Writing Drivers for the DP8390 NIC Family of Ethernet Controllers

National Semiconductor
Application Note 874
July 1993

INTRODUCTION

This document provides detailed information for writing drivers for the NIC Family of Ethernet Controllers; DP8390 NIC, DP83901 SNIC™, DP83902 ST-NIC™, and DP83905 AT/LANTIC™. It describes the basic components of the drivers: (1) hardware initialization, (2) initiating transmissions, and (3) servicing receive and transmit interrupts. It includes specific examples of actual network drivers (DriverInitialize, DriverSend, and DriverISR). **We recommend that you become familiar with the individual part Datasheets**.

HARDWARE INITIALIZATION

The initialization procedure supplies configuration parameters for the NIC Controllers to operate in the current system. This involves the CPU loading the proper values into the configuration and address registers and enabling the NIC Controllers onto the network. The following shows a list of parameters that must be initialized before the NIC Controllers become operational.

— data bus width (8 or 16 bits)

— physical address

— types of interrupts that may be serviced

— size of the Receive Buffer Ring

— FIFO threshold

— types of packets that may be received

An example of an initialization routine for a typical 8-bit system is exemplified in DriverInitialize. Note that the DATA CONFIGURATION register must be initialized before all other registers are initialized (except the COMMAND register). Note also the sequencing to enable the DP83902 and DP83905 onto the network.

PACKET TRANSMISSION

The transmit driver is generally partitioned into two parts. The first part (DriverSend) initiates a transmission whenever the upper level software passes a packet to the driver. If the driver is unable to transmit the packet immediately (i.e., the transmitter is busy), the supplied packet is queued in a transmit-pending buffer. After initiating or queuing up the packet, DriverSend returns.

DriverSend operates in conjunction with an interrupt service routine (DriverISR). After completing the transmission, the NIC Controllers interrupt the CPU to signal the end of the transmission and indicate status information in the TRANSMIT STATUS register.

RECEIVE DRIVER

The responsibility of the receive driver is to transfer data from the Receive Buffer Ring to the host's memory. Ideally, this process is done as fast as possible to eliminate any bottlenecks that may be incurred by the driver. The NIC Controllers facilitate removing data from the Ring by providing a Remote DMA channel to transfer data from the Ring to an I/O port which is readable by the host system. It also maintains two pointers to track packets in the Ring: BOUNDARY and CURRENT. These registers respectively point to the last unread packet in the Ring and the next vacant location in memory to receive another packet. Generally, the receive driver removes the next packet pointed to by BOUNDARY, then increments BOUNDARY to the succeeding packet indicated by the Next Page Pointer in the 4-byte NIC Controllers receive header. This process continues until all packets have been removed from the Ring.

The NIC Controllers automate packet removal with the "send packet" command. When this command is issued, the NIC Controllers automatically load the DMA start address with BOUNDARY, load the DMA byte count from the 4-byte receive header, then begin transferring data. At the end of the DMA, the NIC Controllers update BOUNDARY with the Next Page Pointer from the receive header. To remove all packets from the Ring, the receive driver simply issues the "send packet" command until the BOUNDARY and CURRENT registers are equal.

Because of the asynchronous nature of reception, the receive driver must be interrupt driven. Typically, packet reception is given high priority since delaying packet removal may overflow the Receive Buffer Ring. If several packets in the ring have been queued, all packets should be removed in one process (i.e., a software loop which empties the Ring). In heavy traffic conditions, local memory can fill up quickly so it is important that the Ring be large enough to handle these situations.

To find out how many packets are lost due to Ring overflows or network errors, the NIC Controllers have three statistical registers to monitor the network; FRAME ALIGNMENT ERROR tally, CRC ERROR tally, and FRAMES LOST tally. These registers are useful in initially determining the size of the Ring and how many packets are lost due to network related errors (CRC errors and/or frame alignment errors).

EXAMPLE DRIVERS

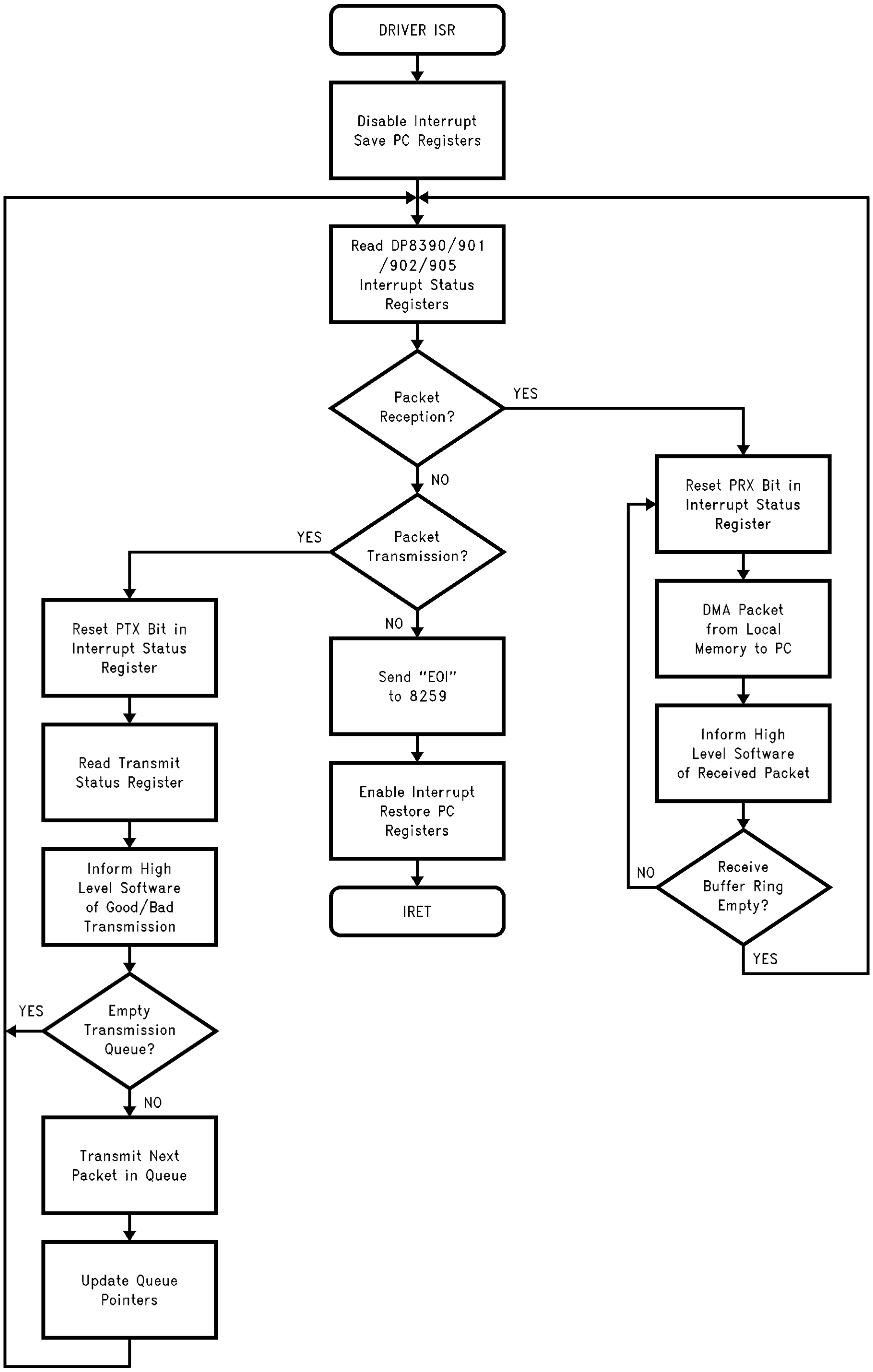
The following transmit and receive drivers are written in assembly for fast execution. The transmit driver is partitioned into two parts, DriverSend and DriverISR, while the receive driver resides entirely within DriverISR. This section gives an overview of DriverISR, followed by a description on how receive and transmit interrupts interact with DriverISR.

Interrupt Service Routine (DriverISR)

DriverISR is concerned with interrupts originating from receptions, transmissions, and errored transmissions. Errored receptions are ignored since these are usually collision fragments and are of no use to the upper layer software. DriverISR *(Figure 2)* consists of (1) a packet transmitted routine and (2) a packet received routine. The basic functions of the routines are as follows:

Packet Transmitted Routine: checks the status of all transmissions and transmits the next packets in the transmit-pending queue.

AT/LANTIC™, SNIC™ and ST-NIC™ are trademarks of National Semiconductor Corporation.


DRIVER ISR
Disable Interrupt Save PC Registers
Read DP8390/901 /902/905 Interrupt Status Registers
Packet Reception?
YES
NO
Packet Transmission?
YES
NO
Reset PTX Bit in Interrupt Status Register
Read Transmit Status Register
Inform High Level Software of Good/Bad Transmission
Empty Transmission Queue?
YES
NO
Transmit Next Packet in Queue
Update Queue Pointers
Send "EOI" to 8259
Enable Interrupt Restore PC Registers
IRET
Reset PRX Bit in Interrupt Status Register
DMA Packet from Local Memory to PC
Inform High Level Software of Received Packet
Receive Buffer Ring Empty?
NO
YES

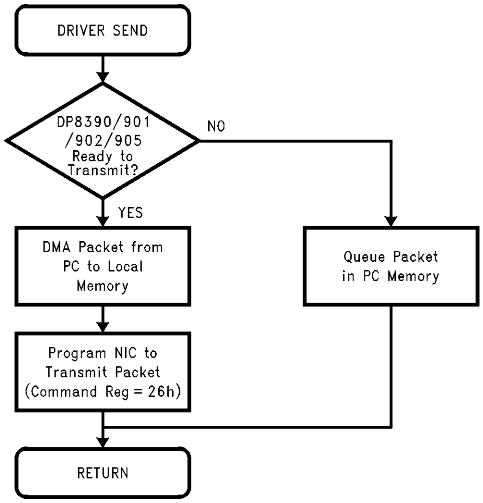
TL/F/11785–2

FIGURE 2. Interrupt Service Routine

Packet Received Routine: removes all packets in the receive buffer ring by using the "send packet" command of the NIC Controllers.

Transmit Driver

The transmit drivers consist of two parts. The first part, DriverSend *(Figure 3)*, initiates transmission when called by the upper layer software. DriverSend checks if the NIC Controllers are ready to transmit by reading the COMMAND register (TXP bit is zero). If ready, the DriverSend using the DP8390's Remote DMA channel, transfers from the PC's memory to local memory, then issues the transmit command and returns. Otherwise, if the NIC Controllers are busy (TXP bits equal one) DriverSend queues the packet in the transmit-pending queue, then returns.


DRIVER SEND
DP8390/901 /902/905 Ready to Transmit?
NO
YES
DMA Packet from PC to Local Memory
Queue Packet in PC Memory
Program NIC to Transmit Packet (Command Reg = 26h)
RETURN

TL/F/11785–3

FIGURE 3. Driver Send Routine

After a transmission is completed, DriverISR services the interrupt from the NIC Controllers and (1) reports status information by reading the TRANSMIT STATUS register and (2) transmits the next packet in the transmit-pending queue, if any. Thus, for a transmit interrupt, DriverISR executes the following steps:

1. Reset PTX bit in INTERRUPT STATUS register.
2. Check for good transmission by reading the TRANSMIT STATUS Register.
3. If there are more packets in the transmit-pending queue, transmit the next packet; otherwise go to 4.
4. Read INTERRUPT STATUS register for any pending interrupts.

Receiver Driver

Since the receiver driver must be interrupt driven, it resides completely within the DriverISR. When the receive interrupt occurs, one or more packets may be buffered into the Ring by the NIC Controllers. The DriverISR removes packets from the Ring and then passes them up to the host. Using the "send packet" command, packets are removed until the Ring is empty, that is, when CURRENT and BOUNDARY registers are equal. The sequence of the receive packet routine is shown below.

1. Reset the PRX bit in the INTERRUPT STATUS register.
2. Remove the next packet in the receive buffer using the "send packet" command.
3. Check to see if the receive buffer ring is empty: BOUNDARY register = CURRENT PAGE register
4. If the Ring is not empty, go to 1; otherwise read INTERRUPT STATUS register for any more pending interrupts.

OTHER SOFTWARE CONSIDERATIONS

The NIC Ethernet Controllers require some special software considerations to operate in all network environments. In particular, the handling of overflow of the receive buffer ring must be handled EXACTLY as described in the data sheet and Design Tips.

The most efficient manner to remove packets from the transmit-pending queue is to use Driver Send to initiate transmission of the very first packet in the queue; then upon completion, use the DriverISR to transmit the remaining packets. Using this method, the DriverISR examines the queue, transmits the next available packet, then exits. The DriverISR transmits the next packet after the NIC Controllers issue the next transmit interrupt.

```
;***********************************************************************
                DriverInitialize

;  Initializes the NIC for a typical network system.
;  Receive Buffer Ring = 2600h to 4000h
;  Transmit Buffer     = 2000h to 2600h
;
;        Entry:  none
;***********************************************************************
;***********************Equates for NIC Registers***********************

COMMAND                 equ    300h
PAGESTART               equ    COMMAND+1
PAGESTOP                equ    COMMAND+2
BOUNDARY                equ    COMMAND+3
TRANSMITSTATUS          equ    COMMAND+4
TRANSMITPAGE            equ    COMMAND+4
TRANSMITBYTECOUNT0      equ    COMMAND+5
NCR                     equ    COMMAND+5
TRANSMITBYTECOUNT1      equ    COMMAND+6
INTERRUPTSTATUS         equ    COMMAND+7
CURRENT                 equ    COMMAND+7         ;in page 1
REMOTESTARTADDRESS0     equ    COMMAND+8
CRDMA0                  equ    COMMAND+8
REMOTESTARTADDRESS1     equ    COMMAND+9
CRDMA1                  equ    COMMAND+9
REMOTEBYTECOUNT0        equ    COMMAND+0ah
REMOTEBYTECOUNT1        equ    COMMAND+0bh
RECEIVESTATUS           equ    COMMAND+0ch
RECEIVECONFIGURATION    equ    COMMAND+0ch
TRANSMITCONFIGURATION   equ    COMMAND+0dh
FAE_TALLY               equ    COMMAND+0dh
DATACONFIGURATION       equ    COMMAND+0eh
CRC_TALLY               equ    COMMAND+0eh
INTERRUPTMASK           equ    COMMAND+0fh
MISS_PKT_TALLY          equ    COMMAND+0fh

PSTART                  equ    46h
PSTOP                   equ    80h

CGroup  group  Code
Code    segment para public 'Code'
        assume  cs:CGroup, ds:CGroup, es:nothing, ss:nothing

rcr   db      0               ;value for Recv config. reg
tcr   db      0               ;value for trans. config. reg
dcr   db      58h             ;value for data config. reg
imr   db      0bh             ;value for intr. mask reg
```

```
;***********************************************************************
DriverInitialize proc near
      public      DriverInitialize

      mov   al,21h                        ;stop mode
      mov   dx,COMMAND
      out   dx,al

      mov   al,dcr
      mov   dx,DATACONFIGURATION          ;data configuration register
      out   dx,al

      mov   dx,REMOTEBYTECOUNT0
      xor   al,al
      out   dx,al                         ;low remote byte count
      mov   dx,REMOTEBYTECOUNT1
      out   dx,al                         ;high remote byte count

      mov   al,rcr
      mov   dx,RECEIVECONFIGURATION       ;receive configuration register
      out   dx,al

      mov   al,20h
      mov   dx,TRANSMITPAGE               ;transmit page start
      out   dx,al

      mov   al,02
      mov   dx,TRANSMITCONFIGURATION
      out   dx,al                         ;temporarily go into Loopback mode

      mov   al,26h
      mov   dx,PAGESTART                  ;page start
      out   dx,al

      mov   dx,BOUNDARY                   ;boundary register
      out   dx,al
      mov   al,40h

      mov   dx,PAGESTOP                   ;page stop
      out   dx,al

      mov   al,61h                        ;go to page 1 registers
      mov   dx,COMMAND
      out   dx,al

      mov   al,26h
      mov   dx,CURRENT                    ;current page register
      out   dx,al

      mov   al,22h                        ;back to page 0, start mode
      mov   dx,COMMAND
      out   dx,al

      mov   al,0ffh
      mov   dx,INTERRUPTSTATUS            ;interrupt status register
      out   dx,al

      mov   al,imr
      mov   dx,INTERRUPTMASK              ;interrupt mask register
      out   dx,al

      mov   dx,TRANSMITCONFIGURATION
      mov   al,tcr
      out   dx,al                         ;TCR in normal mode, NIC is now
                                          ;ready for reception
            ret
      DriverInitialize endp

      Code  ends
            end
```

```
;***********************************************************************
;                       DriverSend

; Either transmits a packet passed to it or queues up the
; packet if the transmitter is busy (COMMAND register = 26h).
; Routine is called from upper layer software.

;        Entry: ds:si => packet to be transmitted
                   cx => byte count of packet
;***********************************************************************
;**********************Equates for NIC Registers***********************

COMMAND                 equ       300h
PAGESTART               equ       COMMAND+1
PAGESTOP                equ       COMMAND+2
BOUNDARY                equ       COMMAND+3
TRANSMITSTATUS          equ       COMMAND+4
TRANSMITPAGE            equ       COMMAND+4
TRANSMITBYTECOUNT0      equ       COMMAND+5
NCR                     equ       COMMAND+5
TRANSMITBYTECOUNT1      equ       COMMAND+6
INTERRUPTSTATUS         equ       COMMAND+7
CURRENT                 equ       COMMAND+7      ;in page 1
REMOTESTARTADDRESS0     equ       COMMAND+8
CRDMA0                  equ       COMMAND+8
REMOTESTARTADDRESS1     equ       COMMAND+9
CRDMA1                  equ       COMMAND+9
REMOTEBYTECOUNT0        equ       COMMAND+0ah
REMOTEBYTECOUNT1        equ       COMMAND+0bh
RECEIVESTATUS           equ       COMMAND+0ch
RECEIVECONFIGURATION    equ       COMMAND+0ch
TRANSMITCONFIGURATION   equ       COMMAND+0dh
FAE_TALLY               equ       COMMAND+0dh
DATACONFIGURATION       equ       COMMAND+0eh
CRC_TALLY               equ       COMMAND+0eh
INTERRUPTMASK           equ       COMMAND+0fh
MISS_PKT_TALLY          equ       COMMAND+0fh
IOPORT                  equ       COMMAND+10h

PSTART                  equ       46h
PSTOP                   equ       80h
TRANSMITBUFFER          equ       40h

.CODE

DriverSend      proc    near
     public     DriverSend
     cli                         ;disable interrupts
     mov        dx,COMMAND
     in         al,dx            ;read NIC command register
     cmp        26h              ;transmitting?
     je         QueueIt          ;if so, queue packet
     push       cx                ;store byte count
     mov        ah,TRANSMITBUFFER
     xor        al,al             ;set page to transfer packet to
     call       PCtoNIC           ;transfer packet to NIC buffer RAM
     mov        dx,TRANSMITPAGE
     mov        al,TRANSMITBUFFER
     out        dx,al             ;set NIC transmit page
     pop        cx                ;get byte count back
     mov        dx,TRANSMITBYTECOUNT0
     mov        al,cl
     out        dx,al             ;set transmit byte count 0 on NIC
     mov        dx,TRANSMITBYTECOUNT1
     mov        al,ch
     out        dx,al             ;set transmit byte count 1 on NIC
     mov        dx,COMMAND
     mov        al,26h
     out        dx,al             ;issue transmit to COMMAND register
     jmp        Finished

QueueIt:
                call Queue_packet
Finished:                         ;enable interrupts
                sti
                ret
DriverSend      endp
```

```
;***********************************************************************
;                                           PCtoNIC
;
; This routine will transfer a packet from the PC's RAM
; to the local RAM on the NIC card.
;
;     assumes: ds: si = packet to be transferred
;              cx     = byte count
;              ax     = NIC buffer page to transfer to
;***********************************************************************
        public  __PCtoNIC
PCtoNIC proc     far
        push     ax                       ; save buffer address
        inc      cx                       ; make even
        and      cx,0ffeh
        mov      dx,REMOTEBYTECOUNT0      ; set byte count low byte
        mov      al,cl
        out      dx,al
        mov      dx,REMOTEBYTECOUNT1      ; set byte count high byte
        mov      al,ch
        out      dx,al
        pop      ax                       ; get our page back
        mov      dx,REMOTESTARTADDRESS0
        out      dx,al                    ; set as lo address
        mov      dx,REMOTESTARTADDRESS1
        mov      al,ah                    ; set as hi address
        out      dx,al
        mov      dx,COMMAND
        mov      al,12h                   ; write and start
        out      dx,al
        mov      dx,IOPORT
        shr      cx,1                     ; need to loop half as many times
Writing_Word:                             ;because of word-wide transfers
        lodsw                             ;load word from ds:si
        out      dx,ax                    ;write to IOPORT on NIC board
        loop     Writing_Word
        mov      cx,0
        mov      dx,INTERRUPTSTATUS
CheckDMA:
        in       al,dx
        test     al,40h                   ; dma done ???
        jnz      toNICEND                 ; if so, go to NICEND
        jmp      CheckDMA                 ;loop until done
toNICEND:
        mov      dx,INTERRUPTSTATUS
        mov      al,40h                   ;clear DMA interrupt bit in ISR
        out      dx,al
        clc
        ret
PCtoNIC          endp
```

```
;************************************************************************
;                                           NICtoPC
;
; This routine will transfer a packet from the RAM
; on the NIC card to the RAM in the PC.
;
;     assumes: es: di = packet to be transferred
;              cx     = byte count
;              ax     = NIC buffer page to transfer from
;************************************************************************
        public  _NICtoPC
_NICtoPC        proc    far
        push    ax                      ; save buffer address
        inc     cx                      ; make even
        and     cx,0fffeh
        mov     dx,REMOTEBYTECOUNT0
        mov     al,cl
        out     dx,al
        mov     dx,REMOTEBYTECOUNT1
        mov     al,ch
        out     dx,al
        pop     ax                      ; get our page back
        mov     dx,REMOTESTARTADDRESS0
        out     dx,al                   ; set as low address
        mov     dx,REMOTESTARTADDRESS1
        mov     al,ah
        out     dx,al                   ; set as hi address
        mov     dx,COMMAND
        mov     al,0ah                  ; read and start
        out     dx,al
        mov     dx,IOPORT
        shr     cx,1                    ; need to loop half as many times
Writing_Word:                           ;because of word-wide transfers
        in      ax,dx
        stosw                           ;read word and store in es:di
        loop    Reading_Word
        mov     dx,INTERRUPTSTATUS
CheckDMA:
        in      al,dx
        test    al,40h
        jnz     ReadEnd
        jmp     CheckDMA
ReadEnd:
        out     dx,al                   ; clear RDMA bit in NIC ISR
        ret
_NICtoPC        endp
```

```
;************************************************************************
;                                          DriverISR

; This interrupt service routine responds to transmit, transmit error, and
; receive interrupts (the PTX, TXE, and PRX bits in the INTERRUPT STATUS
; register) produced from the NIC. Upon transmit interrupts, the upper
; layer software is informed of successful or erroneous transmissions;
; upon receive interrupts, packets are removed from the Receive Buffer
; Ring (in local memory) and transferred to the PC.

;************************************************************************

;***********************Equates for NIC Registers************************

COMMAND                equ   300h
PAGESTART              equ   COMMAND+1
PAGESTOP               equ   COMMAND+2
BOUNDARY               equ   COMMAND+3
TRANSMITSTATUS         equ   COMMAND+4
TRANSMITPAGE           equ   COMMAND+4
TRANSMITBYTECOUNT0     equ   COMMAND+5
NCR                    equ   COMMAND+5
TRANSMITBYTECOUNT1     equ   COMMAND+6
INTERRUPTSTATUS        equ   COMMAND+7
CURRENT                equ   COMMAND+7      ;in page 1
REMOTESTARTADDRESS0    equ   COMMAND+8
CRDMA0                 equ   COMMAND+8
REMOTESTARTADDRESS1    equ   COMMAND+9
CRDMA1                 equ   COMMAND+9
REMOTEBYTECOUNT0       equ   COMMAND+0ah
REMOTEBYTECOUNT1       equ   COMMAND+0bh
RECEIVESTATUS          equ   COMMAND+0ch
RECEIVECONFIGURATION   equ   COMMAND+0ch
TRANSMITCONFIGURATION  equ   COMMAND+0dh
FAE_TALLY              equ   COMMAND+0dh
DATACONFIGURATION      equ   COMMAND+0eh
CRC_TALLY              equ   COMMAND+0eh
INTERRUPTMASK          equ   COMMAND+0fh
MISS_PKT_TALLY         equ   COMMAND+0fh

PSTART                 equ   46h
PSTOP                  equ   80h

CGroup  group   Code
Code    segment para public 'Code'
        assume  cs:CGroup, ds:CGroup, es:nothing, ss:nothing
;  External routines
    extrn   DriverSend: near
byte_count      dw     ?
imr             db     1bh                  ;image of Interrupt Mask register
```

```
;************************************************************************
;  Begin of Interrupt Service Routine
;************************************************************************
netisr proc near
      public netisr
      cli
      push  ax                    ;save regs
      push  bx
      push  cx
      push  dx
      push  di
      push  si
      push  ds
      push  es
      push  bp
      mov   al,0bch
      out   21h,al                ;turn off IRQ3
      sti
      mov   ax,CGroup
      mov   ds,ax                 ;ds=cs

;************************************************************************
;   Read INTERRUPT STATUS REGISTER for receive packets, transmitted
;   packets and errored transmitted packets.
;************************************************************************

poll:
      mov   dx,INTERRUPTSTATUS
      in    al,dx
      test  al,1                  ;packet received?
      jnz   pkt_recv_rt
      test  al,0ah                ;packet transmitted?
      jz    exit_isr              ;no, let's exit
      jmp   pkt_tx_rt

exit_isr:
      mov   dx,INTERRUPTMASK      ;disabling NIC's intr
      mov   al,0
      out   dx,al
      cli
      mov   al,0b4h               ;turn IRQ3 back on
      out   21h,al
      mov   al,63h                ;send 'EOI' for IRQ3
      out   20h,al
      sti

      mov   dx,INTERRUPTMASK      ;NOTE: intr from the NIC
      mov   al,imr                ; are enabled at this point so
      out   dx,al                 ; that the 8259 interrupt
                                  ; controller does not miss any
                                  ; IRQ edges from the NIC
                                  ; (IRQ is edge sensitive)

      pop   bp
      pop   es
      pop   ds
      pop   si
      pop   di
      pop   dx
      pop   cx
      pop   bx
      pop   ax
      iret
```

```
;**********************************************************************
;   Packet Receive Routine (pkt_recv_rt) - clears out all good
;   packets in local receive buffer ring. Bad packets are ignored.
;**********************************************************************

pkt_recv_rt:
      mov   dx,INTERRUPTSTATUS
      in    al,dx
      test  al,10h                ;test for a Ring overflow
      jnz   ring_ovfl
      mov   al,1
      out   dx,al                 ;reset PRX bit in ISR
      mov   ax,next_packet
      mov   cx,packet_length
      mov   es,seq recv_pc_buff
      mov   di,offset recv_pc_buff
      NICtoPC

;**********************************************************************
;
;   Inform upper layer software of a received packet to be processed
;
;**********************************************************************
;   checking to see if receive buffer ring is empty
check_ring:
      mov   dx,BOUNDARY
      in    al,dx
      mov   ah,al                 ;save BOUNDARY in ah
      mov   dx,COMMAND
      mov   al,62h
      out   dx,al                 ;switched to pg 1 of NIC
      mov   dx,CURRENT
      in    al,dx
      mov   bh,al                 ;bh = CURRENT PAGE register
      mov   dx,COMMAND
      mov   al,22h
      out   dx,al                 ;switched back to pg 0
      cmp   ah,bh                 ;recv buff ring empty?
      jne   pkt_recv_rt
      jmp   poll
;**********************************************************************
;
;   The following code is required to recover from a Ring overflow.
;   See Sec. 2.0 of datasheet addendum.
;
;**********************************************************************

ring_ovfl:
      mov   dx,COMMAND
      mov   al,21h
      out   dx,al                 ;put NIC in stop mode

      mov   dx,REMOTEBYTECOUNT0
      xor   al,al
      out   dx,al
      mov   dx,REMOTEBYTECOUNT1
      out   dx,al

      mov   dx,_INTERRUPTSTATUS
      mov   cx,7fffh              ;load time out counter
```

```
wait_for_stop:
      in    al,dx
      test  al,80h               ;look for RST bit to be set
      loop  wait_for_stop        ; if we fall thru this loop, the RST bit may not get
                                 ; set because the NIC was currently transmitting
      mov   dx,TRANSMITCONFIGURATION
      mov   al,2
      out   dx,al                ;into loopback mode 1
      mov   dx,COMMAND
      mov   al,22h
      out   dx,al                ;into stop mode

      mov   ax,next_packet
      mov   cx,packet_length
      mov   es,seg recv_pc_buff
      mov   di,offset recv_pc_buff
      NICtoPC

      mov   dx,INTERRUPTSTATUS
      mov   al,10h
      out   dx,al                ;clear Overflow bit

      mov   dx,TRANSMITCONFIGURATION
      mov   al,tcr
      out   dx,al                ;put TCR back to normal mode
      jmp   check_ring

;************************************************************************
;
;   packet_transmit_routine (pkt_tx_rt) -determine status of
;   transmitted packet, then checks the transmit-pending
;   queue for the next available packet to transmit.
;
;************************************************************************

pkt_tx_rt:
      mov   dx,INTERRUPTSTATUS
      mov   al,0ah
      out   dx,al                ;reset PTX and TXE bits in ISR

      mov   dx,TRANSMITSTATUS    ;check for erroneous TX
      in    al,dx
      test  al,38h               ;is FU, CRS, or ABT bits set in TSR
      jnz   bad_tx

;************************************************************************
;   Inform upper layer software of successful transmission
;************************************************************************

      jmp   chk_tx_queue
bad_tx:                          ;in here if bad TX

;************************************************************************
;
;   Inform upper layer software of erroneous transmission
;
;************************************************************************
```

```
chk_tx_queue:
      call  Check_Queue         ; see if a packet is in queue
                                ; assume Check_Queue will a non-zero
      cmp   cx,0                ; value in cx and pointer to the
      je    poll                ; packet in DS:SI if packet is
      call  DriverSend          ; available. Returns cx = 0 otherwise
      jmp   poll
netisr endp
```

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.