REALTEK SINGLE CHIP FAST ETHERNET CONTROLLER WITH POWER MANAGEMENT RTL8100 PROGRAMMING GUIDE

This document is intended for use by the software engineer when programming for the Realtek RTL8100 series NIC controller chips. Information pertaining to the hardware design of products using these chips is contained in a separate document.

Though every effort has been made to assure that this document is current and accurate, more information may have become available subsequent to the production of this programming guide. In that event, please contact your Realtek representative for additional information which can help in the development process.

1 Packet Transmission

1.1 Architecture

The transmit path of the RTL8100 uses 4 descriptors, each descriptor with a fixed IO address offset. The 4 descriptors are used in a round-robin fashion. As a descriptor is written, PCI operations start and move packets in the memory which the descriptor specifies to the Transmit FIFO. The transmit FIFO is a 2k byte buffer in the chip which holds the data which will be moved to the line (cable). Data in the Transmit FIFO starts to move to the line when the early transmit threshold is met. The early transmit threshold is also specified in the descriptor.

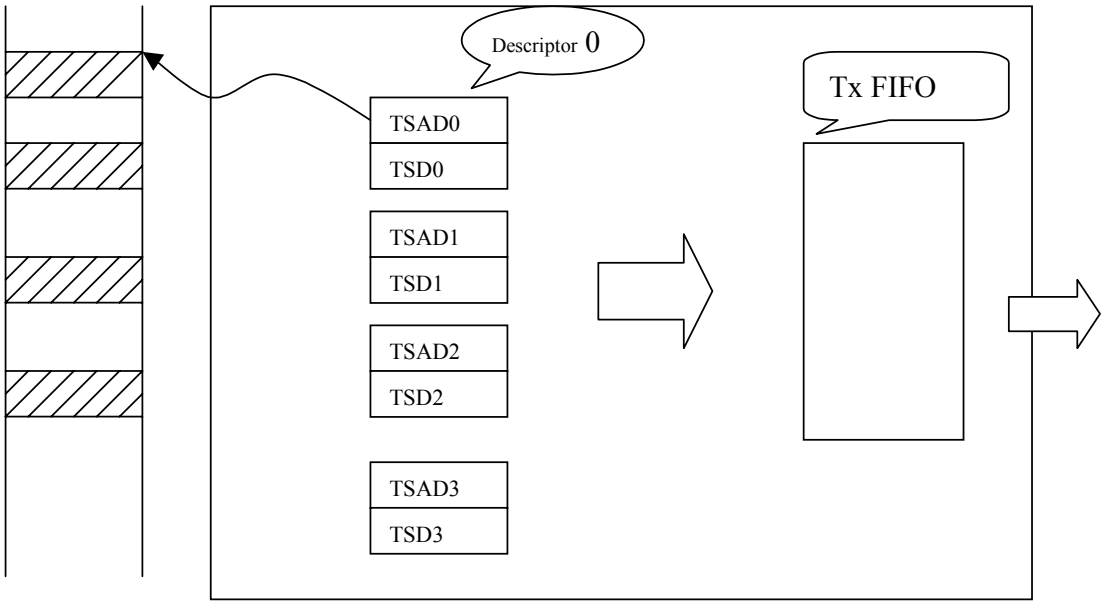
1.2 Transmit Descriptors

A transmit descriptor consist of 2 registers, which are specified below.

Register 1: Transmit Start Address (TSAD0-3) register. The physical address of each packet (Note: the packet must be in a continuous physical memory)

Register 2: Transmit Status (TSD0-3) register. A detailed description of this register is listed below.

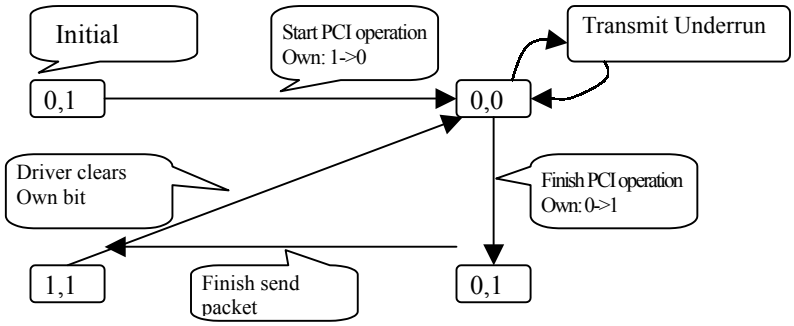
Bit	R/W	Symbol	Description
31	R	CRS	**Carrier Sense Lost:** This bit is set to 1 when the carrier is lost during transmission of a packet.
30	R	TABT	**Transmit Abort:** This bit is set to 1 if the transmission of a packet was aborted. This bit is read only, writing to this bit is not affected.
29	R	OWC	**Out of Window Collision:** This bit is set to 1 if the RTL8100 encountered an "out of window" collision during the transmission of a packet.
28	R	CDH	**CD Heart Beat:** The same as RTL8029(AS). This bit is cleared in the 100Mbps mode.
27-24	R	NCC3-0	**Number of Collision Count:** Indicates the number of collisions encountered during the transmission of a packet.
23-22	-	-	**Reserved**
21-16	R/W	ERTXTH5-0	**Early Tx Threshold:** Specifies the threshold level in the Tx FIFO to begin the transmission. When the byte count of the data in the Tx FIFO reaches this level, (or the FIFO contains at least one complete packet) the RTL8100 will transmit this packet. 000000 = 8 bytes These fields count from 000001 to 111111 in unit of 32 bytes. This threshold must be avoided from exceeding 2K byte.
15	R	TOK	**Transmit OK:** Set to 1 indicates that the transmission of a packet was completed successfully and no transmit underrun occurs.
14	R	TUN	**Transmit FIFO Underrun:** Set to 1 if the Tx FIFO was exhausted during the transmission of a packet. The RTL8100 can re-transfer data if the Tx FIFO underruns and can also transmit the packet to the wire successfully even though the Tx FIFO underruns. That is, when TSD<TUN>=1, TSD<TOK>=0 and ISR<TOK>=1 (or ISR<TER>=1).
13	R/W	OWN	**OWN:** The RTL8100 sets this bit to 1 when the Tx DMA operation of this descriptor was completed. The driver must set this bit to 0 when the Transmit Byte Count (bit0-12) is written. The default value is 1.
12-0	R/W	SIZE	**Descriptor Size:** The total size in bytes of the data in this descriptor. If the packet length is more than 1792 byte (0700h), the Tx queue will be invalid, i.e. the next descriptor will be written only after the OWN bit of that long packet's descriptor has been set.



1.3 The Transmission Process

The following process describes the transmission of a packet.

1. The packet is copied to a physically continuous buffer in memory.
2. The appropriate descriptor is written as follows.
   a. Enter the physical address of this buffer into the Start Address register.
   b. Enter the size of this packet, and the early transmit threshold into the Transmit Status register. Also, clear the OWN bit in TSD. This starts the PCI operation.
3. As the data moved into the FIFO meets the early transmit threshold, the chip starts to move data from the FIFO to the line.
4. When the whole packet is moved to the FIFO, the OWN bit is set to 1.
5. When the whole packet is moved to the line, the TOK (in TSD) is set to 1.
6. If TOK (IMR) is set to 1 and TOK (ISR) is set, then an interrupt is triggered.
7. When an interrupt service routine is called, the driver should clear TOK (ISR) State Diagram: (TOK,OWN)



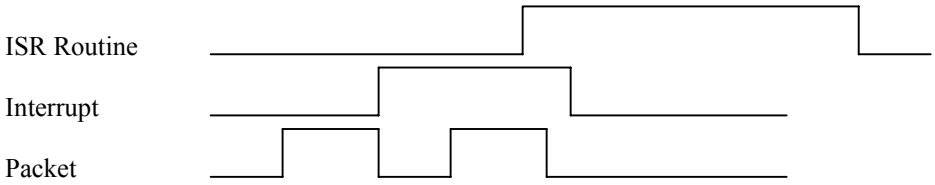
1.4 Registers Involved

1. TSAD0-3
2. TSD0-3
3. ISR (TOK,TER),IMR (TOK,TER)
4. TCR: Transmit Configuration register
5. TSAD: Reflects the corresponding bits in the TSD0-3.

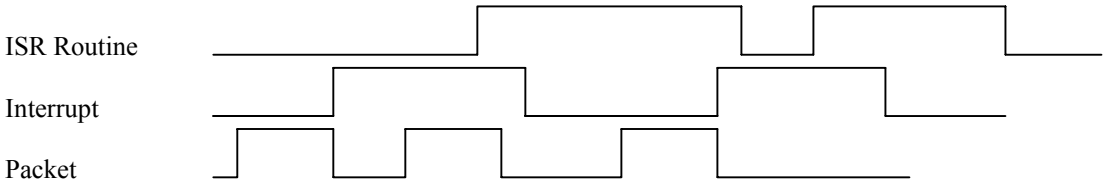
1.5 Software Issues

This section covers the handling of Interrupts. When the driver is processing a transmit interrupt, the following two cases should be managed properly.

Case 1: More than one interrupt between TOK and when ISR routine called.
=>Drivers have to check as many descriptor as possible.



Case 2: No packet TOK needs to be handled, but the ISR routine is called.



1.6 Configuration

The Maximum DMA burst size (MXDMA) per Tx DMA burst should be considered carefully. It is recommended to use the value of 1024 bytes.

1.7 Sample Code

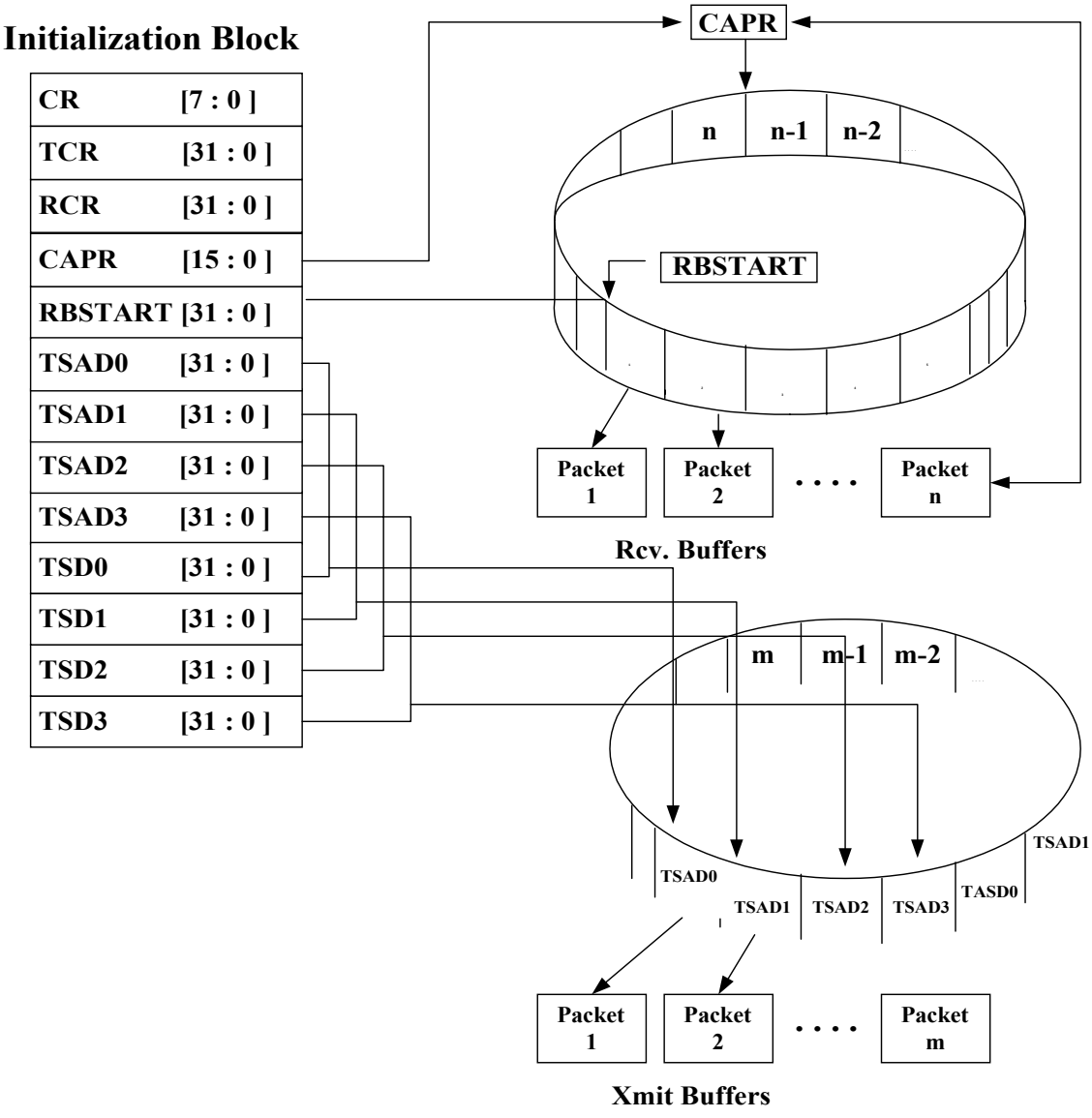
```
unsigned char
NextDesc(
        unsigned char CurrentDescriptor
    )
{
//    (CurrentDescriptor == TX_SW_BUFFER_NUM-1) ? 0 : (1 + CurrentDescriptor);
    if(CurrentDescriptor == TX_SW_BUFFER_NUM-1)
    {
        return  0;
    }
    else
    {
        return ( 1 + CurrentDescriptor);
    }
}
unsigned char
CheckTSDStatus(
    unsigned char           Desc
    )
{
    ULONG       Offset = Desc << 2;
    ULONG       tmpTSD;

    tmpTSD=inpdw(IOBase + TSD0 + Offset);
    switch ( tmpTSD & (TSD_OWN | TSD_TOK) )
    {
        case (TSD_OWN | TSD_TOK):           return   TSDSTATUS_BOTH;
        case (TSD_TOK)                  :       return  TSDSTATUS_TOK;
        case (TSD_OWN)                  :       return  TSDSTATUS_OWN;
        case 0                          :       return  TSDSTATUS_0;
    }
    return 0;
}
void
IssueCMD(unsigned char descriptor)
{
        unsigned long offset = descriptor << 2;
        outpdw(IOBase + TSAD0 + offset, TxDesc[TxHwSetupPtr].PhysicalAddress);
        outpdw(IOBase + TSD0 + offset , TxDesc[TxHwSetupPtr].PacketLength);
}
int
SendPacket(
        PPACKET pPacket
)
{
    disable();
    if( TxHwFreeDesc>0  )
    {
        TxDesc[TxHwSetupPtr].PacketLength=
                CopyFromPacketToBuffer( pPacket , TxDesc[TxHwSetupPtr].buffer);
        IssueCMD(TxHwSetupPtr);
        TxHwSetupPtr = NextDesc(TxHwSetupPtr);
        TxHwFreeDesc--;
        enable();
        return TRUE;//success
    }
    else
    {
        enable();
        return FALSE;//out of resource
    }
}
void
TxInterruptHandler()
{
    while( (CheckTSDStatus(TxHwFinishPtr) == TSDSTATUS_BOTH  ) &&
            (TxHwFreeDesc < 4                                 )  )
    {
        //can Release this buffer now
        TxHwFinishPtr = NextDesc(TxHwFinishPtr);
        TxHwFreeDesc++;
    }
}
```

2 Packet Reception

2.1 Architecture

The receive path of the RTL8100 is designed as a ring buffer. This ring buffer is a physical continuous memory structure. Data coming from the line is first stored in a Receive FIFO in the chip, and then moved to the receive buffer when the early receive threshold is met. The register CBA keeps the current address of the data moved to the buffer. CAPR is then a read pointer which keeps the address of data that the driver had read. The receiving packet status is stored in front of the packet (packet header).



2.2 The Packet Header

Bit	R/W	Symbol	Description
15	R	MAR	**Multicast Address Received:** This bit set to 1 indicates that a multicast packet is received.
14	R	PAM	**Physical Address Matched:** This bit set to 1 indicates that the destination address of this packet matches the value written in ID registers.
13	R	BAR	**Broadcast Address Received:** This bit set to 1 indicates that a broadcast packet is received. BAR, MAR bit will not be set simultaneously.
12-6	-	-	**Reserved**
5	R	ISE	**Invalid Symbol Error:** (100BASE-TX only) This bit set to 1 indicates that an invalid symbol was encountered during the reception of this packet.
4	R	RUNT	**Runt Packet Received:** This bit set to 1 indicates that the received packet length is smaller than 64 bytes (i.e. media header + data + CRC < 64 bytes)
3	R	LONG	**Long Packet:** This bit set to 1 indicates that the size of the received packet exceeds 4k bytes.
2	R	CRC	**CRC Error:** When set, indicates that a CRC error occurred on the received packet.
1	R	FAE	**Frame Alignment Error:** When set, indicates that a frame alignment error occurred on this received packet.
0	R	ROK	**Receive OK:** When set, indicates that a good packet is received.

2.3 The Transmission Process

The following process describes the reception of a packet.

1. Data received from the line is stored in the receive FIFO.
2. When the early receive threshold is met, data is moved from the FIFO to the receive buffer.
3. After the whole packet is moved from the FIFO to the receive buffer, the receive packet header (receive status and packet length) is written in front of the packet. CBA is updated to the end of the packet.
4. CMD (BufferEmpty) and ISR (TOK) is set.
5. An ISR routine is called and then the driver clears ISR (TOK) and updates CAPR.

2.4 Registers Involved

1. RBStart: Receive Buffer start address.
2. CR (BufferEmpty): Indicates if driver is empty.
3. CAPR: Buffer read pointer.
4. CBP: Buffer write pointer.
5. ISR/IMR (ROK, RER, RxOverflow, RxFIFOOverflow)
6. RCR: Receive Configuration register.
7. Packet Header.

2.5 Software Issues

This section covers the handling of various data reception topics.

1. Handling a Receive Buffer Overflow:
   The Rx DMA (FIFO to buffer) is stopped. The CAPR must be updated first to dismiss the ISR (RxBufferOverflow) event. The correct actions to process RxBufOvw are:
   a. Update CAPR.
   b. Write a '1' to ISR (ROK).
   The Rx DMA resumes after step b.

2. Handling RxFIFOOvw:
   When RxFIFOOvw occurs, all incoming packets are discarded. Clearing ISR (RxFIFOOvw) doesn't dismiss the RxFIFOOvw event. To dismiss the RxFIFOOvw event, the ISR (RxBufOvw) must be written with a '1'.

3. Rx FIFO early threshold:
   No early (FIFO->Buffer DMA starts when the whole packet is in the FIFO). If an incoming packet is larger than the size of the FIFO(2K), RxFIFOOvw will be set, but Rx DMA will never start, so the receive path is disabled.

   Note: Never set Rx FIFO early threshold to NoEarly.

4. Suggested handling:

```
if (RxFIFOOvw | RxBufOvw | ROK)
{
        clear ISR(RxFIFOOvw | RxBufOvw | ROK)
}
if (ROK)
{
        while(BufEmpty=0)
        {
                read one packet then update CAPR
        }
}
```

2.6 Configuration

The Maximum DMA burst size (MXDMA) per Rx DMA burst should be considered carefully. It is recommended to use the value of 1024 bytes.

When WRAP is enabled, the RTL8100 will move the reset of the packet data immediately after the buffer. This will make the last packet in the buffer continuous. However, the Receive buffer has to leave 1.5k additional space for this packet.

2.7 Sample Code

```
BOOLEAN
PacketOK(
        PPACKETHEADER pPktHdr
)
{
   BOOLEAN BadPacket = pPktHdr->RUNT ||
                       pPktHdr->LONG ||
                       pPktHdr->CRC  ||
                       pPktHdr->FAE;
   if( ( !BadPacket )   &&
       ( pPktHdr->ROK )   )
   {
       if ( (pPktHdr->PacketLength > RX_MAX_PACKET_LENGTH ) ||
            (pPktHdr->PacketLength < RX_MIN_PACKET_LENGTH )    )
       {
          return(FALSE);
       }
       PacketReceivedGood++;
       ByteReceived += pPktHdr->PacketLength;
       return TRUE ;
   }
   else
   {
       return FALSE;
   }
}

BOOLEAN
RxInterruptHandler(
   )
{
   unsigned char  TmpCMD;
   unsigned int   PktLength;
   unsigned char  *pIncomePacket, *RxReadPtr;
   PPACKETHEADER  pPacketHeader;

   while (TRUE)
   {
       TmpCMD = inportb(IOBase + CR);
       if (TmpCMD & CR_BUFE)
       {
          break;
       }
       do
       {
          RxReadPtr      = RxBuffer + RxReadPtrOffset;
          pPacketHeader = (PPACKETHEADER)  RxReadPtr;
          pIncomePacket = RxReadPtr + 4;
          PktLength      = pPacketHeader->PacketLength;       //this length include CRC
          if ( PacketOK( pPacketHeader ) )
          {
              if ( (RxReadPtrOffset + PktLength) > RX_BUFFER_SIZE )
              {     //wrap around to end of RxBuffer
                 memcpy( RxBuffer + RX_BUFFER_SIZE ,  RxBuffer,
                             (RxReadPtrOffset + PktLength - RX_BUFFER_SIZE)  );
              }
              //copy the packet out here
              CopyPacket(pIncomePacket,PktLength - 4);//don't copy 4 bytes CRC

              //update Read Pointer
              RxReadPtrOffset = (RxReadPtrOffset + PktLength + 4 + 3) & RX_READ_POINTER_MASK;
                      //4:for header length(PktLength include 4 bytes CRC)
                      //3:for dword alignment
              outport( IOBase + CAPR, RxReadPtrOffset - 0x10);     //-4:avoid overflow
          }
          else
          {
//                ResetRx();
              break;
          }
          TmpCMD = inportb(IOBase + CR);
       } while (!(TmpCMD & CR_BUFE));
   }
   return (TRUE);            //Done
}
```

3 Initialization

Though transmit reset and Receive reset can be done individually, there are three steps in the initialization procedure.

1. Enable Transmit/Receive (RE/TE in CommandRegister)
2. Configure TCR/RCR.
3. Enable IMR.

Additional Notes

This section covers some information on the source code provided.

1. This sample code has been developed under Borland C 3.0, and the debugging process was accomplished under Softice for DOS. All testing is done under DOS(win98).
2. To enable source code debugging under Softice, the compiling/linking process needs to generate a '.map. file. Softice provide a 'msym' program to translate '.map' files to '.sym' files. After the '.sym' file is generated, load the demo program with 'Ldr demo'.

Realtek Semiconductor Corp.
Headquarters
1F, No. 2, Industry East Road IX, Science-based
Industrial Park, Hsinchu, 300, Taiwan, R.O.C.
Tel: 886-3-5780211 Fax: 886-3-5776047
WWW: www.realtek.com.tw