Installation instructions for the Crynwr Packet Driver Collection

Document conventions

All numbers in this document are given in C-style representation. Decimal is expressed as 11, hexadecimal is expressed as 0x0B, octal is expressed as 013. All reference to network hardware addresses (source, destination and multicast) and demultiplexing information for the packet headers assumes they are represented as they would be in a MAC-level packet header being passed to the send_pkt() function.

Using the packet drivers

The packet driver must be installed prior to use. Since each packet driver takes only a few thousand bytes, this is best done in your autoexec.bat. Since the Ethernet boards typically have jumpers on board, the packet driver must be informed of the values of these jumpers (auto-configure is possible, but can disturb other boards). The first parameter is the entry point used to communicate with the packet driver. And again, because each board is different, the rest of the parameters will be different.

All parameters must be specified in C-style representation. The same number is expressed in decimal as 11, hexadecimal is expressed as 0x0B, octal is expressed as 013. Any numbers that the packet driver prints will be in the same notation.

Before installing the packet driver, you must choose an entry point (software interrupt) number in the range between 0x60 and 0x7e inclusive. Some people have reported trouble with dBase when using interrupts in the low 60's. These problems go away when they switch to an interrupt in the high 70's (e.g. 0x7e).

Interrupt 0x67 is unavailable because it's used by the EMS interface interrupt. Interrupts 0x70 through 0x77 are unavailable because the second interrupt controller uses them for IRQ 8 through IRQ 15. Interrupts 0x7f and 0x80 are unavailable because at least one package, when locating a packet driver, stops searching before 0x7f.

Running a packet driver with no specifications will give a usage message. The parameters for each packet driver are documented below.

Options

-d -- Most drivers can also be used in a PROM boot environment, see PROMBOOT.NOT for how to use -d and -n options for that purpose. This switch delays the adapter's initialization until the first time the packet driver is accessed.

-n -- NetWare can use two different framing types on Ethernet, "IEEE 802.3" and Ethernet II. The BYU packet driver shell requires Ethernet II. However, the Crynwr packet drivers can convert Ethernet II into Novell's version of IEEE 802.3 (and back) when the -n switch is used.

-p -- A certain small level of security can be achieved by disabling promiscuous mode with the -p switch. Do not mistake this for real security, however.

-w -- A switch used with Windows, obsoleted by the creation of winpkt. If you think you need the -w switch, or you used to run it, then consider running winpkt instead. Winpkt actually solves the problem that -w only attempts to solve. Winpkt (and -w) are only needed for non-resident DOS TCP stacks, e.g. NCSA Telnet, PC-Gopher, etc.

-i -- A switch used with client software that expects to find an IEEE 802.3 packet driver. Many Crynwr Ethernet packet drivers implement both IEEE 802.3 (class 11) and Ethernet II aka Bluebook (class 1) framing. The packet driver specification only allows a driver to report one class. The default is to report Ethernet II. Using -i switches the reported class to IEEE 802.3.

3Com 3C501

usage: 3C501 [options] packet_int_no [hardware_irq [io_addr]]

The 3c501 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

3Com 3C503

usage: 3C503 [options] packet_int_no [int_level(2-5) [io_addr [cable_type]]]

The 3c503 driver requires up to three additional parameters -- the hardware interrupt number, the I/O address, and the cable type. The 3c503 can be attached to thick or thin Ethernet cables, and the selection is made in software. The cable type is automatically determined at start-up, but may be forced to external transceiver (AUI/Thickwire) by specifying zero or internal transceiver (Thinwire/10BaseT/10Base2) or one for thin. The defaults are 2, 0x300, and 65535 (automatic). The 3c503 can use shared memory, but the driver automatically determines that parameter from the hardware.

3Com 3c505

usage: 3c505 [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The 3c505 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x300 and 0xd000.

3Com 3c507

usage: 3c507 [options] packet_int_no io_addr

The 3c507 will determine its parameters by reading the board. The only time you would need to specify the parameters is when you have multiple 3c507s in the same machine.

The 3c507 driver will use three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address.

3Com 3c509

usage: 3c509 [options] packet_int_no [id_port] | [io_port] | [board_num]

The 3c509 will determine its parameters by reading the board. The only time you would need to specify the parameters is when you have multiple 3c509s in the same machine, or if you have an I/O conflict with the default id_port (0x110).

The 3c509 driver will use three additional parameters -- the id port, or the I/O port, or the board number. If the number is between 0 and 0xff, it is the board number. If between 0x100 and 0x1ff, it is an ID port. Otherwise it is an I/O port number.

3Com 3c523

usage: 3c523 [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The 3c523 driver requires no additional parameters. It gets the board's parameters out of the Microchannel POS registers.

AQUILA

usage: aquila [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The AQUILA driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 3 and 0x360 and 0xd000.

ARCETHER

usage: arcether [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The ARCNET driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x2e0 and 0xd800. Note that a packet driver client must specifically support ARCNET. The only known client is Phil Karn's (KA9Q) networking package, NOS.

ARCNET

usage: arcnet [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The ARCNET driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x2e0 and 0xd800. Note that a packet driver client must specifically support ARCNET. The only known client is Phil Karn's (KA9Q) networking package, NOS.

AT1500

usage: at1500 [options] packet_int_no [io_addr]

The Allied Telesis AT1500 packet driver will automatically search for the adapter's I/O address. If you are using two boards, or the automatic search fails, then you should specify the proper I/O address.

AT1700

usage: at1700 [options] packet_int_no [io_addr]

The Allied Telesis AT1700 packet driver will automatically search for the adapter's I/O address. If you are using two boards, or the automatic search fails, then you should specify the proper I/O address.

AT&T

usage: at&t [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The AT&T driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000. This driver supports the StarLAN 1, StarLAN 10 NAU, EN100 and StarLAN Fiber NAU.

AT&T_LP 0x62 2 0x360 0xd000 0 0

usage: at&t_lp [options] <packet_int_no> [<hardware_irq> [<io_addr> [<base_addr> [<media_sel> [<li_enabl>]]]]]

The AT&T_LP driver requires five additional parameters -- the hardware interrupt number, the I/O address, the memory base address, media select, and link integrity. The defaults are 2 and 0x360 and 0xd000. This driver supports the ATStarStation, ATStarLAN 10 LanPACER+ NAU, ATStarLAN 10 LanPACER NAU and ATMicroelectronics T7231 evaluation board.

The final two numbers are new attributes.

0 0 --> This is for AUI setting
0 1 --> This is also for AUI setting
1 0 --> This is for TP setting, no link integrity
1 1 --> This is for TP setting, link integrity enabled

For the LP NAU only, "0 0" and "0 1" are invalid as there is no AUI port on that NIC.

David Systems Inc (DSI)

usage: davidsys [options] <packet_int_no> <hardware_irq> <io_addr> <delay_mult>

The DSI driver requires three additional parameters, the hardware interrupt number, the I/O port and the delay multiplier. Delay_mult is a system dependent timing loop used for I/O to the card. A reasonable value is calculated during initialization, but on some fast systems it may need to be somewhat larger. The multipler is divided by ten, then multiplied by the calculated delay. The default multiplier is 10 (actually 1.0).

D-Link DE-600

usage: de600 [options] packet_int_no

The D-Link Pocket Lan Adapter packet driver requires no additional parameters.

Digital Equipment Corporation DEPCA

usage: depca [options] <packet_int_no> [<hardware_irq> [<io_addr> [<mem_addr>]]]

The DEPCA packet driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x300 and 0xd000. The packet driver will resolve the io_addr automatically if io_addr is set to '?', e.g. depca 0x7e 5 ? 0xd000. The driver requires that you set the jumpers to enable the boot prom.

Digital Equipment Corporation VAXMATE

usage: vaxmate [options] <packet_int_no> [<hardware_irq> [<io_addr> [<mem_addr>]]]

The VAXMATE packet driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x300 and 0xd000. The packet driver will resolve the io_addr automatically if io_addr is set to '?', e.g. depca 0x7e 2 ? 0xd000.

EtherSLIP

usage: ethersl [options] packet_int_no [-h] [hardware_irq]
    [io_addr] [baud_rate] [send_buf_size] [recv_buf_size]

The EtherSLIP driver is a simulated Ethernet adapter. It appears to the application software to be an Ethernet driver, but it transmits and receives SLIP packets on the serial line.

The parameters are as follows. The -h flag is included if you wish to use hardware handshaking (the packet driver will then suspend the transmission of characters while CTS is low). The hardware_irq is the hardware interrupt number, defaults to 4 (COM1). The io_addr is the hardware I/O address, defaults to 0x3f8 (COM1). The baud_rate defaults to 4800 baud. The send_buf_size and recv_buf_size default to 3000 each.

Fujitsu dk86960.com

usage: dk86960 [options] packet_int_no [hardware_irq [io_addr]]

The dk86960 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

Fujitsu dk86965.com

usage: dk86965 [options] packet_int_no

The dk86965 driver requires no additional parameters. It always searches for the proper I/O address.

HP Ethertwist

usage: hppclan [options] packet_int_no [hardware_irq [io_addr]]

The hppclan driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

IBM Token Ring

usage: ibmtoken [options] packet_int_no [adapter_no]

The IBM Token Ring packet driver requires one additional parameters -- the adapter number. The default is zero. See IBMTOKEN.DOC for more information.

ICL EtherTeam16

usage: ETHIIE [options] packet_int_no [int_level [io_addr [cable_type]]]

The ETHIIE driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the cable type. The interrupt levels supported by the adapter are 5, 9 (2), 12 and 15. The Ethenet IIe can be attached to thick or thin Ethernet cables, and the selection is made in software. The cable type parameter should be zero for thick, and one for thin. With the Twisted Pair (TP) version of the adapter, you must set interface to the value 1 (thin).

The defaults are 9 (2), 0x300 and 1 (thin).

Please note, that the adapter can be used only in a 16-bit slot of your computer.

Intel EtherExpress

usage: exp16 [options] <packet_int_no> [<io_addr>]

The Intel EtherExpress packet driver has one optional parameter. The <io_addr> is only needed if there is more than one EtherExpress card in your system. Otherwise, the driver will search for adapter and get its parameters from it.

BICC Data Networks' ISOLAN 4110 ethernet

usage: isolan [options] packet_int_no [hardware_irq [base_addr]]

The BICC Isolan requires two additional parameters -- the hardware interrupt number and the memory base address. The defaults are 2 and 0xb800h.

BICC Data Networks' ISOLAN 4112/3 ethernet

usage: isolink [options] packet_int_no [hardware_irq [dma_no [io_addr]]]

The BICC Isolan requires three additional parameters -- the hardware interrupt number, the dma channel number and the I/O base address. The defaults are 10 and 0, and to automatically search.

Kodiak Raven and Kombo

usage: kodiak8 [options] packet_int_no [hardware_irq [io_addr]]
usage: kodiak16 [options] packet_int_no [hardware_irq [io_addr]]
usage: kodiakk [options] packet_int_no [hardware_irq [io_addr]]

The Kodiak drivers require three additional parameters -- the hardware interrupt number and the I/O base address. The defaults are 2 and 0x300.

LocalTalk

usage: localtlk [options] <packet_int_no> [<IP address>]

The LocalTalk packet driver requires atalk.sys to be installed. Because it is not an Ethernet class driver, it requires special code in the client. See LOCALTLK.NOT for more details.

Microdyne EXOS205T

usage: exos205 [options] <packet_int_no> [hardware_irq] [io_addr] [base_addr]

This Packet Driver supports the EXOS205T with 256K or 512K Byte RAM. It has not been tested with the old EXOS205E with 128K Byte.

Where the last three arguments are optional. If you do not supply them, the driver uses '0x02 0x310 0xcc00', these are the EXOS defaults.

The Interrupt must be set by a jumper on the card. The Packet Driver does not check for a valid setting. Possible values are 2 (default), 3, 4, 5, 6, 7, 10, 11, 12, 13 and 14.

Five bytes of i/o address space are used by the EXOS205. A jumper on the EXOS205 board sets the starting address. Possible values are 0x300, 0x310 (default), 0x320 and 0x330. The Packet Driver fails if it does not find an EXOS205 card at the specified address.

The EXOS205 uses a shared memory to interface the Intel 82586 Ethernet chip to the host's address space. The EXOS205 memory can be 256 K or 512 K large. The Packet Driver uses a 16 K Byte window to access the EXOS205 memory. The location of this window is set by software. The following segments are possible:

0xa000 0xc000 0xc400
0xc800 0xcc00 (default) 0xd000
0xd400 0xd800 0xdc00

If you install a BOOT-PROM on the EXOS205 take care that you do not use the same address for the PROM and for the shared memory.

The SQE check jumper is ignored by the EXOS205 Packet Driver.

Mitel Express

usage: express [options] <packet_int_no> [-n] [<driver_class> [<hardware_irq>]]

The Mitel Express packet driver has one optional switch, and two optional parameters. The <driver_class> defaults to SLIP, and the <hardware_irq> defaults to 7. The -n switch instructs card to be an NT. The <driver_class> should be SLIP or a number.

Multitech EN-301

usage: en301 [options] packet_int_no [hardware_irq [io_addr]]

The Multitech driver runs the EN-301 cards. The Multitech driver requires two additional parameters, the hardware interrupt number, and the I/O port.

Mylex LNE-390B

usage: mylex [options] packet_int_no [int_level [io_addr [mem_base]]]

The Mylex driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are pulled out of the EISA configuration registers for the first board found.

NCR ET-105

usage: ncret105 [options] <packet_int_no> <hardware_irq> <base_addr> <Ethernet_address>

The NCR ET-105 driver requires four additional parameters -- the hardware interrupt number, the I/O address, the memory base address, and the Ethernet address. The Ethernet address assigned to any particular board is printed on sticky labels that come with the board.

Netbios

usage: nb [options] packet_int_no ip.ad.dr.ess [receive queue size]

The netbios packet driver transports IP packets over NetBIOS.

Novell IPX

usage: ipxpkt [options] packet_int_no [-n [no_bytes]]

The ipxpkt packet driver simulates Ethernet on Novell IPX protocol.

Novell ne/2

usage: ne2 [options] <packet_int_no>

The ne/2 driver requires no additional parameters.

Novell ne1000

usage: ne1000 [options] packet_int_no [hardware_irq [io_addr]]

The ne1000 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

Novell ne2000

usage: ne2000 [options] packet_int_no [hardware_irq [io_addr]]

The ne2000 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 2 and 0x300.

Novell ne2100 and ne1500

usage: ne2100 [options] packet_int_no [hardware_irq [io_addr [dma_no]]]

The ne2100 Ethernet card is software compatible with the ne1500 card. The ne2100 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the DMA channel number. The defaults are 3, 0x300, and 5.

Racal-Interlan (Formerly Interlan) ES3210

usage: es3210 [options] packet_int_no [int_level [io_addr [mem_base]]]

The es3210 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. There are no defaults.

Racal-Interlan (Formerly Interlan) NI5010

usage: NI5010 [options] packet_int_no [hardware_irq [io_addr]]

The NI5010 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

Racal-Interlan (Formerly Micom-Interlan) NI5210

usage: ni5210 [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The NI5210 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000.

Racal-Interlan NI6510

usage: ni6510 [options] packet_int_no [hardware_irq [io_addr]]

The ni6510 driver has two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 2 and auto-sense. These parameters do not need to be set unless the auto-sense routine fails, or otherwise disrupts operation of your PC.

Racal-Interlan (Formerly Micom-Interlan) NI9210

usage: ni9210 [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The ni9210 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000.

NTI 16

usage: nti16 [options] packet_int_no [hardware_irq [io_addr [base_addr]]]

The nti16 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 3 and 0x338 and 0xd000.

SLIP8250

usage: SLIP8250 [options] packet_int_no [-h] [driver_class] [hardware_irq]
    [io_addr] [baud_rate] [recv_buf_size]
    The driver_class should be SLIP, KISS, AX.25, or a
    number.

The SLIP8250 driver is not strictly an Ethernet adapter, however some software packages (such as KA9Q's NET and NCSA Telnet) support Serial Line IP (SLIP). SLIP must be specially supported because it doesn't use ARP and has no hardware addresses prepended to its packets. The PDS is not clear on this, but the packet driver does the SLIP encoding.

The parameters are as follows. The -h flag is included if you wish to use hardware handshaking (the packet driver will then suspend the transmission of characters while CTS is low). The driver_class is the class that is returned to a client of the packet driver spec in the driver_info call. The hardware_irq is the hardware interrupt number, defaults to 4 (COM1). The io_addr is the hardware I/O address, defaults to 0x3f8 (COM1). The baud_rate defaults to 4800 baud. The recv_buf_size defaults to 3000.

Thomas-Conrad tcenet

usage: tcenet [options] packet_int_no [int_no [io_addr]]

The tcenet driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and autosense.

Ungermann-Bass NIC-PC

usage: ubnicpc [options] <packet_int_no> <hardware_irq> <base_addr>

The UB NIC-PC driver requires two additional parameters, the hardware interrupt number, and the memory base address.

Ungermann-Bass NIC-PS/2

usage: ubnicps2 [options] <packet_int_no> <hardware_irq> <io_addr> <base_addr>

The UB NIC-PS/2 requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are the contents of the POS registers, so the only time you would need to use the parameters is if you're using two NIC-PS/2 boards in one machine.

SMC (formerly Western Digital) (also IBM) SMCWD

```
usage: smc_wd [options] packet_int_no [-o] [int_level [io_addr [mem_base]]]
```

The SMC_WD driver runs the SMC (formerly Western Digital) E, EBT, EB, ET/A, and E/A Ethernet cards (but not the Ultra), and also on the IBM Microchannel Ethernet cards with POS ID's 0xEFE5, 0xEFD4 and 0xEFD5. The ISA SMC_WD requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The ISA defaults are 3 and 0x280 and 0xd000. The MCA SMC_WD picks up its default parameters from the POS registers, so you only need specify them it you have multiple adapters. The smc_wd cards do not enable their memory until configuration time. Some 386 memory mappers will map memory into the area that the card intends to use. You should be able to configure your software to leave this area of memory alone. Also the driver will refuse to map memory into occupied memory. The occupied memory test fails on some machines, so the optional switch -o allows you to disable the check for occupied memory.

If you get the error "PROM ADDRESS Invalid", use EZSETUP to set all the parameters again (to the same values). Occasionally wayward programs will write to locations that don't belong to them. This can corrupt the EEPROM checksum on the card. EZSETUP will restore the correct checksum.

Tiara Lancard

```
usage: tiara [options] packet_int_no [hardware_irq [io_addr]]
```

The Tiara driver runs the Tiara LANCARD/E cards, both eight and sixteen bit cards. The Tiara driver requires two additional parameters, the hardware interrupt number, and the I/O port.

Zenith Data Systems Z-Note

```
usage: znote [options] packet_int_no
```

The Z-Note packet driver also works on the IBM Thinkpad 300. The Z-Note packet driver has no parameters beyond the packet driver software interrupt number. It picks up its parameters from the bios. This driver also turns the hardware on when it starts, and off when it exits, so you do not need to enable the adapter. In fact, you should leave it disabled, so that you save the power when the driver is not installed.

Errorlevels

Some of the packet drivers return error codes. Some of these error codes indicate fatal errors, and some are merely warnings. For the moment, you must consult the source to see what the errorcodes mean. For example, pktchk returns 0 if a packet driver exists at a given address, and 1 if not. You might use it in a batch file that only installs a packet driver if one is not found.

```
    rem only install the packet driver if there isn't one
    rem already.
    pktchk 0x7e
    if errorlevel 0 goto gotit
    ni5210 0x7e

:gotit
```

The "errorlevel" test is true if the errorlevel is less than or equal to the parameter.

Utility Programs

There are also several utility programs for packet drivers:

PKTADDR

usage: pktaddr packet_int_no [ethernet_addr]

If the second argument is given, the Ethernet address of the given packet driver is set. The Ethernet address is printed out.

PKTALL

usage: pktall packet_int_no [-v] [-p] [-a et:he:rn:et:ad:dr]

All packets are received and discarded from the given packet driver. This program is of most use with PKTMODE and TRACE. The -v switch causes the packet contents to be printed. The -p switch causes the driver to enter promiscuous mode (receives all packets regardless of destination address). The -a switch lets you filter out all but a specific address.

PKTCHK

usage: pktchk packet_int_no [packet_int_no]

Test for existance of a packet driver. Returns with errorlevel 0 if the specified interrupt has a packet driver. If the second argument is given, all interrupts in the range are checked for a packet driver. If no packet driver is found at all, errorlevel 1 is returned.

PKTMODE

usage: pktmode packet_int_no [receive_mode]

If the second argument is given, the receive mode of the given packet driver is set. A decimal number from the list of modes should be used. All the possible modes are printed out. Unimplemented modes are marked with "xx", and the current mode is marked with "->".

PKTMULTI

usage: pktmulti packet_int_no [-f filename | address ...]

The specified addresses are set as allowed multicast addresses. If no list of addresses is given, then the current list of addresses is printed. The addresses may either be specified on the command line, or in a file using
the -f option. When a file is used, any whitespace in the file is ignored.

PKTSTAT

usage: pktstat first_int_no [last_int_no]

The statistics for all packet drivers in the given range are printed. The default range is 0x60 through 0x80. The meanings of the columns are given below.

pkt_in	is the number of packets ever received by this driver.
pkt_out	is the number of packets ever transmitted by this driver.
byt_in	is the number of bytes ever received by this driver.
byt_out	is the number of bytes ever transmitted by this driver.
pk_drop	Packets dropped because there was no handler for that Ethernet packet type.

err_in	Dependent upon the packet driver.
err_out	Dependent upon the packet driver.

PKTSEND

usage: pktsend packet_int_no [-r] [-d delay] [-f filename | packet]

The specified packet is sent using the specified packet driver. The -r option says to repeat sending as fast as possible. You shouldn't use this option very often. The -d option inserts a system-dependent delay between sending packets. Without -r, the program waits for a key before sending a packet. The packet may either be specified on the command line, or in a file using the -f option. When a file is used, any whitespace in the file is ignored.

PKTTRAF

usage: pkttraf packet_int_no

Graphically display traffic on an EGA or VGA screen. The first twenty Ethernet addresses encountered are assigned a node number. The traffic between each pair of nodes is displayed as a line of varying intensity. When any line reaches maximum intensity, the intensities of all lines are halved.
A cursor highlights one of the nodes. The Ethernet address of the highlighted node is printed in the lower-right corner. The cursor is moved using space and backspace.

PKTWATCH

usage: pktwatch packet_int_no [-a et:he:rn:et:ad:dr]

Pktwatch runs the driver in promiscuous mode, and prints all packets recieved on the screen. The -a switch lets you filter out all but a specific address.

TERMIN

usage: termin [-s] packet_int_no

The specified packet driver is terminated, and its memory recovered.

The s-option (stop) is used to prepare for termination. The in-use flag for all handles are cleared. This prevents upcalls to handlers that are to be removed and also makes it possible to later terminate the packet driver even though handles are not released. Actually, doing termin -s after prom boot is like cutting the branch you are sitting on. Recipe for removing packet driver, IPX and NET:

```
pktdrvr 0x7c ....
MARKNET C:\IPX.MRK
PDIPX
NET3
. . .
NET3 u                  ; unload netx to avoid communication timeout
TERMIN -s 0x7c          ; pkt drvr no longer calls any nonexistent rcvrs
RELNET C:\IPX.MRK       ; IPX is "removed"
TERMIN 0x7c             ; It is now safe to terminate the packet driver
```

TRACE

usage: trace packet_int_no [buffer_size]

Trace is very useful for debugging packet driver troubles. Trace lets you trace all transactions between a user program and the packet driver. The transactions are stored in a memory buffer whose size is set with buffer_size. The default size is 10,000 bytes.

When you run trace, it sets itself up and then spawns COMMAND.COM so that you can run a network program that uses the packet driver. After you quit your network session, you issue an "EXIT" command. This returns you to trace, which writes the transaction log to "TRACE.OUT". The following program, DUMP, interprets TRACE.OUT.

DUMP

usage: dump

Interprets the contents of TRACE.OUT as written by TRACE.

WINPKT

usage: winpkt <packet_int_no>

Provides a Packet Driver interface between Windows 3 Enhanced mode applications and a real Packet Driver. This attempts to solve the problem of Windows moving DOS applications around in memory willy nilly. It replaces the -w flag hack. Winpkt (and -w) are only needed for non-resident DOS TCP stacks, e.g. NCSA Telnet, PC-Gopher, etc.

Previous versions of winpkt had two parameters and required that you use different interrupts for the virtual packet driver and the real packet driver. This caused confusion when the software used the wrong packet driver. This version requires that you use the same packet_int_no as the existing packet driver.

Install WINPKT after the Packet Driver and before starting Windows.

Appendix A

Interrupt usage in the range 0x60 through 0x80, from Ralf Brown's interrupt list.

```
60 -- -- reserved for user interrupt
60 -- -- FTP Driver - PC/TCP Packet Driver Specification
60 01 FF FTP Driver - DRIVER INFO
60 02 -- FTP Driver - ACCESS TYPE
60 03 -- FTP Driver - RELEASE TYPE
60 04 -- FTP Driver - SEND PACKET
60 05 -- FTP Driver - TERMINATE DRIVER FOR HANDLE
60 06 -- FTP Driver - GET ADDRESS
60 07 -- FTP Driver - RESET INTERFACE
60 11 -- 10-NET - LOCK AND WAIT
60 12 -- 10-NET - LOCK
60 13 -- 10-NET - UNLOCK
60 20 -- FTP Driver - SET RECEIVE MODE
60 21 -- FTP Driver - GET RECEIVE MODE
60 24 -- FTP Driver - GET STATISTICS
61 -- -- reserved for user interrupt
62 -- -- reserved for user interrupt
63 -- -- reserved for user interrupt
64 -- -- reserved for user interrupt
65 -- -- reserved for user interrupt
66 -- -- reserved for user interrupt
67 -- -- LIM EMS
          ...
67 DE 00 Virtual Control Program Interface - INSTALLATION CHECK
          ...
68 01 -- APPC/PC
          ...
69 -- -- unused
6A -- -- unused
6B -- -- unused
6C -- -- system resume vector (CONVERTIBLE)
6C -- -- DOS 3.2 Realtime Clock update
6D -- -- VGA - internal
6E -- -- unused
6F -- -- Novell NetWare - PCOX API (3270 PC terminal interface)
6F 00 -- 10-NET - LOGIN
          ...
70 -- -- IRQ8 - AT/XT286/PS50+ - REAL-TIME CLOCK
71 -- -- IRQ9 - AT/XT286/PS50+ - LAN ADAPTER 1
72 -- -- IRQ10 - AT/XT286/PS50+ - RESERVED
73 -- -- IRQ11 - AT/XT286/PS50+ - RESERVED
74 -- -- IRQ12 - PS50+ - MOUSE INTERRUPT
75 -- -- IRQ13 - AT/XT286/PS50+ - 80287 ERROR
76 -- -- IRQ14 - AT/XT286/PS50+ - FIXED DISK
77 -- -- IRQ15 - AT/XT286/PS50+ - RESERVED
78 -- -- not used
79 -- -- not used
7A -- -- Novell NetWare - LOW-LEVEL API
7A -- -- AutoCAD Device Interface
7B -- -- not used
7C -- -- not used
7D -- -- not used
7E -- -- not used
```

7F -- -- HDILOAD.EXE - 8514/A VIDEO CONTROLLER INTERFACE
7F -- -- HLLAPI (High-Level Language API)
80 -- -- reserved for BASIC