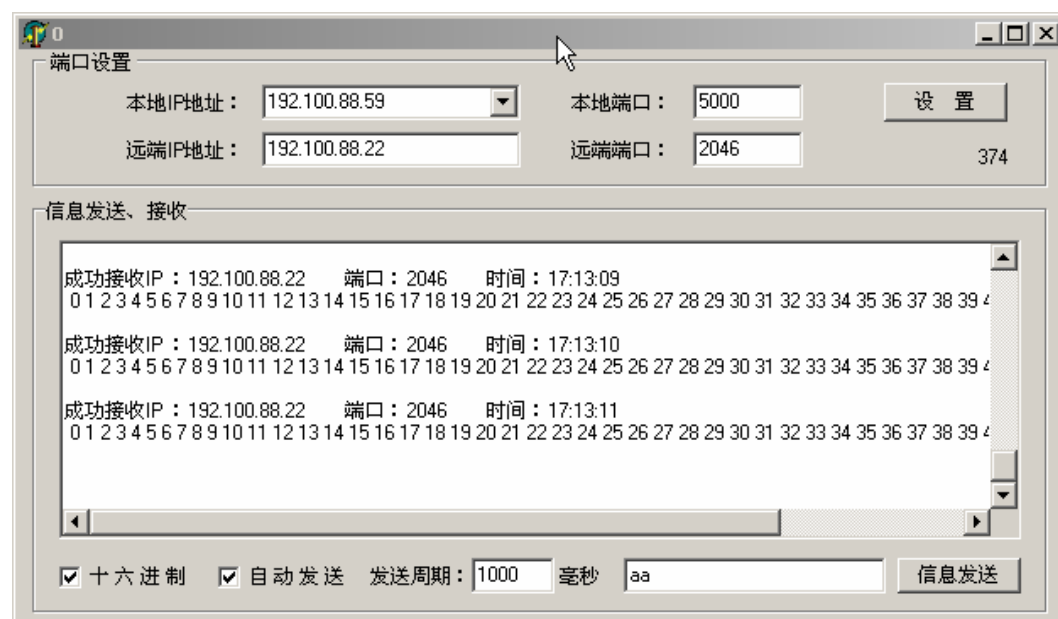
## 2.2.Udp 协议测试

打开软件 udptest.Exe 按照下图设置 远程端口必须是 2046，远程 ip 地址是你的板子地址，参看前面的设置！

选取自动发送 aa 则返回 aa

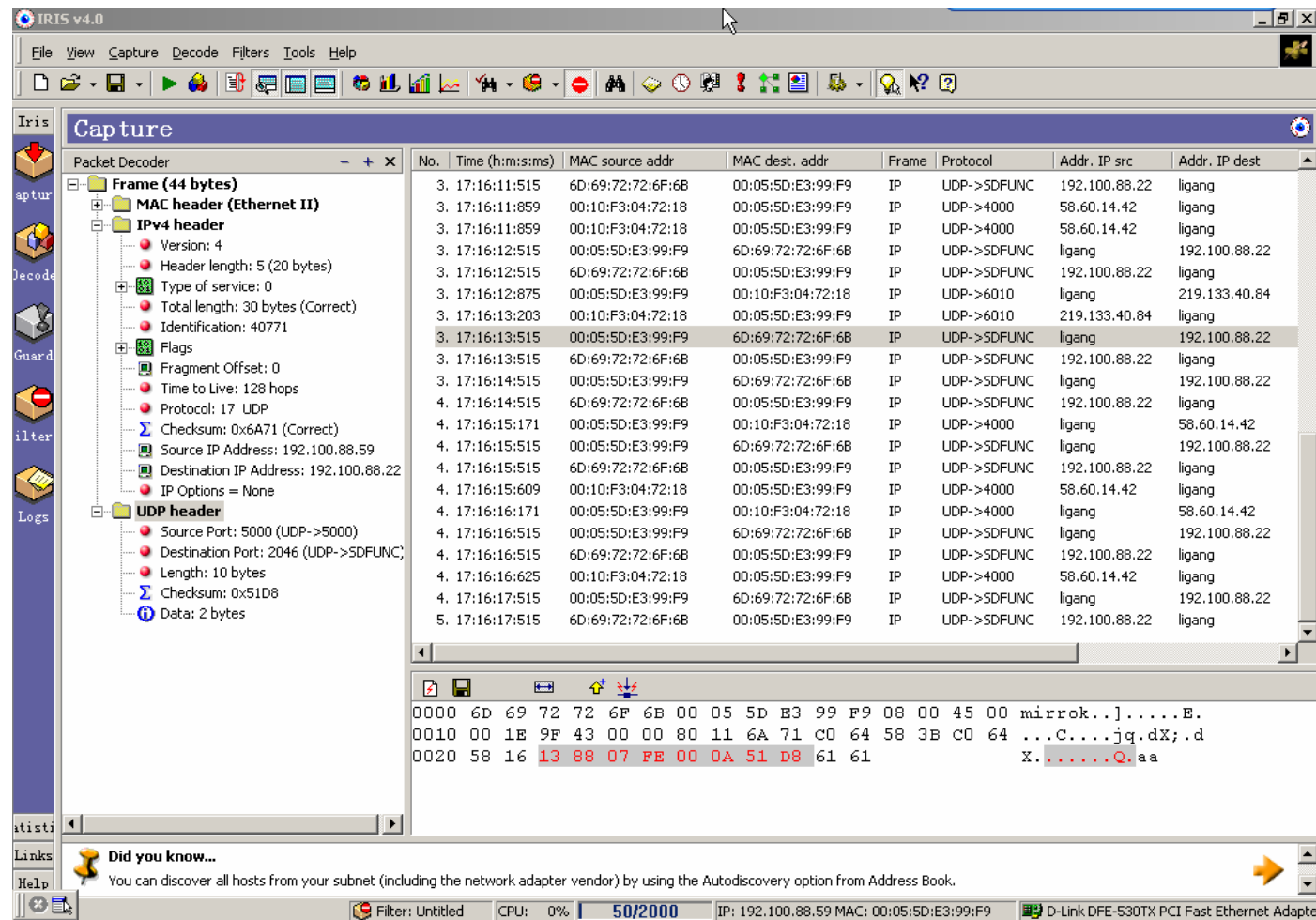


选取自动发送 0xaa 则返回 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49



上述则说明 udp 协议测试成功！！

这个是 pc 发送的 udp 数据包 发送 aa 时 数据包的最后是 aa 0x61 0x61



IRIS v4.0

File View Capture Decode Filters Tools Help

Packet Decoder

Frame (44 bytes)

- MAC header (Ethernet II)
  - Version: 4
  - Header length: 5 (20 bytes)
  - Type of service: 0
  - Total length: 30 bytes (Correct)
  - Identification: 40771
  - Flags
    - Fragment Offset: 0
    - Time to Live: 128 hops
    - Protocol: 17 UDP
    - Checksum: 0x6A71 (Correct)
    - Source IP Address: 192.100.88.59
    - Destination IP Address: 192.100.88.22
    - IP Options = None
- UDP header
  - Source Port: 5000 (UDP->5000)
  - Destination Port: 2046 (UDP->SDFUNC)
  - Length: 10 bytes
  - Checksum: 0x51D8
  - Data: 2 bytes

No.	Time (h:m:s.ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src	Addr. IP dest
3.	17:16:11:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:11:859	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
3.	17:16:11:859	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
3.	17:16:12:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3.	17:16:12:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:12:875	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->6010	ligang	219.133.40.84
3.	17:16:13:203	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->6010	219.133.40.84	ligang
3.	17:16:13:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3.	17:16:13:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:14:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:14:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:15:171	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->4000	ligang	58.60.14.42
4.	17:16:15:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:15:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:15:609	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
4.	17:16:16:171	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->4000	ligang	58.60.14.42
4.	17:16:16:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:16:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:16:625	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
4.	17:16:17:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
5.	17:16:17:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang

0000 6D 69 72 72 6F 6B 00 05 5D E3 99 F9 08 00 45 00 mirrok.].....E.  
0010 00 1E 9F 43 00 00 80 11 6A 71 C0 64 58 3B C0 64 ...C....jq.dX;.d  
0020 58 16 13 88 07 FE 00 0A 51 D8 61 61 X.....Q.aa

Did you know...  
You can discover all hosts from your subnet (including the network adapter vendor) by using the Autodiscovery option from Address Book.

Filter: Untitled CPU: 0% 50/2000 IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9 D-Link DFE-530TX PCI Fast Ethernet Adapter



这个是开发板发送给 pc 的 udp 数据包 数据包的最后是 aa 0x61 0x61 不足 60 字节后面增加到 60 字节

IRIS v4.0

File View Capture Decode Filters Tools Help

Capture

Packet Decoder

Frame (60 bytes)

- MAC header (Ethernet II)
  - Version: 4
  - Header length: 5 (20 bytes)
  - Type of service: 0
  - Total length: 30 bytes (Correct)
  - Identification: 40771
  - Flags
    - Fragment Offset: 0
    - Time to Live: 128 hops
    - Protocol: 17 UDP
    - Checksum: 0x6A71 (Correct)
    - Source IP Address: 192.100.88.22
    - Destination IP Address: 192.100.88.59
    - IP Options = None
- UDP header
  - Source Port: 2046 (UDP->SDFUNC)
  - Destination Port: 5000 (UDP->5000)
  - Length: 10 bytes
  - Checksum: 0x51D8
  - Data: 2 bytes

No.	Time (h:m:s.ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src	Addr. IP dest
3.	17:16:11:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:11:859	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
3.	17:16:11:859	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
3.	17:16:12:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3.	17:16:12:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:12:875	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->6010	ligang	219.133.40.84
3.	17:16:13:203	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->6010	219.133.40.84	ligang
3.	17:16:13:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3.	17:16:13:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
3.	17:16:14:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:14:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:15:171	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->4000	ligang	58.60.14.42
4.	17:16:15:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:15:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:15:609	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
4.	17:16:16:171	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->4000	ligang	58.60.14.42
4.	17:16:16:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
4.	17:16:16:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4.	17:16:16:625	00:10:F3:04:72:18	00:05:5D:E3:99:F9	IP	UDP->4000	58.60.14.42	ligang
4.	17:16:17:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
5.	17:16:17:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang

0000 00 05 5D E3 99 F9 6D 69 72 72 6F 6B 08 00 45 00 ..]...mirrok..E.  
0010 00 1E 9F 43 00 00 80 11 6A 71 C0 64 58 16 C0 64 ...C....jq.dX..d  
0020 58 3B 07 FE 13 88 00 0A 51 D8 61 61 63 64 65 66 X;.....Q.aacdef  
0030 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 ghijklmnopqr

The content of the packet editor window will be decoded in this window. You can also click on an item to see its position in the packet editor window.

Filter: Untitled CPU: 0% 50/2000 IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9 D-Link DFE-530TX PCI Fast Ethernet Adapte

这个是 pc 发送的 udp 数据包 发送 aa 时 数据包的最后是 0xaa

IRIS v4.0

File View Capture Decode Filters Tools Help

IRIS

Capture

Packet Decoder

Frame (43 bytes)

- MAC header (Ethernet II)
  - Destination Address: 6D:69:72:72:6F:6B
  - Source Address: 00:05:5D:E3:99:F9
  - Type: 08-00 DoD IP
- IPv4 header
  - Version: 4
  - Header length: 5 (20 bytes)
  - Type of service: 0
  - Total length: 29 bytes (Correct)
  - Identification: 41552
  - Flags
    - Fragment Offset: 0
    - Time to Live: 128 hops
    - Protocol: 17 UDP
    - Checksum: 0x6765 (Correct)
    - Source IP Address: 192.100.88.59
    - Destination IP Address: 192.100.88.22
    - IP Options = None
- UDP header
  - Source Port: 5000 (UDP->5000)
  - Destination Port: 2046 (UDP->SDFUNC)
  - Length: 9 bytes
  - Checksum: 0x93B
  - Data: 1 bytes

No.	Time (h:m:s.ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src	Addr. IP dest
1	17:20:45:203	00:0C:6E:05:BB:D9	FF:FF:FF:FF:FF:FF	IP	UDP->NETBI...	192.100.88.80	192.100.88.255
2	17:20:45:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3	17:20:45:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4	17:20:46:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
5	17:20:46:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
6	17:20:47:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
7	17:20:47:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
8	17:20:47:859	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->15000	ligang	60.191.55.29
9	17:20:48:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
1.	17:20:48:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang

0000 6D 69 72 72 6F 6B 00 05 5D E3 99 F9 08 00 45 00 mirrok...E.  
0010 00 1D A2 50 00 00 80 11 67 65 C0 64 58 3B C0 64 ...P....ge.dX;.d  
0020 58 16 13 88 07 FE 00 09 09 3B AA X.....;

The content of the packet editor window will be decoded in this window. You can also click on an item to see its position in the packet editor window.

Filter: Untitled CPU: 0% 10/2000 IP: 192.100.88.59 MAC: 00:05:5D:E3:99:F9 D-Link DFE-530TX PCI Fast Ethernet Adapte

这个是开发板发送给 pc 的 udp 数据包 数据包的是 0x00 -----0x31 即 00—49

The screenshot displays the IRIS v4.0 network analysis interface. The main window is titled 'Capture' and shows a list of captured packets. The selected packet (No. 3) is expanded in the 'Packet Decoder' pane on the left, showing the following details:

- Frame (92 bytes)**
  - MAC header (Ethernet II)**
    - Destination Address: 00:05:5D:E3:99:F9
    - Source Address: 6D:69:72:72:6F:6B
    - Type: 08-00 DoD IP
  - IPv4 header**
    - Version: 4
    - Header length: 5 (20 bytes)
    - Type of service: 0
    - Total length: 78 bytes (Correct)
    - Identification: 41552
    - Flags
    - Fragment Offset: 0
    - Time to Live: 128 hops
    - Protocol: 17 UDP
    - Checksum: 0x6734 (Correct)
    - Source IP Address: 192.100.88.22
    - Destination IP Address: 192.100.88.59
    - IP Options = None
  - UDP header**
    - Source Port: 2046 (UDP->SDFUNC)
    - Destination Port: 5000 (UDP->5000)
    - Length: 58 bytes
    - Checksum: 0x5866
    - Data: 50 bytes

The packet list on the right shows the following data:

No.	Time (h:m:s:ms)	MAC source addr	MAC dest. addr	Frame	Protocol	Addr. IP src	Addr. IP dest
1	17:20:45:203	00:0C:6E:05:BB:D9	FF:FF:FF:FF:FF:FF	IP	UDP->NETBI...	192.100.88.80	192.100.88.255
2	17:20:45:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
3	17:20:45:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
4	17:20:46:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
5	17:20:46:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
6	17:20:47:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
7	17:20:47:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang
8	17:20:47:859	00:05:5D:E3:99:F9	00:10:F3:04:72:18	IP	UDP->15000	ligang	60.191.55.29
9	17:20:48:515	00:05:5D:E3:99:F9	6D:69:72:72:6F:6B	IP	UDP->SDFUNC	ligang	192.100.88.22
1.	17:20:48:515	6D:69:72:72:6F:6B	00:05:5D:E3:99:F9	IP	UDP->SDFUNC	192.100.88.22	ligang

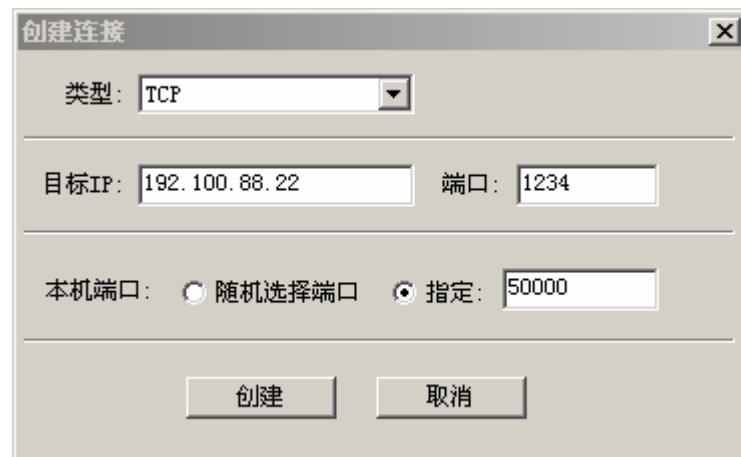
The packet editor window at the bottom shows the raw data of the selected packet (No. 3) in hexadecimal and ASCII format:

```
0000 00 05 5D E3 99 F9 6D 69 72 72 6F 6B 08 00 45 00 ..]...mirrok..E.
0010 00 4E A2 50 00 00 80 11 67 34 C0 64 58 16 C0 64 .N.P....g4.dX..d
0020 58 3B 07 FE 13 88 00 3A 58 66 00 01 02 03 04 05 X;.....Xf.....
0030 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 .....
0040 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 .....! "$ %
0050 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 &* () *+,-./01
```

The status bar at the bottom indicates the current filter is 'Untitled', CPU usage is 0%, and the packet count is 10/2000. The selected packet is No. 3, with source IP 192.100.88.59 and destination IP 192.100.88.22.

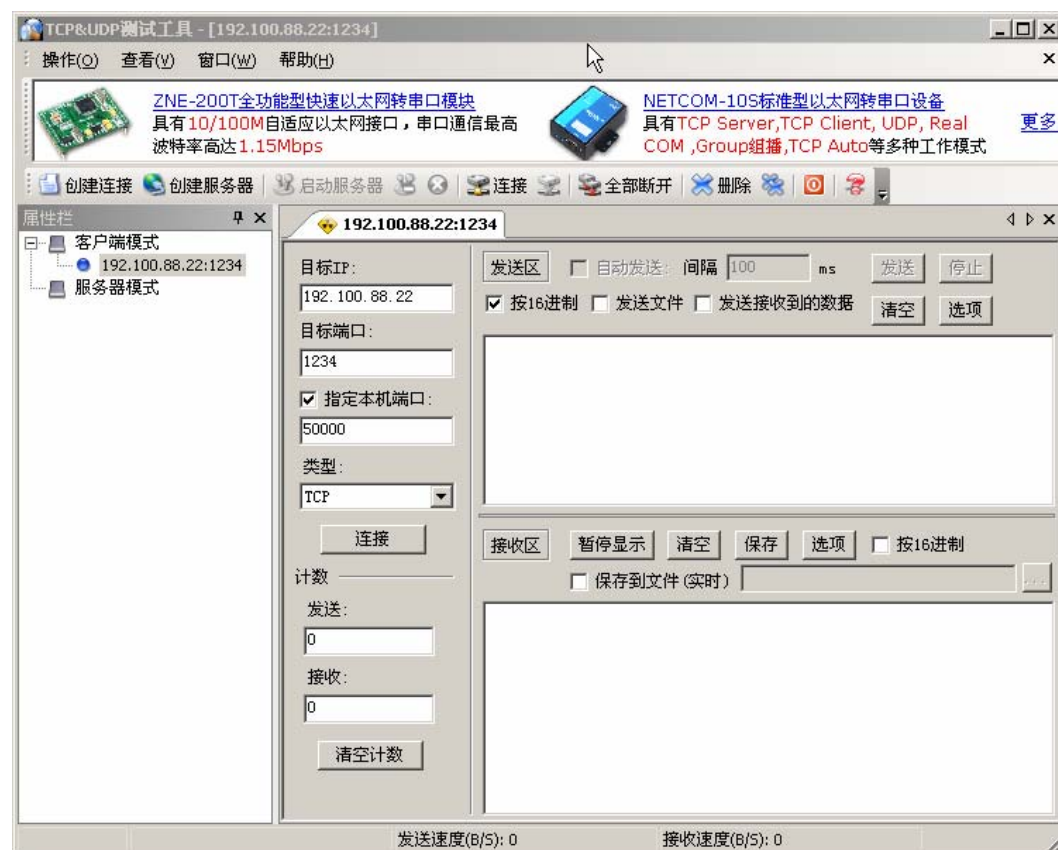
### 2.3.Tcp 测试

选取--**创建连接**

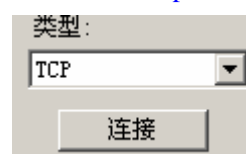


输入目标地址 192.100.88.22 端口 1234 本机指定端口 50000

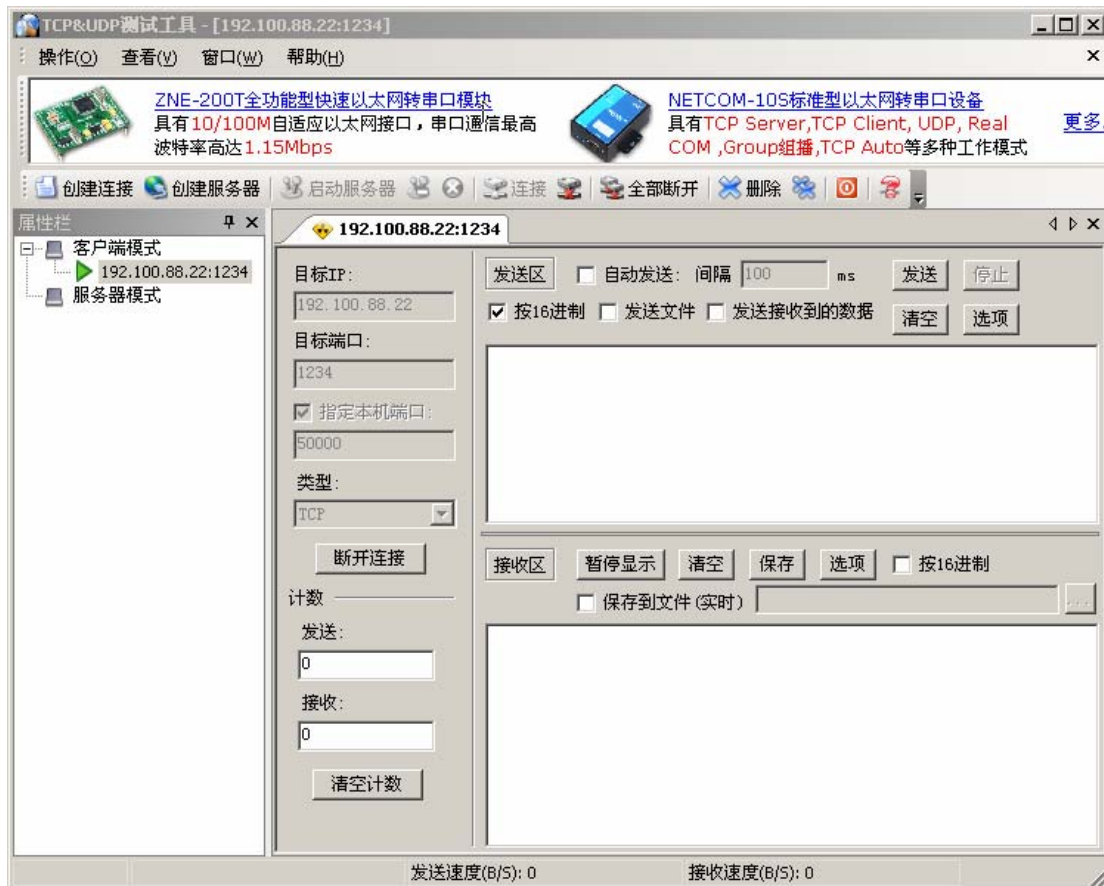
选择——> 创建，出现下图



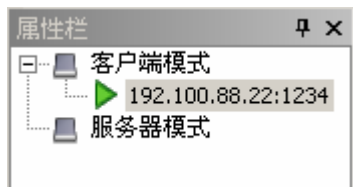
类型选取 tcp



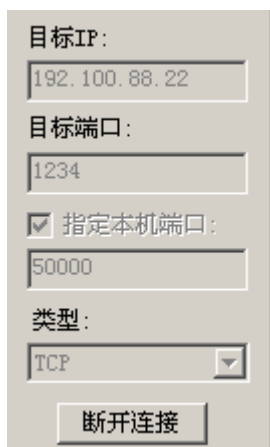
然后选取 ----> 连接



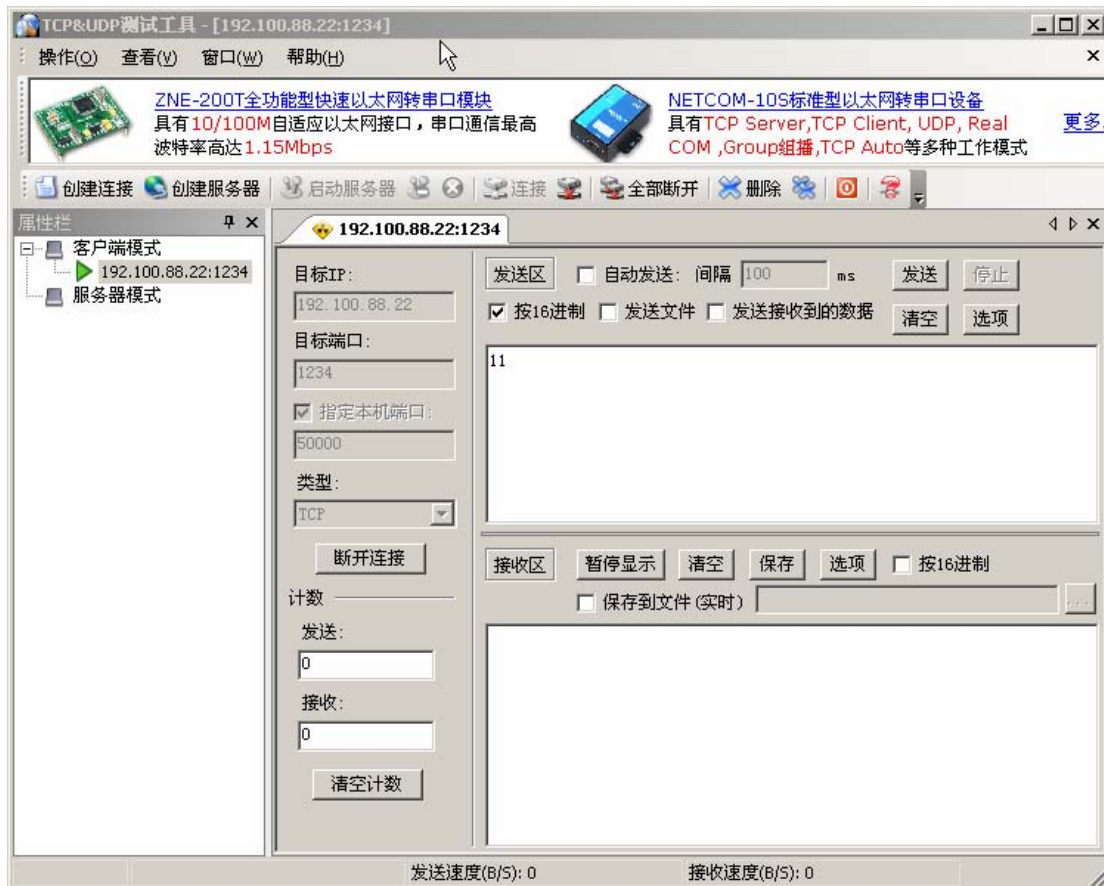
此时左边状态栏出现下图：绿色箭头表示连接成功了！！



连接后目标地址和目标端口 和本机端口 变灰，表示不能修改



此时就可以进行 tcp 协议的测试了!!! ^-^

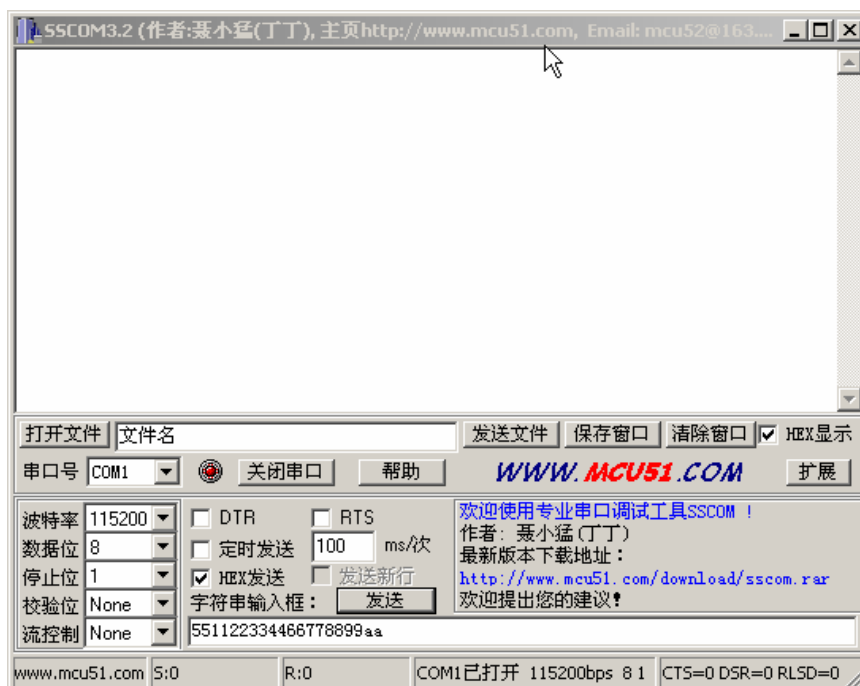


在发送缓冲区 输入 11 22 选取自动发送 如上图

此时打开串口调试助手 波特率设置为 115200，连接开发板的串口和 pc 开发板

PC

<b>TXD (PIN2)</b>	-----	<b>RXD (PIN 3)</b>
<b>RXD (PIN3)</b>	-----	<b>TXD (PIN 2)</b>
<b>GND (PIN5)</b>	-----	<b>GND (PIN5)</b>



采用 hex (16 进制发送), 定时为 100ms 发送一次, 此处**需要在 pc 上运行的 TCP&UDP 测试程序的自动发送时间需要比 串口的发送时间小, 也就是说 轮训的时间比串口的发送时间快!!**

发送一段时间后, 串口和 TCP&UDP 软件有如下界面:





由上图可以看出，在串口软件中发送 1150 个字节，在 TCP&UDP 调试软件中也接收了 1150 字节，由此可以看出 TCP 协议由建立到数据发送都测试通过！！

上述 **tcp 的测试是以 pc 为服务器轮训客户端的数据的范例！**

如果用户需要做串口和 tcp 的转换器需要在提供的代码基础上，进行修改，以达到高速的要求，提高接收速率，缩短处理时间，这样才能做出串口服务器！

## 三、软件入门：

在本开发板配套光盘中我们已经提供如下的测试代码，供使用者学习 uip 和以太网的开发，使使用者能从最简单的测试开始 到实现 uip 的基础测试！！

已经提供的代码如下：

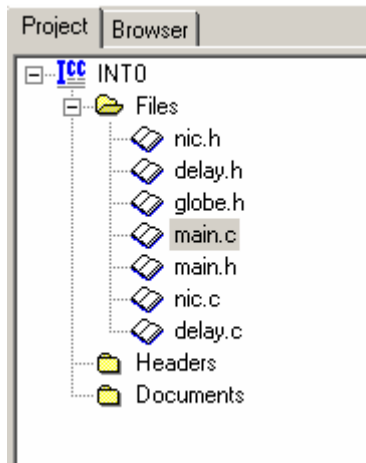
1. arp 和 icmp-ok 测试程序
2. RS232TOTCP
3. test\_int01getpacket
4. test\_int01getpacket 加溢出处理
5. test\_int0 加溢出处理的中断函数
6. test\_int0 中断测试程序
7. uIP-AVR-0900001
8. uip 修改 udpok

我们让使用者了解以太网开发的基础流程，测试的思路是这样的：

- |                      |  |
|----------------------|--|
| 1. 首先，测试网络芯片产生中断；    | 对应程序 test_int0 中断测试程序                                |
| 2. 测试网络芯片产生中断并加溢出中断； | 对应程序 test_int0 加溢出处理的中断函数                            |
| 3. 测试以太网收包，          | 对应程序 test_int01getpacket 和 test_int01getpacket 加溢出处理 |
| 4. 测试简单的以太网处理        | 对应程序 arp 和 icmp-ok 测试程序                              |
| 5. 测试 uip 协议栈，       | 对应程序 uip 修改 udpok 和 uIP-AVR-0900001                  |
| 6. 测试 uip 的应用        | 对应程序 RS232TOTCP                                      |

### 3.1. test\_int0 中断测试程序

在 iccavr 中打开目录下面的项目文件，在程序的右侧列出了项目包含的文件 具体如下图：



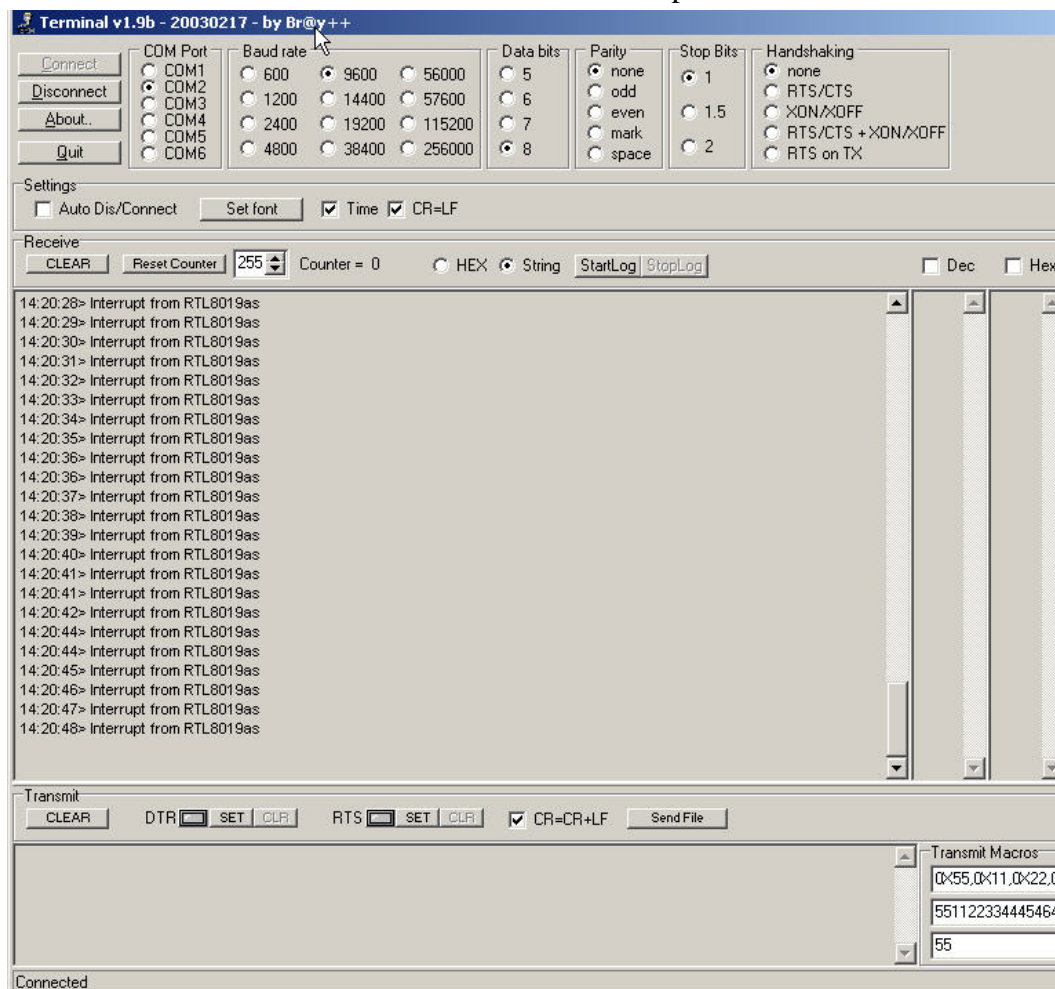
Nic.h/.c RTL8019AS 网络芯片的驱动函数定义包含读写和初始化函数等

Delay.h/.c 延时函数的定义

Globe.h 全局变量的函数

Main.h/.c 主函数定义 包含 RTL8019AS 的中断函数

此项目中没有设置开发板的 ip 地址，是测试 rtl8019as 芯片产生中断的情况，具体的输出如下图：（图片在项目目录下面的 pic 目录下可以找到）

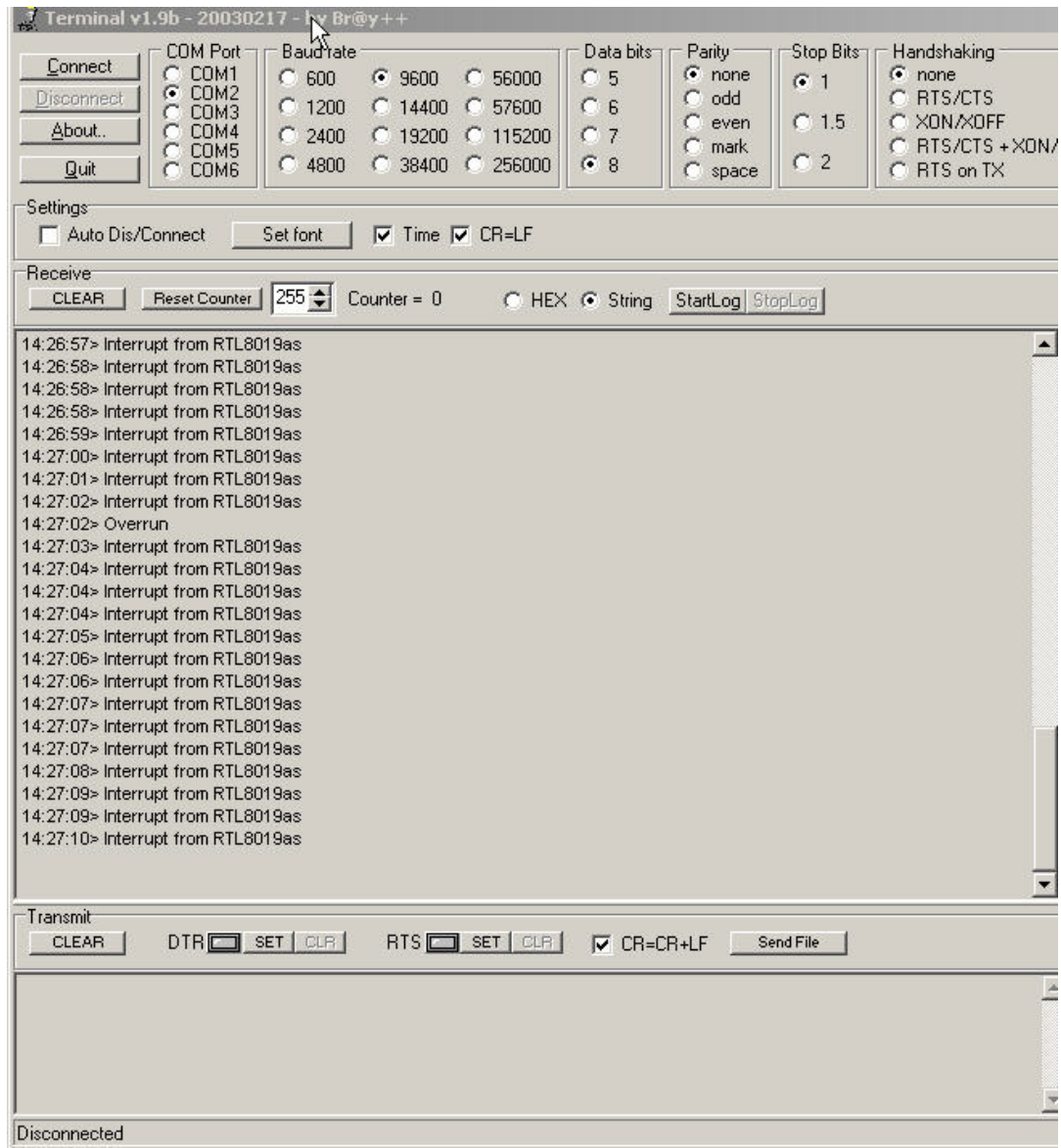


此项目主要检测 RTL8019AS 的中断产生，在产生中断后，MEGA32L 就会发送一句 Interrupt from RTL8019as，在接收到一定的中断后就不再产生中断，这个是

由于 RTL8019AS 已经溢出造成的，这个就需要随后的测试！

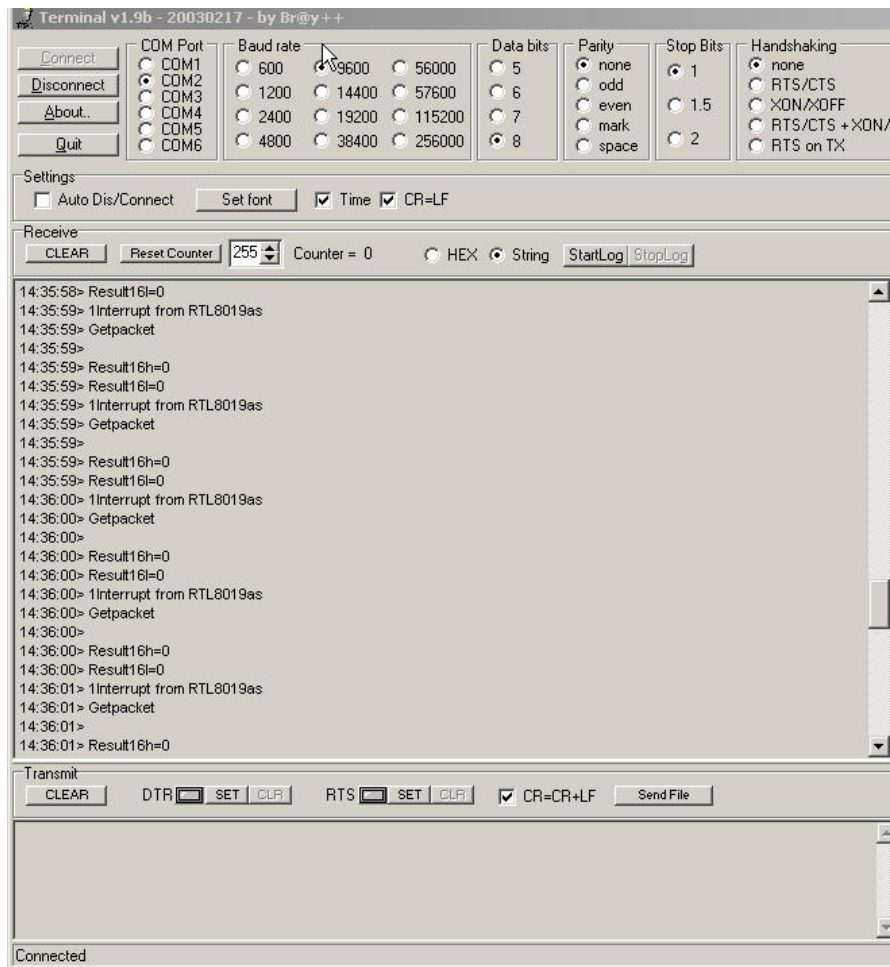
### 3.2. test\_int0 加溢出处理的中断函数

这个项目是增加了溢出处理，RTL8019AS 可以连续产生中断，当 RTL8019AS 溢出时会打印出 OVERRUN，图片显示如下：（图片在项目目录下面的 pic 目录下可以找到）

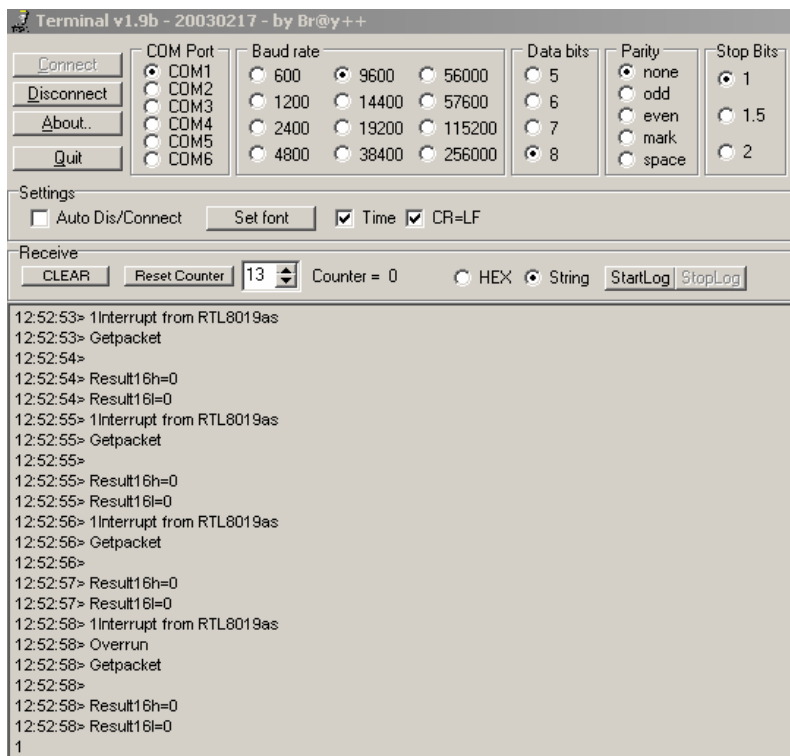


### 3.3. test\_int01getpacket

下面我们开始测试以太网的收包，主函数中通过中断函数读取 RTL8019AS 的中断，然后再调用收包 函数！测试的时候同时 ping 192.100.88.25 -t （ip 地址是任意的）



接到一段那个数量的数据包后，就会溢出出现 OVERRUN 后就会停止收包!!



### 3.4. test\_int01getpacket 加溢出处理

在本项目文件中增加了溢出处理，可以连续的收包!! 具体处理见 **OVERRUN** 函数!!

### 3.5. arp 和 icmp-ok 测试程序

这个项目是检测收到的 IP 数据包是哪种类型的

主要判断在 `mypackproc.c` 中 `void WY_Getpacket(void)` 函数判断,

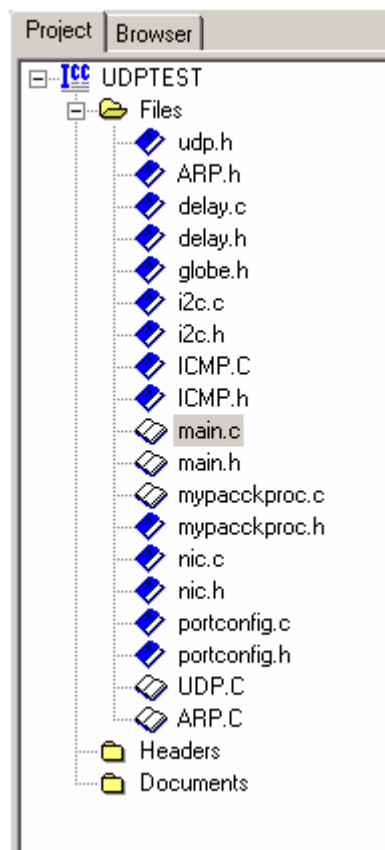
```
switch (uip_buf[0x17])
{
    case 0x01 :
        WY_Icmp(); //icmp 数据包
        break;
    case 0x06:
        WY_Tcp(); //tcp 数据包
        break;
    case 0x11:
        WY_Udp(); //udp 数据包
        break;
    default:
        break;
}
```

在程序中开发板的地址设置成 192.168.0.198，实现简单的 icmp 和 arp 协议，tcp 和 udp 没有处理，ip 的修改在 `mypacckproc.c` 中，需要修改的话修改红色部分为你需要的 ip

```
u8_t ArpTable[] = {0x08, 0x06, 0x00, 0x01, 0x06, 0x04, 0x01, 192, 168, 0, 198};
```

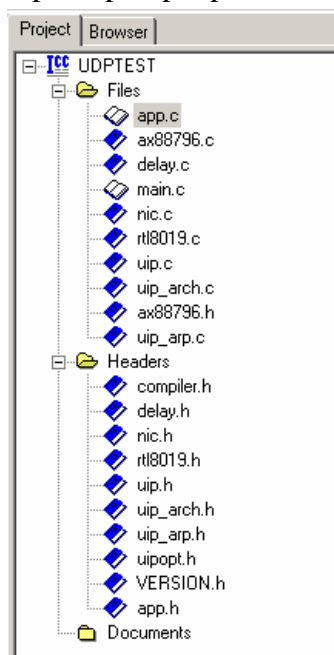
[illegible]

项目中的主要文件如下



### 3.6. uip 修改 udpok

此项目测试 uip 的 arp/icmp/udp/tcp 这个是 uip 的第一个程序, tcp 的应用在 app.c 中实现了, udp 的应用程序在 uip.c 实现了一个简单的范例, 其余的协议 arp/icmp/udp/tcp 主要在 uip.c 中, 主要程序列表如下



其中 ax88796.h/.c 可以从项目中删除, 下面看一下每个 c 文件的函数:



## Uip.c

Project	Browser
[-] uip.c	
[-] Variables	
[-] uip_log(194)	size:22 address:0x0A08
[-] uip_init(201)	size:166 address:0x0A13
[-] uip_connect(229)	size:390 address:0x0A66
[-] uip_udp_new(297)	size:300 address:0x0B29
[-] uip_unlisten(338)	size:84 address:0x0BBF
[-] uip_listen(349)	size:82 address:0x0BE9
[-] uip_add_rcv_nxt(497)	size:86 address:0x0C12
[-] uip_process(508)	size:6262 address:0x0C3D
[-] htons(1739)	size:20 address:0x1878
[-] uip_udpchksum(1744)	size:234 address:0x1882

## App.c

[-] app.c	
[-] Variables	
[-] aborted(42)	size:2 address:0x016B
[-] timedout(43)	size:2 address:0x016C
[-] closed(44)	size:2 address:0x016D
[-] connected(47)	size:26 address:0x016E
[-] newdata(56)	size:48 address:0x017B
[-] acked(72)	size:146 address:0x0193
[-] example1_init(101)	size:48 address:0x01DC
[-] example1_app(113)	size:188 address:0x01F4

## Delay.c

[-] delay.c	
[-] delay_ms(5)	size:46 address:0x067A

## Main.c

[-] main.c	
[-] Variables	
[-] initTimer(133)	size:18 address:0x0691
[-] SIG_OVERFLOW0(149)	size:28 address:0x069A
[-] USART_init(180)	size:98 address:0x06A8
[-] RS485COM(211)	size:30 address:0x06D9
[-] main(305)	size:288 address:0x06E8

## Nic.c 网络层函数

[-] nic.c	
[-] nic_init(25)	size:6 address:0x0778
[-] nic_send(31)	size:104 address:0x077B
[-] nic_poll(56)	size:64 address:0x07AF

## Rtl8019.c RTL8109AS 驱动函数



[-] rt8019.c
+ Variables
+ rt8019Write(78) size:28 address:0x07CF
+ rt8019Read(134) size:28 address:0x07DD
+ rt8019SetupPorts(167) size:20 address:0x07EB
+ rt8019BeginPacketSend(309) size:110 address:0x07F5
+ rt8019SendPacketData(344) size:44 address:0x082C
+ rt8019EndPacketSend(354) size:14 address:0x0842
+ rt8019BeginPacketRetreive(376) size:240 address:0x0849
+ rt8019RetreivePacketData(446) size:176 address:0x08C1
+ rt8019EndPacketRetreive(473) size:52 address:0x0919
+ rt8019Ovrrun(489) size:138 address:0x0933
+ rt8019Init(525) size:266 address:0x0978
+ rt8019ProcessInterrupt(604) size:22 address:0x09FD

Uip\_arch.c ip 的 checksum 生成函数

[-] uip_arch.c
+ uip_add32(45) size:164 address:0x18F7
+ uip_chksum(72) size:120 address:0x1949
+ uip_ipchksum(98) size:12 address:0x1985
+ uip_tcpchksum(113) size:270 address:0x198B

Uip\_arp.c arp 协议处理函数

[-] uip_arp.c
+ Variables
+ uip_arp_init(130) size:66 address:0x1A12
+ uip_arp_timer(147) size:124 address:0x1A33
+ uip_arp_update(163) size:386 address:0x1A71
+ uip_arp_ipin(238) size:154 address:0x1B32
+ uip_arp_arpin(290) size:390 address:0x1B7F
+ uip_arp_out(365) address:0x1C42

这个项目中uipopt.h中设置了本地的ip、地址网关和子网掩码如果需要修改参考[PAGE4](#)进行修改

默认的 ip 地址是 192.168.0.100

默认的网关是 192.168.0.1

默认的子网掩码是 255.255.255.0

默认的连接主机地址 192.168.0.199

`uip_ipaddr(ipaddr, 192,168,0,199);`

默认的连接主机 端口是 4001

`uip_connect(ipaddr, HTONS(4001));`

`uip_listen(HTONS(4001)); // TCP 连接端口`

默认的 udp 连接端口是 2046

`uip_listen(HTONS(2046)); // UDP 连接端口`

上述的代码在 `void example1_init(void)` 可以找到!!!

### 3.7. uIP-AVR-0900001

也是一个非常简单的 uip 范例 具体在 app.c 中

```

/*****
*****

```

```

* "A Very Simple Application" from the uIP 0.9 documentation
*****
*****/

```

```

#include "app.h"

```

```

void example1_init(void)
{
    uip_listen(HTONS(1234));
}

void example1_app(void)
{
    if(uip_newdata() || uip_rexmit())
    {
        uip_send("ok\n", 3);
    }
}

```

测试方法[参考 3.6](#)

### 3.8. RS232TOTCP

这是一个把串口数据通过开发板发送到远程 ip 对应端口的范例，  
在主函数中 串口接收 以 0x55 开始 和以 0xaa 结束的数据包，然后在结束完成后置位 RxOK  
标志,在 main()函数中判断这个标志并进行处理!!

```

if(RxOK==1)RS485process();

```

在函数 `static void newdata(void)` 中判断

```

if(UartRxCount!=0)
{
    s->dataptr=&TxBuf[0];
    s->datalen=UartRxCount;
}

```

然后发送再置位相应的标志!!!

默认的 ip 地址是 192.168.0.100

默认的子网掩码是 255.255.255.0

默认的网关是 192.168.0.1

需要修改请参考 PAGE4

默认连接主机 ip 地址 192.168.0.199

默认 tcp 端口 1234

默认 udp 端口 2046

`void example1_init(void)` 可以修改连接端口内容

具体测试参考 page4!