NETWORKING

WITH THE

PROTON PLUS COMPILER

AND

PROTON-NET

Crownhill Associates

smart electronic solutions

Crownhill Associates

smart electronic solutions

Proton-Net Packing List

Item	Qty	Description	Packed By
1	1	AC-AC adaptor Input: AC230V 50Hz 9W Output: AC9V 500mA 4.5VA	
2	1	CAT5 Straight Through Cable, 1 Mtr	
3	1	CAT5 Cross Over Cable, 1Mtr	
4	1	Straight Through Serial Cable	
5	1	3.5” Floppy Disk containing: source code, examples and manual	
6	1	Printed Manual	

Note:

Longer CAT 5 cables are available from Crownhill, Visit www.crownhill.co.uk or contact sales@crownhill.co.uk to order your custom length CAT 5 cables.

If you wish to manufacture your own cables, follow the details at the end of the manual. Cable, Connectors and Assembly tools are available from sales@crownhill.co.uk

Crownhill Associates
smart electronic solutions

PICBASIC PLUS networking

1. Introduction

This manual describes the Iosoft Ltd. PB_UDP source-code package, which provides networking functionality for the Crownhill Associates Proton-Net board, using Crownhill's ***Proton Plus*** PIC BASIC programming language.

Due to the limitations of the BASIC programming language, a full TCP/IP software stack (e.g. Embedded Web server) is not included; instead, a low-level interface employing standard UDP (User Datagram Protocol) communications is provided, that allows for simple communications over a local network or the Internet.

To demonstrate the networking capability, the source-code package includes client & server code for communication with a local PC running the Iosoft 'datagram' utility.

This manual was written for version 1 of the software. You may have been supplied with an updated version, for details see the revision header at the top of the main source file. Although the new version will have improvements and corrections, the underlying principles described in this manual will still apply.

It is impossible for us to provide a complete description of TCP/IP networking within the confines of this manual; newcomers are recommended to read an introductory book on the subject prior to tackling this challenging area, for example 'TCP/IP Lean: Web Servers for Embedded Systems' by Jeremy Bentham (2nd edition ISBN 1-57820-108-X).

For sales and support information on Iosoft products, refer to the Iosoft Ltd. Web site, **www.iosoft.co.uk**
For the Crownhill Proton Plus PIC BASIC Compiler, see **www.crownhill.co.uk** or **www.picbasic.org**
Crownhill build and sell other Iosoft products, see the Iosoft web site for technical detail and **www.crownhill.co.uk** for sales

2. Development environment

Development environment

To use the PB_UDP package, you will need:

1. The source files PD_UDP.BAS, UDP_DATA.INC and UDP_VARS.INC; they may have been supplied within a Zip file, in which case it must be unzipped before putting it in an appropriate directory (e.g. c:\projects\picbasic). There is a second file PB_UDP2.BAS that is only required when demonstrating board-to-board communications (see section 5).

2. A recent version of the Crownhill Proton Plus compiler. The software was tested with v1.24e, which does include support for 32-bit 'long' variables.

3. A Crownhill Proton-Net target board, with PIC16F877 device.

4. A PICmicro device programmer, that can load the compiled HEX file into a PIC16F877 device.

5. A PC running Windows, with an Ethernet card configured to run TCP/IP.

6. **Either** a crossover Ethernet cable **or** a hub and two straight-through Ethernet cables to link the PC to the target board. An older 10 megabit or a newer 10/100 megabit auto-switching hub may be used.

7. The Iosoft Ltd. 'datagram' network utility for the PC.

8. A straight-through serial cable to connect the target board to an unused serial port on the PC.

9. A terminal-emulator program for the PC (such as Hyperterm supplied with Windows), configured to 9600 baud, no parity, 1 stop bit.

3. Using the software

Getting started

Before looking at the structure of the software, it is worth getting it running in order to check out the development environment.

It is recommended that you set up the target board & PC on their own isolated network, rather than using an existing office network, as it simplifies the initial testing, and avoids the possibility of disrupting office communications.

In order for the PC and target board to communicate, they must not only be physically connected (via a crossover Ethernet cable, or a hub), but they must also have addresses that are in the same **domain**, i.e. must have similar addresses. By default, the PB_UDP package uses an address of 10.1.1.99, which is defined at the top of the Basic file:

```
SYMBOL MYIP1    = 10
SYMBOL MYIP2    = 1
SYMBOL MYIP3    = 1
SYMBOL MYID     = 99
```

It will also try to contact a server (your PC) at the address 10.1.1.3

```
SYMBOL REMIP1   = 10
SYMBOL REMIP2   = 1
SYMBOL REMIP3   = 1
SYMBOL REMIP4   = 3
```

To check whether your PC has a compatible address, look at the TCP/IP properties (details will vary depending on the Windows version).

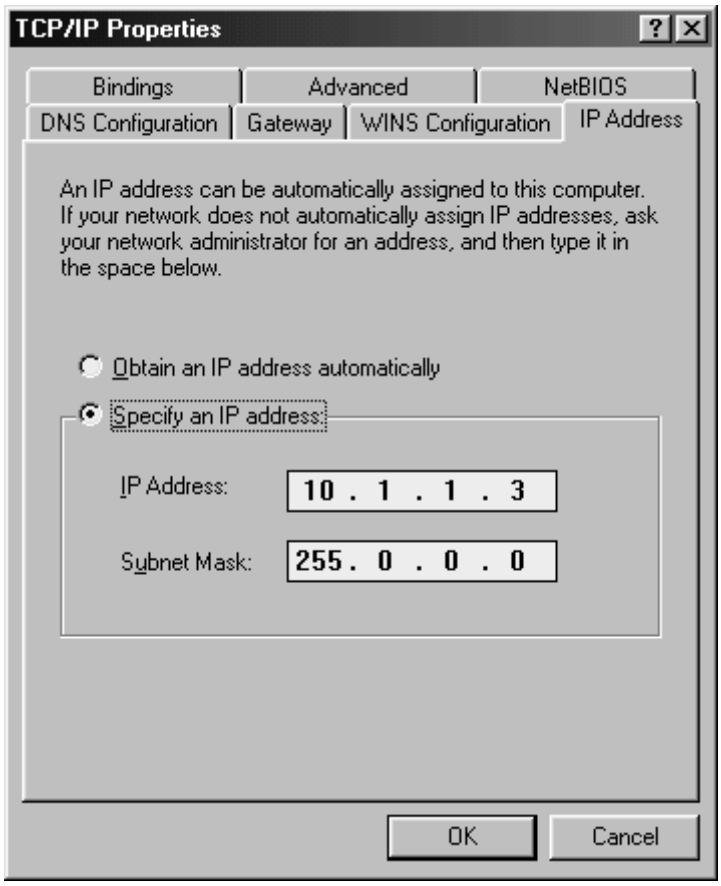


Figure 1: Windows TCP/IP properties

If the PC is set to obtain the IP address automatically, then it won't function in our simplified network, so a fixed IP address should be specified. Be warned that changing the network configuration of a PC is a non-trivial task; it is important to record the existing configuration before any changes (if in doubt, contact the System Administrator for your network before proceeding).

Alternatively, the target board configuration can be changed to reflect the PC settings; for example if the PC is at address 192.168.0.1, then change the Basic code to:

```
SYMBOL MYIP1    = 192
SYMBOL MYIP2    = 168
SYMBOL MYIP3    = 0
SYMBOL MYID     = 99

SYMBOL REMIP1   = 192
SYMBOL REMIP2   = 168
SYMBOL REMIP3   = 0
SYMBOL REMIP4   = 1
```

The addresses given throughout this manual will also need to be adjusted to match.

It is important that the target board address isn't in use by any other device (or any other target board). The easiest way to guarantee this is to ensure no other devices are physically connected to the network. When connecting to an existing network, extreme care is necessary; an address may appear to be unused (e.g. there is no response to a 'ping') but this may just be because the

device is currently powered down; when it is powered up, it will clash with the target board and cause problems. There is normally one individual (the network administrator) who is responsible for allocating addresses on the network, and he/she should be consulted prior to adding any new devices.

Programming the PICmicro

The PB_UDP source file should be loaded into the compiler and compiled, and the resulting PB_UDP.HEX file programmed into a PIC16F877. There are various types of programming adaptor, so details cannot be given here, but it is important to check the configuration fuse settings before programming the device, namely:

- The watchdog must be disabled.
- Low-voltage programming should be disabled
- The crystal type must be HS (high speed)
- In-circuit debugging must be disabled

The programmed device should be inserted into the target board, and the serial and network cables connected. A terminal emulator running on the PC, set to 9600 baud, no parity, 1 stop bit. On power-up, something similar to the following should be observed:

```
PB_UDP
My IP 10.1.1.99, host 10.1.1.3, gate 10.1.1.100
Tx ARP request
Rx len 60 pcol 806 ARP resp
Tx time request
Rx len 70 pcol 800 IP ICMP Destination unreachable
Tx time request
Rx len 70 pcol 800 IP ICMP Destination unreachable
```

All three network indicators on the target board should be lit, the Tx and Rx LEDs flashing every second, and the system LED toggling on or off every second.

Other possible indications are:

- **No activity: system LED doesn't flash**. The PIC16F877 device programming and/or configuration fuse settings are probably incorrect.
- **The 'link status' LED doesn't light**. The target board Ethernet connection is faulty; the cable may be of the wrong type (crossed vs. uncrossed). If using a hub, check that the corresponding link status LED for the port you are using; communication can only take place if the status LEDs at both ends of the link are illuminated.
- **The serial link only shows 'Tx ARP request' every second.** The target board is trying to contact the PC, but is receiving no response. Check the target board addresses as printed out on start-up, and also the PC address.

If you are in doubt about the PC network configuration, run the utility WINIPCFG, select the network adaptor you are using, and you should see a display similar to the following:

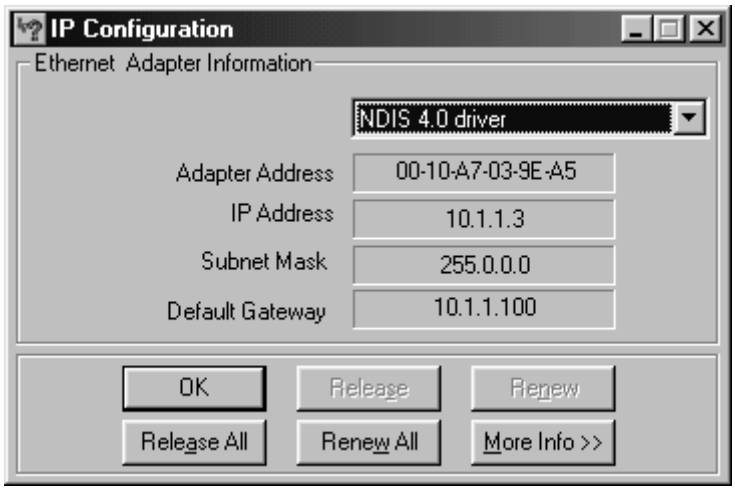


Figure 2: WINIPCFG display

PC time server: the DATAGRAM utility

The target board is contacting the PC every second in an attempt to find out the current time; the 'destination unreachable' message indicates that the PC has received the request, but doesn't know how to respond to it. To give the PC the capability to act as a time server, the Iosoft DATAGRAM utility must be run on the PC. On start-up, the utility presents a small control window, divided into 4 main areas:

- **Local UDP server**. This controls which of 3 services the utility will offer to the network.
- **Destination address**. The utility can initiate network communications, and this area gives the destination address and port number. There is also a timer value which, if non-zero, will cause the communication to be repeated.
- **Transmit UDP data**. The data to be transmitted, in text and hexadecimal notation.
- **Receive UDP data**. Any responses to the outgoing network messages.

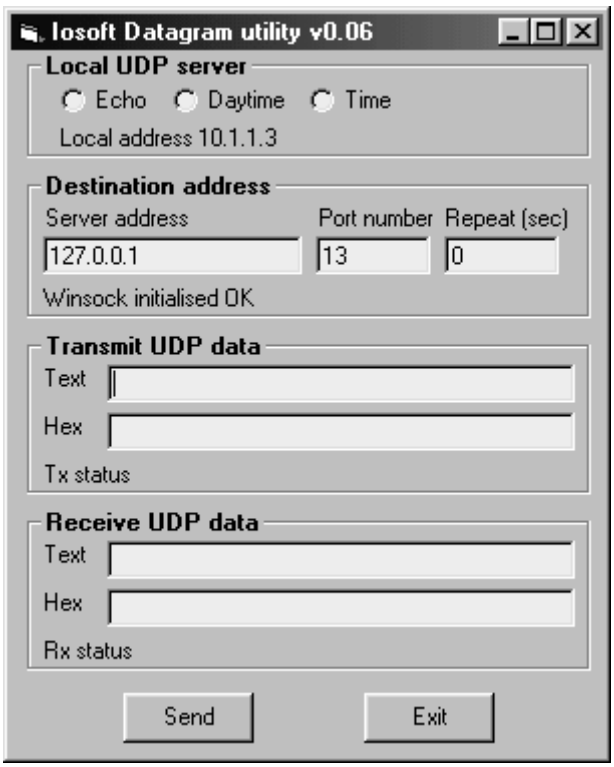


Figure 3: The Iosoft DATAGRAM utility

To enable server operation, it is only necessary to click one of the radio buttons in the top area, we require the 'time' service, so the right-hand button is clicked. The UDP server display briefly changes to 'listening on port 37', then to a count of request that changes every second, e.g. '17 Rx, last from 10.1.1.99 port 1024 len 0'. This confirms that the PC is receiving the time requests, and is returning the time value; the target serial link shows:

```
Rx len 60 pcol 800 IP UDP time
Tx time request
Rx len 60 pcol 800 IP UDP time
Tx time request
```

..and so on, updated every second. The time value received from the PC is displayed on the top right-hand side of the target board LCD, in 24-hour notation, e.g.

```
15:11:04
```

This shows how a simple target board can fetch the current time over the network, instead of having its own real-time clock chip with battery backup, and a user interface to set the time. At a network level, it demonstrates how the client software on the target board can contact the server on the PC, and receive a response.

Target board server: message handler

The target board software also has the ability to process incoming messages, and transmit a reply. To demonstrate this, the DATAGRAM utility is used to send a message to the target board, and display the response.

The datagram utility is set up as follows:

- The server address is set to the target board address 10.1.1.99
- The server port number is set to 4321
- The transmit data is set to the desired text or hex values
- The 'send' button is clicked.

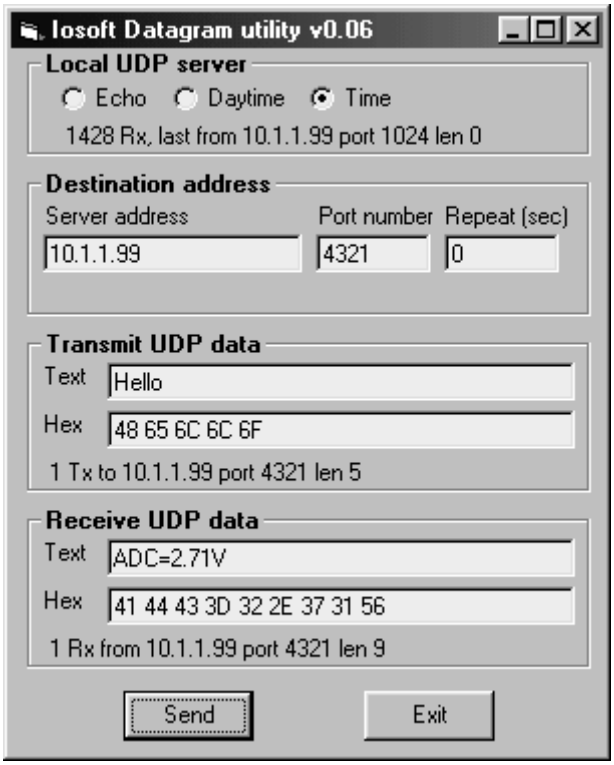


Figure 4: Sending message to target board

The transmitted text ('Hello') appears on the lower line of the target LCD display, and the datagram utility shows a response that is the current voltage measured on the first ADC channel (2.71 volts), which is set by a potentiometer on the board. To see the value change, enter '1' in the 'repeat' box, click 'send', and the voltage value will be redisplayed every second.

Ping diagnostics

The 'ping' utility is often used to diagnose network problems, and the PB_UDP software can respond to pings. Open up a DOS box on the PC, and enter 'ping' plus the target address:

```
C:\WINDOWS>ping          10.1.1.99
Pinging 10.1.1.99 with 32 bytes of data:

Reply from 10.1.1.99: bytes=32 time=35ms TTL=128
Reply from 10.1.1.99: bytes=32 time=34ms TTL=128
Reply from 10.1.1.99: bytes=32 time=34ms TTL=128
Reply from 10.1.1.99: bytes=32 time=34ms TTL=128

Ping statistics for 10.1.1.99:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 34ms, Maximum =  35ms, Average =  34ms
```

Due to buffer size constraints, the maximum ping data size the PB_UDP software can handle is 32 bytes.

Using a gateway

So far, all communications have been restricted to a small desk-top network, but the PB_UDP software has the ability to send & receive messages across larger network, including the Internet.

To do this, it is programmed with the address of a 'gateway', which will forward any outgoing messages to a wider network, and return any responses. The gateway address in given at the top of the source file, by default 10.1.1.100

```
SYMBOL GWIP1    = 10
SYMBOL GWIP2    = 1
SYMBOL GWIP3    = 1
SYMBOL GWIP4    = 100
```

There is also a netmask (default 255.255.255.0), which is used to determine if the desired target address is on the local subnet (no gateway required) or on a remote network (all requests to go though the gateway).

```
SYMBOL MASKIP1  = 255
SYMBOL MASKIP2  = 255
SYMBOL MASKIP3  = 255
```

In theory, all that is necessary is to set the gateway IP address to the correct value for your gateway, the remote IP address to the desired timeserver on the Internet (e.g. the Demon Internet time server 158.152.1.76), and the PB_UDP software will automatically send the time requests out on the Internet, and show the results. In practice, setting up a connection from an office network to the Internet can be quite complicated, as there may be various security systems ('firewalls') in place to stop unwanted messages entering or leaving the site, so it is important to liase with the network administrator before attempting Internet communications.

4. Structure of the software

The PB_UDP software encompasses the following main components:

- Definitions
- Hardware initialisation
- Transmission
- Reception
- Client code
- Server code

Definitions

Extensive use is made of the SYMBOL command to define symbolic constants, for example:

```
SYMBOL LCD_LINE2    = $c0
```

Thereafter, whenever LCD_LINE2 is used the compiler substitutes a value of 0C0h:

```
b = LCD_LINE2: GOSUB LCD_cmd
```

It is convenient to be able to redefine certain byte locations as 16 or 32-bit words, and there is a compiler trick that allows this. For example, the 13th and 14th bytes of the received network frame are a 16-bit protocol value, with the most-significant byte first. The following definition allows these two bytes in the receive buffer to be accessed as if they were one word.

```
DIM rxbuff[RXBUFFLEN] AS BYTE
SYMBOL pcol_            = rxbuff#13
SYMBOL pcol_h           = rxbuff#12
DIM pcol          AS pcol_.WORD
. . .
IF pcol = PCOL_ARP THEN
. . .
```

IP ADDRESS

Symbol definitions are also used for the main IP address definitions

```
SYMBOL MYIP1    = 10
SYMBOL MYIP2    = 1
SYMBOL MYIP3    = 1
'My ID number; last byte of MAC & IP address
SYMBOL MYID     = 99

'4 bytes of remote IP address
SYMBOL REMIP1     = 10
SYMBOL REMIP2     = 1
SYMBOL REMIP3     = 1
SYMBOL REMIP4   = 3
```

```
'4 bytes of gateway IP address
SYMBOL  GWIP1       =  10
SYMBOL  GWIP2       =  1
SYMBOL  GWIP3       =  1
SYMBOL GWIP4   = 100

'3 bytes of IP address mask
SYMBOL MASKIP1  = 255
SYMBOL MASKIP2  = 255
SYMBOL MASKIP3  = 255
```

If several PB_UDP implementations are on the same network, they must each have a unique value for 'MYID', which is used as the least-significant byte of the address.

MAC ADDRESS

In addition to having a unique IP address, the board must also have a unique hardware (MAC) address. There are various ways this could be achieved, the easiest being to use the 'locally administered' MAC address space provided by the IEEE. If the most-significant byte has a value of 2, then the remaining bytes aren't IEEE-administered, so can be any value we choose (so long as they don't happen to coincide with those of someone else's choosing). The 'MYID' value is used as the least-significant byte of the MAC address; providing it has been set to a unique value for each board on the network, then no two boards will have the same MAC address.

Hardware initialisation

To set up the Ethernet Network Interface Controller (NIC), a large number of hardware registers have to be initialised with specific data values. This has been implemented using DATA statements, with pairs of values representing the register and the byte value, for example

```
SYMBOL RBCR0   = $0a
SYMBOL RBCR1   = $0b
SYMBOL RCR     = $0c
SYMBOL TCR     = $0d
SYMBOL DCR     = $0e
. . .
DATA DCR,        $48          'Data config
DATA RBCR0,      $00          'Clear RBCR
DATA RBCR1,    $00
DATA RCR,      $20       'Rx monitor mode
```

In essence, the software has to fetch each pair of values, and put the second into the register specified by the first:

```
SYMBOL  NICADDR   =  PORTB
SYMBOL  NICDATA   =  PORTD
SYMBOL  NIC_IOR   =  PORTE.0
SYMBOL NIC_IOW  = PORTE.1
SYMBOL  NIC_RST   =  PORTE.2
. . .
REPEAT
      READ a: READ b
      NICADDR = a: NICDATA = b: GOSUB out_nic
UNTIL a >= T_STOP
```

The out_nic subroutine asserts the write-enable line, so that the data value is written to the required address.

```
'Do a NIC write cycle
out_nic:
        OUTPUT PORTD
        NIC_IOW = 0
        DELAYUS 3
        NIC_IOW = 1
        INPUT  PORTD
RETURN
```

TIME DELAY

It takes a finite time for the NIC to respond to some settings, so delays have to be inserted at specific points. This is handled by a including a dummy register value (outside the normal 0-1F hex range for NIC registers) to indicate a delay is required, e.g.

```
SYMBOL T_WAIT       = $fd 'Wait x ms
. . .
DATA CMDR,      $21         'Stop, DMA abort (wait 2ms)
DATA T_WAIT,    2           'Wait 2ms
DATA DCR,       $48         'Data config
```

TASK BLOCKS

This idea of dummy register values to indicate special functions has been used extensively, for example there is a dummy value NIC_INIT to mark the start of the initialisation data, and T_STOP to mark the end. This means that the data table can contain several functional blocks, each with a defined beginning and end.

```
DATA NIC_INIT,  0       'NIC initialisation:
DATA RESPORT,   $01         'Reset NIC
DATA T_WAIT,    2           'Wait 2ms
DATA CMDR,      $21         'Stop, DMA abort (wait 2ms)
. . .
DATA T_STOP,    0           'End of this task

DATA TX_ARPREQ, 0       'Tx broadcast ARP request
DATA ISR,       $40         'Clear remote DMA int flag
DATA RSAR0,       0             'Set remote DMA addr
DATA RSAR1,     TXSTART
. . .
DATA T_STOP,    0           'End of this task
```

Each block is referred to as a ‘task’, and is essentially a block of coded instructions that must be executed to initiate the required action, such as setting up the hardware, or transmitting a specific message. To action a task, the following code is used:

```
task = NIC_INIT: GOSUB do_nic_task          'Initialise NIC
. . .
task = TX_ARPREQ: GOSUB do_nic_task         'Send ARP request
```

The subroutine do_nic_task searches the data table for the appropriate code block, and then executes it:

```
do_nic_task:
        IF task > NIC_INIT OR task < TASKBASE THEN RETURN
        RESTORE
        REPEAT                          'Search for task marker
                READ a: READ b
                IF a = T_END THEN RETURN
        UNTIL a = task
        REPEAT                          'Do task
                READ a: READ b
                IF a <= RESPORT THEN NICADDR = a: NICDATA = b: GOSUB out_nic
                IF a = T_WAIT THEN DELAYMS b
                IF a>=T_VOUT1 AND a<=T_VOUT6 THEN GOSUB vout
        UNTIL a >= T_STOP
RETURN
```

Transmission

A transmission task block contains all the commands to set up the NIC, and all the data to be transmitted, for example an ARP request:

```
DATA TX_ARPREQ, 0       'Tx broadcast ARP request
DATA ISR,       $40     'Clear remote DMA int flag
DATA RSAR0,     0       'Set remote DMA addr
DATA RSAR1,     TXSTART
DATA CMDR,      $12     'Start DMA remote write
DATA DPORT,     $ff     'Broadcast addr
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA T_VOUT6,   V_LOC_MAC   'Local MAC addr
DATA DPORT,     $08     'Ether pcol 0806: ARP
DATA DPORT,     $06
DATA DPORT,     $00     'ARP hardware type 0001: Ethernet
DATA DPORT,     $01
DATA DPORT,     $08     'ARP protocol type 0800: IP
DATA DPORT,     $00
DATA DPORT,     MACLEN      'ARP hardware addr len
DATA DPORT,     IPADDRLEN   'ARP pcol addr len
DATA DPORT,     $00     'ARP operation 0001: request
DATA DPORT,     $01
DATA T_VOUT6,   V_LOC_MAC   'Local MAC addr
DATA T_VOUT4,   V_LOC_IP    'Local IP addr
DATA DPORT,     $ff     'Target MAC addr: broadcast
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA DPORT,     $ff
DATA T_VOUT4,   V_REMGATE_IP
DATA T_STOP,    0       'End of this task
```

After the initial register setting, the packet data is put in the NIC buffer (you may care to review the ARP format as described in 'TCP/IP Lean' chapter 3):

> 6-byte destination MAC address: all 1's (broadcast) 6-byte source MAC address: my hardware address Protocol word: 0806h (ARP)
> Hardware type word: 0001h (Ethernet)
> ARP protocol word: 0800 (IP)
> Hardware address length byte: 6 (Ethernet address length)
> IP address length byte: 4

ARP operation word: 0001 (ARP request)
6-byte local MAC address
4-byte local IP address
6-byte broadcast MAC address
4-byte destination IP address

DATA VARIABLES

You will note that yet more dummy register values have been used to represent variables such as the source & destination addresses, so for example the data pair

```
DATA T_VOUT4,   V_LOC_IP
```

Indicates that a 4-byte value should be sent to the NIC data register, consisting of the local IP address. This is handled in the vout subroutine:

```
NICADDR = DPORT
IF b = V_LOC_IP THEN
      NICDATA = MYIP1: GOSUB out_nic
      NICDATA = MYIP2: GOSUB out_nic
      NICDATA = MYIP3: GOSUB out_nic
      NICDATA = MYID: GOSUB out_nic
ENDIF
```

GATEWA Y

The data variable V_REMGATE_IP represents the destination IP address, and has an additional feature; if the destination is outside the network, the ARP request will be sent to the gateway, instead of the true destination. The destination address is XORed with the local address and then masked using the netmask value to decide if the gateway IP or remote IP is to be used.

```
a = ((REMIP1^MYIP1)&MASKIP1) | ((REMIP2^MYIP2)&MASKIP2) |
((REMIP3^MYIP3)&MASKIP3)
      IF b = V_REMGATE_IP AND a <> 0 THEN
            NICDATA = GWIP1: GOSUB out_nic
            NICDATA = GWIP2: GOSUB out_nic
            NICDATA = GWIP3: GOSUB out_nic
            NICDATA = GWIP4: GOSUB out_nic
            RETURN
      ENDIF
      IF b = V_REM_IP OR b = V_REMGATE_IP THEN
            i = 0
            REPEAT
                  NICDATA = rem_ip[i]: GOSUB out_nic: INC I
            UNTIL I > IPADDRLEN-1:
      ENDIF
```

LENGTH CALCULATION

It is necessary to know in advance how much data is being sent to the NIC, in order to set the registers appropriately. A subroutine get_datalen finds a given task, computes the amount of data it contains, and then sets the NIC length registers accordingly.

```
get_datalen:
        IF task > NIC_INIT OR task < TASKBASE THEN RETURN
        RESTORE
        REPEAT                          'Search for task marker
                READ a: READ b
                IF a = T_END THEN RETURN
        UNTIL a = task
        datalen = 0
        REPEAT                          'Tally data lengths
                READ a: READ b
                IF a = DPORT THEN datalen = datalen + 1
                IF a = T_VOUT2 THEN datalen = datalen + 2
                IF a = T_VOUT4 THEN datalen = datalen + 4
                IF a = T_VOUT6 THEN datalen = datalen + 6
        UNTIL a >= T_STOP
        NICADDR = RBCR0: NICDATA = datalen: GOSUB out_nic
RETURN
```

CLIENT/SERVE R REMOTE/RESPONSE ADDRESSES

The PB_UDP code must handle two types of transaction:

CLIENT: initiate a request and wait for a response

SERVER: wait for a request, and send a response

As supplied, the software has two interlinked clients:

ARP CLIENT: requests the MAC address of the time server (or of the gateway, if the server isn't on the local network).

TIME CLIENT: once the MAC address is found, the time is requested at one-second intervals.

There are also three servers:

ARP SERVER: responds to ARP requests for the local IP address.

PING SERVER: responds to pings (ICMP echo requests)

MESSAGE SERVER: accepts message for display on the LCD, returns the current ADC voltage value.

The important point is that the PB_UDP software has no control over the timing of any incoming messages; if the time client has just requested the time value, there is no guarantee that the next message received will be the time response; it could be an ARP or ping request instead. Due to the limited storage space, all messages must be acted upon as soon as they arrive, so the ARP or ping response must be sent even though a time response is expected.

For this reason, two types of destination addresses are used: a **response** address and a **remote** address. The former is used by the server for all its replies, and is just a copy of the incoming source address (i.e. it is used for 'return to sender' replies), while the remote address is a fixed value corresponding to, say, the time server address.

COPYING SOURCE DATA

A frequent requirement is to copy data from an incoming message to the outgoing message. An instance of this is the ARP server, where the original source IP address has to be copied into the outgoing destination IP address field. This is achieved by using the T_VOUT4 4-byte variable definition with the offset of the original source address within the receive buffer:

```
DATA T_VOUT4,   28        'Target IP addr
```

This copies 4 bytes from the receive buffer location 28 to the transmit buffer in the NIC. The same technique can be used with other data sizes, for example copying an incoming 2-byte UDP port number into the outgoing data

```
DATA T_VOUT2,   34     'Destination port (= incoming source port)
```

UDP MESSAGE RE SPONSE

The most complex transmission is the UDP message response, as it must conform to UDP & IP formatting, and also include a dynamically-changing voltage value.

```
DATA TX_MSGRSP, 0     'Tx UDP message response
DATA RSAR0,     0         'Set remote DMA addr
DATA RSAR1,     TXSTART
DATA CMDR,      $12       'Start DMA remote write
DATA T_VOUT6,   V_RESP_MAC  'Send to response MAC addr
DATA T_VOUT6,   V_LOC_MAC   'Local MAC addr
DATA DPORT,     $08       'IP protocol
DATA DPORT,     $00
DATA DPORT,     $45       'IP v4, header len 20 bytes
DATA DPORT,     0         'Normal service
DATA T_VOUT2,   V_IPLEN     'Length of IP datagram
DATA DPORT,     0         'Identifier
DATA DPORT,     1
DATA DPORT,     0         'No fragmentation
DATA DPORT,     0
DATA DPORT,     128       'Time to live
DATA DPORT,     17        'Protocol UDP
DATA DPORT,     $0        'Dummy checksum
DATA DPORT,     $0
DATA T_VOUT4,   V_LOC_IP    'Source IP addr
DATA T_VOUT4,   26        'Dest IP addr
DATA DPORT,     MSGPORT/256 'Source port
DATA DPORT,     MSGPORT//256
DATA T_VOUT2,   34          'Dest port (= incoming source port)
DATA T_VOUT2,   V_UDPLEN     'UDP length (header + data)
DATA DPORT,     0         'Disable checksum
DATA DPORT,     0
DATA DPORT,     "A"       'UDP data: voltage value
DATA DPORT,     "D"
DATA DPORT,     "C"
DATA DPORT,     "="
DATA T_VOUT4,   V_VOLTAGE
DATA DPORT,     "V"
DATA T_STOP,    0         'End of this task
```

The points to note are:

Dynamic data. A special variable value V_VOLTAGE has been defined, that returns a 4-character string containing the ADC value in volts, e.g. “1.23”.

Data length. The IP and UDP data length has to be computed before executing this task, as follows

```
task = TX_MSGRSP: GOSUB get_datalen 'Get response length
iplen = datalen - MACHDR_LEN        'Calc IP & UDP len
udplen = iplen - IPHDR_LEN
```

Checksum. The UDP checksum is set to zero to disable it, but the IP checksum can not be disabled. A subroutine is called to compute the checksum just prior to transmission:

```
GOSUB do_nic_task
GOSUB calc_tx_ip_csum: GOSUB transmit
```

Port number. An arbitrary port number has been chosen for the message server:

```
'Port to be used for incoming messages
SYMBOL MSGPORT  = 4321
```

Reception

The subroutine ‘receive’ is used to check the Network Interface Controller (NIC) to see if any frames have been received. The NIC can store up to 6KB of incoming data in its own circular buffer, so it isn’t necessary to use interrupts to signal data arrival.

The process by which the data is extracted from the NIC packet buffer is known as ‘remote DMA’, and is too complex to discuss here; if you need more information, see the National Semiconductor DP8390 family data sheets, since this device is the heart of the NE2000-compatible network controller.

A relatively small buffer is used to store the incoming data, due to the severe RAM constraints of the PIC16.

```
'Size of receive buffer
SYMBOL RXBUFFLEN= 74
DIM rxbuff[RXBUFFLEN] AS BYTE
```

For an incoming UDP packet, the sizes are:

Ethernet header:	14 bytes (2 x 6-byte MAC addresses, protocol word)
IP header:	20 bytes
UDP header:	8 bytes
UDP data:	Up to 32 bytes

A much larger buffer size could be used on a PIC18xxx processor, though a buffer larger than 255 bytes would necessitate some code modifications, as single-byte length values have been used in various places.

If a frame has been received, it is copied into the receive buffer, and the ‘rxlen’ variable is set to the length; if the frame exceeds the buffer size it is truncated. The frame is then analysed using the check_rx subroutine.

```
GOSUB receive
IF rxlen > 0 THEN
        HRSOUT "Rx len ", DEC rxlen, " pcol ", HEX pcol, " "
        GOSUB check_rx
        HRSOUT $d, $a
ENDIF
```

The check_rx subroutine has to classify the incoming frame, which may be ARP (request or response), IP ICMP (ping request) or IP UDP (message request or time response). If the frame is unrecognised, it is discarded.

During packet reception and decoding, a significant amount of information is printed out to the serial link. Whilst this may be very useful for initial testing, it does significantly increase the response time, so it is recommended that the diagnostic printouts be removed when developing a real-world application.

ARP HANDLER

The ARP message handler must check that the destination IP address matches the local IP address, then branch to handle a request (from a remote host) or a response (to a request we have sent). In the latter case, the remote MAC address is stored for later use, and a flag is set to show the ARP cycle is complete.

```
IF pcol = PCOL_ARP THEN
        idx = ETHLEN + 24 : GOSUB match_myip
        IF matched = 1 THEN
                IF arp_op = $0001 THEN          'ARP request
                        HRSOUT "ARP req"
                        txlen = 0               'Send ARP response
                        task = TX_ARPRSP
                        GOSUB get_datalen: GOSUB do_nic_task
                        GOSUB transmit
                ENDIF
                IF arp_op = $0002 THEN          'ARP response
                        idx = ETHLEN + 14:
                        GOSUB match_remip: GOSUB match_gwip
                        IF matched = 1 THEN
                                HRSOUT "ARP resp"       'Get MAC address
                                idx = ETHLEN + 8: GOSUB get_remmac
                                arped = 1
                        ENDIF
                ENDIF
        ENDIF
ENDIF
```

IP HANDLER

An incoming IP message (known as a ‘datagram’) is checked to ensure the destination IP address matches the local address, then is split into ICMP or UDP handlers.

```
IF pcol = PCOL_IP THEN                  'IP protocol
        idx = 30 : GOSUB match_myip             'Match my address
        IF matched = 1 THEN
                HRSOUT "IP "
                IF ip_pcol = 1 THEN             'ICMP protocol
                        HRSOUT "ICMP "
                        . . .
                ENDIF
                IF ip_pcol = 17 THEN            'UDP datagram
                        HRSOUT "UDP "
                        . . .
                ENDIF
        ENDIF
ENDIF
```

ICMP HANDLER

The main ICMP message we'll receive is an Echo Request (ping request), where the incoming data must be echoed back. The Destination Unreachable message is also displayed, since this will be received if we try to access a closed UDP port (i.e. the PC we're contacting isn't running a time server application).

```
IF ip_pcol = 1 THEN             'ICMP protocol
        HRSOUT "ICMP "
                IF icmp_type = 8 AND rxiplen < 256 THEN
                        iplen = rxiplen         'ICMP echo request
                        task = TX_ICMPRSP: GOSUB get_datalen: GOSUB do_nic_task
                        icmp_type = 0           'Send echo response
                        icmp_csum = icmp_csum + 8       'Adjust checksum
                        IF icmp_csum < 8 THEN icmp_csum = icmp_csum + 1
                        NICADDR = RBCR0: NICDATA = iplen-20: GOSUB out_nic
                        NICADDR = RBCR1: NICDATA = 0 : GOSUB out_nic
                        NICADDR = CMDR: NICDATA = $12 : GOSUB out_nic
                        NICADDR = DPORT         'Copy header & data
                        FOR i=0 TO iplen-21
                                NICDATA = rxbuff[i+34]: GOSUB out_nic
                        NEXT
                        GOSUB calc_tx_ip_csum   'Do IP checksum
                        GOSUB transmit          'Send the frame
                ENDIF
                IF icmp_type = 3 THEN           'ICMP dest unreachable
                        HRSOUT "Destination unreachable"
                ENDIF
        ENDIF
ENDIF
```

UDP HANDLER

The port numbers are used to determine the type of incoming UDP message (confusingly, also known as a 'datagram'). If the **source** port is a time server, then it is a time response, while if the **destination** port is my message port, it is a text message.

```
IF rx_srce_port = 37
        HRSOUT "time "              'Response to time request
        . . .
ENDIF
IF rx_dest_port = 4321
        HRSOUT "message "           'Incoming text message
        . . .
ENDIF
```

UDP TIME HANDLER

An incoming time message consists of the a 32-bit value containing the number of seconds since January 1 1900, with the most-significant byte first. We're only interested in the time, which can be extracted using simple modulo arithmetic, and displayed on the LCD. A minor complication is that the compiler treats 32-bit values as **signed** integers, and the current time value is large (C1C964DC hex at the time of writing), which will look like a negative value. To correct this, an integer number of days (24,000) is subtracted to make the seconds count positive without altering the time value.

```
HRSOUT "time "              'Response to time request
rx_udp_dw = rx_udp_dw - $7b98a000 'Subtract days to make +ve
s = rx_udp_dw // 60         'Get hrs, min, sec
rx_udp_dw = rx_udp_dw / 60
m = rx_udp_dw // 60
rx_udp_dw = rx_udp_dw / 60
h = rx_udp_dw // 24
b = LCD_TIMEPOS: GOSUB LCD_cmd  'Display on top LCD line
a = h: GOSUB disp_dec2
b = $3a: GOSUB disp_char
a = m: GOSUB disp_dec2
b = $3a: GOSUB disp_char
a = s: GOSUB disp_dec2
```

UDP MESSAGE HANDLER

An incoming message to UDP port 4321 is displayed on the 2nd line of the LCD

```
idx = UDP_DATA_START        'Display on 2nd LCD line
b = LCD_LINE2: GOSUB LCD_cmd
FOR i=0 to 15               '16 chars on LCD
        b = rxbuff[idx]
        IF idx<rxlen AND b>0 THEN   'Maybe fewer in message
                GOSUB disp_char
                idx = idx + 1
        ELSE
                b = $20: GOSUB disp_char
        ENDIF
NEXT                        'Respond with ADC value..
```

A response is sent, giving the current voltage on ADC channel 0. The code is relatively simple, since the outgoing message is pre-formatted using DATA statements, as discussed earlier.

```
centivolts = adcval ** 32000        'Get ADC volts / 100
task = TX_MSGRSP: GOSUB get_datalen 'Get response length
iplen = datalen - MACHDR_LEN        'Calc IP & UDP len
udplen = iplen - IPHDR_LEN
GOSUB do_nic_task               'Send response
GOSUB calc_tx_ip_csum: GOSUB transmit
```

5. Communication between target systems

So far, all communication has been between a target board and the PC; to demonstrate board-to-board communications, a second file PB_UDP2.C has been included in the package. It is essentially the same as PB_UDP.C, with minor modifications:

- **The IP address has been changed from 10.1.1.99 to 10.1.1.98**. This is essential to avoid an IP address clash between the boards
- **The remote IP address has been changed from 10.1.1.3 to 10.1.1.99**. Instead of requesting data from a PC, the second board will request data from the first board.
- **The remote port number has been changed from 37 to 4321**. Instead of making time requests, the second board will be sending message requests to the first board, which will return its ADC value.

These changes are sufficient for the second board to poll the first, obtaining its ADC value every second. The UDP message handler also been modified to handle the new incoming messages:

```
IF rx_srce_port = MSGPORT
      HRSOUT "message resp "        'Incoming message response
      idx = UDP_DATA_START          'Display on 2nd LCD line
      b = LCD_LINE2: GOSUB LCD_cmd
      i = 0: REPEAT                 '16 chars on LCD
            b = rxbuff[idx]
            IF idx<rxlen AND b>0 THEN   'Maybe fewer in message
                  PRINT b
                  INC idx
            ELSE
                  PRINT " "
            ENDIF
            INC i
      UNTIL I > 15
ENDIF
```

This is very similar to the other UDP message handler, in that the incoming text (e.g. “ADC=1.23V”) is copied to the second line of the display.

Rotating the upper potentiometer on the first board will change the voltage value displayed on the second board, demonstrating that the two boards have been linked via the network.

6. The DATAGRAM utility

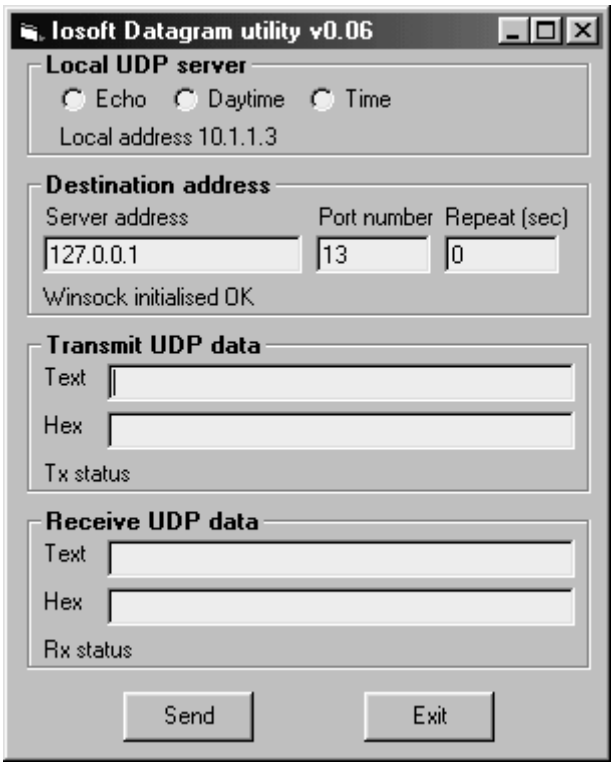


Figure 5: The Iosoft DATAGRAM utility

The Iosoft Datagram utility can perform the following server functions:

- An 'echo' server on UDP port 7. All incoming data is echoed back to the sender.
- An RFC 867 'daytime' server on UDP port 13. The incoming data is discarded, and a date & time string is returned to the sender.
- An RFC 868 'time' server on UDP port 37. The incoming data is discarded, and the number of seconds since 1st Jan 1900 is returned to the sender as an unsigned 32-bit value.

It also has the following UDP client capabilities:

- Sending UDP text or hex strings to a server.
- Receiving UDP text or hex strings in response.

A simple way of testing these capabilities is to employ the 'loopback' IP address 127.0.0.1; for example, enable the daytime server then send a blank (null) message to 127.0.0.1 port 13

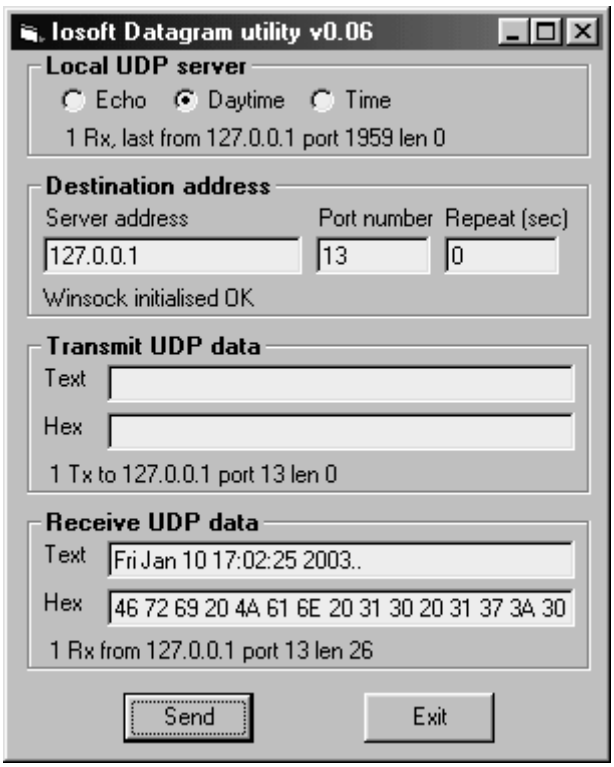


Figure 6: Using the loopback address

Instead of the message going out on the network, it is looped back internally to the datagram utility, which responds as if it had received a normal network request, so returns a date and time string, which is displayed as if it had been received over the network.

Similarly, if you enable the echo server, then send an arbitrary string of characters to 127.0.0.1 port 7, then that string will be echoed back, and displayed as received UDP data.

If a normal (non-loopback) IP address is used, then the UDP message will be sent out on the network, using the normal Windows TCP/IP settings.

JPB 23/1/03 --ends--

Crownhill Associates

smart electronic solutions

Proton-Net Schematic

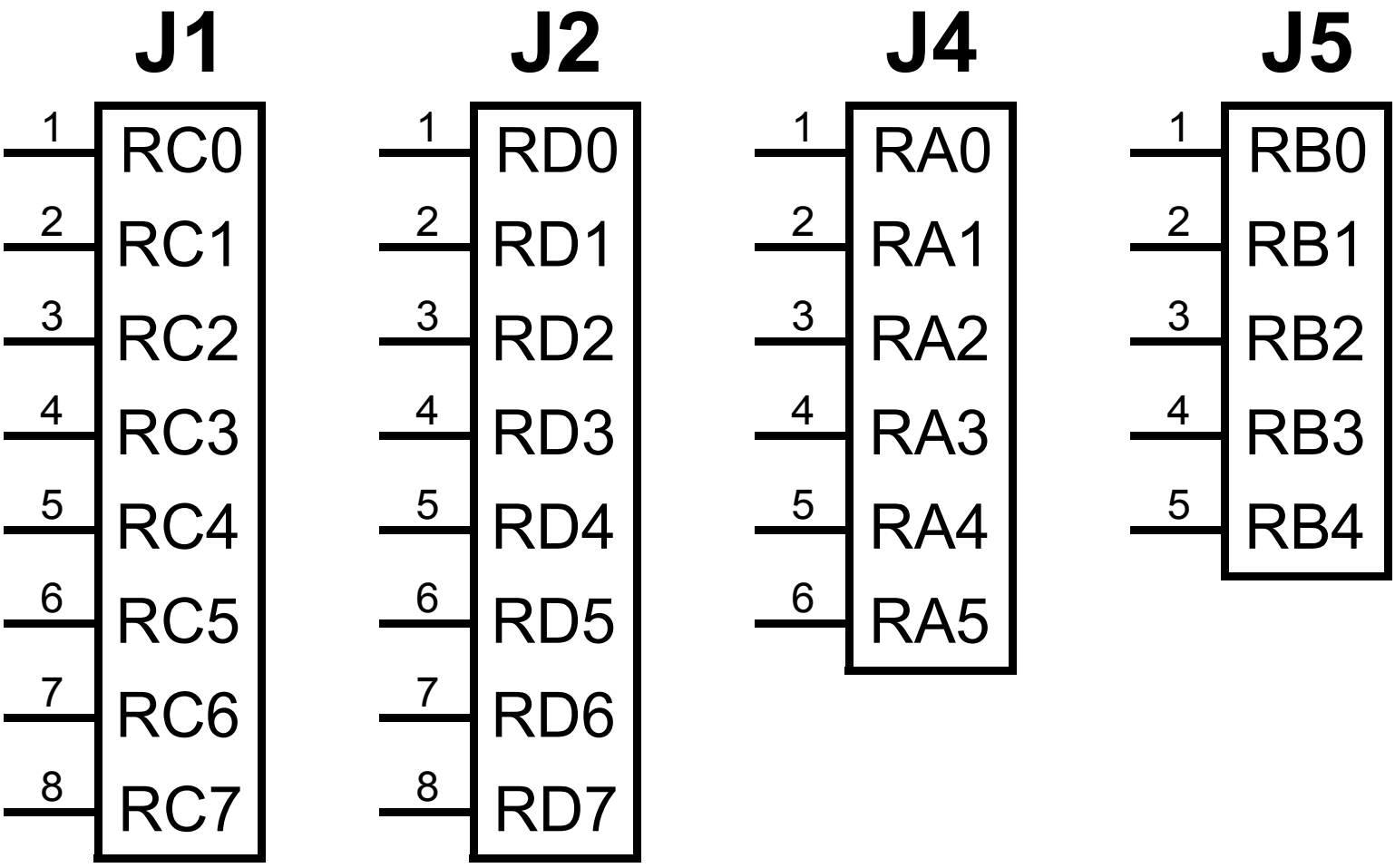
The following pages are schematics for the various components parts of the proton-Net PCB

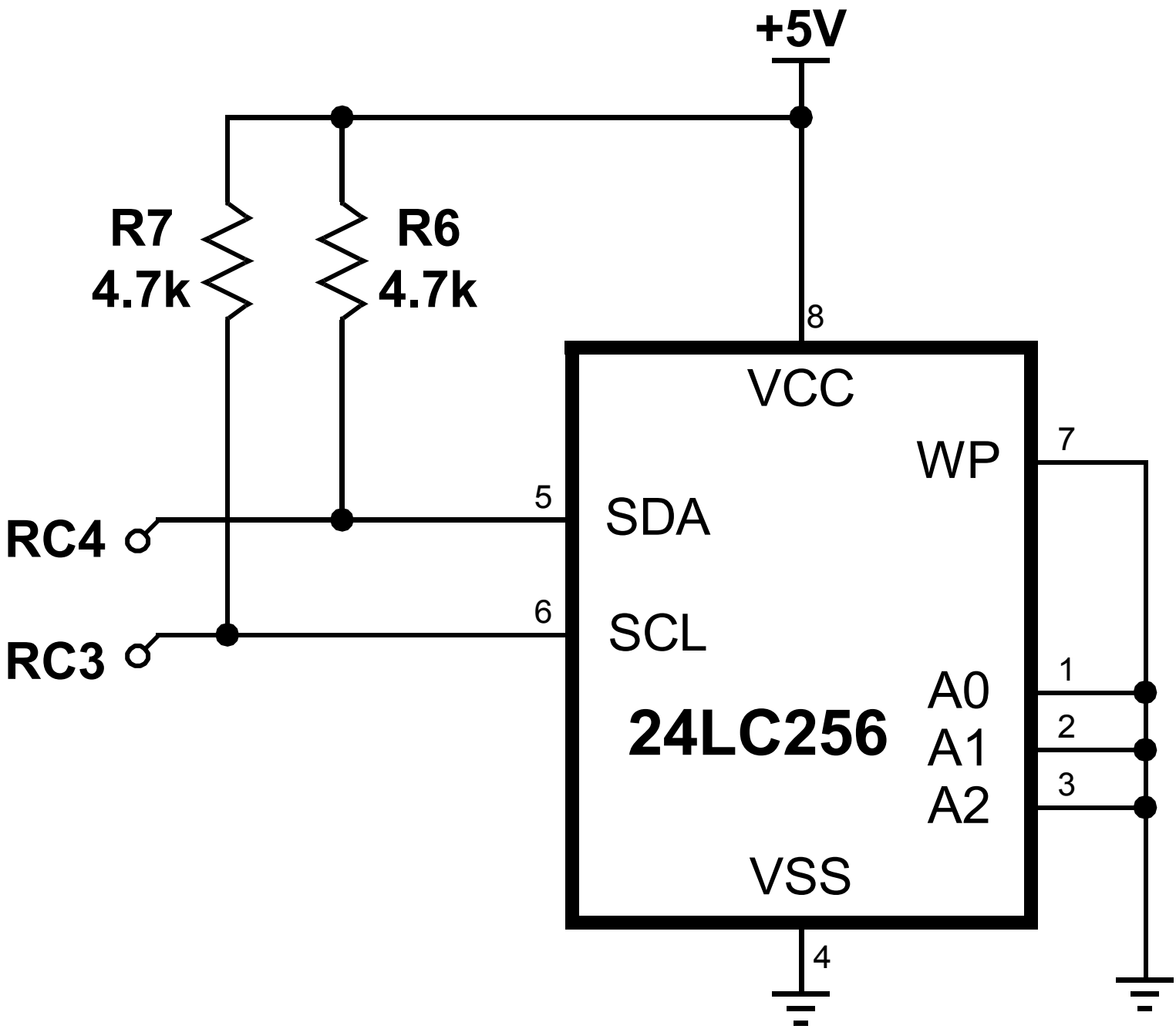
Item	Description
1	Connectors, J1,2,4&5
2	24LC256 eeprom, used to hold configuration and web page data. (PICDEM-NET® mode)
3	Alphanumeric LCD
4	LED (RTL status)
5	16F877
6	ADC test voltage supply
7	Power regulation
8	The full circuit
9	RS232 I/O
10	RTL8019AS connections

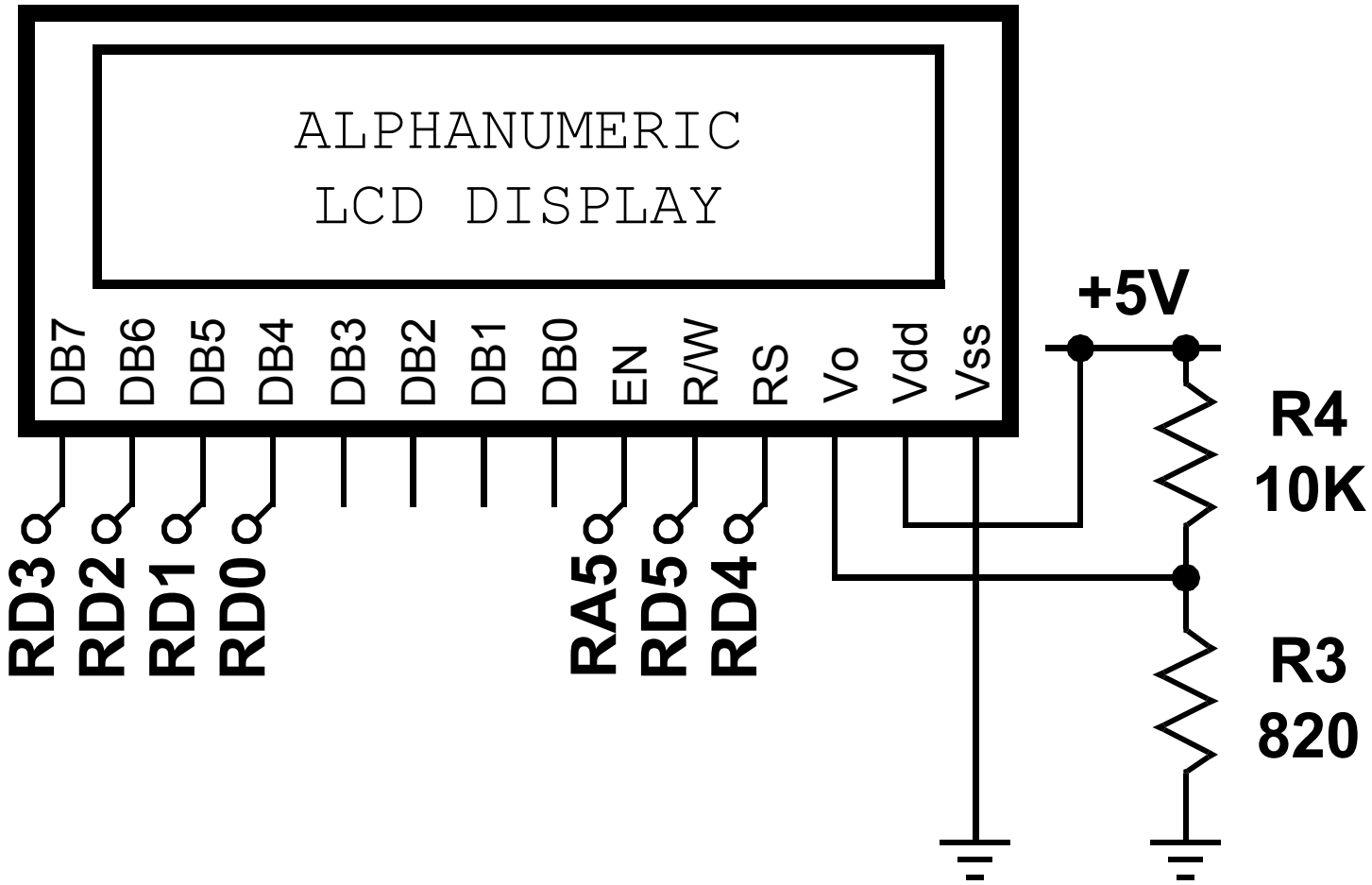
The I/O functionality of the Proton –NET development system is compliant and compatible with the MicroChip® PICDEM NET board as described in Jeremy Benthams Book: “TCP/IP Lean Web Servers for embedded systems” ISBN: 1-57820-108-x

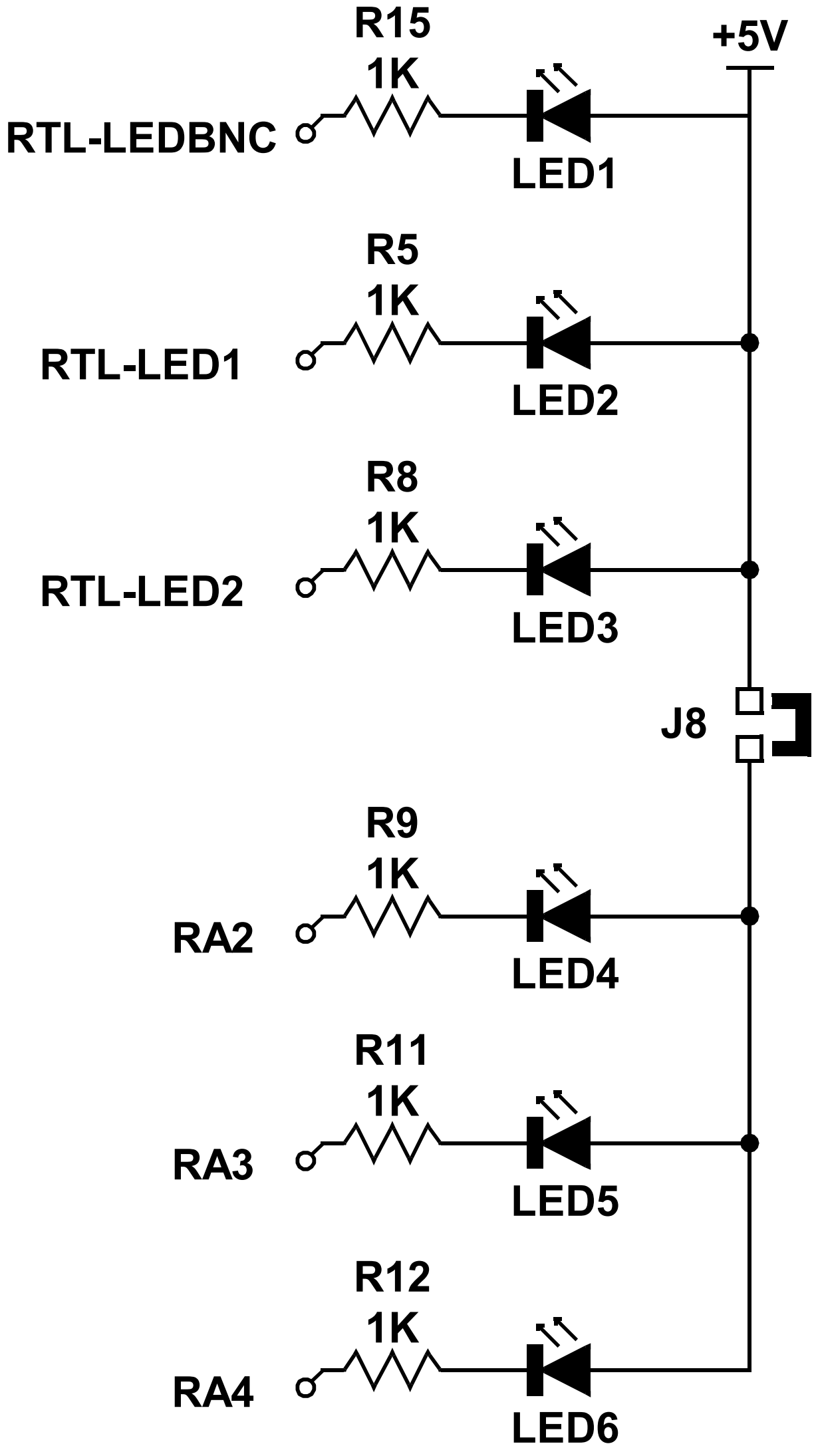
Note that when shipped U3 is programmed to allow demonstration of Jeremy Betham’s TCP/IP web server application, this functionality is NOT supported by PIC BASIC.

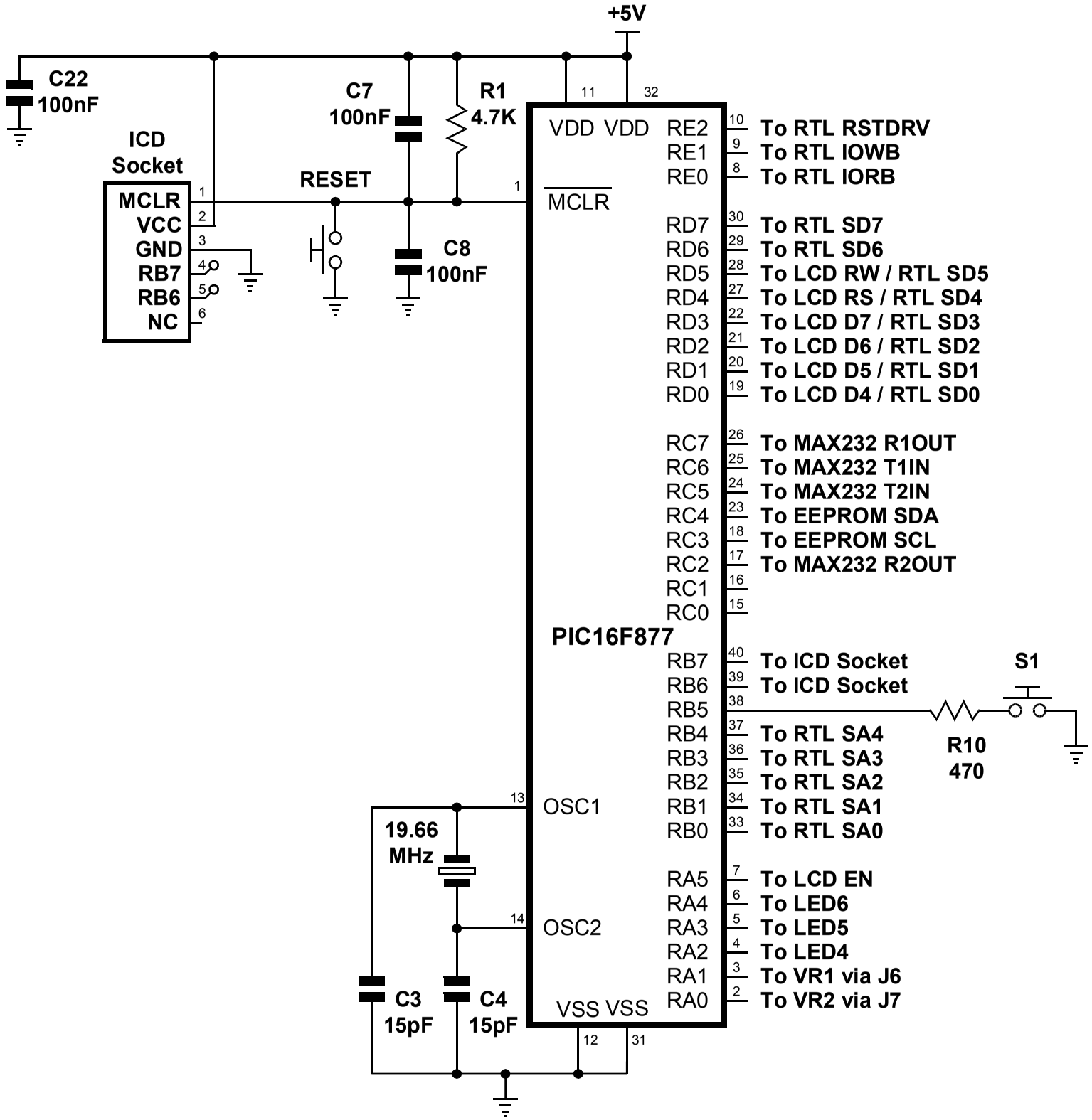
For production use, see Proton-Net *Light*, a small PCB comprising of RJ11 socket,RTL controller, Transformer, Xtal, LED’s for Link status Tx and RX optionally: PIC16f877 or PIC 16F452, Configuration eeprom for RTL controller, The board is in 40Pin DIL format, utilising Surface mount components to provide a small footprint, ready built solution for embedded Network comm’s, see www.crownhill.co.uk for more information.

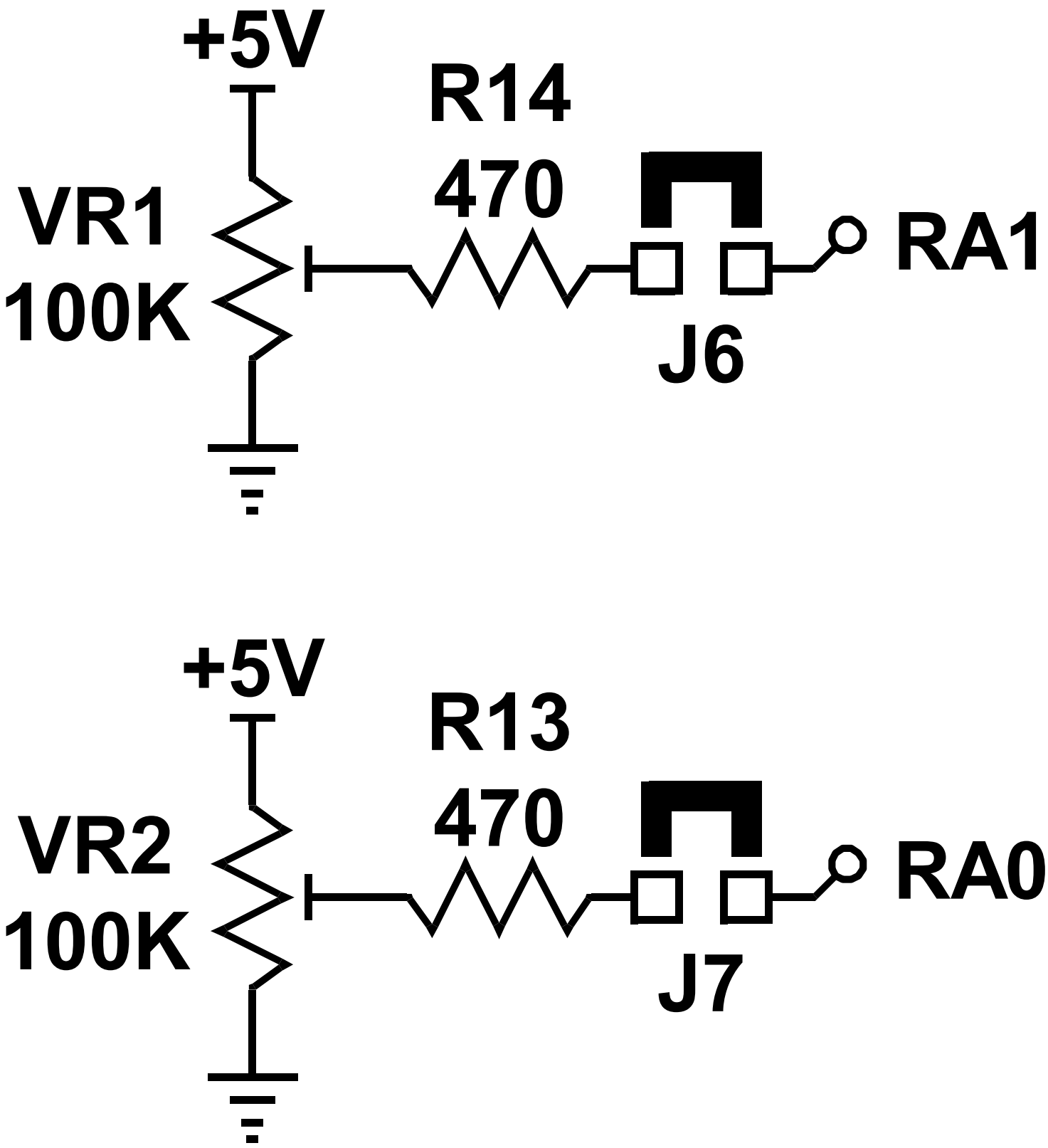

J1
1
RC0
2
RC1
3
RC2
4
RC3
5
RC4
6
RC5
7
RC6
8
RC7
J2
1
RD0
2
RD1
3
RD2
4
RD3
5
RD4
6
RD5
7
RD6
8
RD7
J4
1
RA0
2
RA1
3
RA2
4
RA3
5
RA4
6
RA5
J5
1
RB0
2
RB1
3
RB2
4
RB3
5
RB4

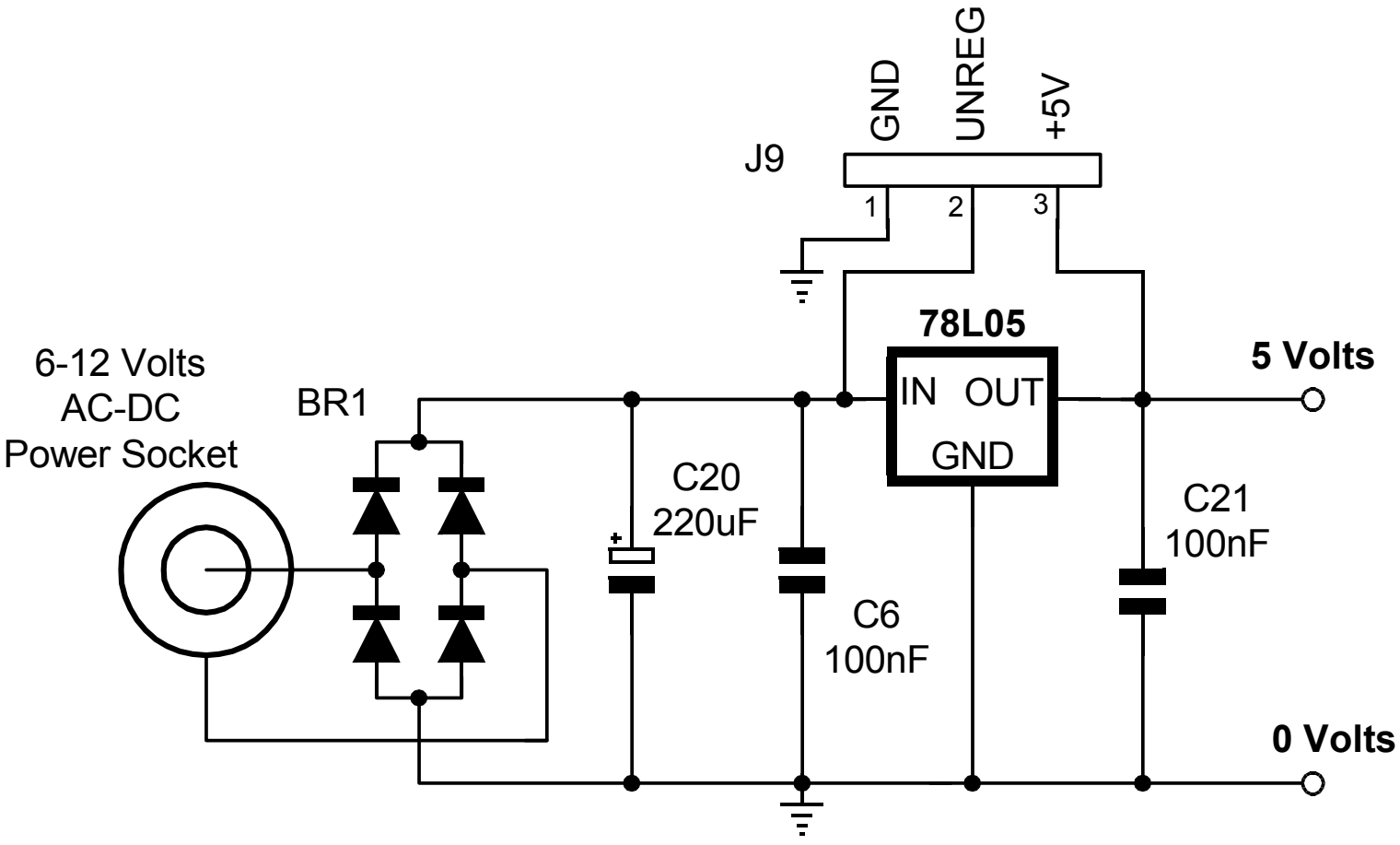

+5V
R7
4.7k
R6
4.7k
8
VCC
WP
7
5
SDA
RC4
6
SCL
RC3
24LC256
A0
A1
A2
1
2
3
VSS
4

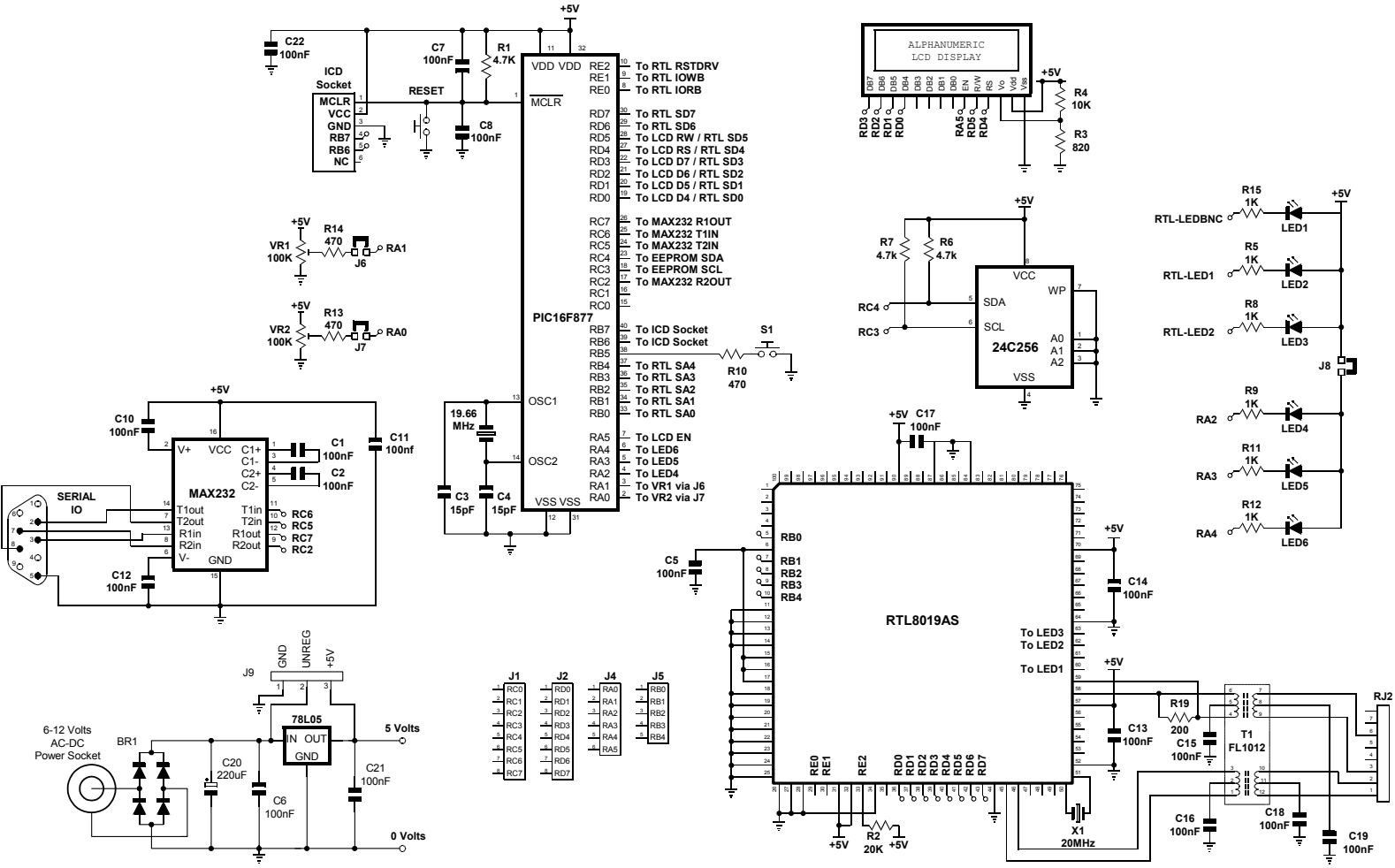

ALPHANUMERIC
LCD DISPLAY
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
EN
R/W
RS
Vo
Vdd
Vss
+5V
R4
10K
R3
820
RD3
RD2
RD1
RD0
RA5
RD5
RD4

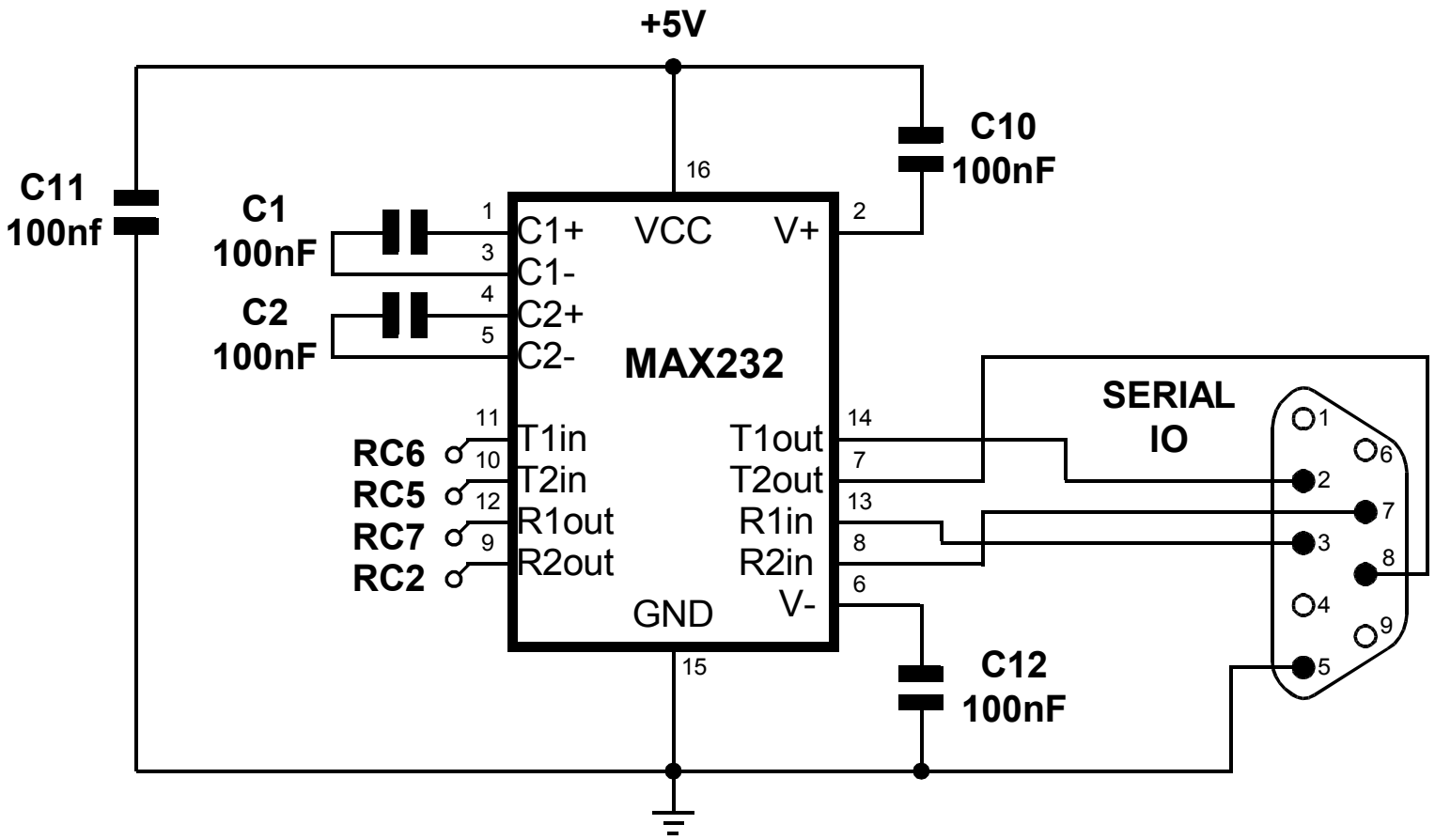

R15
1K
+5V
RTL-LEDBNC
LED1
R5
1K
RTL-LED1
LED2
R8
1K
RTL-LED2
LED3
J8
R9
1K
RA2
LED4
R11
1K
RA3
LED5
R12
1K
RA4
LED6

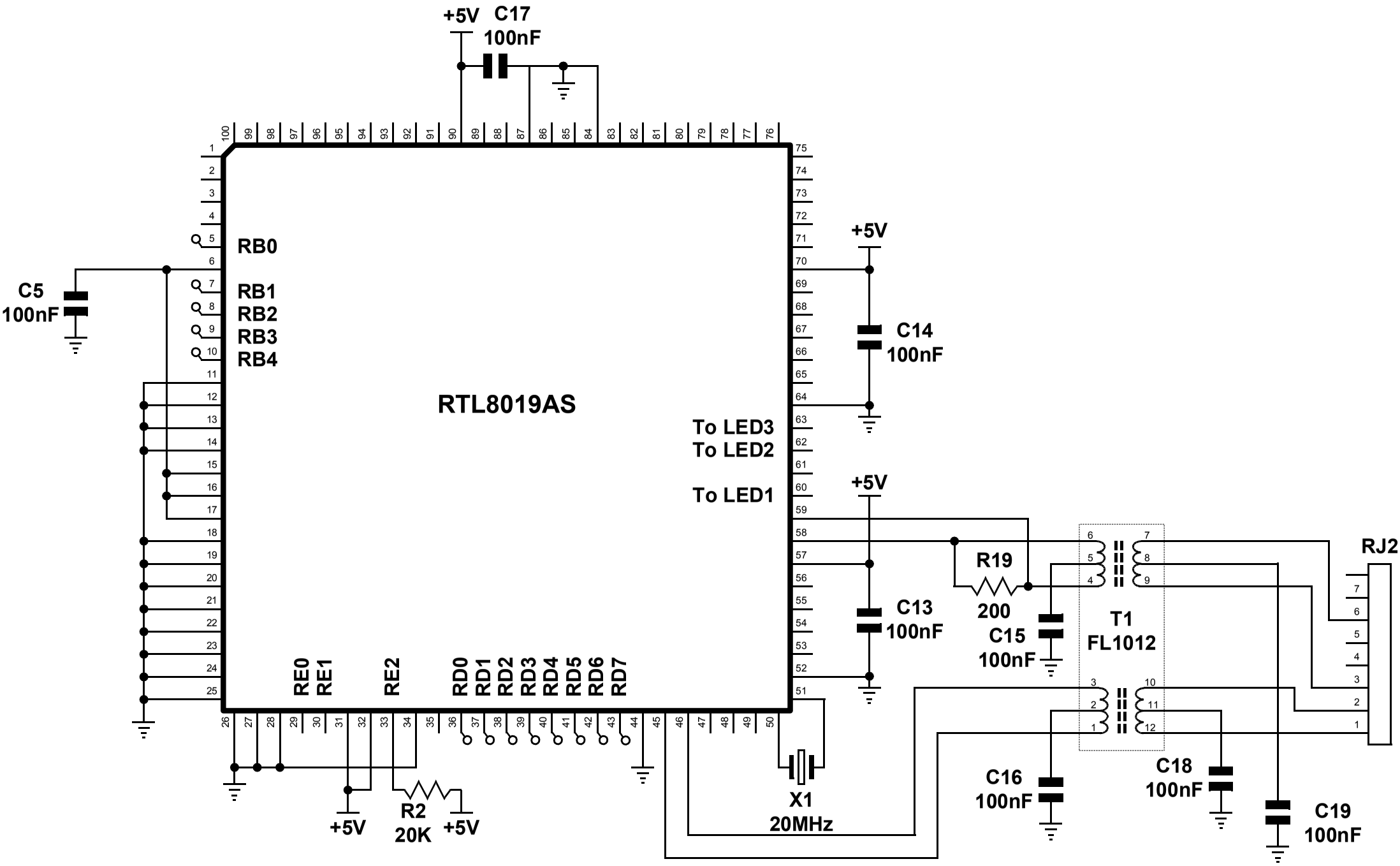

+5V
C22
100nF
C7
100nF
R1
4.7K
ICD
Socket
MCLR
VCC
GND
RB7
RB6
NC
RESET
C8
100nF
11
32
VDD VDD
1
MCLR
RE2
RE1
RE0
RD7
RD6
RD5
RD4
RD3
RD2
RD1
RD0
RC7
RC6
RC5
RC4
RC3
RC2
RC1
RC0
PIC16F877
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
RA5
RA4
RA3
RA2
RA1
RA0
13
OSC1
14
OSC2
12
31
VSS VSS
To RTL RSTDRV
To RTL IOWB
To RTL IORB
To RTL SD7
To RTL SD6
To LCD RW / RTL SD5
To LCD RS / RTL SD4
To LCD D7 / RTL SD3
To LCD D6 / RTL SD2
To LCD D5 / RTL SD1
To LCD D4 / RTL SD0
To MAX232 R1OUT
To MAX232 T1IN
To MAX232 T2IN
To EEPROM SDA
To EEPROM SCL
To MAX232 R2OUT
To ICD Socket
To ICD Socket
To RTL SA4
To RTL SA3
To RTL SA2
To RTL SA1
To RTL SA0
To LCD EN
To LED6
To LED5
To LED4
To VR1 via J6
To VR2 via J7
S1
R10
470
19.66
MHz
C3
15pF
C4
15pF


+5V
R14
470
VR1
100K
RA1
J6
+5V
R13
470
VR2
100K
RA0
J7


GND
UNREG
+5V
J9
1
2
3
78L05
IN
OUT
GND
5 Volts
0 Volts
6-12 Volts
AC-DC
Power Socket
BR1
C20
220uF
C6
100nF
C21
100nF


+5V
C22 100nF
C7 100nF
R1 4.7K
ICD Socket
MCLR
VCC
GND
RB7
RB6
NC
RESET
C8 100nF
PIC16F877
VDD VDD
MCLR
RE2 To RTL RSTDRV
RE1 To RTL IOWB
RE0 To RTL IORB
RD7 To RTL SD7
RD6 To RTL SD6
RD5 To LCD RW / RTL SD5
RD4 To LCD RS / RTL SD4
RD3 To LCD D7 / RTL SD3
RD2 To LCD D6 / RTL SD2
RD1 To LCD D5 / RTL SD1
RD0 To LCD D4 / RTL SD0
RC7 To MAX232 R1OUT
RC6 To MAX232 T1IN
RC5 To MAX232 T2IN
RC4 To EEPROM SDA
RC3 To EEPROM SCL
RC2 To MAX232 R2OUT
RC1
RC0
RB7 To ICD Socket
RB6 To ICD Socket
RB5
RB4 To RTL SA4
RB3 To RTL SA3
RB2 To RTL SA2
RB1 To RTL SA1
RB0 To RTL SA0
RA5 To LCD EN
RA4 To LED6
RA3 To LED5
RA2 To LED4
RA1 To VR1 via J6
RA0 To VR2 via J7
OSC1
OSC2
VSS VSS
19.66 MHz
C3 15pF
C4 15pF
S1
R10 470
VR1 100K
R14 470
J6
RA1
VR2 100K
R13 470
J7
RA0
ALPHANUMERIC LCD DISPLAY
D7 D6 D5 D4 D3 D2 D1 D0 R/W RS Vo Vdd Vss
RD3 RD2 RD1 RD0 RA5 RD5 RD4
R4 10K
R3 820
R7 4.7k
R6 4.7k
RC4
RC3
24C256
VCC
WP
SDA
SCL
A0
A1
A2
VSS
R15 1K
RTL-LEDBNC
LED1
R5 1K
RTL-LED1
LED2
R8 1K
RTL-LED2
LED3
J8
R9 1K
RA2
LED4
R11 1K
RA3
LED5
R12 1K
RA4
LED6
C10 100nF
C11 100nf
MAX232
V+ VCC C1+ C2+ C2- V-
T1out T2out R1in R2in
T1in T2in R1out R2out
GND
C1 100nF
C2 100nF
C12 100nF
RC6
RC5
RC7
RC2
SERIAL IO
C17 100nF
RB0
RB1
RB2
RB3
RB4
C5 100nF
RTL8019AS
To LED3
To LED2
To LED1
C14 100nF
C13 100nF
R19 200
C15 100nF
T1 FL1012
RJ2
C16 100nF
C18 100nF
C19 100nF
X1 20MHz
R2 20K
RE0 RE1 RE2 RD0 RD1 RD2 RD3 RD4 RD5 RD6 RD7
J1 RC0 RC1 RC2 RC3 RC4 RC5 RC6 RC7
J2 RD0 RD1 RD2 RD3 RD4 RD5 RD6 RD7
J4 RA0 RA1 RA2 RA3 RA4 RA5
J5 RB0 RB1 RB2 RB3 RB4
J9
GND UNREG +5V
78L05
IN OUT GND
5 Volts
0 Volts
6-12 Volts AC-DC Power Socket
BR1
C20 220uF
C6 100nF
C21 100nF


+5V
C11
100nf
C1
100nF
C2
100nF
C10
100nF
C12
100nF
MAX232
C1+
C1-
C2+
C2-
VCC
V+
GND
V-
T1in
T2in
R1out
R2out
T1out
T2out
R1in
R2in
RC6
RC5
RC7
RC2
SERIAL
IO
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


+5V
C17
100nF
RTL8019AS
RB0
RB1
RB2
RB3
RB4
C5
100nF
+5V
C14
100nF
To LED3
To LED2
To LED1
+5V
C13
100nF
R19
200
C15
100nF
T1
FL1012
RJ2
RE0
RE1
RE2
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
+5V
R2
20K
+5V
X1
20MHz
C16
100nF
C18
100nF
C19
100nF

Crownhill Associates
smart electronic solutions

Proton-NET & RJ-45 Cables

There are different grades, or categories, of twisted-pair cabling. Category 5 is the most reliable, the cables supplied with your Proton-NET development system are Category 5 rated. We recommend that you use Category 5 cables through out your network.

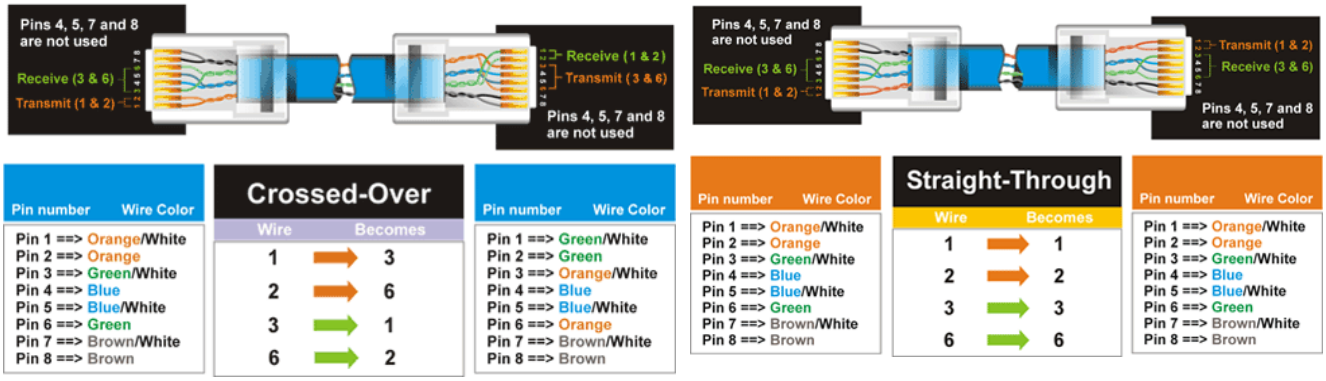
Two types of cable are supplied with your Proton-NET development system:

Straight-through, used for connecting to a hub.
Crossover, used for connecting to your computer (terminal).

Crownhill can supply pre-made Category 5 cabling, however you may prefer to cut and crimp your own. When ordering Category 5 cables, you will need to specify the length required and either straight-through or crossover cables.

The following notes will assist you in making your own cables:
A Category 5 cable has 8 thin, colour-coded wires inside that run from one end of the cable to the other. All 8 wires are used. In a straight-through cable, wires 1, 2, 3, and 6 at one end of the cable appear ai the same positions at the other end of the cable, i.e. 1, 2, 3, and 6. In a crossover cable, the order of the wires change from one end to the other: wire 1 appears as wire 3, and 2 appears as wire 6.
Two wire colour-code standards apply: EIA/TIA 568A and EIA/TIA 568B, so long as both ends of the cable use the same colour code, either colour code can be used.

It is important to note that you cannot reliably use a flat-untwisted telephone cable for your network cable. You must use twisted pairs to connect a set of transmitter pins to their corresponding receiver pins and you cannot use a wire from one pair and another wire from a different pair.



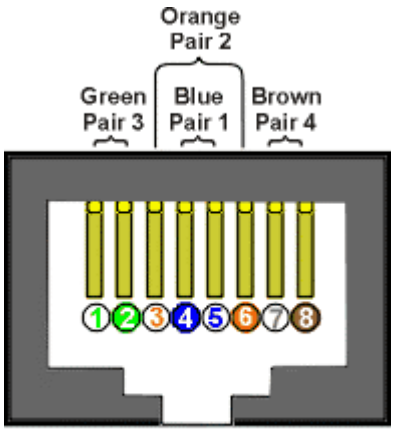
To determine which wire is wire number 1, hold the cable so that the end of the plastic RJ-45 tip (the part that goes into a wall jack first) is facing away from you. Face the clip down so that the copper side faces up (the springy clip will now be parallel to the floor).When looking down on the copper side, wire 1 will be on the far left.

RJ-45 Colour Chart

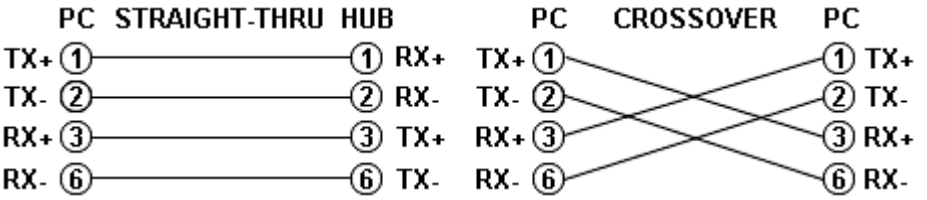
Wire 1	⇨	Orange / White
Wire 2	⇨	Orange
Wire 3	⇨	Green / White
Wire 4	⇨	Blue
Wire 5	⇨	Blue / White
Wire 6	⇨	Green
Wire 7	⇨	Brown / White
Wire 8	⇨	Brown

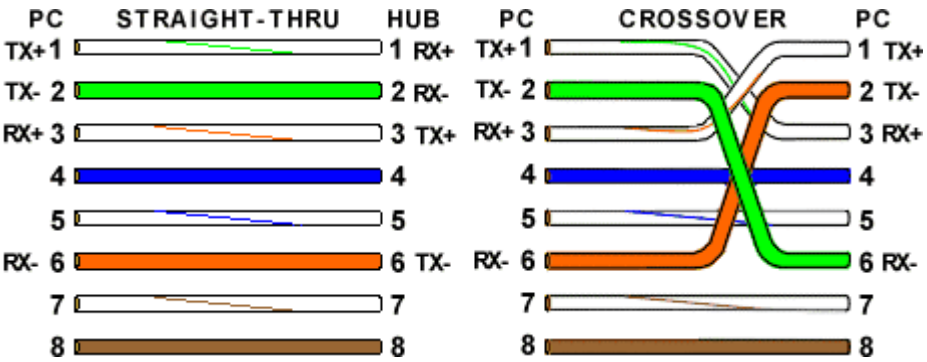
Proton-NET & RJ-45 Cables

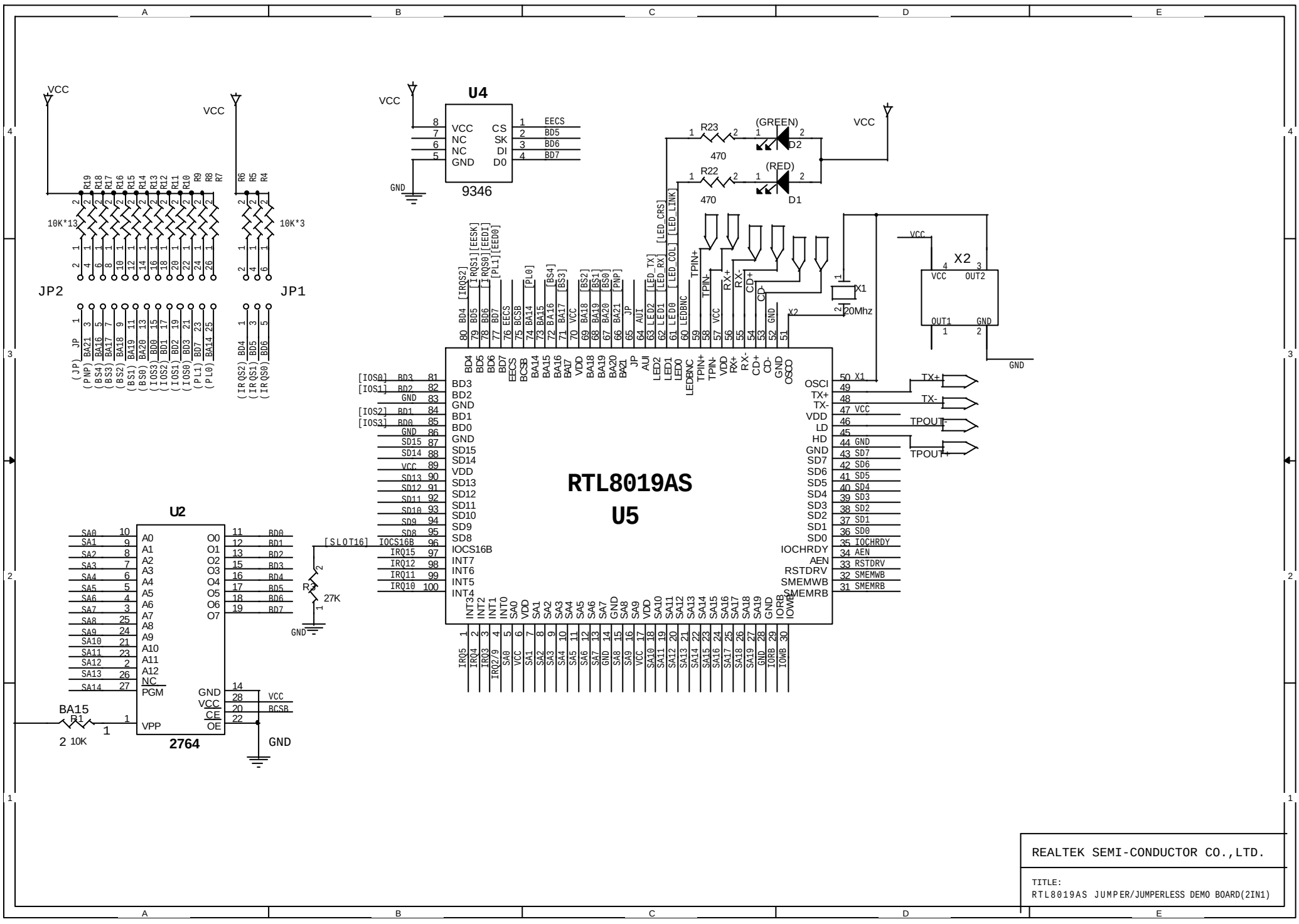
Alternative colour coding



RJ-45 JACK
EIA/TIA 568A STANDARD

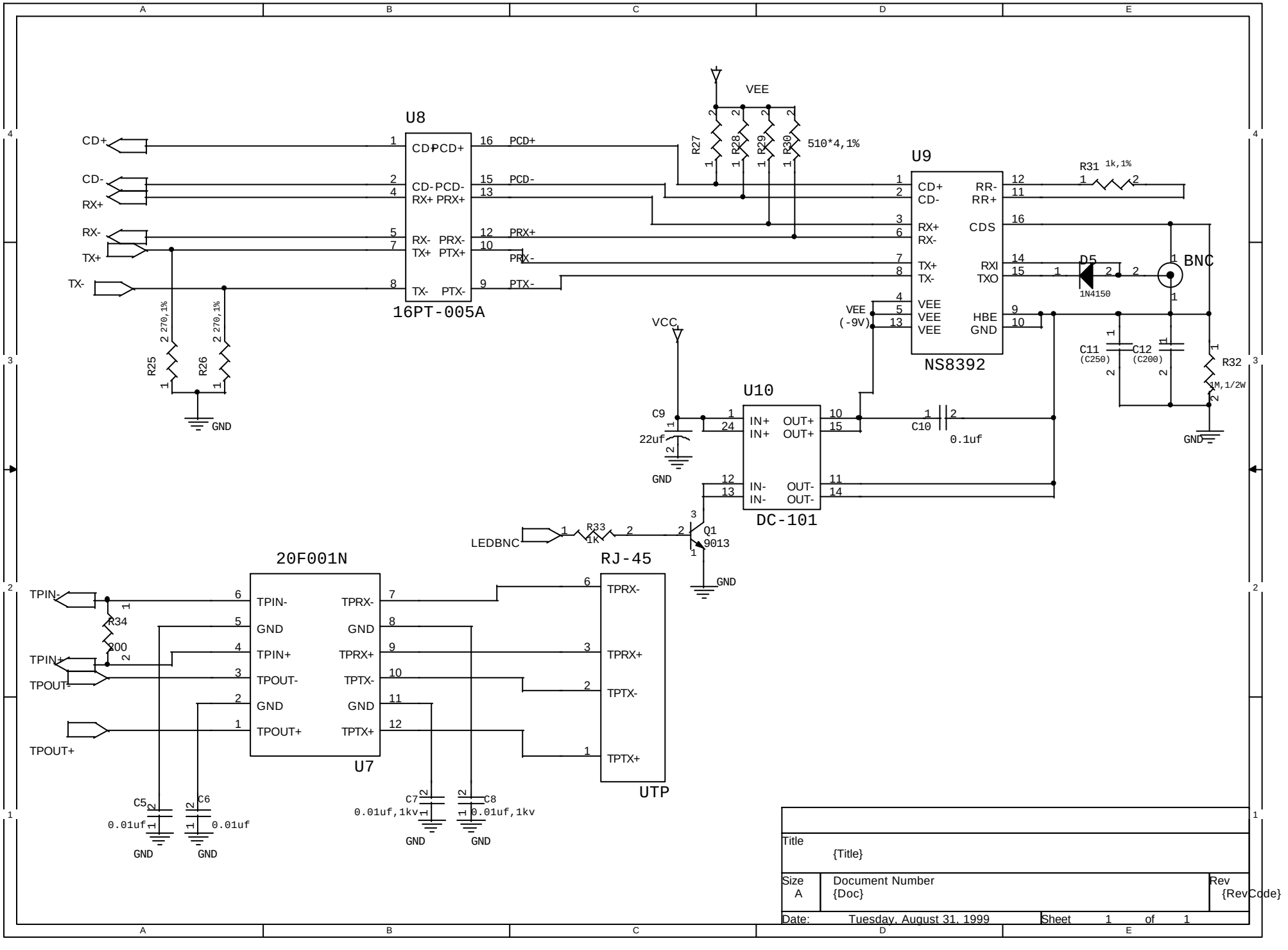






REALTEK SEMI-CONDUCTOR CO.,LTD.

TITLE:
RTL8019AS JUMPER/JUMPERLESS DEMO BOARD(2IN1)



Title	{Title}	
Size A	Document Number {Doc}	Rev {RevCode}
Date: Tuesday, August 31, 1999	Sheet 1 of 1	

RTL8019AS

Realtek Full-Duplex Ethernet Controller with Plug and Play Function (RealPNP)

SPECIFICATION

REALTEK SEMI-CONDUCTOR CO., LTD.

HEAD OFFICE

NO. 2, INDUSTRY E. RD. IX, SCIENCE-BASED
INDUSTRIAL PARK, HSINCHU 30077, TAIWAN, R.O.C.
TEL:886-3-5780211 FAX:886-3-5776598

OFFICE

3F, NO. 56, WU-KUNG 6 RD.,
TAIPEI HSIEN, TAIWAN, R.O.C.
TEL: 886-2-22980098 FAX: 886-2-22980094, 22980097

CONTENTS

1. FEATURES

- 100-pin PQFP
- RTL8019 software compatible
- Supports PnP auto detect mode (RTL8019AS only)
- Compliant to Ethernet II and IEEE802.3 10Base5, 10Base2, 10BaseT
- Software compatible with NE2000 on both 8 and 16-bit slots
- Supports both jumper and jumperless modes
- Supports Microsoft‘s Plug and Play configuration for jumperless mode
- Supports Full-Duplex Ethernet function to double channel bandwidth
- Supports three level power down modes:
  - Sleep
  - Power down with internal clock running
  - Power down with internal clock halted
- Built-in data prefetch function to improve performance
- Supports UTP, AUI & BNC auto-detect (RTL8019AS only)
- Supports auto polarity correction for 10BaseT
- Support 8 IRQ lines
- Supports 16 I/O base address options

  and extra I/O address fully decode mode (RTL8019AS only)
- Supports 16K, 32K, 64K and 16K-page mode access to BROM (up to 256 pages with 16K bytes/page)
- Supports BROM disable command to release memory after remote boot
- Supports flash memory read/write (RTL8019AS only)
- 16k byte SRAM built in (RTL8019AS only)
- Use 9346 (64*16-bit EEPROM) to store resource configurations and ID parameters
- Capable of programming blank 9346 on board for manufacturing convenience
- Support 4 diagnostic LED pins with programmable outputs

2. General Description

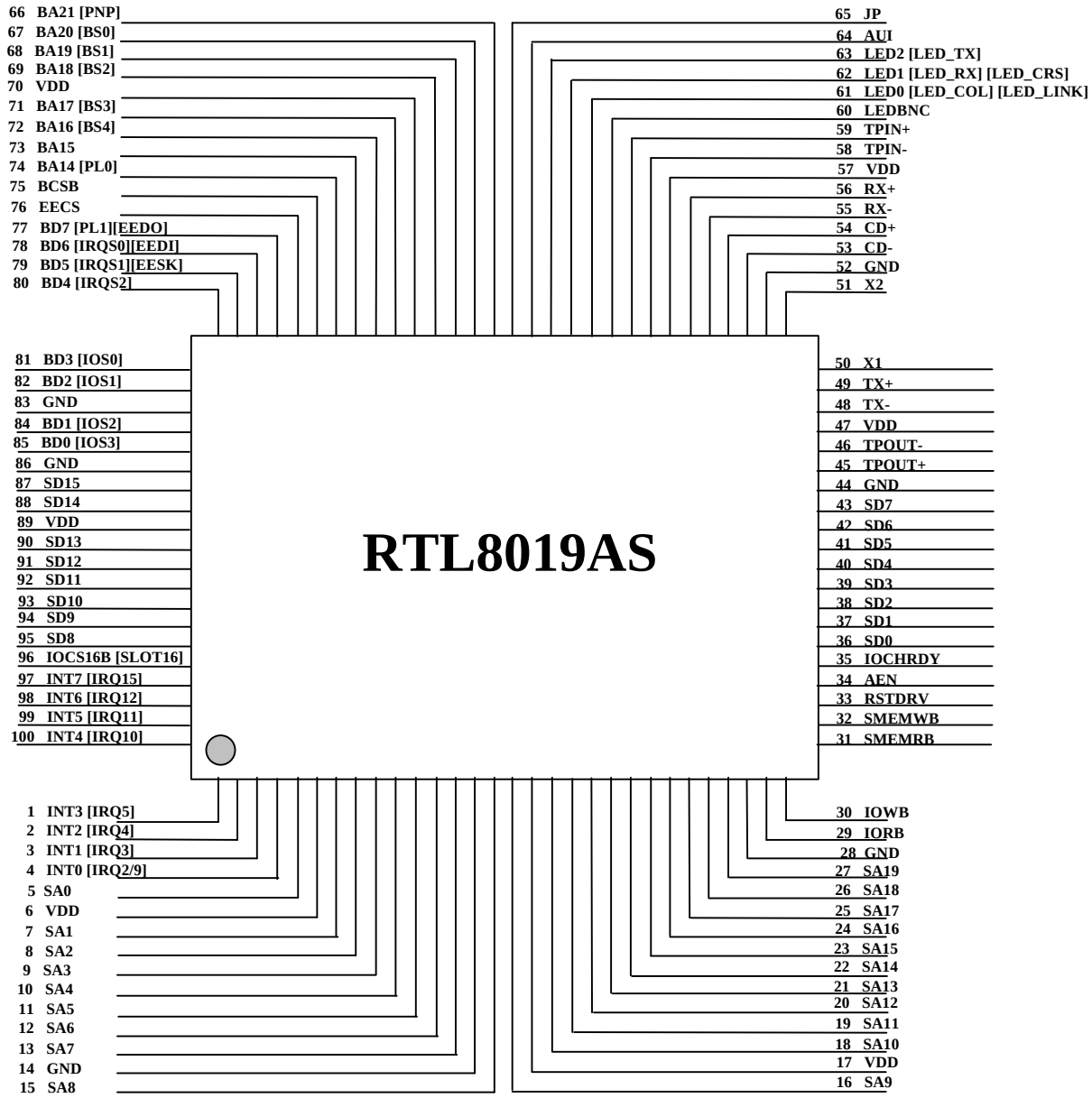
The RTL8019AS is a highly integrated Ethernet Controller which offers a simple solution to implement a Plug and Play NE2000 compatible adapter with full-duplex and power down features. With the three level power down control features, the RTL8019AS is made to be an ideal choice of the network device for a GREEN PC system. The full-duplex function enables simultaneously transmission and reception on the twisted-pair link to a full-duplex Ethernet switching hub. This feature not only increases the channel bandwidth from 10 to 20 Mbps but also avoids the performance degrading problem due to the channel contention characteristics of the Ethernet CSMA/CD protocol. The Microsoft's Plug and Play function can relieve the users from pains of taking care the adapter's resource configurations such as IRQ, I/O, and memory address, etc. However, for special applications not to be used as a Plug and Play compatible device, the RTL8019AS also supports the jumper and proprietary jumperless options.

To offer a fully *plug and play* solution, the RTL8019AS provides the auto-detect capability between the integrated 10BaseT transceiver, BNC and AUI interface. Besides, the 10BaseT transceiver can automatically correct the polarity error on its receiving pair. Furthermore, 8 IRQ lines and 16 I/O base address options are provided for grand resource configuration flexibility.

The RTL8019AS supports 16k, 32k & 64k byte BROM and fiash memory interface. It also offers the page mode function which can support up to 4M-byte BROM within only 16k-byte system memory space. Besides, the BROM disable command is provided to release the BROM memory space for other system usage (e.g. EMM386, etc.) after the BROM program is loaded.

The RTL8019AS is built in with 16K-byte SRAM in a single chip. It is designed not only to provide more friendly functions but also to save the effort of SRAM sourcing and inventory.

3. PIN CONFIGURATION



4. PIN DESCRIPTIONS

4.1. Power Pins

No.	Name	Type	Description
6, 17, 47, 57, 70, 89	VDD	P	+5V DC power
14, 28, 44, 52, 83, 86	GND	P	Ground

4.2. ISA Bus Interface Pins

No.	Name	Type	Descriptions
34	AEN	I	Address Enable. This ISA signal must be low for a valid I/O command.
97-100, 1-4	INT7-0	O	Interrupt request lines which are mapped to IRQ15, IRQ12, IRQ11, IRQ10, IRQ5, IRQ4, IRQ3, IRQ2/9 respectively. Only one line is selected to reflect the interrupt requests at one time. All other lines are tri-stated. The RTL8019AS also uses these pins as inputs to monitor the actual state of the corresponding interrupt lines on ISA bus. The result is recorded in the INTR register, which may be used by software to detect interrupt conflict.
35	IOCHRDY	O	This ISA signal is driven low to insert wait cycles to current host read/write command.
96	IOCS16B [SLOT16]	O	Upon power-on reset, this pin acts as an input named SLOT16 to detect whether a 16-bit or 8-bit slot is in use. To do this, it is connected to a pull-down resistor (about 27KW) externally. At the falling edge of RSTDRV, the RTL8019AS senses this pin's state. If it is sensed high, the adapter is thought to be placed on a 16-bit slot where this pin is connected to the host's IOCS16B pin, which is typically pulled up by a 300W resistor on the mother board. If it is sensed low, the adapter is thought to be placed on an 8-bit slot where this pin is merely pulled low by the 27KW resistor. After having latched the input state, this pin is switched as the IOCS16B signal which is an open-drain output and is driven low during a 16-bit host data transfer. It is decoded from AEN and SA9-0.
29	IORB	I	Host I/O read command.
30	IOWB	I	Host I/O write command.
33	RSTDRV	I	High active hardware reset signal from the ISA bus. Pulses with high level less than 800ns are ignored.
27-18, 16-15, 13-7, 5	SA19-0	I	Host address bus. SA10 is added to implement the fully decode of PnP ports, address 279h and A79h. In RTL8019, SA10 is not decoded. In RTL8019AS, SA10 should be 0 for a valid access to PnP ports.
87-88, 90-95, 43-36	SD15-0	I/O	Host data bus.

31	SMEMRB	I	Host memory read command.
32	SMEMWB	I	Host memory write command. This pin is added to decode the write command of a flash memory.

4.3. Memory Interface Pins (including BROM, EEPROM)

No.	Name	Type	Description
75	BCSB	O	BROM chip select. Active low signal, asserted when BROM is read. RTL8019AS drives this pin low when SA19-14 matches the selected BROM memory base address and either of the 2 conditions below meets: (1) SMEMRB is low (2) SMEMWB is low and RTL8019AS's flash memory write function is enabled.
76	EECS	O	9346 chip select. Active high signal, asserted when 9346 is read/write.
66-69, 71-74	BA21-14	O	BROM address.
77-82, 84-85	BD7-0	I/O	BROM data bus.
[79]	[EESK]	O	9346 serial data clock
[78]	[EEDI]	O	9346 serial data input
[77]	[EEDO]	I	9346 serial data output
[66]	[PNP]	I	The following pins are defined for jumper options. Their states are latched at the falling edge of RSTDRV, then they are changed to serve as the SRAM bus. Each of them is internally pulled down by a 100KW resistor. Therefore, the input will be low when left open and high when pulled up by a 10K resistor externally. When it is high in jumperless mode (i.e. JP=low), the RTL8019AS is forced into Plug and Play mode regardless of the contents of 9346. The following pins are don't care in jumperless mode (JP=low).
[72-71, 69-67]	[BS4-0]	I	Select BROM size and base address.
[85-84, 82-81]	[IOS3-0]	I	Select I/O base address.
[77, 74]	[PL1-0]	I	Select network medium type.
[80-78]	[IRQS2-0]	I	Select one interrupt line among INT7-0.
65	JP	I	When high, this pin selects jumper mode. When low, it selects jumperless modes (including RT jumperless and Plug and Play).

After RTL8019AS latches all jumper status upon power on reset, these pins ***always**** reflect the value of BPAGE register directly in BROM page mode. In normal mode, BA16-21 are not used and BA14-15 act as:

BROM Size	BA14	BA15
16K	high	high
32K	SA14	high
64K	SA14	SA15

*Note: RTL8019AS doesn't drive BA14-21 until the SMEMRB goes from high to low.

4.4. Medium Interface Pins

No.	Name	Type	Description
64	AUI	I	This input is used to detect the usage of an external MAU on the AUI interface. The input should be driven low for embedded BNC and high for external MAU. When the input is high, RTL8019AS sets the AUI bit (bit5) in CONFIG0 and drives LEDBNC low to disable the BNC. If this pin is not used, it should be connected to GND such that RTL8019AS acts like RTL8019. Please refer to section 5.1.2.2. CONFIG0 for more details.
54,53	CD+,CD-	I	This AUI collision input pair carries the differential collision input signal from the MAU.
56,55	RX+,RX-	I	This AUI receive input pair carries the differential receive input signal from the MAU.
49,48	TX+,TX-	O	This AUI transmit output pair contains differential line drivers which send Manchester encoded data to the MAU. These outputs are source followers and require 270 ohm pull-down resistors to GND.
59,58	TPIN+, TPIN-	I	This TP input pair receives the 10 Mbits/s differential Manchester encoded data from the twisted-pair wire.
45,46	TPOUT+, TPOUT-	O	This pair carries the differential TP transmit output. The output Manchester encoded signals have been pre-distorted to prevent overcharge on the twisted-pair media and thus reduce jitter.
50	X1	I	20Mhz crystal or external oscillator input.
51	X2	O	Crystal feedback output. This output is used in crystal connection only. It must be left open when X1 is driven with an external oscillator.

4.5. LED Output Pins

No.	Name	Type	Description
60	LEDBNC	O	This pin goes high when RTL8019AS's medium type is set to 10Base2 mode or auto-detect mode with link test failure. Otherwise, this pin is low. This pin can be used to control the power of the DC convertor for CX MAU and connected to an LED to indicate the used medium type.
61	LED0	O	When LEDS0 bit (in CONFIG3 register of RTL8019AS Page3) is 0, this pin acts as LED_COL. When LEDS0=1, it acts as LED_LINK.

62,63	LED1,LED2	O	When LEDS1 bit (in CONFIG3 register of RTL8019AS Page3) is 0, these 2 pins act as LED_RX & LED_TX respectively. When LEDS1=1, these pins act as LED_CRS & MCSB. Please refer to section 6.5 for details of the lightening behavior of all LEDs.

5. Register Descriptions

The registers in RTL8019AS can be roughly divided into two groups by their address and functions -- one for NE2000, the other for Plug and Play (PnP).

5.1. Group 1: NE2000 Registers

This group includes 4 pages of registers which are selected by bit PS0 & PS1 in the CR register. Each page contains 16 registers. Besides those registers compatible with NE2000, the RTL8019AS defines some registers for software configuration and feature enhancement.

5.1.1. Register Table

No (Hex)	Page0		Page1	Page2	Page3	
	[R]	[W]	[R/W]	[R]	[R]	[W]
00	CR	CR	CR	CR	CR	CR
01	CLDA0	PSTART	PAR0	PSTART	***9346CR***	***9346CR***
02	CLDA1	PSTOP	PAR1	PSTOP	***BPAGE***	***BPAGE***
03	BNRY	BNRY	PAR2	-	***CONFIG0***	-
04	TSR	TPSR	PAR3	TPSR	***CONFIG1***	***CONFIG1***
05	NCR	TBCR0	PAR4	-	***CONFIG2***	***CONFIG2***
06	FIFO	TBCR1	PAR5	-	***CONFIG3***	***CONFIG3***
07	ISR	ISR	CURR	-	-	***TEST***
08	CRDA0	RSAR0	MAR0	-	***CSNSAV***	-
09	CRDA1	RSAR1	MAR1	-	-	***HLTCLK***
0A	***8019ID0***	RBCR0	MAR2	-	-	-
0B	***8019ID1***	RBCR1	MAR3	-	***INTR***	-
0C	RSR	RCR	MAR4	RCR	-	***FMWP***
0D	CNTR0	TCR	MAR5	TCR	***CONFIG4***	-
0E	CNTR1	DCR	MAR6	DCR	-	-
0F	CNTR2	IMR	MAR7	IMR	-	-
10-17	Remote DMA Port					
18-1F	Reset Port					

Notes: "-" denotes reserved. Registers with names typed in ***bold italic*** format are RTL8019AS defined registers and are not supported in a standard NE2000 adapter.

Page 0 (PS1=0, PS0=0)

No.	Name	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	CLDA0	R	A7	A6	A5	A4	A3	A2	A1	A0
	PSTART	W	A15	A14	A13	A12	A11	A10	A9	A8
02H	CLDA1	R	A15	A14	A13	A12	A11	A10	A9	A8
	PSTOP	W	A15	A14	A13	A12	A11	A10	A9	A8
03H	BNRY	R/W	A15	A14	A13	A12	A11	A10	A9	A8
04H	TSR	R	OWC	CDH	0	CRS	ABT	COL	-	PTX
	TPSR	W	A15	A14	A13	A12	A11	A10	A9	A8
05H	NCR	R	0	0	0	0	NC3	NC2	NC1	NC0
	TBCR0	W	TBC7	TBC6	TBC5	TBC4	TBC3	TBC2	TBC1	TBC0
06H	FIFO	R	D7	D6	D5	D4	D3	D2	D1	D0
	TBCR1	W	TBC15	TBC14	TBC13	TBC12	TBC11	TBC10	TBC9	TBC8
07H	ISR	R/W	RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX
08H	CRDA0	R	A7	A6	A5	A4	A3	A2	A1	A0
	RSAR0	W	A7	A6	A5	A4	A3	A2	A1	A0
09H	CRDA1	R	A15	A14	A13	A12	A11	A10	A9	A8
	RSAR1	W	A15	A14	A13	A12	A11	A10	A9	A8
0AH	8019ID0	R	0	1	0	1	0	0	0	0
	RBCR0	W	RBC7	RBC6	RBC5	RBC4	RBC3	RBC2	RBC1	RBC0
0BH	8019ID1	R	0	1	1	1	0	0	0	0
	RBCR1	W	RBC15	RBC14	RBC13	RBC12	RBC11	RBC10	RBC9	RBC8
0CH	RSR	R	DFR	DIS	PHY	MPA	0	FAE	CRC	PRX
	RCR	W	-	-	MON	PRO	AM	AB	AR	SEP
0DH	CNTR0	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	TCR	W	-	-	-	OFST	ATD	LB1	LB0	CRC
0EH	CNTR1	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	DCR	W	-	FT1	FT0	ARM	LS	LAS	BOS	WTS
0FH	CNTR2	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	IMR	W	-	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Page 1 (PS1=0, PS0=1)

No.	Name	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	PAR0	R/W	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
02H	PAR1	R/W	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
03H	PAR2	R/W	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
04H	PAR3	R/W	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
05H	PAR4	R/W	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
06H	PAR5	R/W	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40
07H	CURR	R/W	A15	A14	A13	A12	A11	A10	A9	A8
08H	MAR0	R/W	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
09H	MAR1	R/W	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
0AH	MAR2	R/W	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
0BH	MAR3	R/W	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
0CH	MAR4	R/W	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
0DH	MAR5	R/W	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
0EH	MAR6	R/W	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
0FH	MAR7	R/W	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

Page 2(PS1=1, PS0=0)

No.	Name	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	PSTART	R	A15	A14	A13	A12	A11	A10	A9	A8
02H	PSTOP	R	A15	A14	A13	A12	A11	A10	A9	A8
03H	-									
04H	TPSR	R	A15	A14	A13	A12	A11	A10	A9	A8
05H | 0BH	-									
0CH	RCR	R	-	-	MON	PRO	AM	AB	AR	SEP
0DH	TCR	R	-	-	-	OFST	ATD	LB1	LB0	CRC
0EH	DCR	R	-	FT1	FT0	ARM	LS	LAS	BOS	WTS
0FH	IMR	R	-	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Page 3(PS1=1, PS0=1)

No.	Name	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	9346CR	R	EEM1	EEM0	-	-	EECS	EESK	EEDI	EEDO
		W	EEM1	EEM0	-	-	EECS	EESK	EEDI	-
02H	BPAGE	R/W	BP7	BP6	BP5	BP4	BP3	BP2	BP1	BP0
03H	CONFIG0	R	*VerID1*	*VerID0*	*AUI*	*PNPJP*	JP	BNC	0	0
04H	CONFIG1	R	IRQEN	IRQS2	IRQS1	IRQS0	IOS3	IOS2	IOS1	IOS0
		W*	IRQEN	-	-	-	-	-	-	-
05H	CONFIG2	R	PL1	PL0	BSELB	BS4	BS3	BS2	BS1	BS0
		W*	PL1	PL0	BSELB	-	-	-	-	-
06H	CONFIG3	R	PNP	FUDUP	LEDS1	LEDS0	-	SLEEP	PWRDN	ACTIVEB
		W*	-	-	-	-	-	SLEEP	PWRDN	-
07H	TEST	R/W	Reserved, Do not write							
08H	CSNSAV	R	CSN7	CSN6	CSN5	CSN4	CSN3	CSN2	CSN1	CNS0
09H	HLTCLK	W	HLT7	HLT6	HLT5	HLT4	HLT3	HLT2	HLT1	HLT0
0AH	-	-	Reserved							
0BH	INTR	R	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
0CH	*FMWP*	*W**	*Flash Memory Write Protect*							
0DH	CONFIG4	R	-	-	-	-	-	-	-	IOMS
0EH | 0FH	-	-	Reserved							

Note: The registers marked with type='W*' can be written only if bits EEM1=EEM0=1.

5.1.2. Register Functions

5.1.2.1. NE2000 Compatible Registers

CR: Command Register (00H; Type=R/W)
This register is used to select register pages, enable or disable remote DMA operation and issue commands.

Bit	Symbol	Description
7, 6	PS1, PS0	
5-3	RD2-0	
2	TXP	This bit must be set to transmit a packet. It is internally reset either after the transmission is completed or aborted. Writing a 0 has no effect.
1	STA	The STA bit controls nothing. It only reflects the value written to this bit. POWER UP=0.
0	STP	This bit is the STOP command. When it is set, no packets will be received or transmitted. POWER UP=1.

PS1	PS0	Register Page	Remark
0	0	0	NE2000 compatible
0	1	1	NE2000 compatible
1	0	2	NE2000 compatible
1	1	3	RTL8019AS Configuration

RD2	RD1	RD0	Function
0	0	0	Not allowed
0	0	1	Remote Read
0	1	0	Remote Write
0	1	1	Send Packet
1	*	*	Abort/Complete remote DMA

STA	STP	Function
1	0	Start Command
0	1	Stop Command

ISR: Interrupt Status Register (07H; Type=R/W in Page0)

This register reflects the NIC status. The host reads it to determine the cause of an interrupt. Individual bits are cleared by writing a "1" into the corresponding bit. It must be cleared after power up.

Bit	Symbol	Description
7	RST	This bit is set when NIC enters reset state and is cleared when a start command is issued to the CR. It is also set when receive buffer overflows and is cleared when one or more packets have been read from the buffer.
6	RDC	Set when remote DMA operation has been completed.
5	CNT	Set when MSB of one or more of the network tally counters has been set.
4	OVW	This bit is set when the receive buffer has been exhausted.
3	TXE	Transmit error bit is set when a packet transmission is aborted due to excessive collisions.
2	RXE	This bit is set when a packet received with one or more of the following errors: - CRC error - Frame alignment error -Missed packet
1	PTX	This bit indicates packet transmitted with no errors.
0	PRX	This bit indicates packet received with no errors.

IMR: Interrupt Mask Register (0FH; Type=W in Page0, Type=R in Page2)
All bits correspond to the bits in the ISR register. POWER UP=all 0s. Setting individual bits will enable the corresponding interrupts.

DCR: Data Configuration Register (0EH; Type=W in Page0, Type=R in Page2)

Bit	Symbol	Description
7	-	Always 1
6, 5	FT1, FT0	FIFO threshold select bit 1 and 0.
4	ARM	Auto-initialize Remote 0: Send Packet Command not executed. 1: Send Packet Command executed.
3	LS	Loopback Select 0: Loopback mode selected. Bits 1 and 2 of the TCR must also be programmed for Loopback operation. 1: Normal Operation
2	LAS	This bit must be set to zero. NIC only supports dual 16-bit DMA mode. POWER UP =1
1	BOS	Byte Order Select (Not implement) 0: MS byte placed on MD15-8 and LS byte on MD7-0. (32xxx,80x86) 1: MS byte placed on MD7-0 and LS byte on MD15-8. (680x0)
0	WTS	Word Transfer Select 0: byte-wide DMA transfer 1: word-wide DMA transfer

TCR: Transmit Configuration Register (0DH; Type=W in Page0, Type=R in Page2)

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th></tr>
<tr><td>7</td><td>-</td><td>Always 1</td></tr>
<tr><td>6</td><td>-</td><td>Always 1</td></tr>
<tr><td>5</td><td>-</td><td>Always 1</td></tr>
<tr><td>4</td><td>OFST</td><td>Collision Offset Enable.</td></tr>
<tr><td>3</td><td>ATD</td><td>Auto Transmit Disable.
0: normal operation
1: reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.</td></tr>
<tr><td>2, 1</td><td>LB1, LB0</td><td>
<table>
<tr><th>LB1</th><th>LB0</th><th>Mode</th><th>Remark</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Normal Operation</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Internal Lookback</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>External Lookback</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>External Lookback</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>CRC</td><td>The NIC CRC logic comprises a CRC generator for transmitter and a CRC checker for receiver. This bit controls the activity of the CRC logic. If this bit set, CRC is inhibited by transmitter. Otherwise CRC is appended by transmitter.
<table>
<tr><th colspan="2">Conditions</th><th colspan="2">CRC Logic Activities</th></tr>
<tr><th>CRC Bit</th><th>Mode</th><th>CRC Generator</th><th>CRC Checker</th></tr>
<tr><td>0</td><td>normal</td><td>enabled</td><td>enabled</td></tr>
<tr><td>1</td><td>normal</td><td>disabled</td><td>enabled</td></tr>
<tr><td>0</td><td>loopback</td><td>enabled</td><td>disabled</td></tr>
<tr><td>1</td><td>loopback</td><td>disabled</td><td>enabled</td></tr>
</table>
</td></tr>
</table>

TSR: Transmit Status Register (04H; Type=R in Page0)

This register indicates the status of a packet transmission.

Bit	Symbol	Description
7	OWC	Out of Window Collision. It is set when a collision is detected after a slot time (51.2us). Transmissions are rescheduled as in normal collisions.
6	CDH	CD Heartbeat. The NIC watches for a collision signal (i.e. CD Heartbeat signal) during the first 6.4us of the interframe gap following a transmission. This bit is set if the transceiver fails to send this signal.
5	-	Always 1.
4	CRS	Carrier Sense lost bit is set when the carrier is lost during transmitting a packet.
3	ABT	It indicates the NIC aborted the transmission because of excessive collisions.
2	COL	It indicates the transmission collided with some other station on the network.
1	-	Always 1
0	PTX	This bit indicates the transmission completes with no errors.

RCR: Receive Configuration Register (0CH; Type=W in Page0, Type=R in Page2)

Bit	Symbol	Description
7	-	Always 1
6	-	Always 1
5	MON	When monitor mode bit is set, received packets are checked for address match, good CRC and frame alignment but not buffered to memory. Otherwise, packets will be buffered to memory.
4	PRO	If PRO=1, all packets with physical destination address accepted. If PRO=0, physical destination address must match the node address programmed in PAR0-5.
3	AM	If AM=1, packets with multicast destination address are accepted. If AM=0, packets with multicast destination address are rejected.
2	AB	If AB=1, packets with broadcast destination address are accepted. If AB=0, packets with broadcast destination address are rejected.
1	AR	If AR=1, packets with length fewer than 64 bytes are accepted. If AR=0, packets with length fewer than 64 bytes are rejected.
0	SEP	If SEP=1, packets with receive errors are accepted. If SEP=0, packets with receive errors are rejected.

RSR: Receive Status Register (0CH; Type=R in Page0)

Bit	Symbol	Description
7	DFR	Defferring. Set when a carrier or a collision is detected.
6	DIS	Receiver Disabled. When the NIC enters the monitor mode, this bit is set and receiver is disabled. Reset when receiver is enabled after leaving the monitor mode.
5	PHY	PHY bit is set when the received packet has a multicast or broadcast destination address. It is reset when the received packet has a physical destination address.
4	MPA	Missed Packet bit is set when the incoming packet can not be accepted by NIC because of a lack of receive buffer or if NIC is in monitor mode. Increment CNTR2 tally counter.
3	-	Always 1.
2	FAE	Frame Alignment Error bit reflects the incoming packet didn't end on a byte boundary and CRC did not match at last byte boundary. Increment CNTR0 tally counter.
1	CRC	CRC error bit reflects packet received with CRC error. This bit will also be set for FAE errors. Increment CNTR1 tally counter.
0	PRX	This bit indicates packet received with no errors.

CLDA0, 1: Current Local DMA Registers (01H & 02H; Type=R in Page0)

These two registers can be read to get the current local DMA address.

PSTART: Page Start Register (01H; Type=W in Page0, Type=R in Page 2)
The Page Start register sets the start page address of the receive buffer ring.

PSTOP: Page Stop Register (02H; Type=W in Page0, Type=R in Page2)
The Page Stop register sets the stop page address of the receive buffer ring. In 8 bit mode the PSTOP register should not exceed to 0x60, in 16 bit mode the PSTOP register should not exceed to 0x80.

BNRY: Boundary Register (03H; Type=R/W in Page0)
This register is used to prevent overwrite of the receive buffer ring. It is typically used as a pointer indicating the last receive buffer page the host has read.

TPSR: Transmit Page Start Register (04H; Type=W in Page0)
This register sets the start page address of the packet to the transmitted.

TBCR0,1: Transmit Byte Count Registers (05H & 06H; Type=W in Page0)
These two registers set the byte counts of the packet to be transmitted.

NCR: Number of Collisions Register (05H; Type=R in Page0)
The register records the number of collisions a node experiences during a packet transmission.

FIFO: First In First Out Register (06H; Type=R in Page0)
This register allows the host to examine the contents of the FIFO after loopback.

CRDA0, 1: Current Remote DMA Address registers (08H & 09H; Type=R in Page0)
These two registers contain the current address of remote DMA.

RSAR0,1: Remote Start Address Registers (08H & 09H; Type=W in Page0)
These two registers set the start address of remote DMA.

RBCR0,1: Remote Byte Count Registers (0AH & 0BH; Type=W in Page0)
These two registers se the data byte counts of remote DMA.

CNTR0: Frame Alignment Error Tally Counter Register (0DH; Type=R in Page0)

CNTR1: CRC Error Tally Counter Register (0EH; Type=R in Page0)

CNTR2: Missed Packet Tally Counter Register (0FH; Type=R in Page0)

PAR0-5: Physical Address Registers (01H - 06H; Type=R/W in Page1)
These registers contain my Ethernet node address and are used to compare the destination adderss of incoming packets for acceptation or rejection.

CURR: Current Page Register (07H; Type=R/W in Page1)
This register points to the page address of the first receive buffer page to be used for a packet reception.

MAR0-7: Multicast Address Register (08H - 0FH; Type=R/W in Page1)
These registers provide filtering bits of multicast addresses hashed by the CRC logic.

5.1.2.2. RTL8019AS Defined Registers

Page 0 (PS1=0, PS0=0)

Two registers are defined to contain the RTL8019AS chip ID.

No.	Name	Type	Bit7-0
0AH	8019ID0	R	50H (ASCII code of "P")
0BH	8019ID1	R	70H (ASCII code of "p")

Page 3(PS1=1, PS0=1)

Page3 Power Up Values before loading jumper states and 9346 contents

No.	Name	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	0	0	1	0	0	0	0	1
01H	9346CR	R/W	0	0	-	-	*	*	*	*
02H	BPAGE	R/W	0	0	0	0	0	0	0	0
03H	CONFIG0	R/W	0	0	*	*	*	*	0	0
04H	CONFIG1	R/W	1	*	*	*	*	*	*	*
05H	CONFIG2	R/W	*	*	0	*	*	*	*	*
06H	CONFIG3	R/W	*	*	*	*	*	0	0	1
07H	TEST	R/W	-	-	-	-	-	-	-	-
08H	CSNSAV	R	0	0	0	0	0	0	0	0
09H	HLTCLK	W	1	1	1	1	1	1	1	1
0AH	-									
0BH	INTR	R	*	*	*	*	*	*	*	*
0CH	FMWP	W								
0DH	CONFIG4	R	-	-	-	-	-	-	-	*
0EH | 0FH	-									

Page3 Content Descriptions

9346CR: 9346 Command Register (01H; Type=R/W except Bit0=R)

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th></tr>
<tr><td>7-6</td><td>EEM1-0</td><td>These 2 bits select the RTL8019AS operating mode.
<table>
<tr><th>EEM1</th><th>EEM0</th><th>Operating Mode</th></tr>
<tr><td>0</td><td>0</td><td>Normal (DP8390 compatible)</td></tr>
<tr><td>0</td><td>1</td><td>Auto-load:
Entering this mode will make the RTL8019AS load the contents of 9346 like when the RSTDRV signal is asserted.
This auto-load operation will take about 2ms. After it is completed, the RTL8019AS goes back to the normal mode automatically (EEM1=EEM0 =0) and the CR register is reset to 21H.</td></tr>
<tr><td>1</td><td>0</td><td>9346 programming:
In this mode, both the local & remote DMA operation of 8390 are disabled. The 9346 can be directly accessed via bit3-0 which now reflect the states of EECS, EESK,EEDI, & EEDO pins respectively.</td></tr>
<tr><td>1</td><td>1</td><td>Config register write enable:
Before writing to the Page3 CONFIG1-3 registers, the RTL8019AS must be placed in this mode. This will prevent RTL8019AS's configurations from accidental change.</td></tr>
</table></td></tr>
<tr><td>5-4</td><td>-</td><td>Not used.</td></tr>
<tr><td>3</td><td>EECS</td><td rowspan="4">These bits reflect the state of EECS, EESK, EEDI & EEDO pins in auto-load or 9346 programming mode.</td></tr>
<tr><td>2</td><td>EESK</td></tr>
<tr><td>1</td><td>EEDI</td></tr>
<tr><td>0</td><td>EEDO</td></tr>
</table>

BPAGE: BROM Page Register (02H; Type=R/W)

This register selects a BROM page to be read by the host. Totally it can select 256 pages with 16k bytes per page. Thus the maximum BROM size is 256*16k=4M bytes.

CONFIG0: RTL8019AS Configuration Register 0 (03H; Type=R except Bit[7:6]=R/W)

Bit	Symbol	Description
7-6	VERID	Version ID: These two bits are defined as below. Bit7 / Bit6 / Type / Mode 1 / 1 / R / **RTL8019** 0 / 0 / R / **RTL8019A** 0 / 0 / R/W / **RTL8019AS**, these two bits are all "0" when power on, but can be written in RTL8019AS's config write enable mode (EEM0=EEM1=1). Software uses these differences to identify the chip.
5	AUI	This bit is set when external MAU is used on AUI interface. Therefore it is set when in 10Base5 mode or the AUI input pin is high.
4	PNPJP	This bit is set when PNP jumper pin is pulled high externally.
3	JP	This bit reflects the state of JP input. It, when set, indicates the RTL8019 is in jumper mode.
2	BNC	When set, this bit indicates that the RTL8019 is using the 10Base2 thin cable as its networking medium. This bit will be set in the following 2 cases: (1) PL1=PL0=0 (auto-detect) and link test fails (2) PL1=PL0=1 (10 Base 2)
1-0	0	Always 0s.

The following table describes the behavior of bits and pins for cabling media.

Media Type	AUI Input	Selected Media	AUI Bit	BNC Bit	LEDBNC Output	Original BNC bit in 8019 (For reference only)
10Base5	x	AUI	1	0	L	0
10Base2	x	BNC	0	1	H	1
10BaseT Link disabled	x	UTP	0	0	L	0
Auto detect Link OK	x	UTP	0	0	L	0
Auto detect Link fail	L	BNC	0	1	H	1
Auto detect Link fail	H	AUI	1	0	L	1

CONFIG1: RTL8019AS Configuration Register 1 (04H; Type=R except Bit7=R/W)

Bit	Symbol	Description
7	IRQEN	IRQ Enable: This bit controls the state of the interrupt request line selected by IRQS2-0. If this bit is set, the interrupt line goes high upon an interrupt request and will be low when there is no interrupt request. The interrupt line will be forced to tri-state if this bit is reset. This bit's power-up initial value is 1 and may be modified by software if EEM1=EEM0=1 in 9346CR register.
6-4	IRQS2-0	IRQ Select : These 3 bits select one of INT7-0 to reflect the RTL8019AS's interrupt request status. All unselected interrupt lines will be tri-stated.
3-0	IOS3-0	Select I/O base address.

IRQS2	IRQS1	IRQS0	Interrupt Line	Assigned ISA IRQ
0	0	0	INT0	IRQ2/9
0	0	1	INT1	IRQ3
0	1	0	INT2	IRQ4
0	1	1	INT3	IRQ5
1	0	0	INT4	IRQ10
1	0	1	INT5	IRQ11
1	1	0	INT6	IRQ12
1	1	1	INT7	IRQ15

IOS3	IOS2	IOS1	IOS0	I/O Base
0	0	0	0	300H
0	0	0	1	320H
0	0	1	0	340H
0	0	1	1	360H
1	0	0	0	380H
1	0	0	1	3A0H
1	0	1	0	3C0H
1	0	1	1	3E0H
0	1	0	0	200H
0	1	0	1	220H
0	1	1	0	240H
0	1	1	1	260H
1	1	0	0	280H
1	1	0	1	2A0H
1	1	1	0	2C0H
1	1	1	1	2E0H

CONFIG2: RTL8019AS Configuration Register 2 (05H; Type=R except Bit[7:5]=R/W)

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th></tr>
<tr><td>7-6</td><td>PL1-0</td><td>Select network medium types.
<table>
<tr><th>PL1</th><th>PL0</th><th>Medium Type</th></tr>
<tr><td>0</td><td>0</td><td>TP/CX auto-detect (10BaseT link test is enabled)</td></tr>
<tr><td>0</td><td>1</td><td>10BaseT with link test disabled</td></tr>
<tr><td>1</td><td>0</td><td>10Base5</td></tr>
<tr><td>1</td><td>1</td><td>10Base2</td></tr>
</table></td></tr>
<tr><td>5</td><td>BSELB</td><td>This bit, when set, forces the BROM disabled regardless of the contents of BS4-0. Its power-up initial value is 0 and can be modified by software if EEM1=EEM0=1 in 9346CR register.</td></tr>
<tr><td>4-0</td><td>BS4-0</td><td>These bits select the BROM size & memory base address.
<table>
<tr><th>BS4</th><th>BS3</th><th>BS2</th><th>BS1</th><th>BS0</th><th>BROM Base & size</th></tr>
<tr><td>0</td><td>0</td><td>*</td><td>*</td><td>*</td><td>Disabled</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>C000h, 32K</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>C800h, 32K</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>D000h, 32K</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>D800h, 32K</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>C000h, 64K</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>D000h, 64K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>C000h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>C400h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>C800h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>CC00h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>D000h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>D400h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>D800h, 16K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>DC00h, 16K</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>C000h, Page</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>C400h, Page</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>C800h, Page</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>CC00h, Page</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>D000h, Page</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>D400h, Page</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>D800h, Page</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>DC00h, Page</td></tr>
</table>
The RTL8019AS supports a special BROM mode: page mode. In page mode, the BROM always occupies 16K-byte host memory space. However the actual BROM size can be up to 4M bytes.
The BROM is divided into several 16K-byte pages. The power on boot page is set to page 0 and the program in page 0 is responsible to select the other pages by the BPAGE register and load their programs.
In page mode, bits BP7-0 of BPAGE register are mapped to the BA21-14 pins to select the proper BROM page. In other modes, BA21-16 are not used and the BA15-14 outputs are shown in the following table. </td></tr>
</table>

BROM size	BA14	BA15
16K	high	high
32K	SA14	high
64K	SA14	SA15

CONFIG3: RTL8019AS Configuration Register 3 (06H; Type=R except Bit[2:1]=R/W)

Bit	Symbol	Description
7	PNP	This bit is negligible in jumper mode. In jumperless mode it, when set, indicates the RTL8019AS is operating in Plug and Play mode. This bit is set when the PNP pin is high or the PNP bit in 9346 is set in jumperless mode.
6	FUDUP	When this bit is set, RTL8019AS is set to the full-duplex mode which enables simultaneously transmission and reception on the twisted-pair link to a full-duplex Ethernet switching hub. This feature not only increases the channel bandwidth from 10 to 20 Mbps but also avoids the performance degrading problem due to the channel contention characteristics of the Ethernet CSMA/CD protocol.
5-4	LEDS1-0	These two bits select the outputs to LED2-0 pins. **LEDS0** / **LED0 Pin** 0 / LED_COL 1 / LED_LINK **LEDS1** / **LED1 Pin** / **LED2 Pin** 0 / LED_RX / LED_TX 1 / LED_CRS / MCSB Please refer to section 6.5 for the behavior of LEDs. The MCSB signal is defined to put the local buffer SRAM into standby mode while DMA is not in progress and thus save powers.
3	-	Reserved. **Must not** write a 1 to this bit.
2	SLEEP	This bit, when set, puts RTL8019AS into sleep mode. In sleep mode, all LED signals (P.S. MCSB is not an LED signal) except LEDBNC are forced high to turn off the LEDs. The RTL8019AS still handles the network transmission and reception like in normal mode. The LEDBNC is not affected by this bit. This bit's power-up initial value is 0 and can be modified by software when EEM1=EEM0=1.
1	PWRDN	This bit , when set, puts RTL8019AS into power down mode. RTL8019AS supports two kinds of power down modes, which is selected by the contents of the HLTCLK register: (1) mode 1: power down with clock running (2) mode 2: power down with clock halted In both power down modes, the RTL8019AS's serial network interface and transceiver are turned off. All network activities are ignored. All LED signals except LEDBNC are forced high. The LEDBNC is forced low to disable the DC convertor for coaxial transceiver. In power down mode2, the RTL8019AS stops its internal clock for minimal power consumption. Registers except HLTCLK are typically not accessible in this mode. This bit's initial value comes from 9346 and can be modified if EEM1=EEM0=1 in 9346CR register.

0	ACTIVEB	This bit is the inverse of bit 0 in PnP Activate register (index 30H). When RTL8019AS is deactivated, all BROM memory read and I/O accesses to the Group1 registers except the HLTCLK register are ignored. The HLTCLK register and PnP logic work the same as when RTL8019AS is active. Note: The PnP logical device control register is the only way to activate RTL8019AS. Therefore, the HLTCLK register is allowed to be written to prevent RTL8019AS from dying when it is inactive in the clock-halted power-down mode.

CONFIG4 RTL8019AS Configuration Register 4 (0DH; Type=R)

Bit	Symbol	Description
7-1	-	Reserved
0	IOMS	When this bit is set, RTL8019AS uses SA15-SA0 to decode I/O address of NE2000 registers. When this bit is reset, RTL8019AS only decodes SA9-SA0 like the RTL8019 does. This mode is supported for applications which might require to fully decode I/O address. This bit is read-only and comes from the CONFIG4 byte(Offset 03H) of 9346(refer to section 6.3).

CSNSAV: CSN Save Register (08H; Type=R)
This register is provided to backup the CSN assigned to the PnP CSN register.

HLTCLK: Halt Clock Register (09H; Type=W)
This is the only active one of Group1 registers when RTL8019AS is inactivated.
Writing to this register is invalid if RTL8019AS is not in power down mode. (i.e. If PWRDN bit in CONFIG3 register is zero.)
The data written to this register determines the RTL8019AS's power down mode.

Data	Power Down Mode
52H (ASCII code of 'R')	Mode 1 - clock **R**unning
48H (ASCII code of 'H')	Mode 2 - clock **H**alted
Other values	Ignored

INTR: Interrupt Register (0BH; Type=R)
This register reflects the ISA bus states of INT7-0 pins.

FMWP: Flash Memory Write Protect Register (0Ch, Type=W)
This register is write only. A write to this register is valid only when EEM0=EEM1=1. Sequentially writing 2 bytes of data (57H then A8H) to this register enables the flash memory write operation. Writing other data to this register will reset the write sequence and disable the flash write. All flash memory write commands from host are ignored if the write operation is not enabled.

5.2. Group 2: Plug and Play (PnP) Registers

Auto-configuration Ports

Three 8-bit I/O ports are defined for the PnP read/write operations. They are called Auto-configuration ports and are listed below.

Port Name	Type	Location
ADDRESS	W	279H (Printer status port)
WRITE_DATA	W	A79H (Printer status port + 800H)
READ_DATA	R	Relocatable in range 200H to 3FFH

The Plug and Play registers are accessed by first writing the address of the desired register, which is called "***Register Index***" in the following paragraph, to the ADDRESS port, followed by a read of data from the READ_DATA port or a write of data to the WRITE_DATA port. A write to the ADDRESS port may be followed by any number of WRITE_DATA or READ_DATA accesses to the same indexed register without the need to write to the ADDRESS port before each access.

The Address port is also the write destination of the initiation key, which will be described later.

Plug and Play Registers

The Plug and Play registers may be divided into card registers and logical device registers. According to the Plug and Play specification, a PnP card may contain more than one logical devices. The card registers are unique for each card. However, the logical device registers are repeated for each logical device on the card. Furthermore, all card registers are card control registers, while the logical device registers can be divided into logical device control registers and configuration registers. Although an RTL8019AS card contains only one logical device, the following paragraph still depicts the Plug and Play registers by the same PnP categorizing method.
p.s. Those registers or bits not mentioned below are all read only with value=0.

5.2.1. Card Control Registers

Index	Name	Type	Definition
00H	Set RD_DATA port	W	The location of the READ_DATA port is determined by writing to this register. Bits[7:0] become ISA I/O read port address bits[9:2]. Address bits[1:0] of the READ_DATA port are always 1.
01H	Serial Isolation	R	A read to this register causes a PnP card in the ***Isolation*** state to compare one bit of the card's serial ID. This process will be described in more details in section 6.
02H	Config Control	W	Bit[0] - Reset command Setting this bit will reset all logical devices and restore configuration registers to their power-up values. The CSN is preserved. Bit[1] - Wait for Key command Setting this bit makes the PnP card return to the ***Wait for Key*** state. The CSN is preserved. Bit[2] - PnP Reset CSN command Setting this bit will reset the card's CSN to 0. Both the CSN (index 06H) and CSNSAV (index F5H) registers are reset. Note that the hardware will automatically clear the bits and there is no need for software to clear them.
03H	Wake[CSN]	W	A write to this register will cause all cards that have a CSN that matches the write data[7:0] to go from the ***Sleep*** state to either the ***Isolation*** state if the write data for this command is zero or the ***Config*** state if the write data is not zero.
04H	Resource Data	R	A read from this register reads the next byte of resource data. The Status register must be polled until bit[0] is set before this register may be read.
05H	Status	R	Bit[0] when set indicates it is okay to read the next data byte from the Resource Data register.
06H	Card Select Number (CSN)	R/W	A write to this register sets a card's CSN. The CSN is a value uniquely assigned to each ISA PnP card after the serial identification process so that each card may be individually selected during a Wake[CSN] command. The CSN value written to this register will also be recorded to the CSNSAV register located at PnP register index F5H and Group 1 Page3 offset 08H.
07H	Logical Device Number	R	00H (Only one logical device in RTL8019AS).

5.2.2. Logical Device Control Registers

Index	Name	Type	Definition
30H	Activate	R/W	For each logical device there is one Activate register that controls whether or not the logical device is active on the ISA bus. Bit[0], if set, activates the logical device. Before a logical device is activated, I/O range check must be disabled.
31H	I/O Range Check	R/W	This register is used to perform a conflict check on the I/O port range programmed for use by a logical device. Bit[1] - This bt, when set, enables I/O range check. I/O range check is only valid when the logical device is inactive. Bit[0] - If set, this bit forces the logical device to respond to I/O reads of the logical device's assigned I/O range with a 55H when I/O range check is in operation. If clear, the logical device drives AAH.

5.2.3. Logical Device Configuration Registers

Memory Configuration Registers

Index	Name	Type	Definition
40H	BROM base address bits[23:16]	R/W	Bits[23:20] & bit[17] are read only with values=0. All other bits are read/write bits.
41H	BROM base address bits[15:0]	R/W	Bits[13:8] are read only with values=0. All other bits are read/write bits.
42H	Memory Control	R	00H. (Only 8-bit operation is supported for BROM)

Note: The BROM size of RTL8019AS is determined by the 9346 contents but not the memory configuration registers.

I/O Configuration Registers

Index	Name	Type	Definition
60H	I/O base address bits[15:8]	R/W	Bits[15:10] are read only with values=0. All other bits are read/write bits.
61H	I/O base address bits[7:0]	R/W	Bits[4:0] are read only with values=0. All other bits are read/write bits.

Interrupt Configuration Registers

Index	Name	Type	Definition
70H	IRQ level	R/W	Read/write value indicating a selected interrupt level. Bits[3:0] select which ISA interrupt level is used. One selects IRQ1, fifteen selects IRQ15. IRQ0 is not a valid interrupt selection and represents no interrupt selection.

71H	IRQ type	R	Read/Write value indicating which type of interrupt is used for the IRQ selected above. Bit[1] - Level, 1=high, 0=low Bit[0] - Type, 1=level, 0=edge For RTL8019AS, this register is read only with value=02H.

DMA Configuration Registers

Index	Name	Type	Definition
74H	DMA channel select 0	R	04H (indicating no DMA channel is needed)
75H	DMA channel select 1	R	04H (indicating no DMA channel is needed)

Vendor Defined Registers

Index	Name	Type	Definition
F0H	CONFIG0	R	Direct mapping of the Page3 CONFIG0 register.
F1H	CONFIG1	R	Direct mapping of the Page3 CONFIG1 register.
F2H	CONFIG2	R	Direct mapping of the Page3 CONFIG2 register.
F3H	CONFIG3	R	Direct mapping of the Page3 CONFIG3 register.
F4H	-	-	
F5H	CSNSAV	R	Direct mapping of the Page3 CSNSAV register.
F6H	Vendor Control	W	Bit[2] - RT Reset CSN command Setting this bit will reset the card's CSN in the CSN register (index 06H) to 0. The CSNSAV register is not affected. This bit is cleared by hardware automatically.

6. Functional Descriptions

6.1. RTL8019AS Configuration Modes

The RTL8019AS supports 3 configuration modes: jumper, RT jumperless, and PnP.

JP Pin	PnP Pin	9346 Content		Mode	CONFIG0		CONFIG3	
		PNP	ACTIVEB		JP	PNPJP	PNP	ACTIVEB
H	H L	x	x	Jumper	1	1 0	0	0
L	H	x	a (a=0or1)	PnP	0	1	1	a
L	L	1	a (a=0or1)	PnP	0	0	1	a
L	L	0	x	RT jmpless	0	0	0	0

P.S. "x" denotes don't care.

The RTL8019AS's resource configuration informations such as I/O base address, BROM memory base address, and interrupt request line, etc., are stored in the CONFIG3-0 registers in Group1 Page3 as well as the PnP logical device configuration registers. Their power-up default values may come from the states of jumper pins in jumper mode or the contents of 9346 in PnP and RT jumperless mode. Their values can be modified by software via the logical device configuration registers in all 3 modes. The update values will be recorded to the CONFIG3-0 registers, too. This new configuration is only valid temporarily and will be lost after an auto-load command, an active RSTDRV, or PC power off . Permanent changes of configuration must be done by changing the jumper states or the contents of 9346. Note that the BROM size can not be modified temporarily.

The Plug and Play logic can work in all the three configuration modes except that an RT defined initiation key, named RT initiation key, should be used instead of the PnP initiation key. In other words, the RT initiation key is supported in all configuration modes while the PnP initiation key is only supported in the PnP mode. By using the RT initiation key, the software can put RTL8019AS to the PnP ***Config*** state and access the logical device configuration registers even in the jumper and RT jumperless modes.

Power up default ACTIVE state

In RTL8019, the ACTIVEB bit in 93C46 decides the power-up adapter status even in RT jumpless mode. In the standard application when BROM is not enabled, the adapter should be power up inactive in PnP mode and active in RT jumperless mode. However RTL8019's PnP jumper only decides the jumperless mode. The adapter's "ACTIVE" status is not changed properly at the same time when the user changes the PnP jumper state. This causes an application inconsistence when PnP jumper is to be used.

In RTL8019AS, we change RTL8019's original specification into:
The ACTIVEB bit in 9346 is ignored when RTL8019AS is in jumper or RT jumperless mode. The adapter's power-up status is always "ACTIVE" in RT jumperless mode. However, the active status still can be changed via the PnP Activate register.

The differences between the 3 configuration modes are shown in the following table.

Configuration Mode	Resource of Power-up Value	Supported Initiation Key
Jumper	Jumper Pins	RT Initiation Key
RT Jumperless	9346	RT Initiation Key
Plug and Play	9346	RT and PnP Initiation Key

Initial Values of CONFIG1-3 Registers after RSTDRV or Auto-load Command

CONFIG1

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode	**IRQEN**	**IRQS2**	**IRQS1**	**IRQS0**	**IOS3**	**IOS2**	**IOS1**	**IOS0**
Jumper	1	jumper	jumper	jumper	jumper	jumper	jumper	jumper
RT Jumperless Plug and Play	1	9346	9346	9346	9346	9346	9346	9346

CONFIG2

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode	**PL1**	**PL0**	**BSELB**	**BS4**	**BS3**	**BS2**	**BS1**	**BS0**
Jumper	jumper	jumper	0	jumper	jumper	jumper	jumper	jumper
RT Jumperless Plug and Play	9346	9346	0	9346	9346	9346	9346	9346

CONFIG3

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode	**PNP**	**FUDUP**	**LEDS1**	**LEDS0**	**-**	**SLEEP**	**PWRDN**	**ACTIVEB**
Jumper	0	9346	9346	9346	-	0	9346	9346
RT Jumperless Plug and Play	0 1	9346	9346	9346	-	0	9346	9346

6.2. Plug and Play

6.2.1. Initiation Key

The Plug and Play logic is quiescent on power up and must be enabled by software. This is done by a predefined series of writes (32 I/O writes) to the ADDRESS port, which is called the *initiation key*. The write sequence is decoded by RTL8019AS. If the proper series of I/O writes is detected, then the Plug and Play auto-configuration ports are enabled. The write sequence will be reset and must be issued from the beginning if any data mismatch occurs. The exact sequence for the initiation key is listed below in hexadecimal notation.

PnP Initiation Key

6A, B5, DA, ED, F6, FB, 7D, BE,
DF, 6F, 37, 1B, 0D, 86, C3, 61,
B0, 58, 2C, 16, 8B, 45, A2, D1,
E8, 74, 3A, 9D, CE, E7, 73, 39

RT Initiation Key

DA, 6D, 36, 1B, 8D, 46, 23, 91,
48, A4, D2, 69, 34, 9A, 4D, 26,
13, 89, 44, A2, 51, 28, 94, CA,
65, 32, 19, 0C, 86, 43, A1, 50

6.2.2. Isolation Protocol

A simple algorithm is used to isolate each Plug and Play card. This algorithm uses the signals on the ISA bus and requires lock-step operation between the Plug and Play hardware and the isolation software.

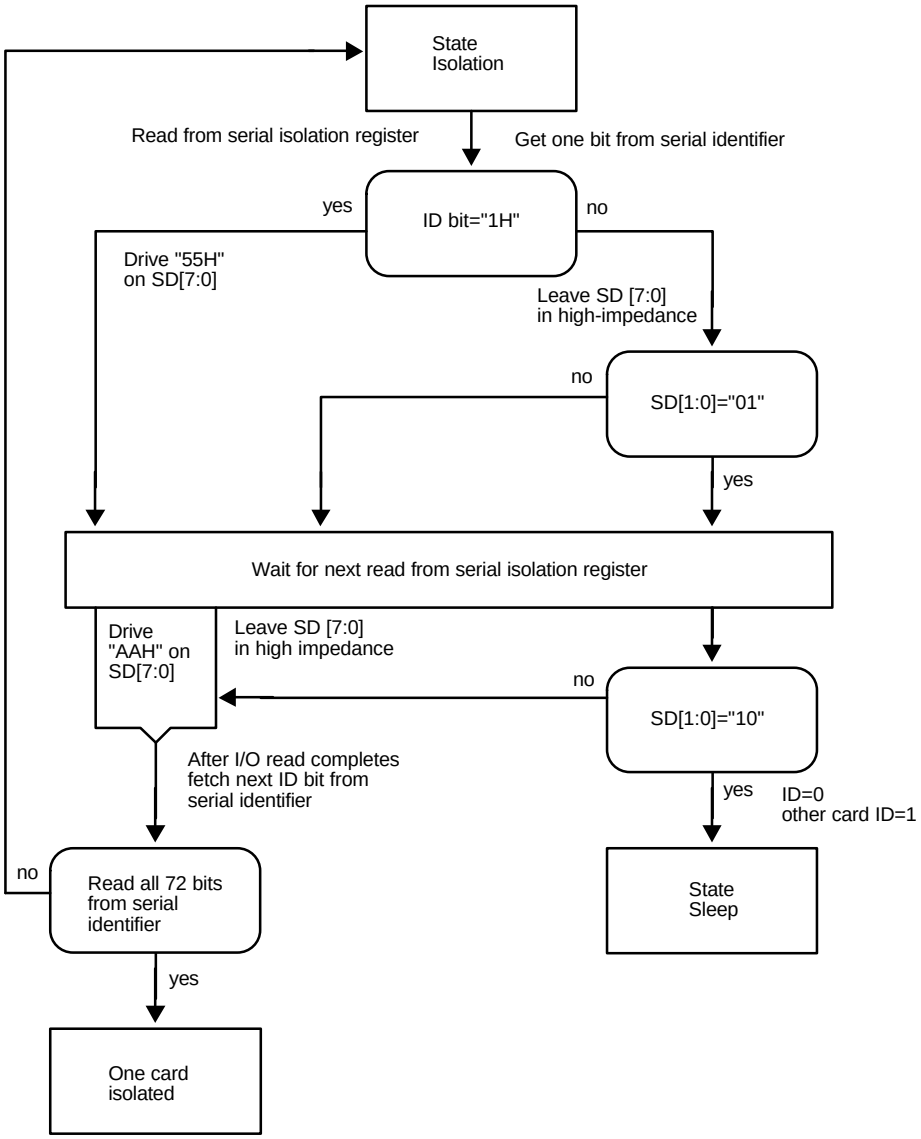


Figure 1. Plug and Play ISA Card Isolation Algorithm

Serial Identifier

The key element of the Plug and Play isolation protocol is that each card contains a unique number, named *serial identifier*. The serial identifier is a 72-bit unique, non-zero number composed of two 32-bit fields and an 8-bit checksum. The first 32-bit field is a vendor identifier. The other 32-bits can be any value, for example, a serial number, part of a LAN address, or a static number, as long as there will never be two cards in a single system with the same 64-bit number. The serial identifier is accessed bit-serially by the isolation logic and is used to differentiate the cards.

Check-sum	Serial Number				Vendor ID			
Byte 0	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
7:0	7:0	7:0	7:0	7:0	7:0	7:0	7:0	7:0

Figure 2. Shifting of Serial Identifier

The shift order for all Plug and Play serial isolation and resource data is defined as bit[0], bit[1], and so on through bit[7].

Hardware Protocol

The isolation protocol can be invoked by the Plug and Play software at any time. The initiation key described earlier, puts all cards into configuration mode. The hardware on each card expects 72 pairs of I/O read accesses to the READ_DATA port. The card's response to these reads depends on the value of each bit of the serial identifier which is being examined one bit at a time, in the sequence shown in Figure 1.

If the current bit of the serial identifier is a "1", then the card will drive the data bus to 55H to complete the first I/O read cycle. If the bit is "0", then the card puts its data bus driver into high impedance. All cards in high impedance will check the data bus during the I/O read cycle to sense if another card is driving SD[1:0] to "01". During the second I/O read, the card(s) that drove the 55H, will now drive a AAH. All high impedance card will check the data bus to sense if another card is driving SD[1:0] to "10."

If a high impedance card sensed another card driving the data bus with the appropriate data during both cycles, then that card ceases to participate in the current iteration of card isolation. Such cards, which lose out, will participate in future iterations of the isolation protocol.

NOTE: *During each read cycle, the Plug and Play hardware drives the entire 8-bit data bus, but only checks the lower 2 bits.*

If a card was driving the bus or if the card was in high impedance and did not sense another card driving the bus, then it should prepare for the next pair of I/O reads. The card shifts the serial identifier by one bit and uses the shifted bit to decide its response.
The above sequence is repeated for the entire 72-bit serial identifier.

At the end of this process, one card remains. This card is assigned a handle referred to as the *Card Select Number* (CSN) that will be used later to select the card. Cards which have been assigned a CSN will not participate in subsequent iterations of the isolation protocol. Cards must be assigned a CSN before they will respond to the other PnP commands.

It should be noted that the protocol permits the 8-bit checksum to be stored in non-volatile memory on the card or generated by the on-card logic in real-time. The checksum algorithm is implemented as a *Linear Feedback Shift Register* (LFSR), which is shown in Figure 3.

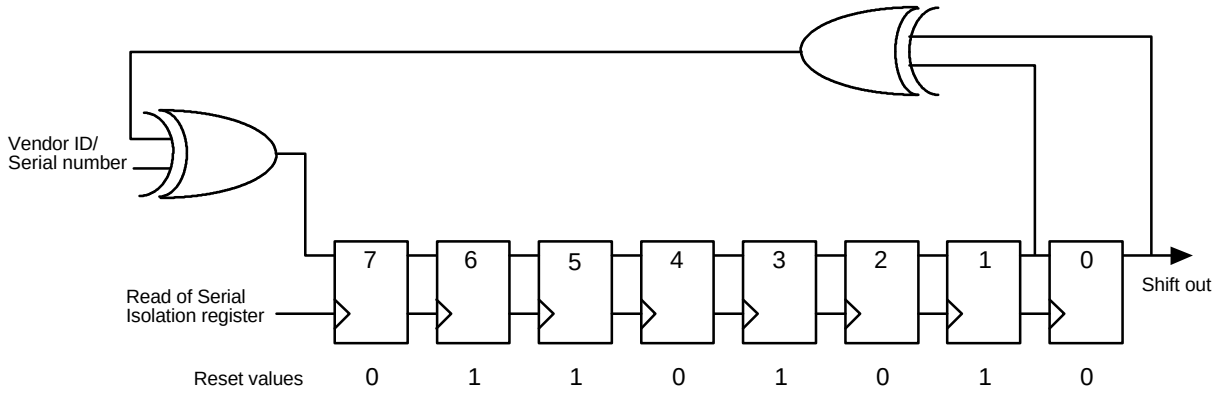


Figure 3. Checksum LFSR

The LFSR resets to 6AH upon receiving the Wake[CSN] command. The next shift value for the LFSR is calculated as LFSR[1] XOR LFSR[0] XOR Serial Data. The LFSR is shifted right one bit at the conclusion of each pair of reads to the Serial Isolation register. LFSR[7] is assigned the next shift value described above.

After the first 64 pairs of reads of the Serial Isolation register, the LFSR will have the value of serial identifier checksum.

Plug and Play cards must not drive the IOCHRDY signal during serial isolation. However, cards may drive IOCHRDY at any other time.

Software Protocol

The Plug and Play software sends the initiation key to all Plug and Play cards to place them into configuration mode. The software is then ready to perform the isolation protocol.

The Plug and Play software generates 72 pairs of I/O read cycles from the READ_DATA port. The software checks the data returned from each pair of I/O reads for the 55H or AAH driven by the hardware. If both 55H or AAH are read back, then the software assumes that the hardware had a "1" bit in that position. All other results are assumed to be a "0".

During the first 64 bits, software generates a checksum using the received data. The checksum is compared with the checksum read back in the last 8 bits of the sequence.

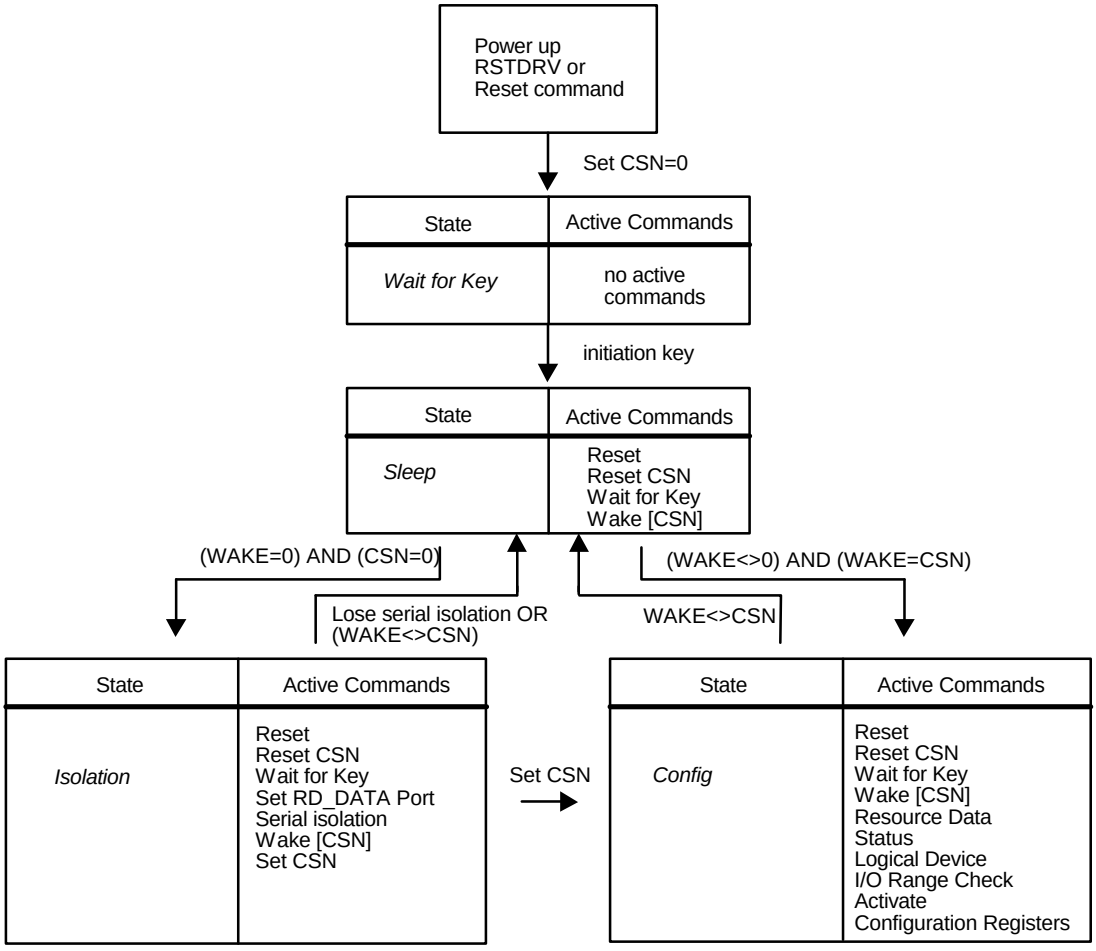
There are two other special considerations for the software protocol. During an iteration, it is possible that the 55H and AAH combination is never detected. It is also possible that the checksum does not match. If either of these cases occur on the first iteration, it must be assumed that the READ_DATA port is in conflict. If a conflict is detected, then the READ_DATA port is relocated. The above process is repeated until a non-conflicting location for the READ_DATA port is found. The entire range between 200H and 3FFH is available, however in practice it is expected that only a few locations will be tried before software determines that no Plug and Play cards are present.

During subsequent iterations, the occurrence of either of these two special cases should be interpreted as the absence of any further Plug and Play cards (i.e. the last card was found in the previous iteration). This terminates the isolation protocol.

NOTE: The software must delay 1 msec prior to starting the first pair of isolation reads, and must wait 250 msec between each subsequent pair of isolation reads. This delay gives the ISA card time to access information from possibly very slow storage devices.

6.2.3. Plug and Play Isolation Sequence

The Plug and Play isolation sequence is divided into four states: ***Wait for Key***, ***Sleep***, ***Isolation***, and ***Config*** states. The state transitions for the Plug and Play ISA card are shown below.



NOTES:

1. CSN= Card Select Number
2. RSTDRV causes a state transition from the current state to Wait for Key and sets all CSNs to zero
3. The Wait for Key command causes a state transition from the current state to Wait for Key
4. The Reset CSN commands include PnP Reset CSN and RT Reset CSN commands. The former sets all ISA PnP cards' CSNs to zero while the latter only sets RTL8019 PnP cards' CSNs to zero. Both commands do not cause a state transition.

Figure 4. Plug and Play ISA Card State Transitions

On power up, all PnP cards detect RSTDRV, set their CSN to 0, and enter the ***Wait for Key*** state. There is a required 2 msec delay from either a RSTDRV or a PnP Reset command to any Plug and Play port access to allow a card to load initial configuration information from a non-volatile device, which is 9346 for RTL8019AS.

Cards in the ***Wait for Key*** state do not respond to any access to their auto-configuration ports until the initiation key is detected. Cards ignore all ISA access to their Plug and Play interface.

When the cards have received the initiation key, they enter the ***Sleep*** state. In this state, the cards listen for a Wake[CSN] command with the write data set to 00H. This wake[CSN] command will send all cards to the ***Isolation*** state and reset the serial identifier/resource data pointer to the beginning.

The first time the cards enter the ***Isolation*** state it is necessary to set the READ_DATA port address using the Set RD_DATA port command. The software should then verify the selected READ_DATA port address is not in conflict with any other devices by the isolation protocol.

Next, 72 pairs of reads are performed to the Serial Isolation register to isolate a card as described previously. If the checksum read from the card is valid, then this means one card has been isolated. The isolated card remains in the ***Isolation*** state while all other cards have failed the isolation protocol and have returned to the ***Sleep*** state. The CSN on this card is set to a unique number. Writing this value causes this card to transition to the ***Config*** state. Sending a Wake[0] command causes this card to transition back to ***Sleep*** state and all cards with a CSN value of zero to transition to the Isolation state. This entire process is repeated until no Plug and Play cards are detected.

6.2.4. Reading Resource Data

Each PnP card supports a *resource data* structure stored in a non-volatile device (e.g. 9346) to describe the resources supported and those requested by the functions on the card. The Plug and Play resource management software will arbitrate resources and setup the logical device configuration registers according to the resource data.

Card resource data may only be read from cards in the ***Config*** state. A card may get to the ***Config*** state by one of two different methods. A card enters the ***Config*** state in response to the card "winning" the serial isolation protocol and having a CSN assigned. The card also enters the ***Config*** state in response to receiving a Wake[CSN] command that matches the card's CSN.

As described above, all Plug and Play cards function as if their serial identifier and their resource data both come from the same serial device. As also stated above, the pointer to the serial device is reset in response to any Wake[CSN] command. This implies that if a card enters the ***Config*** state directly in response to a Wake[CSN] command, the 9-byte serial identifier must be read first before the card resource data is accessed. The Vendor ID and Unique Serial Number is valid; however, the checksum byte, when read in this way, is not valid. A card that enters the ***Config*** state after the isolation protocol has been run has already accessed all 72 bits of the serial identifier and the first read of the Resource Data register will return resource data.

Card resource data is read by first polling the Status register and waiting for bit[0] to be set. When this bit is set it means that one byte of resource data is ready to be read from the Resource Data register. After the Resource Data register is read, the Status register must be polled before reading the next byte of resource data. This process is repeated until all resource data is read. The format of resource data is described in the following section.

The above operation implies that the hardware is responsible for accumulating 8 bits of data in the Resource Data register. When this operation is complete, the status bit[0] is set. When a read is performed on the Resource Data register, the status bit[0] is cleared, eight more bits are shifted into the Resource Data register, then the status bit[0] is set again.

6.2.5. PnP auto detect mode

When using RTL8019, the user needs to setup the card to PnP or jumperless mode according to the host environments. The typical operating modes of a RTL8019 card include:
(1) when used in a non-PnP PC, set the card to RT jumperless mode & power-on active
(2) when used in a PnP PC,
    (2.1) if BROM disabled, set the card to PnP mode & power-on inactive
    (2.2) if BROM enabled, set the card to PnP mode & power-on active
P.S. PCs with PnP BIOS, or Windows 95, or Intel Configuration Manager, etc. are called PnP PCs

If a card in mode(2.1) is put in a non-PnP PC, the drivers will fail to initialize the card. RTL8019AS supports a PnP auto-detect mode to solve the problem. ***The card may be set to a default state: PnP mode & power-on active with BROM disabled.*** If the card is in a non-PnP PC, it will work like a normal jumperless card. If the card is in a PnP PC which requires the card to be power-on inactive, RTL8019AS will change itself into inactive state when the first time a PnP init key is detected.

6.3. 9346 Contents

The 9346 is a 1k-bit EEPROM. Although it is actually addressed by words, we list its contents by bytes below for convenience.

Bytes		Contents	Comments
00H - 03H	**(4 bytes)**		**Power-up initial value of Page3 and PnP logical device configuration registers**
	00H	CONFIG1	
	01H	CONFIG2	
	02H	CONFIG3	
	03H	CONFIG4	
04H - 11H	**(14 bytes)**		**NE2000 IDPROM**
	04H - 09H	Ethernet ID 0-5	Ethernet node address
	0AH - 11H	Product ID 0-7	Assigned by card makers; negligible
12H - 1AH	**(9 bytes)**		**Plug and Play Serial Identifier**
	12H - 15H	Vendor ID 0-3	
	16H - 19H	Serial Number 0-3	
	1AH	Serial ID Checksum	
1BH - 7FH	**(101 bytes)**		**Plug and Play Resource Data**

Detail values of 9346 CONFIG1-3 bytes

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG1	*	IRQS2	IRQS1	IRQS0	IOS3	IOS2	IOS1	IOS0
CONFIG2	PL1	PL0	*	BS4	BS3	BS2	BS1	BS0
CONFIG3	PNP	FUDUP	LEDS1	LEDS0	*	*	PWRDN	ACTIVEB

P.S. '*' denotes don't care.

Example : Plug and Play Resource Data for RTL8019AS (Total 73+5 bytes)

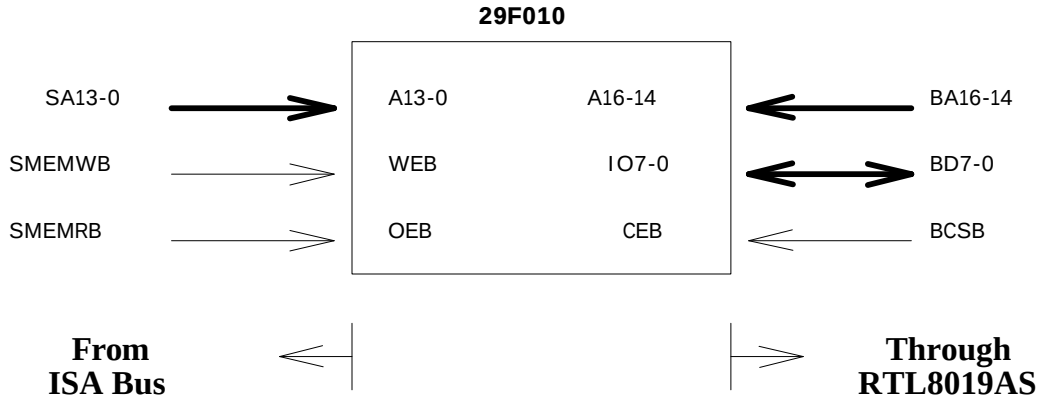
TAG	Plug and Play Version Number		Length: fixed	3 bytes
	Item byte	0AH		
	PnP version	10H		
	Vendor version	10H		
TAG	ANSI Identifier String		Length: variable	37 bytes
	Item byte	82H		
	Length bits 7-0	22H		
	Length bits 15-8	00H		
	Identifier string	'REALTEK PLUG & PLAY ETHERNET CARD', 00H		
TAG	Logical Device ID		Length: fixed	7 bytes
	Item byte	16H		
	Logical device ID0-3	4AH, 8CH, 80H, 19H		
	Flag 0	02H or 03H (use 03H when BROM is enabled)		
	Flag 1	00H		
TAG	Compatible Device ID (NE2000 compatible)		Length: fixed	5 bytes if given

	Item byte	1CH		
	Compatible ID0-3	41H, D0H, 80H, D6H		
TAG	I/O Format		Length: fixed	8 bytes
	Item byte	47H		
	I/O information	00H		
	Min. I/O base bits 7-0	20H		
	Min. I/O base bits 15-8	02H		
	Max. I/O base bits 7-0	80H		
	Max. I/O base bits 15-8	03H		
	Base alignment	20H		
	Range length	20H		
TAG	IRQ Format		Length: fixed	4 bytes
	Item byte	23H		
	IRQ mask bits 7-0	38H		
	IRQ mask bits 15-8	9EH		
	IRQ information	01H		
TAG	Memory Format (optional)		Length: fixed	12 bytes
	Item byte	81H	This example uses 16k-byte BROM.	
	Length bits 7-0	09H		
	Length bits 15-8	00H		
	Memory information	40H		
	Min. base bits 15-8	00H		
	Min. base bits 23-16	0CH		
	Max. base bits 15-8	C0H		
	Max. base bits 23-16	0DH		
	Base alignment bits 7-0	00H		
	Base alignment bits 15-8	40H		
	Range length bits 15-8	40H		
	Range length bits 23-16	00H		
TAG	END Tag		Length: fixed	2 bytes
	Item byte	79H		
	Checksum	2's complement of the sum of all the above resource data i.e. 2's complement of (0AH+10H+10H+.....+79H)		

6.4. Boot ROM

Whether a EPROM or flash memory is used as the BROM, RTL8019AS's BROM read operation is still the same as RTL8019's. The supported BROM size is the same, too.

The write operation of a flash memory is much like the read except that a SMEMWB command is issued instead of SMEMRB. The block diagram below shows the application when an 128k*8bit flash memory (e.g. 29F010) is used as the BROM.



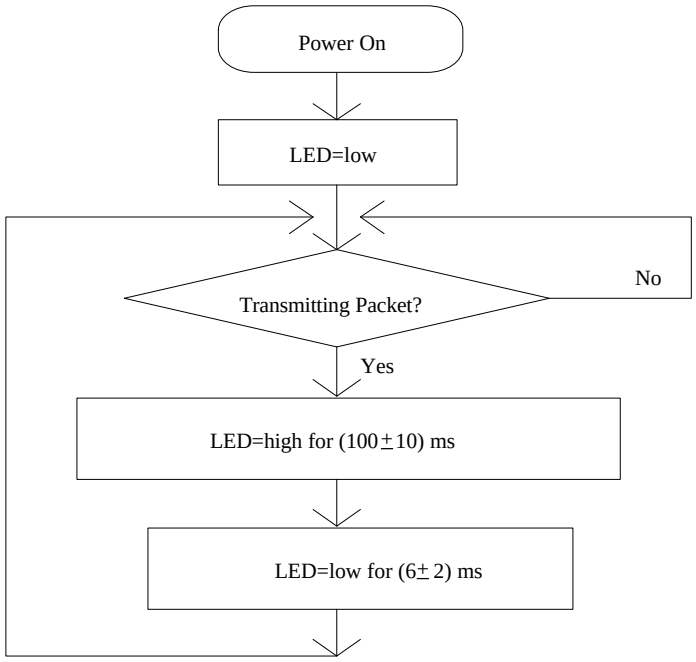
In this case, the BROM page mode is used. Before either to read or write BROM, the appropriate ROM page must be set in the BPAGE (page3, offset 02h) register first. The RTL8019AS will always reflect the content of BPAGE onto the BA14-21 bus. When RTL8019AS decodes a valid BROM read or write command, it asserts BCSB low. Note the flash memory write must be enabled through the RTL8019AS's FMWP register before the host's flash write command.

6.5. LED Behaviors

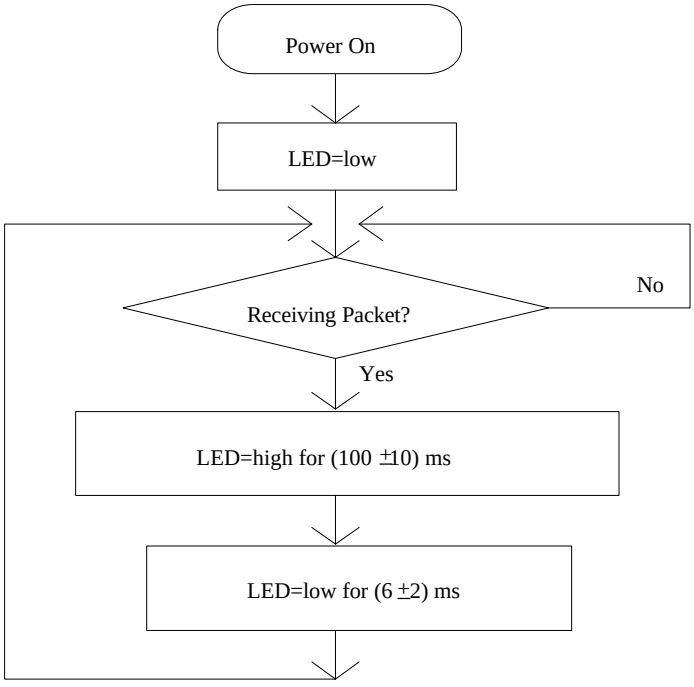
This section describes the lighting behaviors of the LED output signals which may be selected by LEDS1 and LEDS0 bits in the Page3 CONFIG3 register.

P.S. It is assumed that the LED is on when the signal goes low.

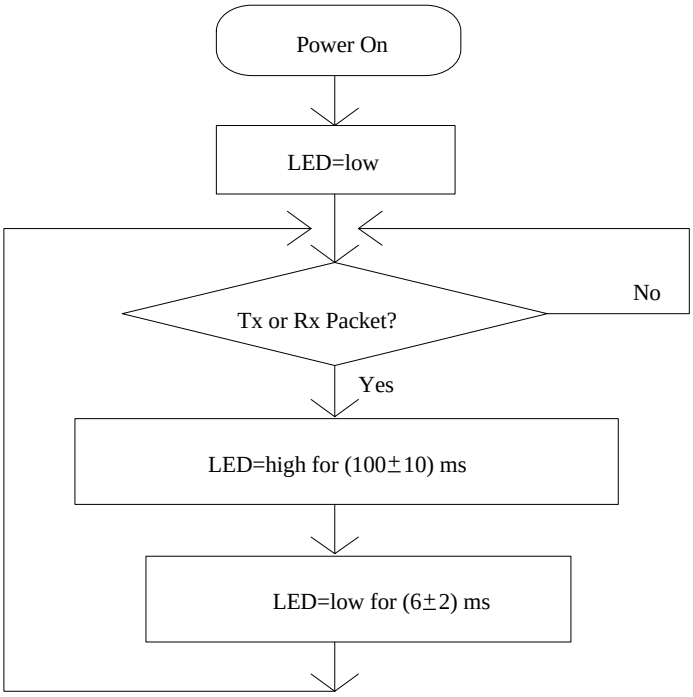
(1) LED_TX: Tx LED



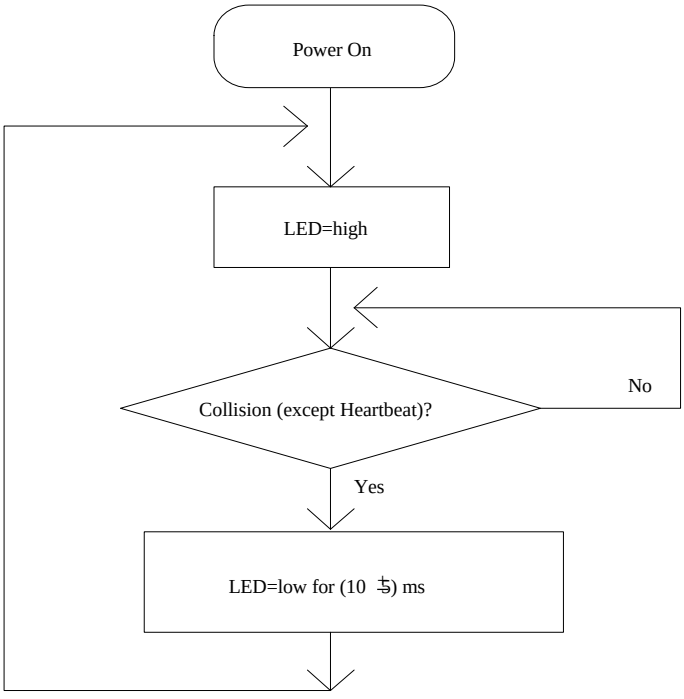
(2) LED_RX: Rx LED



(3) LED_CRS=LED_TX+LED_RX: Carrier Sense LED



(4) LED_COL: Collision LED



LED1 (LED_RX or LED_CRS)

RTL8019's LED_RX or LED_CRS LED sometimes keeps blinking when the media type of a 2-in-1 (UTP+BNC) LAN adapter is set to auto-detect and both UTP and coaxial cable are not connected. In the case, RTL8019 is actually using the BNC because the UTP link test fails. Many 8392 will falsely detect a carrier when the BNC inteface is not properly terminated (e.g. coaxial cable is not connected). That carrier sense will then make RTL8019's LED_RX or LED_CRS blink. The problem is that not all 8392s cause the LED blinking, which makes the phenomenon very ambiguous. Considering the phenomenon is normally awared upon power on, we change RTL8019's original function to solve the problem to some extent.

The new specification is:
The LED_RX or LED_CRS does not reflect the carrier sense when the CR register bit 0 is set (in stop mode). Thus, the false carrier due to cabling problem upon power on will not cause the LED1 to blink anymore.

LED Output States in Power Down Modes

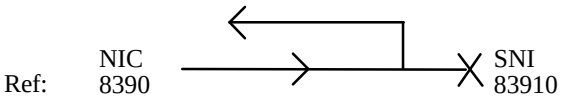
LED Output	Normal Mode / Idle	Sleep Mode	Power Down Mode
LEDBNC	-	-	Low
LED_LINK	-	High	High
LED_COL	High	High	High
LED_TX	Low	High	High
LED_RX	Low	High	High
LED_CRS	Low	High	High

6.6. Loopback Diagnostic Operation

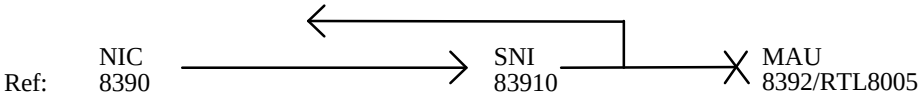
6.6.1. Loopback operation

The RTL8019AS provides 3 loopback modes. By loopback test, we can verify the integrity of data path, CRC logic, address recognition logic and cable connection status.

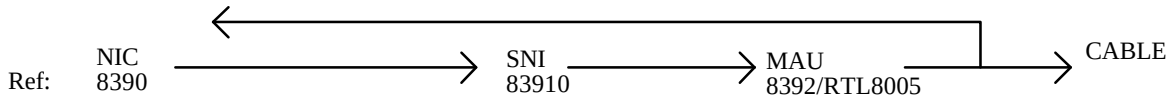
Mode 1: Loopback through the NIC (LB1=0, LB0=1 in TCR).
The NRZ data is not transmitted to the SNI but instead it's loopbacked to the NIC's Rx deserializer. The traffic on the cable is ignored.



Mode 2: Loopback through the SNI (LB1=1, LB0=0 in TCR)
The Manchester encoded data is not transmitted to the MAU. It's loopbacked through the SNI to NIC. The traffic on the cable is ignored.



Mode 3: Loopback through the cable (LB1=1, LB0=1 in TCR)
The packets are transmitted via the MAU onto the network and RTL8009 receives all incoming packets (not only the MAU-loopbacked Tx data) in the meantime.



Alignment of the Reception FIFO

The reception FIFO is an 8-byte ring structure. The first received byte is put at location zero. When the location pointer goes to the end of the FIFO, it wraps to the beginning of the FIFO and overwrites the previous data. At the end of the packet reception, the FIFO contents are in the "order" (from the ring structure's view) as shown below.

(1) CRC enabled (CRC bit in TCR=0)
- 1-byte received packet data
- 4-byte CRC
- 1-byte lower byte count
- 1-byte upper byte count
- 1-byte upper byte count

(2) CRC disabled (CRC bit in TCR=1)
- 5-byte received packet data
- 1-byte lower byte count
- 1-byte upper byte count
- 1-byte upper byte count

6.6.2. To Implement Loopback Test

(1) To verify the integrity of data path

set RCR=00h to accept physical packet
set PAR0-5 to accept packet
set DCR=40h (8-bit slot) or 43h (16-bit slot)
set TCR=02h, 04h, 06h to do loopback test 1, 2, 3 respectively
set CRC enabled (CRC=0 in TCR)
clear ISR
tx a packet and check ISR
check FIFO after loopback

Note: Loopback mode 3 is sensitive to the network traffic, so the values of FIFO may be not correct.

(2) To verify CRC logic

Select a loopback mode (e.g. mode 2) to test

A. To test CRC generator

set RCR=00h to accept physical packet

set PAR0-5 to accept packet
set TCR=04h (CRC enabled)
set DCR=40h (8-bit slot) or 43h (16-bit slot)
clear ISR
tx a packet
check CRC bytes in FIFO after loopback

B. To test CRC checker
set RCR=00h to accept physical packet
set PAR0-5 to accept packet
set TCR=05h (CRC disabled)
set DCR=40h (8-bit slot) or 43h (16-bit slot)
clear ISR
tx a packet with good or bad CRC appended by program
check FIFO, ISR & RSR after loopback
For bad CRC, expected: ISR=06h, RSR=02h (Tx: OK, Rx:CRC error)
For good CRC, expected: ISR=02h, RSR=01h (Tx:OK, Rx: OK)

Note: In loopback mode, the received packets are not stored to SRAM, so PRX bit in ISR isn't set.

(3) To verify the address recognition function

Select a loopback mode (e.g. mode 2) to test

A. Right physical destination address
set RCR=00h to accept physical packet
set PAR0-5 to accept packet
set TCR=04h (CRC enabled)
set DCR=40h (8-bit slot) or 43h (16-bit slot)
clear ISR
tx a packet
check ISR after loopback
Expected: ISR=06h (packets accepted, Rx CRC error)

B. Wrong physical destination address
set RCR=00h to accept physical packet
set PAR0-5 to reject packet
set TCR=04h (CRC enabled)
set DCR=40h (8-bit slot) or 43h (16-bit slot)
clear ISR
tx a packet
check ISR after loopback
Expected: ISR=02h (packets rejected, Rx no response)

(4) To Test Cable Connection

There are four physical medium types in RTL8019.

We perform loopback mode 3 to test the cable connection status.

set RCR=00h to accept physical packet
set PAR0-5 to accept packet
set TCR=06h (CRC enabled)
set DCR=40h (8-bit slot) or 43h (16-bit slot)
clear ISR
tx a packet
check TSR after loopback

A. 10Base2

If cable OK, get TSR=03h (Tx OK).
If cable FAIL, get TSR=0Eh (Collision and Tx aborted).

B. 10Base5

If cable OK, get TSR=03h (Tx OK).
If MAU connected but cable FAIL, get TSR=0Eh (Tx collision and Tx aborted).
If MAU not connected, get TSR=53h (Carrier sense is lost during transmission and CD heartbeat fails.).

C. 10BaseT with link test disabled

RTL8019AS disables link test in this case, so cable OK or FAIL doesn't affect TSR; get TSR=03h.

D. Auto-detection (10BaseT with link test enabled)

RTL8019AS automatically switches from 10BaseT to 10Base 2 if the twisted-pair wire is not connected (10BaseT link test fails).

If twisted-pair wire OK, get TSR=03h (Tx OK) & BNC=0 in CONFIG2
If twisted-pair wire FAIL but coaxial cable OK, get TSR=03h (Tx OK) & BNC=1 in CONFIG2
Otherwise, get TSR=0Eh (same as 10Base2 connection fail).

7. Electrical Specifications and Timings

7.1. Absolute Maximum Ratings

Operating Temperature ..... 0 to 70
Storage Temperature ..... -65 to 140
All Outputs and Supply Voltages, with respect to Ground ..... -0.5V to 7V
Power Dissipation .....

Warning:
Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only.
Functionality at or above these limits is not recommended and extended exposure to "Absolute Maximum Ratings" may affect device reliability.

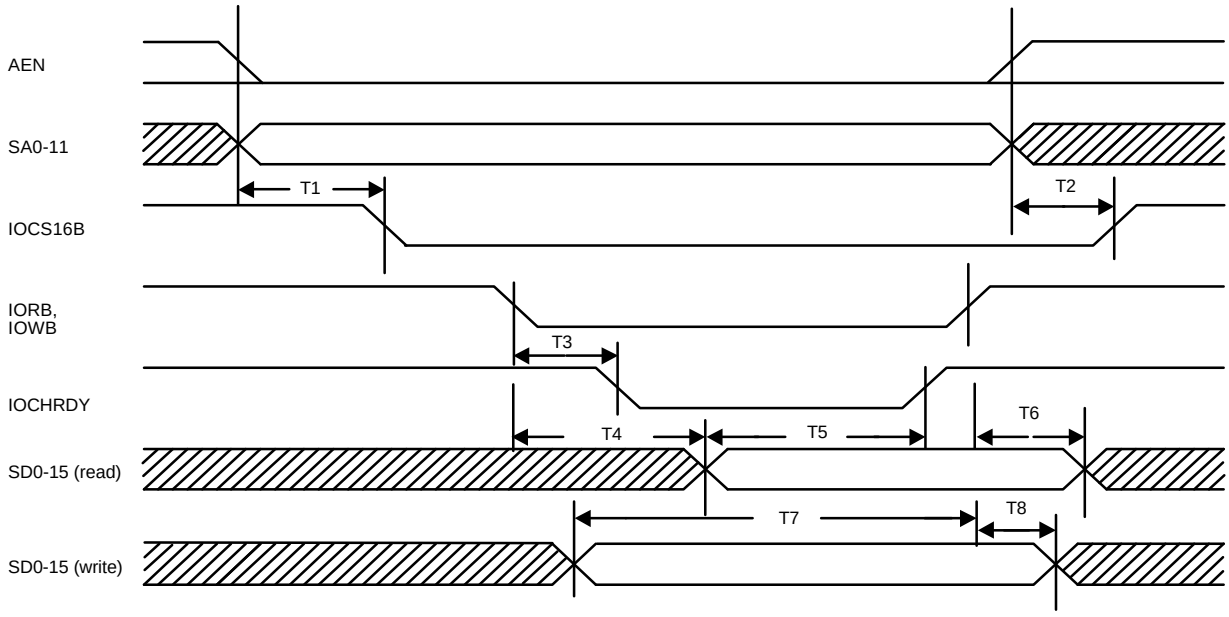
7.2. D.C. Characteristics (Tc=0 to 70 , Vcc=5V±5%)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
Vil	Input Low Voltage			0.8	V	
Vih	Input Low Voltage	2.0			V	
Vol1	Output Low Voltage 1		0.4	0.6	V	Iol=16mA, Note 1
Voh1	Output High Voltage 1	3.0	3.5		V	Ioh=8mA, Note 1
Vol2	Output Low Voltage 2		0.4	0.6	V	Iol=4mA, Note 2
Voh2	Output High Voltage 2	3.5	4.0		V	Ioh=4mA, Note 2
Vol3	Output Low Voltage 3			0.6	V	Iol=24mA, Note 3
Rpull-low	Internal Pull-Low Resistance	50	100	150	KW	
II	Input Leakage Current	-10		10	mA	

Note 1: Apply only to INT7 ~ INT0, SD15 ~ SD0.
Note 2: Apply only to MD7 ~ MD0, MA13 ~ MA0, LED Pins, EECS, MWRB, MRDB, BCSB.
Note 3: Apply only to IOCHRDY, IOCS16B

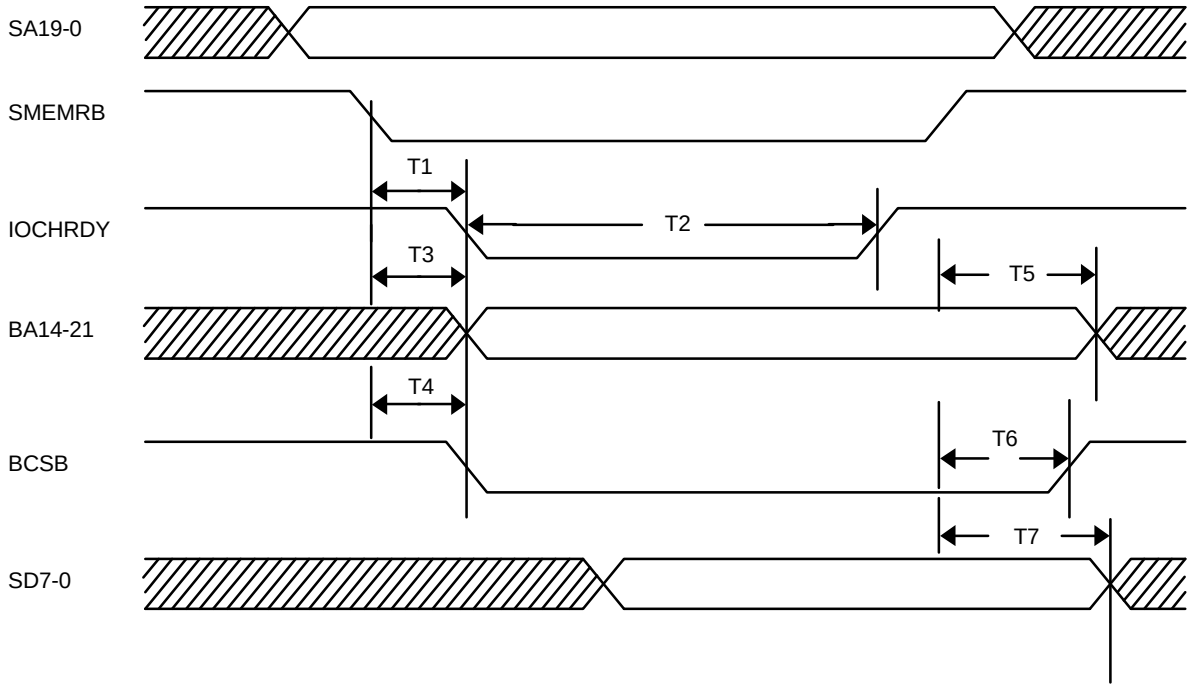
7.3. A.C. Timing Characteristics

(1) ISA I/O Read/Write



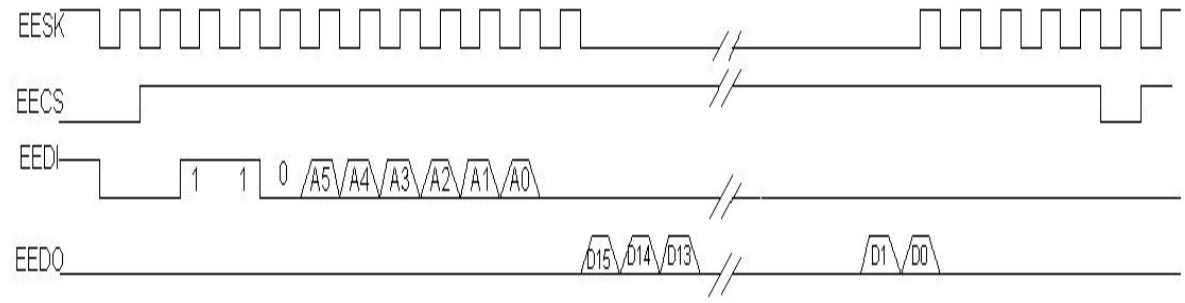
Symbol	Parameter	Min.	Typ.	Max.	Unit
T1	Host address valid to IOCS16B low	8	20	20	ns
T2	Host address invalid to IOCS16B high	4	30	--	ns
T3	IOCHRDY goes low from falling edge of IORB or IOWB when wait state insertion is needed.	--	50	50	ns
T4	Read data valid from falling edge of IORB or IOWB when no wait state insertion is needed.	--	50	60	ns
T5	Read data valid to IOCHRDY high when wait state is needed	25	--	--	ns
T6	Read data hold after IORB rising edge	10	30	30	ns
T7	Write data setup to IOWB rising edge	10	10	--	ns
T8	Write data hold from IOWB rising edge	10	10	--	ns

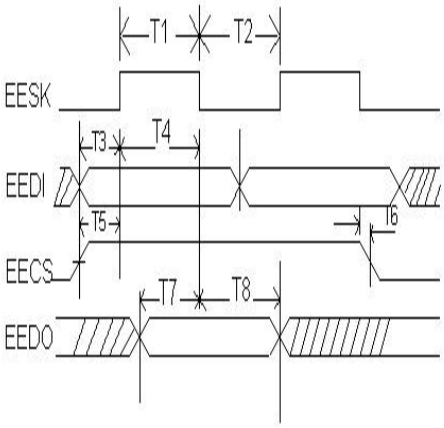
(2) BROM Read



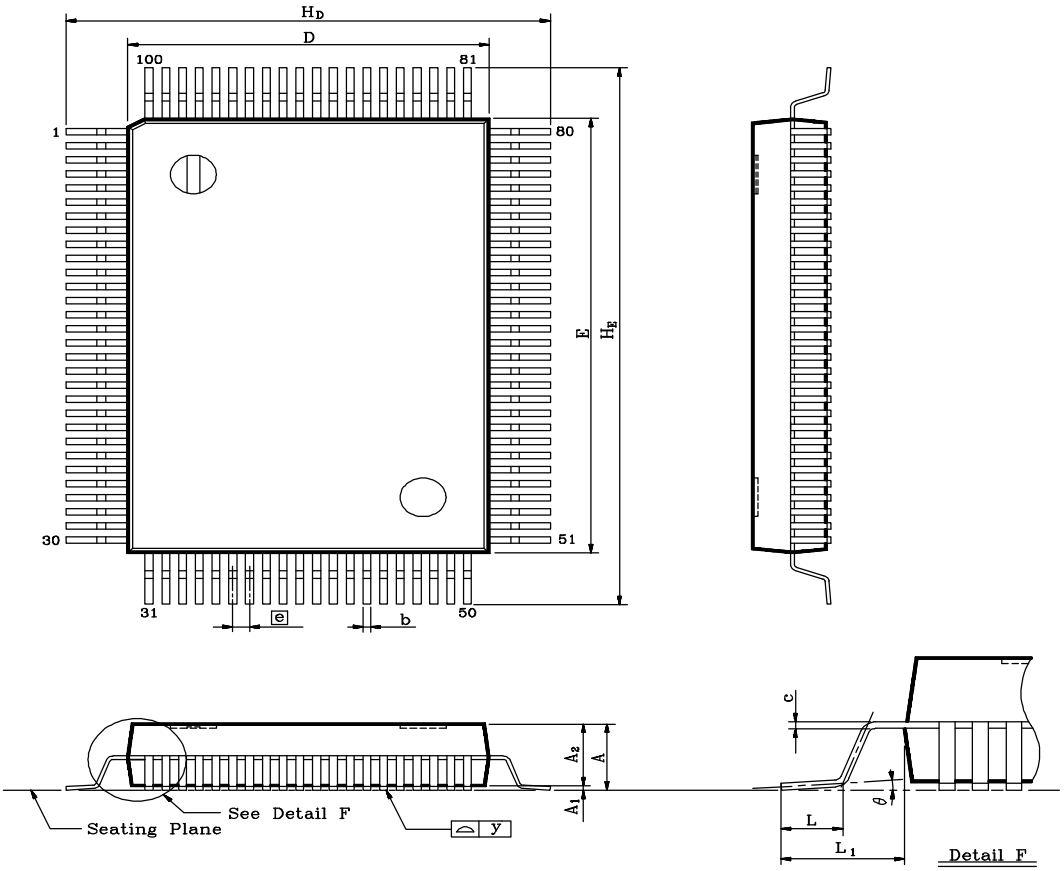
Symbol	Parameter	Min.	Typ.	Max.	Unit
T1	SMEMRB low to IOCHRDY low	-	-	30	ns
T2	IOCHRDY low width	125	200	350	ns
T3	SMEMRB low to BA14-21 valid	-	-	30	ns
T4	SMEMRB low to BCSB valid	-	-	30	ns
T5	BA14-21 hold from SMEMRB rising edge	-	-	30	ns
T6	BCSB hold from SMEMRB rising edge	-	-	30	ns
T7	Read data hold from SMEMRB rising edge	-	-	30	ns

(3) Serial EEPROM (9346) Auto-load





Symbol	Parameter	Min.	Typ.	Max.	Unit
T1	EESK high width	-	3.2	-	ms
T2	EESK low width	-	3.2	-	ms
T3	EEDI setup to EESK rising edge	3.0	-	-	ms
T4	EEDI hold from EESK rising edge	3.0	-	-	ms
T5	EECS goes high to EESK rising edge	3.0	-	-	ms
T6	EECS goes low from EESK falling edge	-	0	-	ns
T7	EEDO setup to EESK falling edge	20	-	-	ns
T8	EEDO hold from EESK falling edge	10	-	-	ns



Symbol	Dimension in mil			Dimension in mm		
	Min	Typ	Max	Min	Typ	Max
A	106.3	118.1	129.9	2.70	**3.00**	3.30
A_1	4.3	20.1	35.8	0.11	**0.51**	0.91
A_2	102.4	112.2	122.0	2.60	**2.85**	3.10
b	7.1	11.8	16.5	0.18	**0.30**	0.42
c	1.6	5.9	10.2	0.04	**0.15**	0.26
D	541.3	551.2	561.0	13.75	**14.00**	14.25
E	777.6	787.4	797.2	19.75	**20.00**	20.25
e	19.7	25.6	31.5	0.50	**0.65**	0.80
H_D	726.4	740.2	753.9	18.45	**18.80**	19.15
H_E	962.6	976.4	990.2	24.45	**24.80**	25.15
L	39.4	47.2	55.1	1.00	**1.20**	1.40
L_1	88.6	94.5	104.3	2.25	**2.40**	2.65
y			3.9			0.10
	0°		12°	0°		12°

Note:

1.Dimension D & E do not include interlead flash.
2.Dimension b does not include dambar protrusion/intrusion.
3.Controlling dimension: Millimeter
4.General appearance spec. should be based on final visual inspection spec.

TITLE : 100L QFP (14x20 mm**2) FOOTPRINT 4.8 mm PACKAGE OUTLINE DRAWING			
LEADFRAME MATERIAL:			
APPROVE		DWG NO.	
		REV NO.	
		SCALE	
CHECK	Ricardo Chen	DATE	
		SHT NO.	1 OF
REALTEK SEMI-CONDUCTOR CO., LTD			